Analytically Defined Spatiotemporal ConvNets for Spacetime Image Understanding

Isma Hadji

A Dissertation submitted to
the Faculty of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

Graduate Program in
Electrical Engineering and Computer Science
York University
Toronto, Ontario

December 2019

Abstract

This dissertation introduces a novel hierarchical spatiotemporal orientation representation for spacetime image analysis. This representation is designed to combine the benefits of the multilayer architecture of Convolutional Networks (ConvNets) and a more controlled approach to spacetime analysis. A distinguishing aspect of the approach is that unlike most contemporary convolutional networks no learning is involved; rather, all design decisions are specified analytically with theoretical motivations. This approach makes it possible to understand what information is being extracted at each stage and layer of processing as well as to minimize heuristic choices in design. Another key aspect of the network is its recurrent nature, whereby the output of each layer of processing feeds back to the input. The multilayer architecture that results systematically reveals hierarchical image structure in terms of multiscale, multiorientation properties of visual spacetime. To illustrate the utility of the proposed research, the designed networks has been tested on two spacetime image understanding tasks, dynamic texture recognition and video object segmentation. Further, the role of learning in the context of the proposed analytic approach to network design is systematically explored, thereby yielding a promising hybrid architecture. Finally, a new, large scale dynamic texture dataset is introduced and used for evaluation.

Acknowledgements

First, I would like to thank my supervisor, professor Richard Wildes, for opening the doors of his lab to me and the confidence he placed in me. His patience, understanding and support in different aspects of this journey made the completion of this dissertation possible. My experience being a PhD student in Professor Wildes' lab has been one of the highlights of my graduate life. Thanks to Professor Wildes' admirable mentorship, my initial inspirations turned into reality in his lab. Working with him over the past four years, I have learned more than I would have ever imagined when I first walked into his lab. For all these reasons and many more I will be eternally grateful.

I would also like to thank my amazing dad for making this intense desire to succeed an integral part of me, my mom for her endless prayers and unconditional love and my brothers for making me smile even in the worst days.

Finally, I wanted to thank my husband, Mahrez, for his unparalleled patience, support and love. His presence in my life alone brings out the best in me.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Multilayer representations have long played an important role in computer vision [146]. For example, standard widely used hand crafted features such as SIFT [110] can be seen as a shallow multilayer representation, which loosely speaking consists of a convolutional layer followed by pooling operations. Moreover, before the deep learning era, state-of-the-art recognition systems typically followed hand-crafted feature extraction with (learned) encodings followed by spatially organized pooling and a learned classifier (*e.g.* [48]), which also is a multilayer representational approach. Modern multilayer architectures push the idea of hierarchical data representation further, via extension to many layers, while typically eschewing hand designed features in favor of learning based approaches [74, 12, 174, 100, 95, 71, 163, 162, 169].

Overall, while the literature tackling multilayer networks is very large, with each

faction advocating for the benefits of one architecture over another, major computer vision research efforts have focused on convolutional neural networks, commonly referred to as ConvNets or CNNs. These efforts have resulted in new state-of-the-art performance on a wide range of classification (e.g [95, 169, 71]) and regression (e.g [199, 108, 42]) tasks. In contrast, while the history of such approaches can be traced back a number of years (e.g [59, 100]), theoretical understanding of how these systems achieve their outstanding results lags. Indeed, currently many contributions in the computer vision field use ConvNets as a black box that works while having a very vague idea for why it works, which is very unsatisfactory from a scientific point of view. In particular, there are two main complementary concerns: **(1)** For learned aspects (e.g convolutional kernels), exactly what has been learned? **(2)** For architecture design aspects (e.g number of layers, number of kernels/layer, pooling strategy, choice of nonlinearity), why are some choices better than others? The answers to these questions not only will improve the scientific understanding of ConvNets, but also increase their practical applicability.

Moreover, current realizations of ConvNets require massive amounts of data for training [90, 95, 100] and design decisions made greatly impact performance [25, 83]. Deeper theoretical understanding should lessen dependence on data-driven design. While empirical studies have investigated the operation of implemented networks, to date, their results largely have been limited to visualizations of internal processing at the different layers of a ConvNet [191, 161, 117].

Overall, there is both strong theoretical and practical motivation for taking a more analytic perspective on ConvNets than typically has been done to date.

1.2 Contributions

In response to the above noted state of affairs, this dissertation explores more controlled approaches to network realization and takes into account domain priors in ConvNet design. The main objective of this dissertation is to build on the success of ConvNets, while addressing their major shortcomings (*i.e.* lack of interpretability and heavy reliance on large scale datasets for training). To achieve these ends, the proposed research makes the following contributions:

- **SOE-Net**: Design of a purely analytically defined ConvNet architecture, where no learning is involved given that all network components are designed based on theoretical considerations. The designed network extracts measurements of Spatiotemporal Oriented Energy and is therefore dubbed, SOE-Net**.**

- **DTDB:** Design, realization and exploitation of the largest currently available Dynamic Texture DataBase. DTDB is used for analyzing the behavior of major classes of spatiotemporal ConvNets, including the proposed SOE-Net, to identify their strengths and weaknesses. The dataset is also made available to the research community.

- **MSOE-two-stream:** Design of an improved learning based spatiotemporal ConvNet architecture via introduction of domain priors into the network's design. In particular, based on the outcome of the analysis performed with DTDB, the most successful learning-based spatiotemporal ConvNet architecture (*i.e.* the two-stream architecture [162]) is improved via introduction of the MSOE motion representation [36] to replace the less optimal optical flow;

the resulting architecture is dubbed MSOE-two-stream. Here, MSOE refers to spatially Marginalized Spatiotemporal Oriented Energy, which is designed to capture pattern dynamics, apart from spatial appearance.

- **MSOE-SO-Net:** Combination of the proposed analytical approach to network design (*e.g.* SOE-Net) and the successful two-stream architecture to yield an analytically defined two-stream ConvNet architecture dubbed MSOE-SO-Net. Here, the Spatiotemporal Oriented Energy Network (SOE-Net) representation is factored into two components: MSOE-Net (spatially Marginalized Spatiotemporal Oriented Energy Network) for capturing dynamics and SO-Net (Spatial Orientation Network) for capturing spatial appearance.

- **Applications:** Validation of the designed architectures via application to two important computer vision tasks, dynamic texture recognition and video object segmentation.

- **Hybrid-SOE-Net:** Analysis of the role of learning in an otherwise completely analytically defined architecture to identify optimal learning strategies in this setting, yielding a hybrid (hand-crafted/learned) architecture.

1.3 Dissertation Overview

This dissertation is structured as follows: The present chapter has motivated the need for a better understanding of convolutional networks and provided a general overview of the most significant contributions of the research described in this dissertation.

Chapter 2 will provide the necessary background for the rest of the dissertation. In particular, it will describe current trends in ConvNet design and discuss the role of the building blocks of typical convolutional networks from both biological and theoretical perspectives. Importantly, it will also describe efforts made toward ConvNet understanding and highlight some critical outstanding shortcomings that remain, thereby highlighting the contributions made in this dissertation. Chapter 3 introduces the proposed analytically defined ConvNet architecture, which constitutes the backbone of this dissertation and sets the stage for the remainder of the document. Chapter 4 introduces the first application tackled in this research, *i.e.* dynamic texture recognition, and demonstrates the representational power of the proposed architecture as well as the role of each building block with application to dynamic texture recognition. With this baseline established, Chapter 5 further studies the properties of the defined architecture in comparison to the currently most popular learning-based networks. This study is enabled via the introduction of a new large scale dataset. Next, the analytically defined framework advocated in this dissertation is further tested on the fine grained task of video object segmentation in Chapter 6. The role of learning is then investigated in Chapter 7 to identify potential ways to realize a hybrid architecture that takes the best of both worlds. Finally, Chapter 8 provides a summary of the dissertation with suggestions for future research.

Chapter 2

Background

2.1 Introduction

This chapter gives a succinct description of the most prominent multilayer architectures used in computer vision and discusses the most notable approaches to understanding such architectures. Notably, while this chapter covers the most important contributions in the literature, it will not provide a comprehensive review, as such is available elsewhere [66]. Instead, the purpose of this chapter is to set the stage for the remainder of the document by clearly identifying where the contributions of this dissertation lie in the realm of the most recent literature.

2.2 Multilayer Architectures

Prior to the recent success of deep learning-based networks, state-of-the-art computer vision systems for recognition relied on three separate but complementary

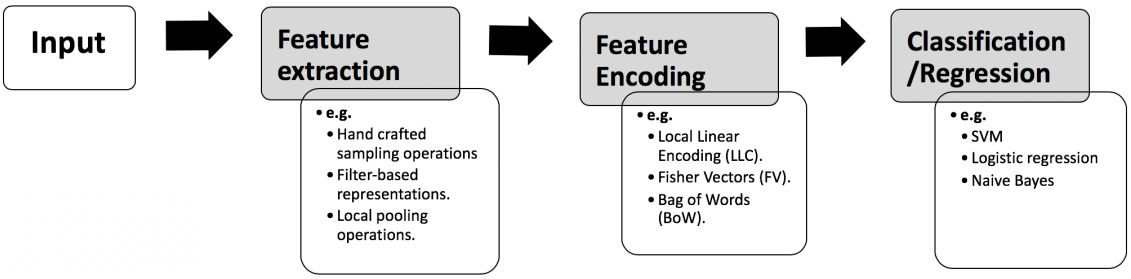


Figure 2.1: Typical pipeline used in computer vision algorithms prior to the wide success of deep learning-based networks.

steps (*e.g.* [47, 48]) as depicted in Figure 2.1. First, the input data is transformed to a suitable form via a set of hand designed operations (*e.g.* convolutions with a basis set). Second, an intermediate level, learning based, local or global encoding approach (*e.g.* improved Fisher Vectors, iFV [138]) is used to transform the data into a representation tuned to the task at hand. The transformations that the input incurs usually entail finding a compact and/or abstract representation of the input data, while injecting several invariances depending on the task at hand. The goal of this transformation is to change the data in a way that makes it more amenable to being readily separated by a classifier. Third, the transformed data is used to train some sort of classifier (*e.g.* Support Vector Machines [31]) to recognize the content of the input signal.

More recent deep-learning based architectures bring about a different outlook on the problem by proposing to learn, not only the classifier or intermediate level encodings, but also all stages of the transformation applied in the input data. This form of learning is commonly referred to as representation learning [99, 11], which when used in the context of deep multilayer architectures is called deep learning.

Multilayer architectures can be defined as computational models that allow for extracting useful information from the input data via multiple levels of abstraction. Generally, multilayer architectures are designed to amplify important aspects of the input at higher layers, while becoming more and more robust to less significant variations. For example, with application to object recognition, exact spatial position of an item may be suppressed to achieve shift-invariance. Such incremental abstraction via multilayer processing has a long history in computer vision (*e.g.* [110, 34, 98, 59, 100]). Most recent multilayer architectures stack simple building block modules with alternating linear and nonlinear functions. Over the years, a plethora of various multilayer architectures have been proposed (*e.g.* [74, 12, 174, 13, 75]). However, in the world of computer vision, Convolutional Networks (ConvNets) have seen the most success for several reasons as outlined in the next section.

2.2.1 Convolutional Networks

Convolutional networks are a particular type of neural network that are especially well adapted to computer vision applications because of their ability to hierarchically abstract representations with local operations. There are two key design ideas driving the success of convolutional architectures in computer vision. First, ConvNets take advantage of the organized structure of visual data (*i.e.* the 2D and 3D structure of images and videos, respectively) and the fact that pixel values within a neighborhood are usually highly correlated. Therefore, ConvNets eschew the use of one-to-one connections between all pixel units (*i.e.* as is the case of most so called

fully connected neural networks) in favor of using grouped local connections. Further, ConvNet architectures rely on feature sharing and each channel (or output feature map) is thereby generated from convolution with the same filter at all locations. This important characteristic of ConvNets leads to an architecture that relies on far fewer parameters compared to standard fully connected neural networks. Second, ConvNets also introduce a pooling step that provides a degree of translation invariance making the architecture less affected by small variations in position. Notably, pooling also allows the network to gradually see larger portions of the input thanks to an increased size of the network's receptive field. The increase in receptive field size (coupled with a decrease in the input's resolution) allows the network to represent more abstract characteristics of the input as the network's depth increases. For example, for the task of object recognition, it is advocated that ConvNets layers move from focusing on edges to parts of the object to finally cover the entire object at higher layers in the hierarchy [191, 195, 10].

Most ConvNets architectures deployed in recent computer vision applications are inspired by the successful architecture proposed by LeCun in 1998, now known as LeNet, for handwriting recognition [100], which in turn was inspired by the seminal work of Fukushima known as the Neocognitron [59]. As described in key literature [83, 101], a classical convolutional network (depicted in Figure 2.2) is made of four basic layers of processing: (i) a convolution layer, (ii) a nonlinearity or rectification layer, (iii) a normalization layer and (iv) a pooling layer. Also, a key component to the success of such an architecture was the use of the backpropagation algorithm [150] for relatively efficient learning of the convolutional parameters. Notably, this

algorithm also was foreseen in earlier investigations, *e.g.* [19, 179].

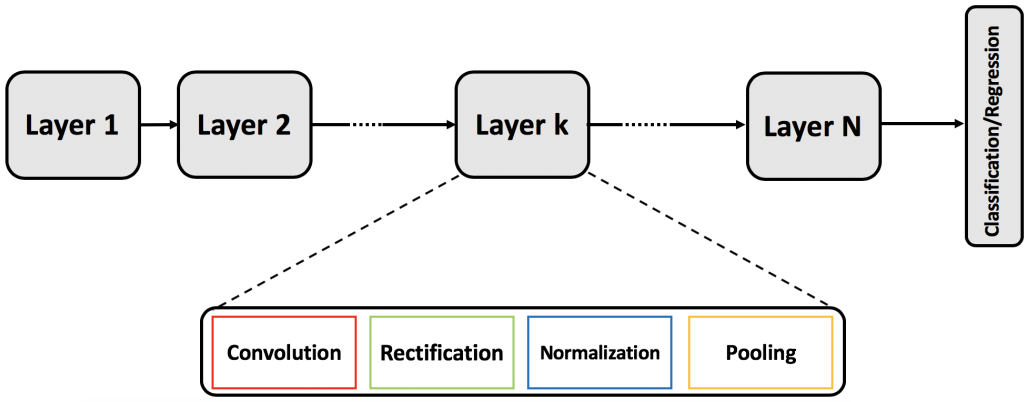


Figure 2.2: Typical ConvNet multilayer architecture with its 4 building blocks.

Although ConvNets allow for an optimized architecture that requires far fewer parameters compared to their fully connected neural network counterpart, their main shortcoming remains their heavy reliance on learning and labeled data. This data dependence is probably one of the main reasons why ConvNets were not widely used until 2012 when the availability of the large ImageNet dataset [151] and concomitant computational resources made it possible to revive interest in ConvNets [95].

In fact, with the availability of large scale datasets and powerful computers for training, the vision community has recently seen a surge in the use of ConvNets for various applications. The used networks vary from simply stacking convolutional layers followed by non-linearities such as Rectified Linear Units (ReLU) as done in AlexNet [95] or VGG [163] to more complex and deeper architectures relying on dimensionality reduction layers using 1×1 convolutions as done GoogLeNet [166] or additional skip connection as done in the now popular Residual Networks (ResNets) [71, 78].

The significant performance boost brought to various image based applications via use of ConvNets sparked interest in extending 2D spatial ConvNets to 3D spatiotemporal ConvNets for video analysis. Generally, the various spatiotemporal architectures proposed in the literature have simply tried to extend 2D architectures from the spatial domain, (x, y), into the spatiotemporal domain, (x, y, t). In the realm of training based spatiotemporal ConvNets, there are three different architectural designs that stand out: LSTM based networks (*e.g.* [128, 39]), which propagate information temporally via use of Long Short Term Memory (LSTM) units [75], 3D ConvNets (*e.g.* [169, 90]), which replace 2D filters with their 3D counterpart, and Two-Stream architectures (*e.g.* [162, 52]), which analyze appearance and motion information in two parallel pathways.

Importantly, while these networks achieve competitive results in many computer vision applications, their main shortcomings remain: the limited understanding of the exact nature of the learned representation, the reliance on massive training datasets, the lack of ability to support precise performance bounds and the lack of clarity regarding the choice of the networks' hyper parameters. These choices include the filters sizes, choice of nonlinearities, pooling functions and parameters as well as the number of layers and architectures themselves. Attempts at understanding the impact of these design choices as well as what is being learned by these ConvNets are discussed in the next section.

2.3 Understanding ConvNets

In general, methods aiming at understanding ConvNets can be divided into three tacks: those that rely on visualizations of the learned filters and the extracted feature maps, those that rely on ablation studies as inspired from biological approaches to understanding the visual cortex and those that rely on minimizing learning by introducing analytic principles into their network's design. Each of these approaches will be briefly reviewed in this section.

2.3.1 Understanding ConvNets via Visualization

Although several methods have been proposed in the literature for visualizing the feature maps extracted by ConvNets, in this section we will focus on the two most prominent approaches and discuss their different variations.

The first approach to ConvNet visualization is known as a dataset-centric approach [189] because it relies on probing the network with input from a dataset to find maximally responding units in the network. One of the earliest approaches falling under this category is known as DeConvNet [191], where visualization is achieved in two steps. First, a ConvNet is presented with several images from a dataset and the feature maps responding maximally to this input are recorded. Second, these feature maps are projected back to the image space using the DeConvNet architecture, which consists of blocks that approximately invert the operations of the ConvNet used. In particular, DeConvNet inverts the convolution operations (*i.e.* performs "de-convolution") via use of the transpose of the learned filters in the Con-

vNet under consideration. Here, it is worth noting that taking the transpose is not guaranteed to invert a convolution operation. For "un-pooling", DeConvNet relies on recording the locations corresponding to max-pooled responses in the ConvNet and uses those locations for "un-pooling" or upsampling the feature maps.

Typical visualizations resulting from these methods generally reveal that earlier layers in the network tend to capture low level features such as oriented bars and edges, *i.e.* filters learned at lower layers are similar to oriented bandpass filters. In contrast, at higher layers features captured progress from simple textures to more complex objects. Interestingly, these visualizations tend to conserve a high level of detail from the images that yielded a high response in the network [191]; see Figure 2 in [191] for sample visualizations. In fact, it seems like these visualization tend to emphasize the edges of the input images and mainly reveal the part of the image that is responsible for the high response (*i.e.* they can be seen as methods for finding the high contrast points of the input images and mainly reveal the part that is responsible for high classification results). Motivated by these observations, other approaches falling under the dataset-centric paradigm proposed even simpler methods to visualize what a network is learning. Examples include methods that progressively remove parts from images yielding high responses to highlight what specific parts are responsible for those high responses [195, 196]. Some of the conclusions that emerged from these approaches are that objects are largely responsible for recognizing scenes [195] or more generally that object detectors emerge as we visualize higher layers of the network [196]. However, this conclusion is likely heavily affected by the fact that most studied networks are trained using large scale object

recognition datasets.

The second approach to ConvNet visualization is known as the network-centric approach [189] because it uses the network parameters only without requiring any additional data for visualization purposes. This approach was first introduced in the context of deep belief networks [44] and later applied to ConvNets [161]. In this case, visualization is achieved by synthesizing an image that will maximize some unit's (or filter's) response. For example, starting from the last layer of a network that yields a class score, S_c, and an image initialized to random noise, I, the goal is to modify the image such that its score for belonging to class c is maximized. This optimization problem is defined in [161] according to

$$\arg\max_I S_c(I) - \lambda||I||_2^2, \tag{2.1}$$

where λ is a regularization parameter. Here, the L_2 regularization is used to penalize large values. Most other methods falling under this paradigm attempt to solve the same optimization problem while enforcing different regularization techniques, *e.g.* total variation regularization to enforce smoothness and avoid high frequency content in the synthesized image [117] or simple clipping of pixels that do not participate strongly in the filter's response and thereby only highlight the patterns responsible for a filter's activation [189]. Another variant falling under the network centric paradigm relies on Generative Adverserial Networks (GANs), *e.g.* [130, 129], and uses the encoding part of the architecture to render an image that will maximize a certain response in the generator.

Usually, visualizations obtained with network centric approaches suggest that the

network is learning high level shapes responsible for discriminating various objects (*e.g.* eyes, nose and ears when the target class is animals faces); however, the exact locations of these features does not matter [161, 117]; see Figures 1 and 6 in [117] for sample visualizations. The only exception here is seen in the GAN based visualization approaches that tend to render more natural images of the target class [130, 129], as it is optimized to achieve such natural visualizations. Generally, however, these visualizations imply that the network learns invariances progressively (*i.e.* it becomes position agnostic at higher layers), as expected from the use of various pooling operations. However, when this network-centric visualization technique is applied to invert lower layers, it does not necessarily show that they are learning lower level features such as edges as opposed to the dataset-centric technique. Instead, when the network centric approach is applied to visualize lower layers, it is usually seen that those lower layers retain most of the photographic information present in the image to be inverted (ie the visualization yields a blurred version of the image to be inverted) [117].

More generally, a major limitation of visualization approaches to understanding ConvNets is the subjective nature of the interpretation, which typically is based on visual inspection by the authors of the method.

2.3.2 Understanding ConvNets via Ablation Studies

Another popular method to shed light on ConvNet capabilities that is being widely used is the so called ablation study of the network. In fact, most prominent ConvNet papers (*e.g.* [25, 191, 162, 176, 50, 33, 84]) include an ablation study in their exper-

imental section, where the goal is to isolate the different components of the network to see how removing or adding any given block affects the overall performance.

These ablation studies have the potential to guide ConvNet practitioners toward "good practices" to achieve higher performance in different applications. For example, one of the earliest ablation studies in the context of ConvNets revealed the importance of proper rectification and normalization even while using randomly initialized networks [83]. Other work revealed the importance of deeper representations while using smaller filters at all layers [25, 163]. Yet other studies investigated the role of pre-training and finetuning as well as the number, location and magnitude of features. These investigations further highlighted the transferability of features extracted from ConvNets across tasks [7].

More interestingly, other work proposed to dissect ConvNets to investigate the impact of each component on the interpretability of the representations learned [10]. This approach relies on a dataset with pixel level annotations, where each pixel is assigned several "concept" labels that include color, texture, object and scene labels. Each unit in a ConvNet under consideration is evaluated for its ability to generate a segmentation mask that matches a certain concept-based segmentation mask L_c. In particular, each feature map, S_k, in the network is converted to a binary mask, M_k, where pixels are set to 1 only if their activation exceeds a certain pre-set threshold. Next, each unit, k, is assigned a score for its ability to segment a given concept, c, according to

$$IoU_{k,c} = \frac{\sum_{dataset} |M_k \cap L_c|}{\sum_{dataset} |M_k \cup L_c|} \tag{2.2}$$

where |.| is the cardinality of the set. With this measure, the interpretability of each unit is defined based on its ability to generate good segmentation masks. This measure revealed that units in lower layers are able to generate better color or texture based masks, whereas higher layers generate better object and scene segmentations. In line with dataset-centric visualization approaches, this observation suggests that lower layers are learning filters that capture lower level concepts while higher layers learn more abstract features such as objects parts. This approach also allowed for a systematic analysis of the effect of various nonlinearities and training strategies on interpretability and revealed that higher performance does not always yield highly interpretable units. For example, it was shown that regularization techniques such as batch normalization non-trivially affect a unit's interpretability as defined in this approach. Also, it was found that networks trained on scenes datasets yield more interpretable units compared to the widely used ImageNet training. Unfortunately, one of the main flaws of this approach is the fact that their definition of interpretability depends highly on the dataset used for evaluation. In other words, ConvNet units capable of capturing concepts that are not represented in the dataset will yield low Intersection over Union (*i.e.* IoU, (2.2)) scores; hence, will be deemed not interpretable by this method, even if the concept is visually interpretable. Therefore, this method can miss other important aspects revealed by the network components responsible for higher performance.

2.3.3 Understanding ConvNets via Controlled Design

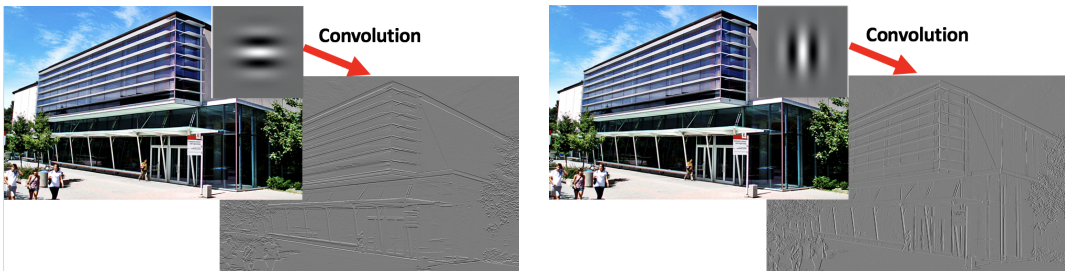
Another method to understand ConvNets is to minimize the number of learned parameters by injecting priors into the network design. For example, some methods reduce the number of learned filters per layer and include transformed versions of the learned filters in each layer to model rotation invariances, *e.g.* [112, 198]. Other approaches rely on replacing learned filters with a basis set and instead of learning filter parameters they learn how to combine the basis set to form the effective filters at each layer, *e.g.* [81, 30, 182, 112, 198]. Yet other approaches, push the idea of injecting priors into their network design even further by either partially or fully hand crafting their network and adapting it to a given task, which yields especially interpretable networks, *e.g.* [18, 133, 64]. This dissertation falls under this paradigm and therefore related work aiming at modeling the various building blocks of a ConvNet will be discussed more thoroughly in the next section.

2.4 Understanding ConvNets Building Blocks

2.4.1 Convolution

The convolutional layer is, arguably, one of the most important steps in ConvNet architectures. Basically, convolution is a linear, shift invariant operation that consists of performing local weighted combination across the input signal. Depending on the set of weights chosen (*i.e.* the chosen point spread function) different properties of the input signal are revealed, as illustrated in Figure 2.3. In the frequency domain, the correlate of the point spread function is the modulation transfer function that tells

how the frequency components of the input are modified through scaling and phase shifting [167]. Therefore, it is of paramount importance to select the right kernels to capture the most salient and important information contained in the input signal that allows for making strong inferences about the content of the signal. This section discusses some of the different ways proposed to tackle the kernel selection step.



(a) (b)

Figure 2.3: Effect of the chosen filter (*i.e.* the chosen point spread function) on the output of the convolution operation. (a) The input image is convolved with a horizontally oriented bandpass filter, thereby highlighting horizontal lines in the input. (b) The input image is convolved with a vertically oriented bandpass filter, thereby highlighting vertical lines in the input.

Biological Perspective

Neurophysiological evidence for hierarchical processing in the mamalian visual cortex provides an underlying inspiration for spatial and spatiotemporal ConvNets. In particular, research that hypothesized a cascade of simple and complex cells that progressively extract more abstract attributes of the visual input [80] has been of particular importance. At the very earliest stages of processing in the visual cortex, the simple cells were shown capable of detecting primitive features such as oriented gratings, bars and edges, with more complicated tunings emerging at subsequent

stages.

A popular choice for modeling the described properties of cortical simple cells is a set of oriented Gabor filters or Gaussian derivatives at various scales. More generally, filters selected at this level of processing typically are oriented bandpass filters. Many decades later most biological models still rely on the same set of simple cells at the initial layers of the hierarchy [145, 157, 158, 85, 9, 57]. In fact, these same Gabor kernels are also extended to the chromatic [192] and temporal [85] domains to account for color and motion sensitive neurons, respectively.

Matters become more subtle, however, when it comes to representing cells at higher areas of the visual cortex and most contributions building on Hubel and Wiesel's work strive to find an appropriate representation for these areas. The HMAX model is one of the most well known models tackling this issue [145]. The main idea of the HMAX model is that filters at higher layers of the hierarchy are obtained through the combination of filters from previous layers such that neurons at higher layers respond to co-activations of previous neurons. Extensions of this work thereafter explicitly introduce learning to model filters at higher layers [157, 158, 85].

Another related, although somewhat different, train of thought suggests that there exist more complex cells at higher levels of the hierarchy that are dedicated to capturing intermediate shape representations, *e.g.* curvature [147, 148]. While the HMAX class of models propose modeling shapes via compositions of feature types from previous layers, these investigations propose an approach that directly models hypercomplex cells (also referred to as endstopped cells) without resorting to learning. In particular, models falling within this paradigm model hypercomplex

cells via combination of simple and complex cells to generate new cells that are able to maximally respond to curvatures of different degrees and signs as well as different shapes at different locations.

More closely related to the approach adopted in this dissertation is another body of research, which advocates that the hierarchical processing (termed $Filter \rightarrow Rectify \rightarrow Filter$) that takes place in the visual cortex deals progressively with higher-order image structures [9, 57, 121]. It is therefore advocated that the same set of kernels present at the first layer (*i.e.* oriented bandpass filters) are repeated at higher layers. However, the processing at each layer reveals different properties of the input signal given that the same set of kernels now operate on different input obtained from a previous layer. Therefore, features extracted at successive layers progress from simple and local to abstract and global while capturing higher order statistics. In addition, joint statistics are also accounted for through the combination of layerwise responses across various scales and orientations.

Theoretical Perspective

More theoretically driven approaches are usually inspired from biology but strive to inject more theoretical justifications into their models. These methods usually vary depending on their kernel selection strategy.

One way of looking at the kernel selection problem is to consider that objects in the natural world are a collection of a set of primitive shapes and thereby adopt a shape based solution [56, 54, 55], in which case, the proposed algorithms start by finding the most primitive shapes in an image (*e.g.* oriented edges) using a bank

of oriented Gabor filters. Using these edges, or more generally parts, the algorithm proceeds by finding potential combinations of parts in the next layers by looking at increasingly bigger neighborhoods around each part, where the choice of the combination is based on the probabilities learned during unsupervised training. In reality, such a shape based approach is more of a proof of concept where only lower layers of the hierarchy can be learned in such an unsupervised way, whereas higher layers need to be learned using category specific images [54].

Another outlook on the kernel selection process is based on the observation that many training based convolutional networks learn redundant filters. Moreover, many of the learned filters at the first few layers of those networks resemble oriented band-pass filters. Therefore, several recent investigations aim at injecting priors into their network design with a specific focus on the convolutional filter selection. One approach proposes learning layerwise filters over a basis set of 2D derivative operators [81]. While this method uses a fixed basis set of filters, it relies on supervised learning to linearly combine the filters in the basis at each layer to yield the effective layer-wise filters and it is therefore dataset dependent. Nonetheless, using a basis set of filters and learning combinations aligns well with biological models, such as HMAX [145] and its successors (*e.g.* [158, 85]), and simplifies the network's architecture, while maintaining a certain level of interpretability. Also, as learning is one of the bottlenecks of modern ConvNets, using a basis set also eases this process by tremendously decreasing the number of parameters to be learned. For these reasons such approaches are gaining popularity in the most recent literature [81, 30, 182, 112, 198].

Interestingly, a common thread across these recent efforts is the aim of reducing

redundant kernels with a particular focus on modeling rotational invariance (although it is not necessarily a property of biological vision). The focus on rotation is motivated by the observation that learned filters often are rotated versions of one another. For example, one effort targeted learning of rotational equivariance by training over a set of circular harmonics [182]. Alternatively, other approaches attempt to hard code rotation invariance by changing the network structure itself such that for each learned filter a set of corresponding rotated versions are automatically generated either directly based on a predefined set of orientations, *e.g.* [198], or by convolving each learned filter with a basis set of oriented Gabor filters [112].

Other approaches push the idea of injecting priors into their network design even further by fully hand crafting their network via casting the kernel selection problem as an invariance maximization problem based on group theory, *e.g.* [18, 133, 30]. For example, kernels can be chosen such that they maximize invariances to small deformations and translations for texture recognition [18] or to maximize rotation invariance for object recognition [133]. This approach is most akin to the work proposed in this dissertation as will be demonstrated throughout the next chapters.

Arguably, the scattering transform network (ScatNet) has one of the most rigorous mathematical definitions to date [18]. The construction of scattering transforms starts from the assertion that a good image representation should be invariant to small, local deformations and various transformation groups depending on the task at hand. The kernels used in this method are a set of dilated and rotated wavelets, ψ_λ, where λ is the center frequency defined as $\lambda = 2^{-j}r$, with 2^{-j} and r representing the dilation and rotation, respectively. The network is constructed by a hierarchy of

convolutions using various wavelets centered around different frequencies, as well as various nonlinearities as discussed in the next section. The center frequencies of the employed kernels are chosen to be lower at each layer.

Loosely speaking both the work proposed in this dissertation and ScatNet fall under the $Filter \rightarrow Rectify \rightarrow Filter$ paradigm advocated by some biologically based models [9]. Because these networks are based on a rigorous mathematical analysis, they also take into account the frequency content of the signal as it is processed in each layer. One of the direct results of this design is the ability to make theory driven decisions regarding the different components involved in these networks.

Discussion

Overall, the ability of human visual cortex in recognizing the world while being invariant to various changes that are unimportant to the task at hand has been the driving force of many researchers in this field. Although, several approaches and theories have been proposed to model the different layers of the visual cortex, a common thread across these efforts is the agreement on the presence of hierarchical processing that splits the vision task into smaller pieces. However, while most models agree on the general choice of kernels at the initial layers, motivated by the seminal work of Hubel and Wiesel [80], modeling areas responsible for recognizing more abstract features seems to be more intricate and controversial.

On the other hand, most theoretically driven approaches to convolutional kernel selection aim at introducing priors into their hierarchical representations with the

ultimate goal of reducing the need for massive training. In doing so, these methods either rely on maximizing invariances through methods grounded in group theory or rely on combinations over basis sets. Interestingly, similar to more biologically inspired instantiations, it also is commonly observed that there is a pronounced tendency to model early layers with filters that have the appearance of oriented bandpass filters. However, the choice for higher layers' kernels remains an open critical question.

As previously mentioned, this dissertation argues in favor of methods supporting the use of the same set of kernels at all layers, even while operating on different input data at each layer. Notably, the input changes at each layer thanks to the presence of other nonlinear operations. The most common nonlinearities used throughout a typical ConvNet architecture and their roles are discussed next.

2.4.2 Rectification

Multilayer networks are typically highly nonlinear and rectification is, usually, the first stage of processing that introduces nonlinearities to the model. Rectification refers to applying a pointwise nonlinearity (also known as an activation function) to the output of the convolutional layer. Use of this term borrows from signal processing, wherein rectification refers to conversion from alternating to direct current. It is another processing step that finds motivation both from biological and theoretical viewpoints. Computational neuroscientists introduce the rectification step in an effort to find the appropriate models that best explain the neuroscientific data at hand. On the other hand, machine learning researchers use rectification to obtain

models that learn faster and better. Interestingly, both streams of research tend to agree, not only on the need for rectification, but they are also converging to the same type of rectification.

Biological Perspective

From a biological perspective, rectification nonlinearities are usually introduced into the computational models of neurons in order to explain their firing rates as a function of the input [35]. A fairly well accepted model for biological neuron's firing rate in general is referred to as *Leaky Integrate and Fire* (LIF) [35]. This model explains that the incoming signal to any neuron has to exceed a certain threshold for the cell to fire. Research investigating the cells in the visual cortex in particular also relies on a similar model, referred to as half wave rectification [80, 122, 72].

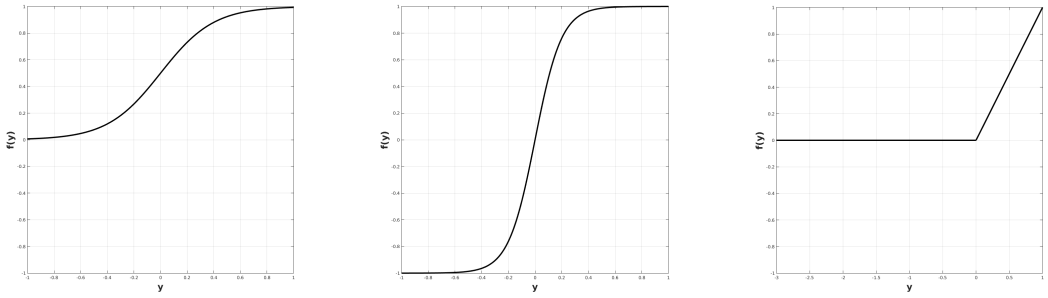
Notably, Hubel and Wiesel's seminal work already presented evidence that simple cells include nonlinear processing in terms of half wave rectification following on linear filtering [80]. As previously mentioned in Section 2.4.1, the linear operator itself can be considered as a convolution operation. It is known that, depending on the input signal, convolution can give rise to either positive or negative outputs. However, in reality cells' firing rates are by definition positive. This is the reason why Hubel and Wiesel suggested a nonlinearity in the form of a clipping operation that only takes into account the positive responses. More in line with the LIF model, other research suggested a slightly different half wave rectification in which the clipping operation happens based on a certain threshold (*i.e.* other than zero) [122]. More in line with the approach adopted in this dissertation, other models also took into account the

possible negative responses that may arise from the filtering operation [72, 73]. In this case, a two-path half wave rectification is suggested where the positive and negative incoming signals are clipped separately and carried in two separate paths. Also, in order to deal with the negative responses both signals are followed by a pointwise squaring operation and the rectification is therefore dubbed half-squaring (although, biological neurons do not necessarily share this property). In this model the cells are regarded as energy mechanisms of opposite phases that encode both the positive and negative outputs.

Theoretical Perspective

From a theoretical perspective, rectification is usually introduced by machine learning researchers for two main reasons. First, it is used to increase the discriminating power of the extracted features by allowing the network to learn more complex functions. Second, it allows for controlling the numerical representation of the data for faster learning. Historically, multilayer networks relied on pointwise sigmoidal nonlinearities using either the logistic nonlinearity or the hyperbolic tangent [100]. Although the logistic function is more biologically plausible given that it does not have a negative output, the hyperbolic tangent was more often used given that it has better properties for learning, such as a steady state around 0 (See Figures 2.4 (a) and (b), respectively). To account for the negative parts of the hyperbolic tangent activation function it is usually followed by a modulus operation (also referred to as *Absolute Value Rectification* AVR) [83]. However, recently the *Rectified Linear Unit* (ReLU), first introduced by Nair *et al*. [127], quickly became the default rectification

nonlinearity in many areas of investigation (*e.g.* [116]) and especially computer vision ever since its first successful application on the ImageNet dataset [95]. In particular, it was shown that the ReLU plays a key role against overfitting and expediting the training procedure, even while leading to better performance compared to traditional sigmoidal rectification functions [95].



(a) Logistic (b) tanh (c) ReLU

Figure 2.4: Typical nonlinear rectification functions used in ConvNets.

The ReLU operator, depicted in Figure 2.4 (c), has two main desirable properties for any learning based network. First, ReLU does not saturate for positive input given that its derivative is one for positive input. This property makes ReLU particularly attractive since it removes the problem of vanishing gradients usually present in networks relying on sigmoidal nonlinearities. Second, given that ReLU sets the output to zero when the input is negative, it introduces sparsity, which has the benefit of faster training and better classification accuracy. In particular, for improved classification it is usually desirable to have linearly separable features and sparse representations are usually more readily separable [62]. However, the hard 0 saturation on negative input comes with its own risks. Here, there are two complementary con-

cerns. First, due to the hard zero activation some parts of the network may never be trained if the paths to these parts were never activated. Second, in a degenerate case where all units at a given layer have a negative input, back propagation might fail and this result will lead to a situation that resembles the vanishing gradient problem. Because of these potential issues many improvements to the ReLU nonlinearity have been proposed to better deal with the case of negative outputs while keeping the advantages of ReLU.

Variations of the ReLU activation function include the *Leaky Rectified Linear Unit* (LReLU) [116] and its closely related *Parametric Rectified Linear Unit* (PReLU) [70] and the *Concatenated Rectified Linear Unit* (CReLU) [159]. A common thread across these variations is the introduction of modification to the ReLU operator such that the negative part of the signal is taken into account. This strategy is motivated by the observation that kernels learned at the initial layers of most ConvNets tend to form negatively correlated pairs (*i.e.* filters that are 180 degrees out of phase). This observation implies that the negative responses eliminated by the ReLU nonlinearity are replaced by learning kernels of opposite phase. By replacing ReLU with functions capable of dealing with the negative signal, the authors were able to demonstrate that a network designed to encode both parts of the signal leads to a better performance, while reducing the number of parameters to be learned by removing redundancies.

Discussion

Overall, it is seen that biologically motivated models of neuronal activation functions have become common practice in today's convolutional network algorithms and are,

in part, responsible for much of their success. Interestingly, the broad class of ReLU nonlinearities clearly became the most popular choice for the rectification stage from a machine learning perspective. Notably, the choice of completely neglecting the negative inputs (*i.e.* as done in ReLU) seems to be more questionable as evidenced by the many contributions proposing alternatives to this choice [116, 70, 88, 29, 159].

Motivated by these observations, which also are in consensus with some biological findings [72], this dissertation proposes a rectification strategy which takes into account both positive and negative parts of the signal as will be discussed in the next chapter.

2.4.3 Normalization

Multilayer architectures are highly nonlinear due to the cascade of nonlinear operations that take place in these networks. In addition to the rectification nonlinearity discussed in the previous section, normalization is another nonlinear block of processing that plays a significant role in ConvNet architectures. The most widely used form of normalization used in ConvNets is the so called *Divisive Normalization* or DN (also known as local response normalization). This section sheds light on the role of the normalization step and describes how it corrects for some of the shortcomings of the previous two blocks of processing (*i.e.* Convolution and Rectification). Once again the role of normalization will be discussed both from biological and theoretical perspectives.

Biological Perspective

Normalization was proposed early on by neurophysiologists to explain the phenomenon of light adaptation in the retina [16] and was later extended to explain the nonlinear properties of neurons in the mammalian visual cortex [72]. Indeed, from a biological point of view, the need for a normalization step stems from two main observations [73, 72]. First, although cells responses were proven to be stimulus specific [80], it was also shown that cell responses can inhibit one another and that there exists a phenomenon of cross-orientation suppression, where the response of a neuron to its preferred stimuli is attenuated if it is superimposed with another less effective stimuli [73, 17, 22]. Neither the linear models (*i.e.* in the convolution step) nor the different forms of rectification discussed in the previous section, such as half-wave rectification proposed by computational neuroscientists, explain this cross-orientation suppression and inhibition behavior. Second, while cell responses are known to saturate at high contrast, a model relying only on convolution and unbounded rectifiers, such as ReLU, will have values that keep increasing with increasing contrast. These two observations suggested the need for a step that discounts the responses of other stimuli in order to keep the specificity of each cell and make it contrast invariant while explaining other inhibition behaviors of cells.

One popular model to deal with these issues includes a divisive normalization block described mathematically as follows

$$\hat{E}_i = \frac{E_i}{\sigma^2 + \sum_j E_j}, \tag{2.3}$$

where E_i is the output of a squared, half wave rectified convolution operation, pooled over a set of orientations and scales, j, and σ^2 is a saturation constant that can be chosen based on either one of two adaptation mechanisms [72]. In the first case, it could be a different value for each cell learned from the cell's response history. The second possibility is to derive it from the statistics of the responses of all cells. This divisive normalization scheme discards information about magnitude of the contrast in favor of encoding the underlying image pattern in terms of relative contrast across the input responses, E_j, in the normalization operation, (2.3). Use of this model seemed to provide a good fit to neuron responses of mammalian visual cortex [73]. It was also shown that it explains well the cross-orientation suppression phenomenon [17].

Theoretical Perspective

From a theoretical perspective, normalization has been explained as being a method of achieving efficient coding when representing natural images [114]. In that work, the normalization step was motivated by findings regarding the statistics of natural images that are known to be highly correlated and for containing very redundant information. In light of these findings, the normalization step was introduced with the goal of finding a representation that minimizes statistical dependencies in images. To achieve this goal a popular derivation (discussed thoroughly elsewhere [114, 113]) starts by representing images using a statistical model based on a Gaussian Scale Mixture. Using this model and an objective function whose role is to minimize dependencies, a nonlinearity is derived in the form of

$$r_i = \frac{x_i - \sum_j a_j x_j}{\sqrt{b + \sum_k c_j (x_j - \sum_k a_k x_k)^2}} \tag{2.4}$$

where x_i and r_i are the input and output images, respectively, while b, a_i and c_i are parameters of the divisive normalization that can be learned from a training set. Notably, there exists a direct relationship between the definition of the divisive normalization introduced to deal with redundancies and high order dependencies in natural images, (2.4), and that suggested to best fit neuron responses in the visual cortex, (2.3). In particular, with a change of variable where we set $y_i = x_i - \sum_j a_j x_j$, we see that the two equations are related, subject to the square root difference, by an elementwise operation, (*i.e.* squaring, with $E_i = y_i^2$), and thereby both models achieve the goal of maximizing independence while satisfying neuroscientific observations.

Another way of looking at normalization in ConvNets in particular is to consider it as a way of enforcing local competition between features [83, 100], similar to the one taking place in biological neurons. This competition can be enforced between adjacent features within a feature map through subtractive normalization or between feature maps through divisive normalization operating at the same spatial locations across feature maps. Alternatively, divisive normalization can be seen as a way of minimizing sensitivity to multiplicative contrast variation [64]. It was also found, on deeper network architectures, that divisive normalization was helpful in increasing the generalization power of a network [95]. A concrete example illustrating the competition across features enforced by divisive normalization is shown in Figure 2.5.

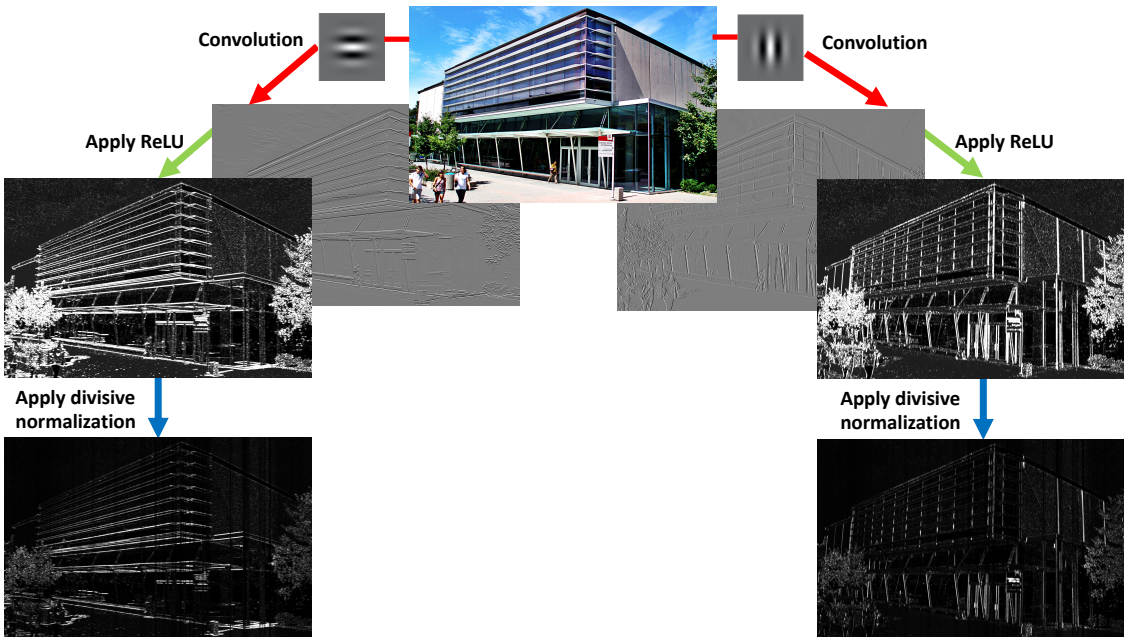


Figure 2.5: An example of divisive normalization enforcing local competition across feature maps. Two different feature maps, E_i, are obtained by applying horizontally and vertically oriented bandpass filters to the same input followed by a ReLU rectification. Divisive normalization, (2.3), is then applied to yield new feature maps, $\hat{E}_i$, which better highlight the horizontal and vertical lines in the input thanks to the local competition across features.

More recent ConvNets rely on a variation of divisive normalization, referred to as *batch normalization* [156]. Batch normalization is another kind of divisive normalization that takes into account a batch of the data to learn normalization parameters, *i.e.* the mean and variance in the transformation

$$\widehat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}}, \tag{2.5}$$

with $E[x^{(k)}]$ being the mini-batch mean calculated as $E[x^{(k)}] = \frac{1}{m}\sum_{i=1}^{m} x_i$ over the m samples of the mini-batch, and $Var[x^{(k)}]$ is the variance of the same mini-batch calculated as $Var[x^{(k)}] = \frac{1}{m}\sum_{i=1}^{m}(x_i - E[x^{(k)}])^2$.

Batch normalization was first introduced as an improvement to traditional divisive normalization with the ultimate goal of reducing the problem of *internal covariate shift*, which refers to the continuous change of the distribution of inputs at each layer [156]. The changing scale and distribution of inputs at each layer implies that the network has to significantly adapt its parameters at each layer and thereby training has to be slow (*i.e.* use of a small learning rate) for the loss to keep decreasing during training (*i.e.* to avoid divergence during training). Therefore, batch normalization was introduced to guarantee more regular distributions at all inputs.

This normalization strategy was inspired by general rules of thumb established for efficient training of ConvNets. In particular, for good generalization performance in ConvNets it is common practice to enforce that all training and testing set samples have the same distribution (*i.e.* through normalization). For example, it has been shown that networks converge faster when the input is always whitened [100, 83]. Batch normalization builds on this idea by considering that each layer can be con-

sidered as a shallow network. Therefore, it would be advantageous to make sure that the inputs keep the same distribution at each layer and this constraint is enforced by learning the distribution of the training data (using mini-batches) and using the statistics of the training set to normalize each input. More generally, it is also important to remember that, from a machine learning perspective, such a normalization scheme can also make features easier to classify. For example, if two different inputs induce two different outputs, they are more easily separable by a classifier if the responses lie within the same range and it is therefore important to process the data to satisfy this condition.

In comparison, batch normalization is somewhat similar to divisive normalization in the sense that they both make the scale of the inputs at each layer similar. However, divisive normalization normalizes the values for each input by dividing it by all other inputs at the same location within the same layer. Batch normalization, on the other hand, normalizes each input with respect to the statistics of the training set at the same location (or, more accurately, the statistics of a mini-batch containing examples from the entire training set). The fact that batch normalization relies on the statistics of a training set may explain the fact that it improves the generalization power of the representation.

One problem with batch normalization is its dependence on the mini-batch size: It might not properly represent the training set at each iteration, if it is chosen to be too small; alternatively, it can have the negative effect of slowing down training, if it is too big (*i.e.* since the network has to see all training samples under the current weights to calculate the mini-batch statistics).

Discussion

Generally, most of the biologically driven studies investigating the role of divisive normalization show that models that include it provide good fits to neural data (*e.g.* [72, 73, 17, 22]). More theoretically driven approaches start from the goal of reducing redundancies in the input as well as bringing it to the same scale, even while casting the problem under different forms. Indeed, while early proposals of divisive normalization, *e.g.* [114], explicitly cast the problem as a redundancy reduction problem, newer proposals, such as batch normalization [156], are also implicitly enforcing this operation through whitening of the data at each layer.

Overall, the common thread across the contributions discussed in this subsection is the fact that they all agree on the important role of normalization in improving the representational power of multilayer architectures and this point will be further demonstrated in this dissertation.

2.4.4 Pooling

Virtually any ConvNet model, be it biologically inspired, purely learning based or completely hand-crafted, includes a pooling step. The goal of the pooling operation is to bring a level of invariance to changes in position and scale as well as to aggregate responses within and across feature maps. Similar to the three building blocks of ConvNets discussed in the previous sections, pooling is also supported by biological findings as well as more theory driven investigations. The major debate when it comes to this layer of processing in convolutional networks is on the choice of the pooling function. The two most widely encountered variations are average and max

pooling. This section explores the advantages and shortcomings of each and discusses other variations as described in related literature.

Biological Perspective

From a biological perspective, pooling is largely motivated by the behavior of cortical complex cells [80, 122, 73, 21]. In the seminal work of Hubel and Wiesel [80], it was found that, similar to simple cells, complex cells are tuned to specific orientations. As opposed to simple cells, however, complex cells exhibit a level of position invariance. They suggest that this result is achieved through some sort of pooling in which the responses of simple cells tuned to the same orientation are pooled across space and/or time.

Some of the early biologically inspired convolutional networks, *e.g.* Fukushima's Neocognitron [59] and the original LeNet network [100], relied on average pooling. Here, average pooling is defined as taking the average response over some support region in the input. In these efforts, average pooling followed by sub-sampling is largely motivated by the findings of Hubel and Wiesel and it is used to decrease the network's sensitivity to position changes. On the other hand, the HMAX [145] class of networks (*e.g.* [157, 125, 158, 85]) rely on max pooling instead. Here, max pooling is defined as taking the maximum response over some support region in the input. Supporters of the max pooling strategy claim that it is more plausible when the input to the pooling operator is a set of Gabor filtered images (*i.e.* the typical model for simple cells). In fact, the authors argue that while a Gaussian scale space (*i.e.* similar to weighted average pooling) reveals new structures at different scales

when applied to a natural image, it causes features to fade away when applied to a Gabor filtered image.

The behavior of complex cells can also be viewed as a type of cross-channel pooling, which is in turn another method of injecting invariances into the representation. Cross channel pooling is achieved through the combination of outputs from various filtering operations at a previous layer. This idea was realized via max pooling across channels at the second layer of a network, where the output of orientation tuned units at the previous layer are pooled across orientations to keep the response of only the maximally responding unit at each spatial position [125].

More recent work investigating center-surround connections in the visual cortex advocates for the presence of similar cross channel connections but across responses tuned for the same orientations [119]. These kinds of connections have been used in the modeling of some recurrent architectures, thereby showing their relevance for complex tasks involving grouping [91] and segmentation [107].

Theoretical Perspective

Pooling has been a component of computer vision representational pipelines for some time, *e.g.* [110, 34, 98, 59, 20, 100, 170], with the goal of introducing some level of invariance to image transformations and better robustness to noise and clutter. From a theoretical perspective, probably one of the most influential works discussing the importance and role of pooling was Koendrink's concept of locally orderless images [92]. This work argued in favor of pooling whereby the exact position of pixels within a Region Of Interest (ROI), *i.e.* a pooling region, can be neglected even while

conserving the global image structure. Currently, virtually all convolutional architectures include a pooling block as part of its processing stages. As with biologically motivated models, more theory driven approaches typically employ either average or max pooling.

Most of the early convolutional architectures relied on average pooling, *e.g.* [59, 100], but it has slowly fallen out of favor in many learning based convolutional architectures and been replaced by max pooling. This trend has been mainly driven by small differences in performance. However, the role of pooling in a network is significant and needs more careful consideration. In fact, early work exploring pooling demonstrated that the type of pooling plays such a significant role in a ConvNet architecture that even an otherwise randomly initialized network yielded competitive results on object recognition, provided appropriate pooling was used [83]. In particular, this work compared average and max pooling and demonstrated that with a randomly initialized network average pooling yields superior performance.

Other work more systematically compared average and max pooling empirically and suggested that there exists a complementarity between the two types of pooling [155]. Yet other work considered the question from a purely theoretical perspective, examining the effect of average versus max pooling on the separability of extracted features [15]. The common conclusions across these efforts imply that ConvNets can benefit from using more than one pooling option throughout the architecture depending on the input type and the transformations it undergoes. For example, both of these efforts argue that max pooling is more suitable when the pooled features are very sparse (*e.g.* when pooling is preceded by a ReLU).

More in line with the approach proposed in this dissertation, recent work approaching their network's design from a purely theory based perspective, *e.g.* ScatNet [18, 133], rely on a form of average pooling. In particular, such networks rely on a weighted sum pooling operation. These approaches tackle the pooling problem from a frequency domain point of view; therefore, their choice of average pooling is motivated by a desire to keep track of the frequency content of the signal. Average pooling allows these networks to act on different frequencies at each layer while downsampling the images to increase invariance and reduce redundancies. At the same time their controlled approach to specifying pooling parameters allows them to avoid aliasing during the pooling operation.

Another type of pooling that gained popularity in the world of ConvNets is so called cross-channel pooling, shown in Figure 2.6. Here, pooling is advocated as a regularization technique that allows for varying the network's structure during training as well as for reducing potential redundancies [63, 105]. This pooling strategy was first introduced as a way to deal with overfitting and correct for the over-complete representation of ConvNets [105] . In particular, due to the large number of kernels used at each layer, it was noticed that many networks often end up learning redundant filters after training. Therefore, cross-channel pooling was introduced to reduce redundancies at each layer by training the network to learn which feature maps to combine either using a weighted linear combination [105] or by setting the output to the maximum across k feature maps on a channelwise basis [63]. Due to the large number of parameters and repeated operations, this type pooling plays a key role and will be investigated as a way to deal with overcomplete representation in this

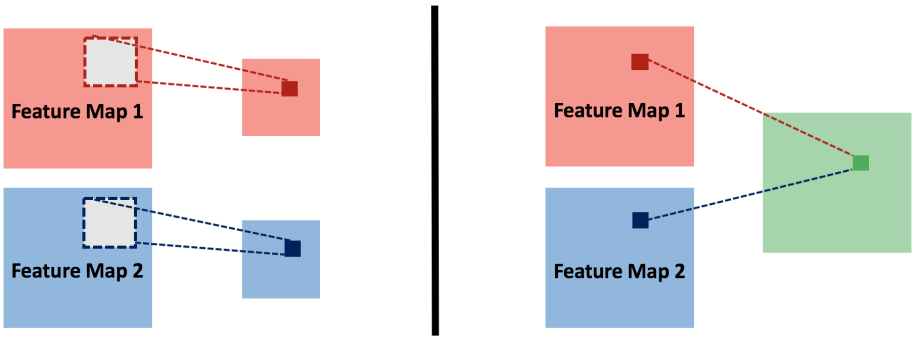


Figure 2.6: Illustration of the pooling operation applied (left) within vs. (right) across feature maps.

dissertation.

It is also worth mentioning under this section one last type of pooling, referred to as global pooling, which takes pooling to its extreme by aggregating measurements across the entire input domain. Global pooling has been used in some prominent ConvNet models in an effort to deal with more practical issues relevant to ConvNet architecture design [69, 105]. For example, it is known that standard ConvNets rely on convolutional layers for feature learning/extraction and fully connected layers followed by a softmax for classification. However, fully connected layers entail the use of a large number of parameters and are thereby prone to overfitting. Many methods were introduced to deal with overfitting induced by fully connected layers, perhaps the most widely used of which is dropout [95]. However, an arguably more elegant way that fits naturally in a convolutional framework is called global average pooling [105]. It simply relies on aggregating the last layer features across the entire feature map support. Another practical consideration for the use of global average pooling is to alleviate the need for using fixed size inputs across an entire dataset to

fit the fully connected layer's requirement, as was done in SPP-Net [69].

Finally, although various pooling operations have been successfully used in different ConvNet architectures as discussed throughout this section, it is important to note here that more recent architectures, known as capsule networks [152], argue strongly against the pooling operation. In that work, it is argued that pooling leads to the loss of important information regarding the relative position of extracted features with respect to each other and it is therefore not included in the more recent capsule based architectures.

Discussion

Overall, pooling should be viewed as a way to summarize information from multiple feature measurements into a compact form that preserves the important aspects of the signal while discarding details. Traditionally, the default functions used in pooling have relied on either the average or max operators. However, several investigations revealed a certain complementary between the two showing that more parameters should be taken into account when choosing the pooling operation. Due to such observations, recent research has been pushing to extend the idea of training to include learning the pooling functions and their parameters. However, this direction entails an increase in the number of parameters to be learned and thereby more chances of overfitting. Importantly, this approach is to be taken with caution, as it would likely further obscure our knowledge and understanding of the learned representations. In complement, pooling parameters can be specified on a theoretical basis for cases where previous stages of processing have adequate analytic detail and

this tack is supported throughout this dissertation.

2.5 Summary and Discussion

This chapter has documented the progress made in understanding ConvNets' outstanding performance, which also is the main motivation for this dissertation. While several of the techniques discussed here are taking good steps to shed light on ConvNets' operations, they all leave open critical questions that remain to be answered.

In fact, the review on the different techniques that aim at explaining ConvNets revealed that the most widely adopted approach relies on visualizations. However, one of the biggest flaws of visualization based techniques is the fact that they reduce understanding of a complex and highly nonlinear model to a small set of images that are open to various potential interpretations. Notably, these visualizations vary according to the adopted technique (*e.g.* dataset-centric vs. network-centric) and usually also depend on the architecture under consideration as well as the training strategy [10], and more importantly, their interpretation also is highly subjective.

Considering ablation studies, while they allow for isolating parts of a network to identify components responsible for better performance, it is difficult for them to elucidate what a network learned as they try to explain ConvNets' highly intertwined components in isolation. More importantly, in their current application, ablation studies typically are simply used as ways to glean a few percentage points in performance without really adding value from an understanding point of view.

Although there are flaws in both visualization and ablation studies approaches

to understanding ConvNets, they still shed some light on ConvNet shortcomings, such as the redundancies in the learned filters (*e.g.* [159]) and importance of certain nonlinearities (*e.g.* [83, 25]). These insights were in turn used in more controlled approaches to ConvNet realization that are less reliant on data, and more importantly, less obscure (*e.g.* [18, 133, 81, 84]). Those same insights were also used as points of departure for the work presented in this dissertation.

Notably, more controlled approaches to network design are emerging as a promising direction for future research as they lend deeper understanding of the operations and representations that these systems employ relative to purely learning based approaches. In turn, they also have the potential to support more rigorous performance bounds based on their precise system specifications. Here interesting ways forward include the following.

- Controlled design of a network's building blocks as well as the network architecture itself (*e.g.* number of layers or filters per layer) via analysis of the input signal properties (*e.g.* the frequency content of the signal). This method can help adapt the architecture's complexity to the application. Understanding a signal's properties as it is processed is the backbone of defining a network in this dissertation and will be discussed thoroughly in Chapter 3.

- Accompanying controlled approaches to network realization with systematic studies of other aspects of ConvNets that usually receive less attention due to the focus on learned parameters. Examples include investigation of various pooling strategies. The analytical approach to network design proposed in this dissertation allows for such a systematic analysis as will be shown in Chapters

4, 5 and 7.

- Theory driven adaptation of the network's design to the peculiarities of a task instead of relying on blind learning. An example for this approach is provided in Chapter 6 of this dissertation.

- Progressively fixing network parameters and analyzing impact on a network's behavior. For example, fixing convolutional kernel parameters (based on some prior knowledge of the task at hand) one layer at a time to analyze the suitability of the adopted kernels at each layer. This progressive approach has the potential to shed light on the role of learning and can also be used as an initialization to minimize training time. This approach is investigated in Chapter 7 of this dissertation.

Chapter 3

An Analytically Defined ConvNet

3.1 Introduction

Chapter 2 introduced ConvNets, highlighted their shortcomings and discussed prominent attempts at understanding ConvNets. That review places the contributions of the present work in the realm of methods that seek to understand ConvNets via controlled design of their building blocks. This chapter describes the proposed learning free approach and details the building blocks of the resulting analytically defined ConvNet.

3.1.1 Motivation

The challenge of extracting useful information (*e.g.* objects, materials and environmental layout) from images has led to incremental recovery of progressively more abstracted representations. Convolutional networks (ConvNets) provide an interesting

contemporary example of this hierarchical paradigm yielding state-of-the-art results on a range of classification and regression tasks, *e.g.* [95, 169, 199, 42]. While such learning-based approaches show remarkable performance, the exact nature of their representations often remains unclear and they typically rely on massive amounts of training data. Deeper theoretical understanding should lessen dependence on data-driven design, which is especially important when training data is limited. In response, this dissertation explores a more controlled approach to network realization. In particular, a small vocabulary of theory motivated, analytically defined filtering operations are repeatedly cascaded to yield hierarchical representations of input imagery. Although the same operations are applied repeatedly, different information is extracted at each layer as the input changes due to the previous layer's operations. Since the primitive operations are specified analytically, they do not require training data and the resulting representations are readily interpretable.

3.1.2 Related Work

As was discussed in Chapter 2, previous work pursuing controlled approaches to hierarchical network realization variously relied on biological inspiration and analytic principles. Here, we provide a succinct summary of research most related to the present work.

Most biologically-based approaches mimic the multilayered architecture of the visual cortex with cascades of simple and complex cells [145, 125, 85]. Variations of this underlying approach consider different learning strategies at higher layers [181, 178, 56], wavelet based filters [76], feature sparsification [79, 181, 125], optimizations

of filter parameters [181, 120] and different pooling strategies [79]. Typically, these approaches still use learning and leave open questions regarding the theoretical basis of their designs.

More theoretically driven approaches typically focus on network architecture optimization, *e.g.* the number of filters/layer or the number of learned weights [100, 53]. Other work makes use of predetermined filters at all layers, as learned via PCA [24] or over a basis of 2D derivatives [81]. Two commonalities appear across these efforts. First, they address only a single aspect of their architecture, while assuming all others are fixed. Second, they rely on learning in optimization.

More closely related to the present work is ScatNet [18]. This network is rigorously defined to increase invariance to signal deformations via hierarchical convolution with filters of different frequency tunings. In its restricted application to 2D spatial images, ScatNet bases its design on optimization with respect to 2D invariances. In contrast, the present approach considers 3D spacetime images, which leads to a spatiotemporal orientation analysis for extracting varying dynamic signal properties across levels, including invariance maximization via a multiscale network. Moreover, while the present approach analytically specifies the number of filters/layer, ScatNet's choice of wavelets limits it from analytically specifying the number of filters/layer, which instead are chosen empirically.

3.1.3 Contributions

In the light of previous work, the contributions of the present research are as follows. **1)** A novel processing network, based on a repeated spatiotemporal oriented

energy analysis, is presented where the layers interact via a recurrent connection so the output feeds back to the input. The resulting multilayer architecture systematically reveals the hierarchical structure in its input. Exposed properties progress from simple and local to abstract and global across layers, but are always interpretable in terms of the network's explicit design. **2)** At every layer, extracted feature maps are combined via cross-channel pooling to yield coherent groups based on the employed filters. This innovation constrains the representation dimensionality while maintaining interpretability and high discriminating power. **3)** Every stage of processing in the network is designed based on theoretical considerations. While the developed approach is not being advocated as a model of biological visual processing, ties to biology are established where relevant. This design approach removes the need for a learning phase, which is not always feasible, *e.g.*, when confronted with modest datasets.

3.2 Technical Approach

The proposed network architecture is designed to capture spatiotemporal image structure across multiple layers of processing as shown in Figure 3.1. The input to the system is a three-dimensional, $\mathbf{x} = (x, y, t)^\top$, spacetime volume, $V(\mathbf{x})$, and the output is a volume of feature maps, $\boldsymbol{F}(\mathbf{x})$, that capture the spatiotemporal structure. Each processing layer, $\mathcal{L}_k$, is comprised of a sequence of four stages: convolution, rectification, normalization and pooling. A key novelty of the approach is the repeated filtering, whereby the final processing stage of each layer (pooling) feeds back

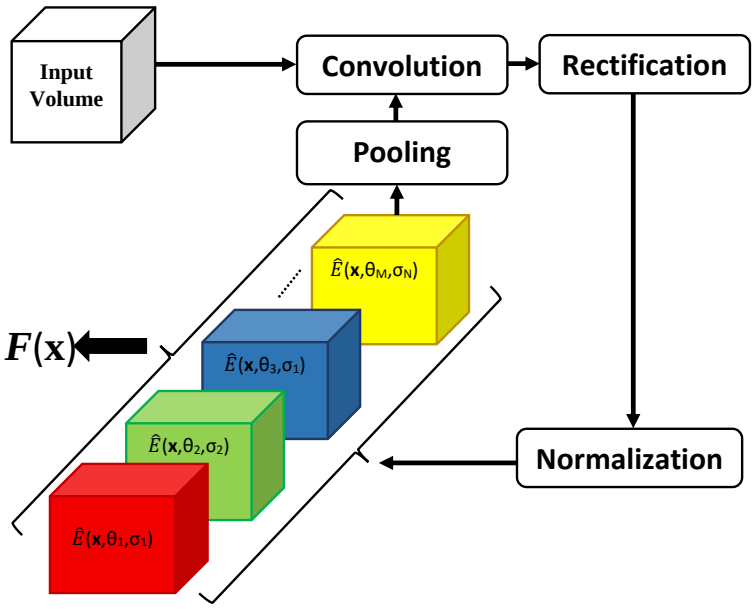


Figure 3.1: Overview of the **SOE-Net** architecture. The same set of operations are repeatedly applied via a recurrent connection; however, different information, $\boldsymbol{F}$, is extracted at each pass as the input changed due to the operations of the previous pass. Here, the different colored blocks correspond to different feature maps, $\hat{E}$, extracted after the normalization block and obtained from filtering with different filters at various orientations, θ_i, and scales, σ_j.

to the initial processing stage (convolution) to yield a subsequent layer of processing, $\mathcal{L}_{k+1}$. The entire process is repeated K times, after which the energy remaining in the signal about to be fed back through has essentially vanished. The final output of the network, $\boldsymbol{F}(\mathbf{x})$, is the set of feature maps extracted at the K^{th} layer. Notably, feature maps extracted after each layer can be used depending on the task at hand. Each layer in the network corresponds to Spatiotemporal Oriented Energy filtering [180]; therefore, the network is dubbed **SOE-Net** [64].

3.2.1 Repeated Filtering

Key to the success of multilayer architectures is their ability to extract progressively more abstract attributes of their input at each successive layer and thereby yield a powerful data representation. As a simple example, a degree of shift-invariance emerges via linear filtering, followed by nonlinear rectification and pooling: The exact position of the extracted features becomes immaterial across the pooling support. A similar interpretation becomes more subtle, however, as the entire filter-rectify-pool block of operations is repeated, especially when using learned filters.

Consider the simple example of Figure 3.2. The left side depicts a sinusoidal spatial pattern alternately moving rightward and staying static across time. On the right, the same spatial pattern is moving rightward behind a similarly textured static picket fence. An initial stage of directional (motion) filtering cannot extract the difference in the overall dynamic behavior of the two patterns. However, further directional filtering that operates on the output of the initial layer of filtering can detect the overall patterns and thereby allows for making the distinction between the

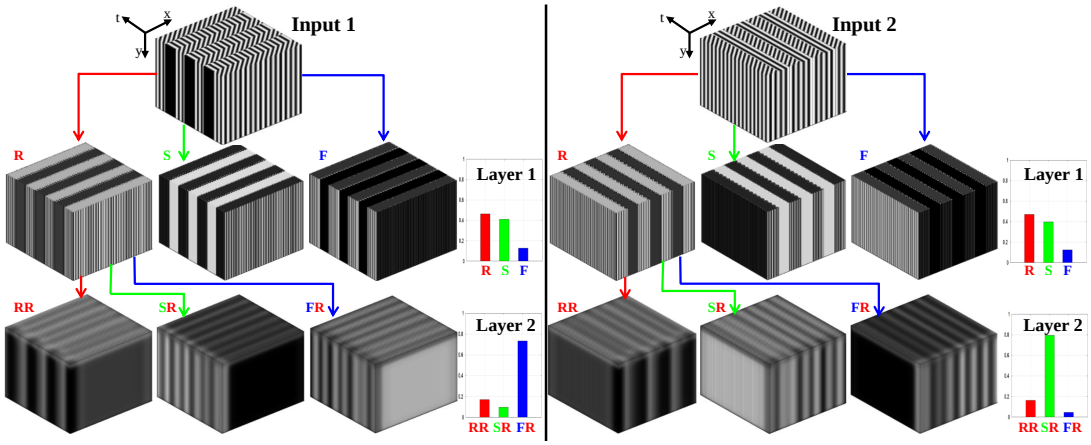


Figure 3.2: Emergence of abstract features via repeated filtering. (Left) Input synthetic sinusoidal pattern alternately moves right (orientation along x-t diagonal) and stays static (orientation parallel to time axis). (Right) Same pattern moving right behind a static picket fence with same spatial pattern (*i.e.* background motion viewed between the spaces in a stationary picket fence). **SOE-Net** is used with filters tuned to **R**ightward motion, **S**tatic (no motion) and **F**lickering (pure temporal change). **L**ayer **1** captures the local rightwardly moving and static portions; but not the more abstract alternating temporal move-stop pattern of the left input, nor the fact that the right input maintains the same spatially interleaved moving and static stripes. Indeed, measurements aggregated over the volumes (shown as histograms on the right) are the same at this layer. The difference is revealed at $\boldsymbol{L_2}$ after applying the same filters on the **R** response of $\boldsymbol{L_1}$: The move-stop behavior of the left texture becomes explicit and yields a large **F** response. In contrast, the constant rightward motion and picket fence of the right texture yield a large **S** response. In practice, more directional filters are applied at each layer; see Section 3.2.2.

two different dynamic textures. More generally, this example shows how powerful features can emerge across multiple layers of processing: The exposed properties progress from simple and local to abstract and global. A more detailed description of the example provided in Figure 3.2 as well as other real life examples can be found in Appendix A.

Motivated by these observations, the proposed SOE-Net is designed to extract progressively more abstract representations of the input signal at each layer, while maintaining interpretability. To achieve these ends, repeated filtering is employed via a recurrent connection, whereby the output of layer $\mathcal{L}_k$, denoted as $\boldsymbol{L}_k$, feeds back to the initial convolutional stage (*i.e.* convolution with the same filter set) to yield a subsequent layer of processing, $\mathcal{L}_{k+1}$. Since the processing at each layer is defined precisely (see following subsections), it will be interpretable. Also, since it is applied repeatedly, abstraction emerges. This process is depicted in Figure 3.3 with an unfolded recurrence, and symbolized as

$$\boldsymbol{L}_{k+1} = \mathcal{L}_{k+1}\left(\boldsymbol{L}_k\right), \tag{3.1}$$

with $\boldsymbol{L}_1 = \mathcal{L}_1\left(V(\mathbf{x})\right)$. Interestingly, biological models have advocated that similar processing (termed $Filter \rightarrow Rectify \rightarrow Filter$) takes place in visual cortex to deal with higher-order image structures [9].

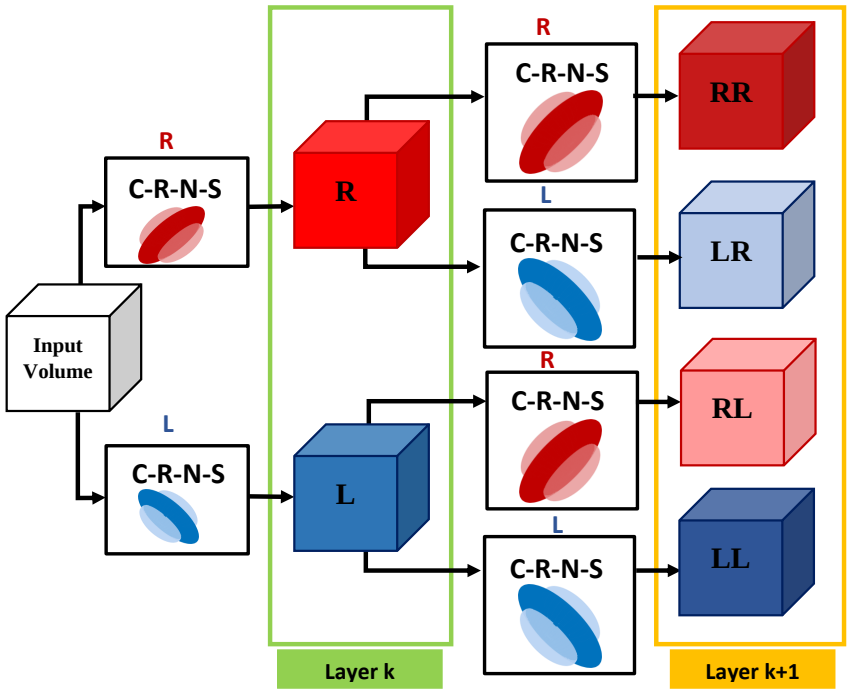


Figure 3.3: Unfolding the **SOE-Net** recurrent connection. Local spatiotemporal features at various orientations are extracted with an initial processing layer, $\mathcal{L}_k$. C-R-N-S indicate Convolution, Rectification, Normalization and Spatiotemporal pooling, as detailed in Secs. 3.2.2, 3.2.3, 3.2.4 and 3.2.5, resp., while R and L indicate rightward vs. leftward filtered data, resp., with symbol strings (*e.g.* LR) indicating multiple filterings. A network with only 2 filters (*i.e.* 2 orientations) is shown for illustration. Each of the feature maps at layer $\mathcal{L}_k$ is treated as a new separate signal and fed back to layer $\mathcal{L}_{k+1}$ to be convolved with the same set of filters but at a different effective resolution due to spatiotemporal pooling.

3.2.2 Convolution

Convolution serves to make local measurements revealing salient properties of its input. Given a temporal sequence of images, local spatiotemporal orientation is of fundamental descriptive power, as it captures the 1^{st}-order correlation structure of the data irrespective of the underlying visual phenomena [6, 180, 36]. Also, such

measures provide a uniform way to capture spatial appearance and dynamic properties of the underlying structure. Spatial patterns (*e.g.* static texture) are captured as the filters are applied within the image plane. Dynamic attributes of the pattern are obtained by filtering at orientations extending into time.

Based on the above theoretical motivations, in this work convolution is designed to extract local measurements of multiscale, spatiotemporal orientation. Still, having committed to convolution that captures spatiotemporal orientation, a variety of options exist for specifying an exact set of filters, *e.g.* Gabor, lognormal, wavelet or Gaussian derivatives. Here, Gaussian derivative filters are selected for two main reasons *cf.* [58]. First, for any given order of Gaussian derivative, a set of basis filters that allow synthesis of the response at an arbitrary orientation can be specified. This property makes it possible to set the exact number of filters used at each layer on a theoretical basis, rather than experiments or learning. Second, these filters are separable, which provides efficient implementation. In particular, 3D Gaussian 3^{rd}-order derivative filters are used, $G_{3D}^{(3)}(\theta_i, \sigma_j)$, with θ_i and σ_j denoting the 3D filter orientation and scale, resp. Given an input spacetime volume, $V(\mathbf{x})$, a set of output volumes, $C(\mathbf{x}; \theta_i, \sigma_j)$, are produced as

$$C(\mathbf{x}; \theta_i, \sigma_j) = G_{3D}^{(3)}(\theta_i, \sigma_j) * V(\mathbf{x}), \tag{3.2}$$

with $*$ denoting convolution. Since 3^{rd}- order Gaussian filter are used, $M = 10$ filters are required to span the space of orientations [58]. The directions, θ, are chosen to uniformly sample 3D orientations as the normals to the faces of an icosahedron, with antipodal directions identified. The number of scales, σ, is determined by the size of

the spacetime volume to be analyzed; details are provided in Section 4.2.2.

3.2.3 Rectification

The output of the convolutional stage, $C(\mathbf{x}; \theta_i, \sigma_j)$, is comprised of positive and negative responses. Both indicate a contrast change along the direction of the oriented filters and hence structure along that direction. It is therefore interesting to keep the distinction between the two responses. Also, in anticipation of the subsequent pooling stage of processing, it is critical to perform some type of rectification. Otherwise, the pooling across positive and negative responses will attenuate the signal. By squaring the individual responses, it is possible to consider the results in terms of spectral energy, *e.g.* [180]. Based on these considerations, the present approach makes use of a nonlinear operation that splits the signal into two paths, as depicted in Figure 3.4: The first path carries the positive responses, while the second carries the negative responses, both of which are pointwise squared to relate to spectral energy measurements to yield

$$\begin{aligned} E^{+}(\mathbf{x}; \theta_i, \sigma_j) &= \left(\max[C(\mathbf{x}; \theta_i, \sigma_j), 0]\right)^2 \\ E^{-}(\mathbf{x}; \theta_i, \sigma_j) &= \left(\min[C(\mathbf{x}; \theta_i, \sigma_j), 0]\right)^2 \end{aligned} \quad . \tag{3.3}$$

Interestingly, biological findings suggest a model for cortical simple cells that includes a nonlinearity in the form of two half wave rectifications that treat the positive and negative outputs in two different paths [72]. Similarly, recent ConvNet analysis also revealed that learned filters tend to form pairs with opposite phases [159].

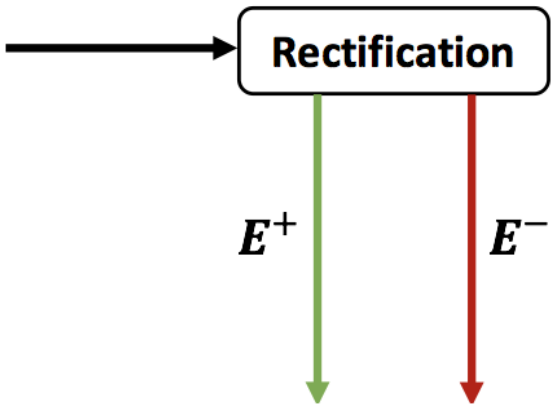


Figure 3.4: Illustration of the employed two-path rectification.

Beyond physical interpretation (signed energy) and relation to biology, use of the squaring function at this stage is advantageous as it allows for keeping track of the frequency content of the processed signal (*i.e.* doubling the maximum frequency present). This information plays an important role in specifying the pooling stage parameters, detailed below. Given that the following processing stages treat the two paths in the exact same way, in the remainder of this dissertation the + and − superscripts will be suppressed when referring to energy measurements, $E(\mathbf{x}; \theta_i, \sigma_j)$.

3.2.4 Normalization

Owing to the bandpass nature of the filtering used in the convolutional stage, (3.2), the responses are invariant to overall additive brightness offsets. Still, the responses remain sensitive to multiplicative contrast variations, *i.e.* the filter response increases with local contrast independent of local orientation structure. Moreover, the responses after rectification, (3.3), are unbounded from above, which makes practical

signal representation challenging. Divisive normalization is a way to correct for these difficulties. Significantly, it also serves to lessen statistical dependencies present in natural images via signal whitening [114]. Therefore, the next stage operates via pointwise division of the rectified measurements by the sum over all orientations to yield

$$\hat{E}(\mathbf{x};\theta_i,\sigma_j) = \frac{E(\mathbf{x};\theta_i,\sigma_j)}{\sum_{m=1}^{M} E(\mathbf{x};\theta_m,\sigma_j) + \epsilon}. \tag{3.4}$$

Here, ϵ serves as a noise floor and to reduce numerical instabilities. It is specified as the standard deviation of the energy measurements across all orientations. Once again, it is interesting to note that biological modeling of visual processing have employed a similar divisive normalization, including use of a signal adaptive saturation constant [72]. Finally, note that features captured at this layer, $\hat{E}(\mathbf{x};\theta_i,\sigma_j)$, are measures of Spatiotemporal Oriented Energy [180, 36]; therefore, the overall network is named **SOE-Net**.

3.2.5 Pooling

Two pooling mechanisms are employed to achieve the desired level of abstraction. First, spatiotemporal pooling is performed following a frequency decreasing path. Thus, the same set of filters in the convolution block operate on lower spatiotemporal frequencies at each subsequent layer thereby revealing new information from the originally input signal. Second, the feature maps extracted for each layer are linearly combined, through cross-channel pooling, to capture more complex structures at each layer while preventing the number of feature maps from exponential increase.

3.2.5.1 Spatiotemporal Pooling

Spatiotemporal pooling serves to aggregate normalized responses, (3.4), over space-time. Aggregation provides for a degree of shift invariance in its output, as the exact position of a response in the pooling region is abstracted. In SOE-Net, spacetime pooling is implemented via low-pass filtering with a 3D Gaussian, $G_{3D}(\gamma)$, followed by downsampling, $\downarrow_\tau (\cdot)$, where γ is the standard deviation and τ is the sampling period,

$$\boldsymbol{S}_k(\mathbf{x};\theta_i,\sigma_j) = \downarrow_\tau \left(G_{3D}(\gamma) * \hat{E}(\mathbf{x};\theta_i,\sigma_j) \right). \tag{3.5}$$

Here, a key question is how to specify the pooling parameters, γ and τ? Typical ConvNets rely on heuristic choices, as their learning based approach does not yield enough theoretical insight for a formal answer. In contrast, since the signal properties of the present architecture have been specified precisely, these parameters can be specified analytically. First, applying a 3^{rd}- order Gaussian derivative filter, $G_{3D}^{(3)}$, with zero mean and standard deviation, σ, in the convolution stage greatly attenuates frequencies $\omega_c > \frac{3\sqrt{3}}{\sigma}$ (*i.e.* a factor of ≈3 beyond the central frequency). However, given the frequency doubling effect from the squaring in the rectification stage, consideration must instead be given to $\eta = 2\omega_c$ as the effective cut-off frequency. Correspondingly, it is appropriate to select the low-pass filter of the pooling stage with a cut-off frequency $\omega_l = \alpha\eta$, with $0 < \alpha < 1$ to ensure operations on lower spatiotemporal frequencies at each layer. This implies taking $\gamma = \frac{3}{\alpha\eta}$ (*i.e.* approximating the cut off frequency to be ≈3 standard deviations away from the central frequency).

To avoid aliasing in downsampling, the sampling theorem is used such that $\omega_s > 2\alpha\eta$. Correspondingly, the sampling period is $\tau = \beta\frac{2\pi}{\omega_s}, 0 < \beta < 1$. In implementation, a conservative $\beta = 0.5$ is employed. Notably, use of low-pass filtering with decreasing frequencies guarantees an energy decay from layer $\mathcal{L}_k$ to layer $\mathcal{L}_{k+1}$.

3.2.5.2 Cross-Channel Pooling

Cross-channel pooling serves to aggregate pooled responses, (3.5), across feature maps. Figure 3.3 unfolds the recurrence to illustrate how once all features maps from layer $\mathcal{L}_k$ have gone separately through the recurrence, the number of feature maps, $\boldsymbol{S}_k$, for each scale, σ_j, used in the convolution block is M^k. This situation is unsatisfactory for two reasons. First, it fails to capture the emergence of common attributes across feature maps that result from applying the same filter orientation, θ_i, to the inputs from $\mathcal{L}_{k-1}$. Second, there is potential for explosion of the representation's size as many layers are cascaded. Both of these concerns can be dealt with by pooling across all feature maps from $\mathcal{L}_k$ that result from filtering with a common orientation, as illustrated in Figure 3.5.

To formalize pooling across feature maps, recall that beyond the very first layer, each $\boldsymbol{S}_k$ derives from input that was itself parameterized by an orientation, θ_m^{k-1}, from filtering at the previous layer, $\mathcal{L}_{k-1}$. This dependence is now captured explicitly by extending its parameterization to $\boldsymbol{S}_k(\mathbf{x}; \theta_i, \sigma_j, \theta_m^{k-1})$. This extension allows the desired cross-channel pooling to produce the final output of $\mathcal{L}_k$ as

$$\boldsymbol{L}_k(\mathbf{x}; \theta_i, \sigma_j) = \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{S}_k(\mathbf{x}; \theta_i, \sigma_j, \theta_m^{k-1}). \tag{3.6}$$

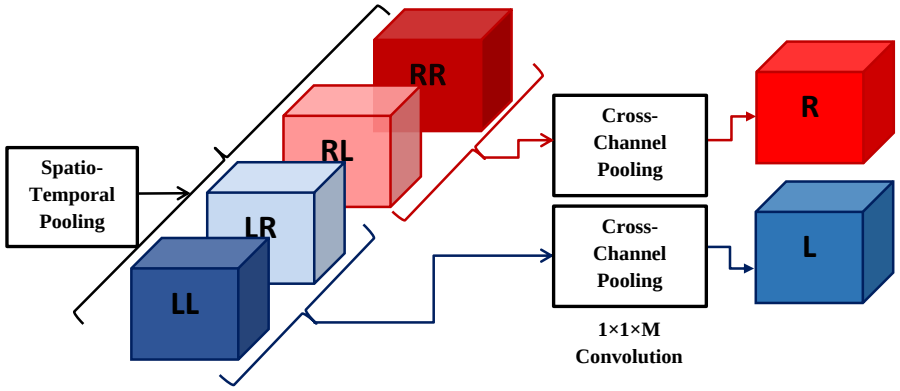


Figure 3.5: Overview of the employed cross-channel pooling.

In implementation this operation is realized as a $1 \times 1 \times M$ convolution with an averaging kernel. Note that the summation holds vacuously at the very first layer of processing.

Theoretically, (3.6) can be seen as an oriented energy generalization of the construction for derivatives of multivalued images *cf*. [38, 154, 149], but is novel in the current context of oriented energy processing. In particular, given a multivalued function $\boldsymbol{f}(\mathbf{x}) = [f_1(\mathbf{x}), ..., f_M(\mathbf{x})]^\top$ with $\mathbf{x} = (x_1, ..., x_N)^\top$ and defining

$$\mathsf{G} = \sum_{m=1}^{M} \nabla f_m(\mathbf{x}) \nabla^\top f_m(\mathbf{x}), \tag{3.7}$$

a measure of change along $\hat{\mathbf{v}}$ is given as $\hat{\mathbf{v}}^\top \mathsf{G} \hat{\mathbf{v}}$. Note that letting $\hat{\mathbf{v}} = \hat{\mathbf{e}}_i$, with $\hat{\mathbf{e}}_i$ the unit vector along x_i, yields

$$\hat{\mathbf{e}}_i^\top \mathsf{G} \hat{\mathbf{e}}_i = \sum_{m=1}^{M} (\frac{\partial f_m}{\partial x_i})^2 \tag{3.8}$$

Comparatively, in (3.6) the multivalued function is just the set of feature maps

obtained from a previous layer, $\mathcal{L}_{k-1}$, that has been differentially filtered and squared in the convolution and rectification blocks, resp. Thus, $\boldsymbol{L}_k(\mathbf{x};\theta_i,\sigma_j)$ captures the energy along direction θ_i taken across the multivalued $\boldsymbol{S}_k(\mathbf{x};\theta_i,\sigma_j,\theta_m^{k-1})$.

Conceptually, the cross-channel pooling, (3.6), produces a set of M feature maps (one for each filter orientation, θ) that capture the amount of structure present in the previous layer at that orientation, irrespective of the source (*i.e.* irrespective of a particular feature map at the previous layer). This approach thereby yields immediate insight into the nature of the representation, in contrast to alternative approaches that rely on random cross-channel combinations, *e.g.* [105]. Reflecting back on the motivating examples, Figure 3.2, the proposed processing was used to generate the feature maps, yielding exactly the desired abstractions.

3.3 Summary and Discussion

In summary, this chapter has presented a novel hierarchical spatiotemporal network for the representation of visual spacetime based on three key ideas. First, a multilayer repeated filtering architecture is employed. Second, design decisions have been theoretically motivated without relying on learning or other empirically driven decisions. Third, in addition to adding insight into convolution, rectification, normalization and spatiotemporal pooling, a novel cross-channel pooling has been introduced that keeps the representation compact while maintaining representational clarity.

The proposed analytical approach has a number advantages in comparison to most ConvNets. The recurrent architecture systematically exposes hierarchical space-

time image structure that is understandable in terms of multiorientation, multiscale properties. The theoretically driven design further makes the operation of each stage of processing easy to understand. Further, by eschewing learning, the approach does not rely on training data. In contrast to the current approach, the operation of learning based ConvNets often remains opaque, even in the presence of strong performance. Moreover, by eschewing parameter learning, the present approach does not rely on massive amounts of training data and attendant need for extensive computational resources. Indeed, due to the limitations of learning all system parameters, typical networks make use of many heuristic choices in their design (*e.g.* number of layers and filters). In contrast, these choices are theoretically based in the present approach.

Owing to the proposed processing architecture's ability to systematically extract hierarchical structure in a sample of visual spacetime (*e.g.* a video), it has potential to serve as the representational substrate for a variety of visual spacetime understanding tasks. Two illustrative examples will be provided in Chapters 4 through 6. Further, the role of learning in conjunction with this analytically defined framework will be explored in Chapter 7.

Chapter 4

Application 1: Dynamic Texture Recognition

4.1 Introduction

Chapter 3 introduced the backbone of this dissertation, which consists of an analytically defined spatiotemporal ConvNet architecture that is referred to as SOE-Net. This chapter validates the benefits of the proposed architecture in the context of dynamic texture recognition.

4.1.1 Motivation

Visual texture, be it static or dynamic, is an important scene characteristic that provides important information for segmentation into coherent regions and identification of material properties. Moreover, it can support subsequent operations

involving background modeling, change detection and indexing. Correspondingly, much research has addressed static texture analysis for single images (*e.g.* [106, 28, 27, 173, 172]). In comparison, research concerned with dynamic texture analysis from temporal image streams (*e.g.* video) has been limited (*e.g.* [64, 141, 186, 142]).

The relative state of dynamic vs. static texture research is unsatisfying because the former is as prevalent in the real world as the latter and it provides similar descriptive power. In fact, many commonly encountered patterns are better described by global dynamics of the signal rather than individual constituent elements. For example, it is more perspicuous to describe the global motion of the leaves on a tree as windblown foliage rather than in terms of individual leaf motion. Further, given the onslaught of video available via on-line and other sources, applications of dynamic texture analysis may eclipse those of static texture.

The goal of this chapter is to tackle the challenge of dynamic texture recognition using the analytically defined ConvNet architecture introduced in Chapter 3 to evaluate its representational power on a relatively simple yet important task.

4.1.2 Related Work

Although dynamic texture has not been as heavily researched as its static counterpart, it nevertheless has received some attention from the vision community and the major contributions tackling this problem are reviewed here.

Broadly speaking, representations used for the task of dynamic texture recognition can be largely categorized into learned vs. hand-crafted. Those in the former category rely on either autoregressive [168] or Linear Dynamical Systems (LDS)

[153]. The more recent trends of research falling under this category, typically rely on LDS with dictionary learning. Variants of this general framework either propose different clustering strategies [143, 123] or different dictionary learning approaches [67, 142, 124]. The main downside of such approaches is the fact that they are strongly tied to the learning process. Therefore, such methods are bound to fail at representing more complex dynamics if they haven't been previously seen during the learning phase.

In contrast, hand-crafted approaches to dynamic texture analysis typically eschew modeling the underlying dynamical system in favor of more directly encoding the spacetime signal with an attempt to balance discriminability and generalizability. Such approaches can be subcategorized according to methods treating each frame of the sequence as a static texture [60, 186], using 3D Local Binary Patterns [194, 144], reliance on optical flow [139, 111] and building more directly on spatiotemporal filtering [36, 185, 86, 184, 41]. This last approach is most akin to the work presented in this dissertation, as it also has at its core the application of spatiotemporal filters to data streams. In contrast, the proposed approach exploits repeated application of such filters and combination of their outputs at each layer to extract progressively more abstract information.

4.1.3 Contributions

In light of previous research, the contributions of this chapter are as follows. **1)** The Spatiotemporal Oriented Energy Network (SOE-Net), resulting from the analytically defined framework introduced in Chapter 3 is used as a novel approach to

representation and recognition of dynamic textures (DTs). **2)** The task of dynamic texture recognition is used to validate the role of each building block of SOE-Net via a detailed ablation study. **3)** In empirical evaluation on standard datasets, the system achieves superior performance to virtually all previously available alternative DT recognition approaches. **4)** The results of the empirical evaluation are used to identify the shortcomings in the area of DT recognition, which subsequently are addressed in Chapter 5.

4.2 Technical Approach

4.2.1 Dynamic Texture Recognition Using SOE-Net

The SOE-Net representation, $\boldsymbol{F}(\mathbf{x}; \theta_i, \sigma_j)$, extracted after K iterations over SOE-Net's recurrent connection, provides a rich hierarchical description of visual spacetime in terms of multiscale spatiotemporal orientation. As such, it has potential to serve as the representational substrate for a wide variety of spacetime image understanding tasks (*e.g.* segmentation, tracking and recognition). As an illustrative example, dynamic texture (DT) recognition is considered here.

For the specific application of dynamic texture recognition, each input spacetime volume, $V(\mathbf{x})$, to be recognized is taken to contain a single pattern. Therefore, the pointwise extracted feature maps, $\boldsymbol{F}(\mathbf{x}; \theta_i, \sigma_j)$, can be aggregated over a region, Ω, that covers the entire spacetime texture sample to be classified to yield a global

feature vector,

$$\mathcal{F}(\theta_i, \sigma_j) = \sum_{\mathbf{x} \in \Omega} \boldsymbol{F}(\mathbf{x}; \theta_i, \sigma_j). \tag{4.1}$$

The resulting global feature vector is normalized to yield the overall descriptor $\hat{\mathcal{F}}(\theta_i, \sigma_j)$. To compare the feature distributions of an input query to database entries, the Bhattacharyya coefficient is used, as it provides a principled way to compare two normalized distributions kept as histograms.

Notably, while previous work made use of spatiotemporal oriented energies for dynamic textures [36], it differs from the current work in five significant ways. First and foremost, previous work did not use repeated filtering, (3.1), for hierarchical abstraction of image structure. Second, it marginalized appearance information so that it did not capture purely spatial structure and therefore was less able than the current approach to capitalize on both appearance and dynamics. Third, it did not separate the opposite phase responses in rectification, (3.3). Fourth, it did not employ multiple scales, σ, (3.2). Fifth, no cross-channel pooling was involved in previous work, while it has a key role in the current work, (3.6).

4.2.2 Implementation Details

Convolution with the Gaussian 3^{rd} derivatives, (3.2), is realized with separable 13-tap filters, with $M = 10$ orientations/scale and $\sigma = 1$. Multiscale filtering is realized by applying the same filters across levels of a Gaussian pyramid, with factor of 2 subsampling between levels. The number of scales, $|\sigma|$, is chosen automatically to avoid undue border effects depending on the size of the input spacetime volume: The

coarsest scale is the last such that the filter can fit entirely within the corresponding pyramid level. Unless otherwise noted, this constraint yields $|\sigma| = 2$ for all datasets considered in the empirical evaluation, except Dyntex++ where $|\sigma| = 1$ due to the very small size of its videos ($50 \times 50 \times 50$). Similarly, the number of iterations over the network, (3.1), *i.e.* the number of layers, K, is stopped when the spatiotemporal support of the signals to be fed back through the recurrence is less than or equal to the filter size; see Section 4.3 for specifics by dataset. Due to the cross-channel pooling, (3.6), the output of the convolution block starting from layer 2 is always $M^2 \times |\sigma|$ feature maps. Also, because the proposed rectification strategy, (3.3), splits the results of each convolutional block into 2 paths, the effective number of feature maps after the rectification block is doubled at each layer. Therefore, the dimension, D_K, of feature vectors extracted after the normalization block of the K^{th} layer is $D_K = M^2 \times |\sigma| \times 2^K$.

4.3 Empirical Evaluation

In this section, the task of dynamic texture recognition is used to analyze SOE-Net's representational power in detail. Later, in Chapter 5, the same task will be used to analyze SOE-Net in comparison to the currently most popular learning based spatiotemporal ConvNets.

SOE-Net is evaluated here according to standard protocols on two standard dynamic texture datasets, YUVL [36] and Dyntex [140]. Prior to the work in this dissertation, YUVL was the largest DT dataset with 610 sequences partitioned in two different ways. The first partition (basic-level or YUVL1) uses all 610 sequences

and groups them into 5 classes based on the number of spacetime orientations present in any given sequence. The second partition (subordinate-level or YUVL2) further subdivides three of the basic level categories to 6 sub-categories based on the specific spacetime orientation present in each sequence. This breakdown uses only 509 sequences from the full dataset. In the present evaluation, a third breakdown of the dataset (YUVL3) is considered where two classes neglected in YUVL2 are reinstated in order to use all 610 sequences divided into 8 different classes. Dyntex is used in its 5 main variations: Dyntex-35 has 35 classes with 10 sequences/class [193]; Dyntex++ has 36 classes with 100 sequences/class [60]; Alpha, Beta and Gamma have 60, 162 and 275 sequences, divided into 3, 10 and 10 classes, resp [41].

The same protocol is used for all datasets. Feature vectors from the last layer of SOE-Net are input to a Nearest-Neighbor (NN) classifier using the leave-one-out procedure [153, 36, 142]. Although NN is not state-of-the-art as a classifier, it is appropriate here where the goal is to highlight the discriminative power of the features without obscuring performance via use of more sophisticated classifiers. Still, for completeness and fair comparison to previous work, results also are reported using Nearest Class Center (NCC) and SVM classifiers in Section 4.3.3.

4.3.1 Component-Wise Validation

SOE-Net's theory based component specifications are now validated empirically. Primarily, classification accuracy on YUVL is used for this goal, as it was the largest available prior to this dissertation work and is well organized according to pattern dynamics.

4.3.1.1 Convolution

The theoretical basis for use of multiscale, spatiotemporally oriented filters, in general, and Gaussian derivative filters, in particular, was given in Section 3.2.2. Nevertheless, a choice remains regarding the order of the derivative used. Table 4.1 compares 2^{nd}-, 3^{rd}- and 4^{th}-order Gaussian derivatives, in terms of classification accuracy, while using a single layer and scale of SOE-Net. The results show $G_{3D}^{(3)}$ as the best performer. This choice provides the right balance between tuning specificity and the numerical stability of relatively low-order filtering. In light of these observations, all subsequent results rely on 3^{rd}-order Gaussian derivatives.

	YUVL1			YUVL2			YUVL3		
SOE-Net (3D)	$G_{3D}^{(2)}$	$G_{3D}^{(3)}$	$G_{3D}^{(4)}$	$G_{3D}^{(2)}$	$G_{3D}^{(3)}$	$G_{3D}^{(4)}$	$G_{3D}^{(2)}$	$G_{3D}^{(3)}$	$G_{3D}^{(4)}$
	83.3	**91.1**	89.8	85.1	**90.2**	90.1	74.1	**84.6**	82.9

Table 4.1: Comparison of 2^{nd}, 3^{rd}, 4^{th} order Gaussian derivatives. In this and all tables of results in this dissertation, bold font is used to indicated top performance.

Also, one could question the benefit of 3D filtering that captures temporal information in recognizing dynamic textures over 2D filtering that captures only spatial appearance. Thus, a 2D version of SOE-Net is used as a baseline comparison. Further, a 2D state-of-the-art hand crafted network, originally proposed for 2D texture analysis, ScatNet [18] is compared. In both cases, 2D frames are supplied to the 2D networks. Table 4.2 shows the decided advantage of 3D over 2D filtering for dynamic texture recognition.

	YUVL1	YUVL2	YUVL3
ScatNet (2D) [18]	68.7	69.7	64.8
SOE-NET (2D)	64.0	70.1	60.8
SOE-Net (3D)	**95.6**	**91.7**	**91.0**

Table 4.2: Benefits of 3D vs. 2D filtering.

4.3.1.2 Multiple Layers and Scales

Table 4.3 documents the benefits of multiscale filtering, (3.2), at each level of the proposed network as well as those of multiple layers, (3.1). The results show that addition of multiple layers and scales consistently improve classification accuracy, with an increase ranging from $\approx 2\%$ to $\approx 6\%$ in going from a single layer and scale to multiple layers and scales. Significantly, the combined multiscale, multilayer results also outperform the best results previously presented on this dataset of 94% and 90% for YUVL1 and YUVL2, respectively [36]. (There are no previously reported results on YUVL3.) These results highlight the pivotal role of the recurrent connection in the proposed network that allows it to decompose the input signal to reveal novel information across scales and layers.

Notably, the network instantiation applied to YUVL is limited to 2 layers and scales due to the small spatiotemporal extent of some sequences in the dataset; see Section 4.2.2 for how dataset extent determines layers and scales. Based on these results, all SOE-Net results presented in the remainder of this chapter are based on feature vectors formed through the concatenation of the last layer's output at all scales.

	YUVL1			YUVL2			YUVL3		
	Sc 1	Sc 2	[Sc 1, Sc 2]	Sc 1	Sc 2	[Sc 1, Sc 2]	Sc 1	Sc 2	[Sc 1, Sc 2]
Layer 1	91.1	87.5	92.9	90.2	85.3	90.9	84.6	80.8	86.2
Layer 2	94.3	91.9	**95.6**	89.0	88.2	**91.7**	86.7	86.9	**91.0**
[36]	-	-	94.0	-	-	90.0	-	-	-

Table 4.3: Classification accuracy on the YUVL dataset using SOE-Net with multiple layers and scales. Here, Sc 1 and Sc 2 refer to Scale 1 and Scale 2, respectively.

4.3.1.3 Two Path Rectification

Next, the advantage of the proposed two path rectification, (3.3), is evaluated. Comparison is made to two alternative rectification approaches: 1) full wave rectification, where the positive and negative signals are combined as its input signal is simply pointwise squared; 2) ReLU rectification, where only the positive part of its input signal is retained. Table 4.4 shows the benefit of the proposed rectification approach. It is seen that not only is there value of not neglecting one signal component (*i.e.* full-wave outperforms ReLU), but moreover additional benefit comes from keeping the positive and negative components separate, which suggest their complementarity.

	YUVL1	YUVL2	YUVL3
Full Wave Rectification	94.2	90.3	89.3
Positive Path (ReLU)	94.2	89.4	88.4
Two Paths	**95.6**	**91.7**	**91.0**

Table 4.4: Benefits of the proposed two path rectification approach.

4.3.1.4 Normalization

Normalization, (3.4), serves to increase invariance to contrast changes. To document this advantage, an instantiation of SOE-Net without the normalization block is compared. The results in Table 4.5 clearly demonstrate the usefulness of this step.

	YUVL1	YUVL2	YUVL3
No Normalization	90.6	87.2	82.8
With Normalization	**95.6**	**91.7**	**91.0**

Table 4.5: Benefits of the normalization step.

4.3.1.5 Spatiotemporal Pooling

The theoretical basis for use of a Gaussian filter in spatiotemporal pooling was given in Section 3.2.5. Here, this choice is validated through comparison to two alternative pooling approaches [83]: 1) the Gaussian filter is replaced with a simple boxcar filter; 2) the more widely used max pooling is considered. Table 4.6 shows the benefit of the proposed spatiotemporal pooling approach that outperforms other pooling strategies by at least 4%.

	YUVL1	YUVL2	YUVL3
boxcar filter	91.7	87.3	87.0
max pooling	91.6	87.4	85.7
Gaussian filter	**95.6**	**91.7**	**91.0**

Table 4.6: Benefits of the used spatiotemporal pooling approach.

4.3.1.6 Cross-Channel Pooling

SOE-Net's novel cross-channel (CC) pooling plays a pivotal role in keeping the dimensionality manageable. Table 4.7 compares the size of the feature vectors with and without CC-pooling for $|\sigma| = 1$. Note that dimensionality reduction does not occur until layer 3, as final feature vector output occurs prior to where CC-pooling would apply and is vacuous at level 1; see Section 3.2.5. To examine the impact of such striking dimensionality reduction on classification, SOE-Net is compared with

	L1	L2	L3	L4	L5
SOE-Net (a)	20	400	8000	160000	3200000
SOE-Net (b)	**20**	**400**	**800**	**1600**	**3200**

Table 4.7: Feature dimensions (a) without vs. (b) with CC-pooling.

and without CC-pooling. Here, a dataset with spatiotemporal size big enough to support 3 layers is needed, as it is the point where dimensionality reduction becomes apparent. Also, comparing beyond layer 3 is not computationally feasible due to dimensionality explosion without CC-pooling, even though it is reasonable with CC-pooling, as done in Sections 4.3.2 and 4.3.3. Since the spatial dimensions of YUVL only support up to layer 2, two variations of Dyntex (Beta and Dyntex_35) are used here. Significantly, on both datasets accuracy of SOE-Net with vs. without CC-pooling is comparable, *i.e.* 97.7% vs. 96.8% on Dyntex_35 and 95.7% vs. 95.1% on Beta, although the feature vectors extracted using CC-pooling are an order of magnitude smaller. These results show that CC-pooling keeps network size manageable while maintaining high discriminating power.

4.3.2 Comparison to a Learned 3D ConvNet

It is interesting to compare the performance of SOE-Net to a learning based 3D ConvNet. For this comparison C3D [169] is used for 3 main reasons. 1) C3D is a popular 3D ConvNet trained end-to-end using 3D filters only without any pre-processing of the input volumes (*e.g.* to extract optical flow). This architecture makes it the most similar trained network to SOE-Net that also relies on 3D convolutions on the raw input volumes. 2) This pre-trained network has shown state-of-the-art performance,

without any fine tuning, on a variety of tasks, including action recognition, object recognition and dynamic scene classification, which is similar to dynamic texture recognition. Indeed, C3D is advocated as a general feature extractor for video analysis tasks without fine tuning (see Sec. 3.3 in [169]). 3) It is not feasible to fine tune a network with any of the available DT datasets prior to this dissertation due to their very limited size. Therefore, alternative ConvNets that require such data for fine tuning prior to testing could not be compared in a meaningful fashion with previously extant datasets. However, a new dataset introduced in this dissertation allows for such experiments and detailed comparisons will be presented later in Section 5.4.

For this experiment, C3D features are extracted (FC-6 activations as specified in [169]) for all versions of the YUVL and Dyntex, except Dyntex++, which is discarded as the small size of its videos ($50 \times 50 \times 50$) precludes application of C3D. For experiments with the considered Dyntex versions, the relatively large size of the sequences makes it possible to push SOE-Net to 3 layers on Dyntex_35 and to 5 layers on Alpha, Beta, Gamma. The results in Table 4.8 show that SOE-Net outperformed C3D on the majority of cases (five out of seven). The larger performance gaps between SOE-Net and C3D on the different variations of the YUVL dataset are of particular note, especially given that SOE-Net uses only 2 layers on YUVL.

	YUVL1	YUVL2	YUVL3	Alpha	Beta	Gamma	Dyntex_35
C3D [169]	88.0	89.8	85.5	100	96.3	95.0	96.3
SOE-Net	**95.6**	**91.7**	**91.0**	98.3	**96.9**	93.6	**97.7**

Table 4.8: Comparison of SOE-Net features versus C3D features.

Significantly, design of YUVL was motivated by interest in building a true *dynamic* texture dataset. Therefore, YUVL groups patterns based on their dynamics, rather than their static (*i.e.* single frame) appearance. For example, Figure 4.1 shows frames from four videos belonging to the same category in YUVL1 (dominant motion). These videos are grouped as they all contain a unique dominant motion direction, even though they look different in a single frame,*i.e.* their commonality is revealed through observing them as temporal sequences. Thus, relative performance on YUVL suggests that SOE-Net is more able to capitalize on dynamic information than C3D. This aspect of C3D is further investigated using the proposed DTDB dataset in Section 5.4. Further, SOE-Net either outperforms or is on par with C3D on datasets that group dynamic textures with more emphasis on visual appearance, *i.e.* Dyntex, suggesting that SOE-Net is able to capitalize on appearance information as well. Overall, these results show that SOE-Net can capture rich, discriminative information, be it dynamic or static appearance, without reliance on extensive and costly training.

humans statues bicycles cars

Figure 4.1: Examples of spacetime textures in the YUVL1 dataset that belong to the same category (*i.e.* dominant motion).

4.3.3 Dynamic Texture Recognition State-of-the-art

Comparison now is made to state-of-the-art approaches designed for dynamic texture recognition using the currently most widely used DT datasets (*i.e.* various breakdowns of Dyntex, as YUVL has received notably less attention). In particular, SOE-Net is compared to 4 learning based [60, 123, 67, 142] and 5 hand-crafted [185, 184, 194, 86, 41] DT descriptors. Following previous research using Dyntex [185, 184, 194, 86, 41, 142], evaluation is performed variously using SVM, Nearest Class Center (NCC) and Nearest Neighbor (NN) classifiers. For SVM, the same protocol used in previous research is followed, whereby 50% of samples per category are used for training and the rest for testing.

Table 4.9 shows that SOE-Net outperforms all other approaches on the Alpha, Beta and Gamma datasets, with sizable margins between $\approx 9-19\%$ using SVM. Using NCC, SOE-Net outperforms all other methods by at least 10%. Overall, these results speak decisively for the high discriminative power of SOE-Net's descriptors.

Similarly, on Dyntex++ SOE-Net extracts stronger features than all hand-crafted methods, achieving an accuracy that is 4.5% higher than previous state-of-the-art approach [185]. Also, under SVM, SOE-Net is on par with state-of-the-art learning based method, with only marginal difference ($\approx 0.3\%$). Moreover, using a NN classifier SOE-Net outperforms all other methods, with an accuracy of 95.6% (results not shown in table, as not reported by others). These results again confirm the discriminative power of the proposed representation. Notably, due to the extremely small size of the Dyntex++ videos, SOE-Net was restricted to only a single scale and 2 layers on this particular dataset.

Method		Alpha		Beta		Gamma		Dyntex_35		Dyntex++
		SVM	NCC	SVM	NCC	SVM	NCC	NN	NCC	SVM
Learning-based	[60]	-	-	-	-	-	-	-	-	63.7
	[123]	-	-		-	-	-	98.6	-	-
	[67]	-	-		-	-	-	-	-	92.8
	[142]	87.8	86.6	76.7	69.0	74.8	64.2	**99.0**	**97.8**	**94.7**
Hand-crafted	[185]	84.9	83.6	76.5	65.2	74.5	60.8	-	97.6	89.9
	[184]	82.8	-	75.4	-	73.5	-	-	96.7	89.2
	[86]	-	-	-	-	-	-	-	96.5	88.8
	[194]	83.3	-	73.4	-	72.0	-	-	97.1	89.8
	[41]	-	85.0	-	67.0	-	63.0	-	-	-
	SOE-NET	**96.7**	**96.7**	**95.7**	**86.4**	**92.2**	**80.3**	97.7	93.1	94.4

Table 4.9: Comparison to state-of-the-art methods on dynamic texture recognition.

Finally, on Dyntex_35 SOE-Net performs slightly worse than state-of-the-art using the NN classifier (a difference of ≈1.3%); however, the difference is greater using the NCC classifier. Interestingly, closer examination of these results reveals that confusions typically occur between slightly different views of the same physical process, which naturally yield visually very similar dynamic textures. Other confusions arise from different physical processes, which happen to yield similar visual appearances. Figure 4.2 shows examples. Overall, the fact that these are the types of confusions made by the network suggests an ability to generalize across viewpoint, which arguably is more important than ability to make fine grained distinctions between viewpoints during texture classification as well as other recognition tasks.

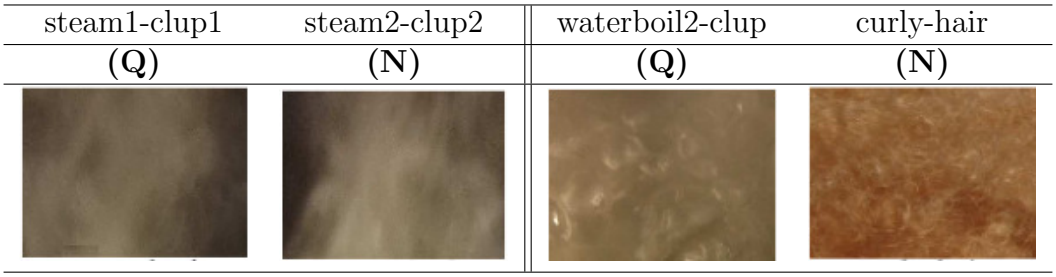


Figure 4.2: Misclassification examples on Dyntex_35 using NCC. **(Q)** is the query class. (**N**) is the nearest class in the database.

4.4 Summary and Discussion

In summary, this chapter presented the second contribution of this dissertation where the principal goal was to validate the capabilities of the analytically defined ConvNet (SOE-Net) introduced in Chapter 3. First, the role and advantage of each of the analytically defined building blocks of SOE-Net were empirically validated in comparison to popular alternatives used in the field. In addition, the benefits of SOE-Net as a spacetime representational substrate were shown with application to the task of dynamic texture recognition, where it extends state-of-the-art on all previously available datasets. Finally, the study performed in this chapter shed light on a major shortcoming in the study of dynamic textures in the ConvNets era. This shortcoming consists in lack of a reasonably sized dataset to allow for an in-depth analysis of learning based architectures. This shortcoming will be addressed in the work presented in the next chapter.

Chapter 5

A New Large Scale Dynamic Texture Database

5.1 Introduction

Chapter 4 provided an in depth analysis of SOE-Net's building blocks with application to dynamic texture recognition. However, it remains interesting to evaluate the capabilities of SOE-Net in the face of other more standard learning based architectures. This comparison was not previously possible given the lack of datasets supporting the use of learning based ConvNets for the task of dynamic textures. This chapter addresses this shortcoming via introduction of a large dual organization dataset for dynamic texture recognition that allows for detailed analysis of the representational power of various networks, including SOE-Net.

5.1.1 Motivation and Related Work

Dynamic texture research has received relatively less attention compared to its static counterpart. Dynamic texture research has been hindered by a number of factors. A major issue is lack of clarity on what constitutes a dynamic texture. Typically, dynamic textures are defined as temporal sequences exhibiting certain temporal statistics or stationary properties in time [153]. In practice, however, the term dynamic texture is usually used to describe the particular case of image sequences exhibiting stochastic dynamics (*e.g.* turbulent water and windblown vegetation). This observation is evidenced by the dominance of such textures in the popular UCLA [153] and DynTex [140] datasets. A more compelling definition describes dynamic texture as any temporal sequence that can be characterized by the same aggregate dynamic properties across its support region [36]. Hence, under this definition, the dominant dynamic textures in UCLA and DynTex are the subclass of textures that exhibit stochastic motion. Another concern with definitions applied in extant datasets is that the classes are usually determined by appearance, which defeats the purpose of studying the *dynamics* of these textures. The only dataset that stands out in this regard is YUVL [36], wherein classes were defined explicitly in terms of pattern dynamics.

The other major limiting factors in the study of dynamic textures are lack of size and diversity in extant datasets. Table 5.1 documents the benchmark datasets used in dynamic texture recognition. It is apparent that these datasets are small compared to what is available for static texture (*e.g.* [27, 32, 131]). Further, limited diversity is apparent, *e.g.* in cases where the number of sequences is greater than

Dataset	Variations	#Videos	#Sequences	#Frames	#Classes
DynTex [140]	Alpha [41]	60	60	>140K	3
	Beta [41]	162	162	>397K	10
	Gamma [41]	264	264	>553K	10
	DynTex-35 [193]	35	350	>8K	35
	DynTex++ [60]	345	3600	>17K	36
UCLA [153]	UCLA-50 [153]	50	200	15K	50
	UCLA-9 [60]	50	200	15K	9
	UCLA-8 [143]	50	92	>6K	8
	UCLA-7 [37]	50	400	15k	7
	SIR [37]	50	400	15K	50
YUVL [36]	YUVL1 [36]	610	610	>65k	5
	YUVL2 [36]	509	509	>55k	6
	YUVL3 [64]	610	610	>65k	8
DTDB [65]	Appearance	**>9K**	**>9K**	**>3.1 million**	**45**
	Dynamics	**>10K**	**>10K**	**>3.4 million**	**18**

Table 5.1: Comparison of the new DTDB dataset with other dynamic texture datasets.

the number videos, multiple sequences were generated as clips from single videos. Diversity also is limited by different classes sometimes being derived from slightly different views of the same physical phenomenon. Moreover, diversity is limited in variations that have a small number of classes.

Over the past few years, increasingly larger sized datasets (*e.g.* [151, 197, 90]) have driven progress in computer vision, especially as they support training of powerful ConvNets (*e.g.* [95, 163, 71]). For video based recognition, action recognition is the most heavily researched task and the availability of large scale datasets (*e.g.* UCF101 [164] and the more recent Kinetics [23]) play a significant role in the progress being made. Therefore, large scale dynamic texture datasets are of particular interest to support use of ConvNets in this domain. Also, as previously mentioned, in addition to describing a dynamic texture based on its appearance, it also is an application

which allows for a characterization that is purely based on the inherent dynamics of a sequence independently of a single frame's appearance. These complementary ways of describing a dynamic texture make it an especially suitable task to analyze the ability of a given representation to capture dynamic as well as appearance information.

5.1.2 Contributions

In response to the above noted state of the affairs, the contributions of this chapter are as follows. **1)** A new large scale dynamic texture dataset that is two orders of magnitude larger than any available is presented. At over 10,000 videos, it is comparable in size to UCF101 [164] that has played a major role in advances to action recognition. **2)** Two complementary organizations of the dataset are provided. The first groups videos based on their dynamics irrespective of their static (single frame) appearance. The second groups videos purely based on their visual appearance. For example, in addition to describing a sequence as containing car traffic, the description is complemented with dynamic information that allows making the distinction between smooth and chaotic car traffic. Figure 5.1 shows frames from the large spectrum of videos present in the dataset and illustrates how videos are assigned to different classes depending on the grouping criterion (*i.e.* dynamics vs. appearance). **3)** The new dataset is used to explore the representational power of different spatiotemporal ConvNet architectures, including the proposed SOE-Net. In particular, the dataset is used for an analysis that examines the relative abilities of architectures that directly apply 3D filtering to input videos [169, 64] vs. two-stream architectures that explicitly separate appearance and motion informa-

tion [162, 49]. The two complementary organizations of the same dataset allow for uniquely insightful experiments regarding the capabilities of the algorithms to exploit appearance vs. dynamic information. **4)** As a byproduct of the availability of a large scale dynamic texture dataset, the analysis of learning based ConvNets is complemented with a novel learning based two-stream architecture that yields superior performance to more standard two-stream approaches on the dynamic texture recognition task. Further utility of the new dataset is documented in Appendix B.

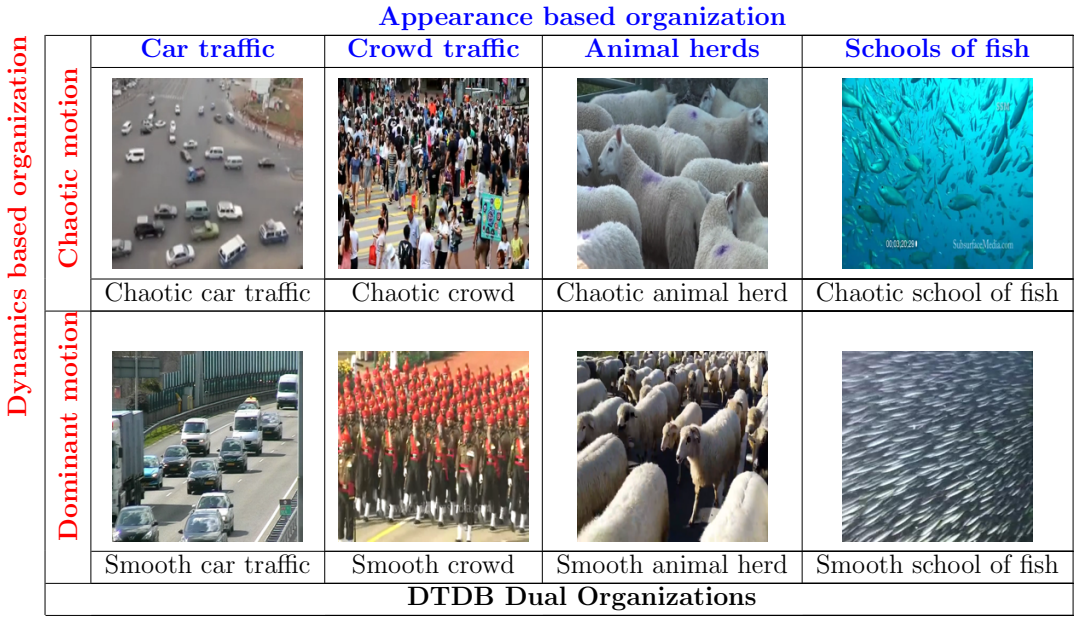


Figure 5.1: Sample frames from the proposed Dynamic Texture DataBase (DTDB) and their assigned categories in both the dynamics and appearance based organizations.

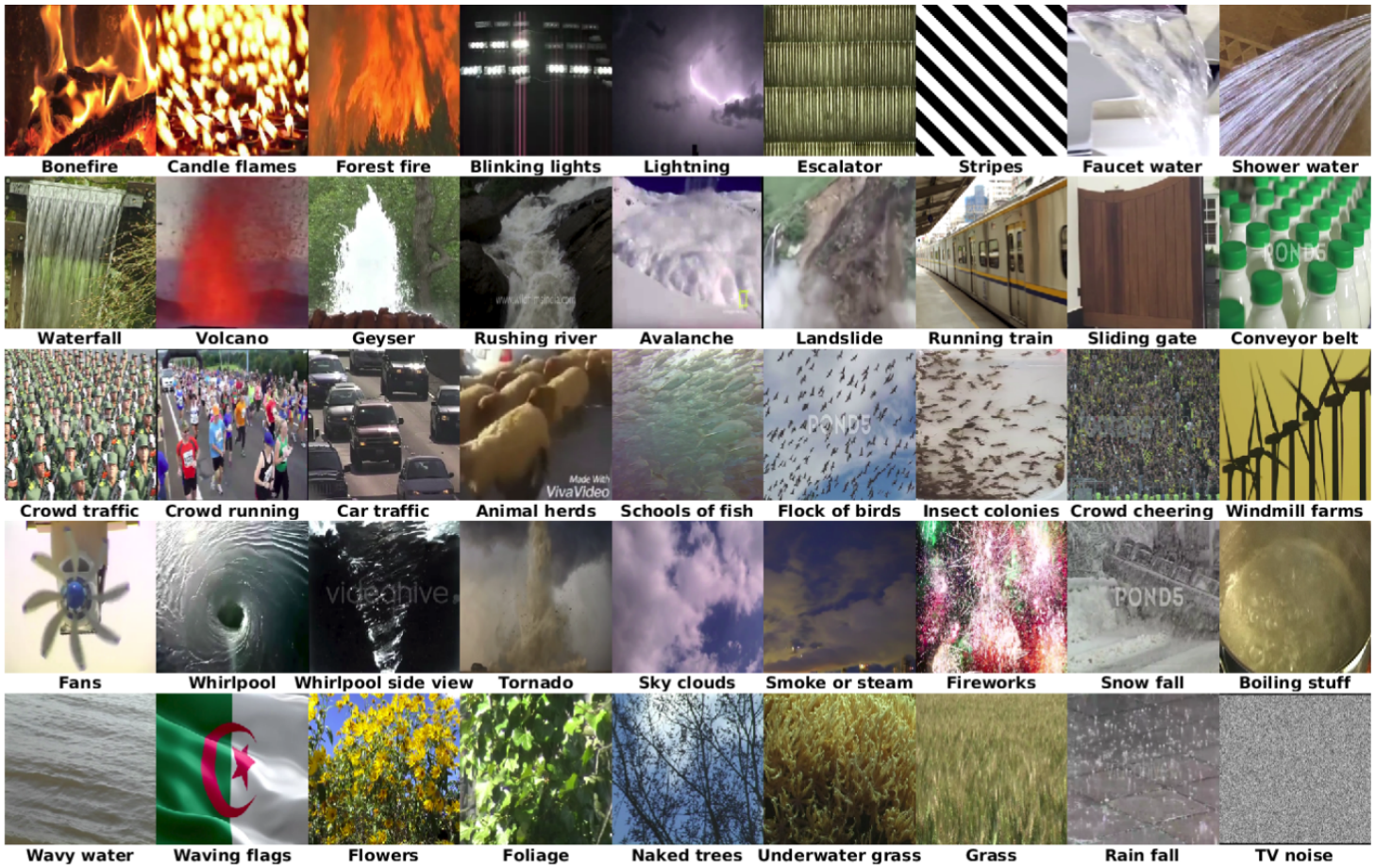


Figure 5.2: Thumbnail examples of the different appearance based dynamic textures present in the new DTDB dataset.

5.2 The Dynamic Texture DataBase (DTDB)

The new dataset, Dynamic Texture DataBase (DTDB) [65], constitutes the largest dynamic texture dataset available with $>$ *10,000 videos* and $\approx$ *3.5 million frames.* As noted above, the dataset is organized in two different ways with 18 dynamics based categories and 45 appearance based categories. Figure 5.1 illustrates how videos are assigned to different classes depending on the grouping criterion (*i.e.* dynamics vs. appearance). Table 5.1 compares the new dataset with previous dynamic texture benchmarks showing the significant improvements compared to alternatives. The videos were collected from various sources, including the web and various handheld cameras, which helps ensure diversity and large intra-class variations. Figure 5.2 provides thumbnail examples from the entire dataset. The remainder of this subsection provides details on the specification and construction of DTDB.

5.2.1 Dynamic Category Specification

The dataset was created with the main goal of building a true *dynamic* texture dataset where sequences exhibiting similar dynamic behaviors are grouped together irrespective of their appearance. Previous work provided a principled approach to defining five coarse dynamic texture categories based on the number of spatiotemporal orientations present in a sequence [36], as given in the left column of Table 5.2. In the present work that enumeration is used as a point departure, but the original categories are further subdivided to yield a much larger set of 18 categories, as given in the middle column of Table 5.2. Note that the original categories are subdivided

in a way that accounts for increased variance about the prescribed orientation distributions in the original classes. For example, patterns falling under *dominant orientation* (*i.e.* sequences dominated by a single spacetime orientation) were split into five sub-categories: (1) Single Rigid Objects, (2) Multiple Rigid Objects, (3) Smooth Non-Rigid Objects, (4) Turbulent Non-Rigid Objects and (5) Pluming Non-Rigid Objects, all exhibiting motion along a dominant direction, albeit with increasing variance (*cf*. [97]); see Figure 5.3 for a visual example. At an extreme, the original category *Isotropic* does not permit further subdivision based on increased variance about its defining orientations, because although it may have significant spatiotemporal contrast, it lacks in discernible orientation(s), *i.e.* it exhibits isotropic pattern structure. See Appendix B for more examples of all categories and a more detailed accompanying discussion.

Original YUVL categories	DTDB categories		
Name/Description	**Name/Description**		**Example sources**
Underconstrained spacetime orientation	↓	**Aperture Problem**	conveyor belt, barber pole
		Blinking	blinking lights, lightning
		Flicker	fire, shimmering steam
Dominant spacetime orientation	↓	**Single Rigid Object**	train, plane
		Multiple Rigid Objects	smooth traffic, smooth crowd
		Smooth Non-Rigid Objects	faucet water, shower water
		Turbulent Non-Rigid Objects	geyser, fountain
		Pluming Non-Rigid Objects	avalanche, landslide
Multi-dominant spacetime orientation	↓	**Rotary Top-View**	fan, whirlpool from top
		Rotary Side-View	tornado, whirlpool from side
		Transparency	translucent surfaces, chain link fence vs. background
		Pluming	smoke, clouds
		Explosion	fireworks, bombs
		Chaotic	swarming insects, chaotic traffic
Heterogeneous spacetime orientation	↓	**Waves**	wavy water, waving flags
		Turbulence	boiling liquid, bubbles
		Stochastic	windblown leaves, flowers
Isotropic	↓	**Scintillation**	TV noise, scintillating water

Table 5.2: Dynamics based categories in the DTDB dataset. A total of 18 different categories are defined by making finer distinctions in the spectrum of dynamic textures proposed originally in [36]. Subdivisions of the original categories occur according to increased variance (indicated by arrow directions) about the orientations specified to define the original categories; see text for details. Appendix B provides more examples and a more detailed discussion.

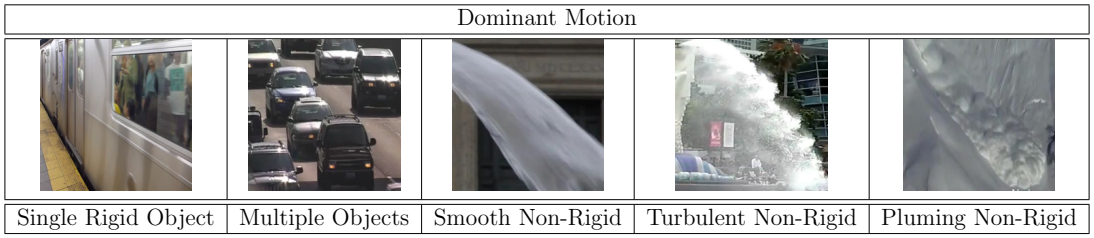


Figure 5.3: Example of the finer distinctions made within dynamic textures falling under the broad dominant motion category. Note the increased level of complexity in the dynamics from left to right. See Appendix B for more examples.

5.2.2 Keywords and Appearance Categories

The definition of appearance categories derived from a reconsideration of the dynamic categories. For each category, a list of scenes, objects and natural phenomena that could contain or exhibit the desired dynamic behavior was brainstormed and their names were used as keywords for subsequent web search. To obtain a large scale dataset, an extensive list of English keywords were generated and augmented with their translations to various languages: Russian, French, German and Mandarin. A visualization of the generated keywords and their frequency of occurrence across all categories is represented as a wordle [2] in Figure 5.4. To specify appearance categories, 45 of the keywords were selected, which taken together covered all the dynamics categories. This approach was possible, since on-line tags for videos are largely based on appearance. The resulting appearance categories are given as sub-captions in Figure 5.2.

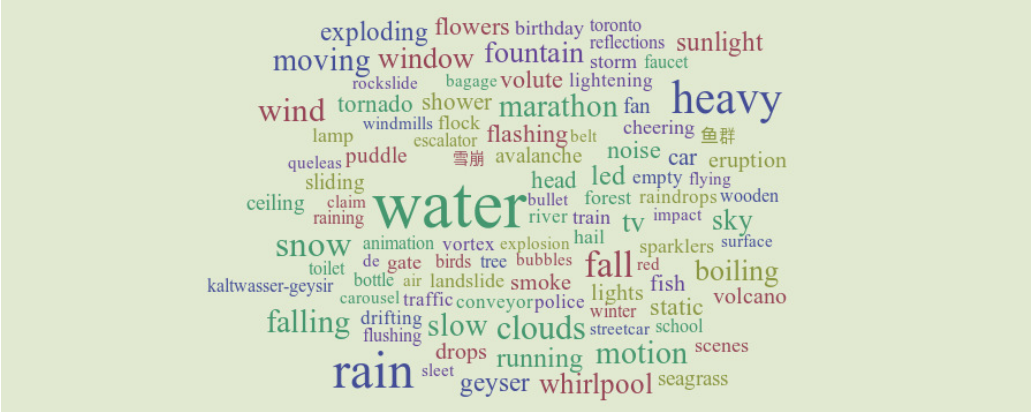


Figure 5.4: DTDB keywords wordle. Bigger font size of a word indicates higher frequency of the keyword resulting in videos in the dataset.

5.2.3 Video Collection

The generated keywords were used to crawl videos from YouTube [5], Pond5 [3] and VideoHive [4]. In doing so, it was useful to specifically crawl playlists. Since playlists are created by human users or generated by machine learning algorithms, their videos share similar tags and topics; therefore, the videos crawled from playlists were typically highly correlated and had a high probability of containing the dynamic texture of interest. Finally, the links (URLs) gathered using the keywords were cleaned to remove duplicates.

5.2.4 Annotation

Annotation served to verify via human inspection the categories present in each crawled video link. This task was the main bottleneck of the collection process and required multiple annotators for good results. Since the annotation required label-

ing the videos according to dynamics while ignoring appearance and vice versa, it demanded specialist background and did not lend itself well to tools such as Mechanical Turk [1]. Therefore, two annotators with computer vision background were hired and trained for this task.

Annotation employed a custom web-based tool allowing the user to view each video according to its web link and assign it the following attributes: a dynamics-based label (according to the 18 categories defined in Table 5.2), an appearance-based label (according to the 45 categories defined in Figure 5.1) and start/end times of the pattern in the video. Each video was separately reviewed by both annotators. When the two main annotators disagreed, a third annotator (also with computer vision background) attempted to resolve matters with consensus and if that was not possible the link was deleted. Following the annotations, the specified portions of all videos were downloaded with their labels.

5.2.5 Dataset Cleaning

For a clean dynamic texture dataset, a decision was made that the target texture should occupy at least 90% of the spatial support of the video and all of the temporal support. Since such requirements are hard to meet with videos acquired in the wild and posted on the web, annotators were instructed to accept videos even if they did not strictly meet this requirement. In a subsequent step, the downloaded videos were visually inspected again and spatially cropped so that the resulting sequences had at least 90% of their spatial support occupied by the target dynamic texture. To ensure the cropping did not severely compromise the overall size of the texture sample, any

video whose cropped spatial dimensions were less than 224×224 was deleted from the dataset. The individuals who did the initial annotations also did the cleaning.

This final cleaning process resulted in slightly over 9000 clean sequences. To obtain an even larger dataset, it was augmented in two ways. First, relevant videos from the earlier DynTex [140] and UCLA [153] datasets were selected (but none from YUVL [36], so that it could be reserved for transfer learning experiments), while avoiding duplicates; second, several volunteers contributed videos that they recorded (*e.g.* with handheld cameras). These additions resulted in the final dataset containing 10,020 sequences with various spatial supports and temporal durations (5-10 seconds).

5.2.6 Dynamics and Appearance Based Organization

All the $10,020$ sequences were used in the dynamics based organization with an average number of videos per category of 556 ± 153. However, because the main focus during data collection was dynamics, it was noticed that not all appearance based video tags generated enough appearance based sequences. Therefore, to keep the dataset balanced in the appearance organization as well, any category containing less than 100 sequences was ignored in the appearance based organization. This process led to an appearance based dataset containing a total 9206 videos divided into 45 different classes with an average number of videos per category of 205 ± 95. The DTDB dataset is available for download at: http://vision.eecs.yorku.ca/research/dtdb/

5.3 Spatiotemporal ConvNets Evaluated

There are largely two complementary approaches to realizing spatiotemporal ConvNets. The first works directly with input temporal image streams (*i.e.* video), *e.g.* [87, 90, 169]. The second takes a two-stream approach, wherein the image information is processed in parallel pathways, one for appearance (RGB images) and one for motion (optical flow), *e.g.* [162, 128, 49]. SOE-Net, as defined in Chapter 3, is an example of the first type of approach, albeit one that is learning free. For the sake of comparison under DTDB, an additional straightforward exemplar is considered for each class that previously has shown strong performance in spatiotemporal image understanding. In particular, C3D [169] is used as an example of working directly with input video and Simonyan and Zisserman Two-Stream [162] as an example of splitting appearance and motion at the input. In addition, a novel two-stream architecture that is designed to overcome limitations of optical flow in capturing dynamic textures is considered in the proposed evaluation. It is worth noting here that SOE-Net has shown state-of-the-art performance on dynamic texture recognition with previously available datasets, as shown in Section 4.3. Importantly, in selecting this set of four ConvNet architectures to compare, it is important to remember that the goal here is not to compare details of the wide variety of instantiations of the two broad classes considered, but more fundamentally to understand the relative power of the single and two-stream approaches. In the remainder of this section, a brief outline of the compared alternatives to SOE-Net is provided, while additional details are provided in Appendix B.

5.3.1 C3D

C3D [169] works with temporal streams of RGB images. It operates on these images via multilayer application of learned 3D, (x, y, t), convolutional filters. It thereby provides a fairly straightforward generalization of standard 2D ConvNet processing to image spacetime. This generalization entails a great increase in the number of parameters to be learned, which is compensated for by using very limited spacetime support at all layers ($3 \times 3 \times 3$ convolutions). Consideration of this type of ConvNet allows for evaluation of the ability of integrated spacetime filtering to capture both appearance and dynamics information.

5.3.2 Two-Stream

The standard Two-Stream architecture [162] operates in two parallel pathways, one for processing appearance and the other for motion. Input to the appearance pathway are RGB images; input to the motion path are stacks of optical flow fields. Essentially, each stream is processed separately with fairly standard 2D ConvNet architectures. Separate classification is performed by each pathway, with late fusion used to achieve the final result. Consideration of this type of ConvNet allows evaluation of the two streams to separate appearance and dynamics information for understanding spatiotemporal content.

5.3.3 MSOE-two-stream

Optical flow is known to be a poor representation for many dynamic textures, especially those exhibiting decidedly non-smooth and/or stochastic characteristics [40, 36]. Such textures are hard for optical flow to capture as they violate the assumptions of brightness constancy and local smoothness that are inherent in most flow estimators. Examples include common real-world patterns shown by wind blown foliage, turbulent flow and complex lighting effects (*e.g.* specularities on water). An illustrative example with a fireworks sequence is provided in Figure 5.5, showing the inability of the optical flow representation to capture motion information in such settings. Thus, various alternative approaches have been used for dynamic texture analysis in lieu of optical flow [26].

A particularly interesting alternative to optical flow in the present context is appearance Marginalized Spatiotemporal Oriented Energy (MSOE) filtering [36]. This approach applies 3D, (x, y, t), oriented filters to a video stream in order to extract the various spacetime orientation present in any given sequence, while not relying on any strong assumptions as is the case for optical flow. As a result this representation is more amenable to representing complex spacetime patterns. To provide an example, the MSOE representation is used to extract dynamics from the same fireworks sequence used earlier. Figure 5.6 shows how to MSOE representation is better able to capture the various dynamic information present in the sequence. In addition, the MSOE representation relies on 3D spacetime convolutions as its core operation and thereby fits naturally in a convolutional architecture. Also, its appearance marginalization abstracts from purely spatial appearance to dynamic information in its output

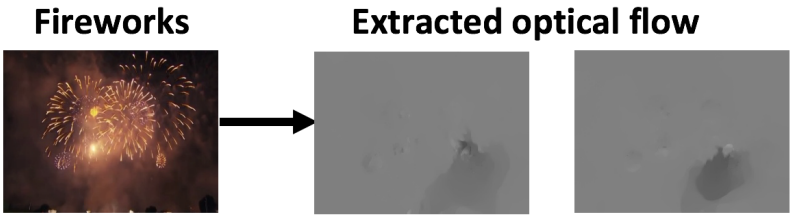


Figure 5.5: Sample optical flow fields extracted from a fireworks sequence.

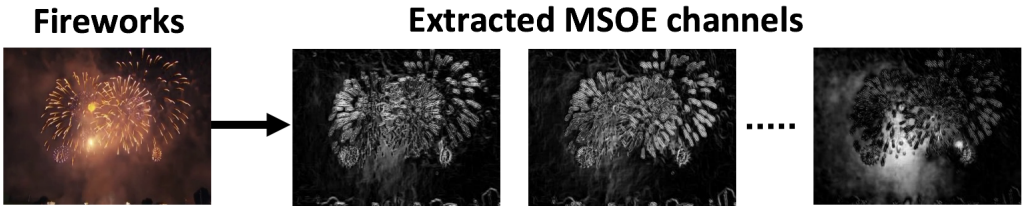


Figure 5.6: Sample MSOE channels capturing 10 directions of motion (only 3 shown here) extracted from the same fireworks sequence.

and thereby provides a natural input to a motion-based pathway. Correspondingly, as a novel two-stream architecture, we replace input optical flow stacks in the motion stream with stacks of MSOE filtering results. Otherwise, the two-stream architecture is the same, including use of RGB frames to capture appearance. The hypothesis here is that the resulting architecture, MSOE-two-stream, will be able to capture a wider range of dynamics in comparison to what can be captured by optical flow, while maintaining the ability to capture appearance.

5.4 Empirical Evaluation

The goals of the proposed dataset in its two organizations are two fold. First, it can be used to help better understand strengths and weaknesses of spatiotemporal ConvNets and thereby guide decisions in the choice of architecture depending on

the task at hand. Second, it can serve as a training substrate to advance research on dynamic texture recognition, in particular, and an initialization for other related tasks, in general. Correspondingly, from an algorithmic perspective, this empirical analysis aims at answering the following questions: **1)** Are spatiotemporal ConvNets able to disentangle appearance and dynamics information? **2)** What are the relative strengths and weaknesses of popular architectures in doing so? **3)** What representations of the input data are best suited for extracting strong representations of image dynamics?

Additional analysis addressing questions from the dataset's perspective is provided in Appendix B.

5.4.1 What Are Spatiotemporal ConvNets Better at Learning? Appearance vs. Dynamics

5.4.1.1 Experimental Protocol

For training purposes each organization of the dataset is split randomly into training and test sets with 70% of the videos from each category used for training and the rest for testing. The C3D [169] and standard two-stream [162] architectures are trained following the protocols given in their original papers. The novel MSOE-two-stream is trained analogously to the standard two-stream, taking into account the changes in the motion stream input (*i.e.* MSOE rather than optical flow). For a fair comparison of the relative capabilities of spatiotemporal ConvNets in capitalizing on both motion and appearance, all networks are trained from scratch on DTDB to

avoid any counfounding variables (*e.g.* as would arise from using the available models of C3D and two-stream as pretrained on different datasets). Training details can be found in Appendix B. SOE-Net is also used in this analysis, however no training is associated with SOE-Net, as all its parameters are specified by design. At test time, the held out test set is used and the reported results are obtained from the softmax scores of each learning based network and the last last layer for SOE-Net as it does not rely on a softmax layer. Note that recognition performance is compared for each organization separately as it does not make sense in the present context to train on one organization and test on the other since the categories are different. However, related transfer learning experiments are reported in Sections B.3.1 and B.3.2. Moreover, the experiments of Section B.3.2 also consider pretrained versions of the C3D and two-stream architectures.

5.4.1.2 Results

Table 5.3 provides a detailed comparison of all the evaluated Networks. To begin, consider the relative performance of the various architectures on the dynamics-based organization. Here, it is seen that the analytically defined SOE-Net of Chapter 3 is the top overall performer. Also, reference to the confusion matrices (Figure 5.7) shows only minor confusions between dynamic categories entailing similar appearances (*e.g.* Chaotic motion and Dominant Multiple Rigid Objects). These results lends strong support to the merits of well designed architectures over those that rely heavily on learning.

Of the learning-based approaches (*i.e.* all but SOE-Net), it is striking that RGB

	DTDB-Dynamics	DTDB-Appearance
C3D [169]	74.9	75.5
Two-Stream [162]	81.5	77.9
SOE-Net [64]	**86.8**	**79.0**

Table 5.3: Recognition accuracy of all the evaluated networks using both organizations of the new Dynamic Texture DataBase.

stream outperforms the Flow stream as well as C3D, even though the latter two are designed to capitalize on motion information. A close inspection of the confusion matrices (Figure 5.7) sheds light on this situation. It is seen that the networks are particularly hampered when similar appearances are present across different dynamics categories as evidenced by the two most confused classes (*i.e.* Chaotic motion and Dominant Multiple Rigid Objects). These two categories were specifically constructed to have this potential source of appearance-based confusion to investigate an algorithm's ability to abstract from appearance to model dynamics; see Figure 5.1 and its corresponding video[1]. Also of note is performance on the categories that are most strongly defined in terms of their dynamics and show little distinctive structure in single frames (*e.g.* Scintillation and motion Transparency). The confusions experienced by C3D and the Flow stream indicate that those approaches have poor ability to learn the appropriate abstractions. Indeed, the performance of the Flow stream is seen to be the weakest of all. The likely reason for the poor Flow stream performance is that its input, optical flow, is not able to capture the underlying dynamics in the videos because they violate standard optical flow assumptions of brightness constancy and local smoothness.

[1]https://www.youtube.com/watch?v=Gj43-hGiso8&feature=youtu.be

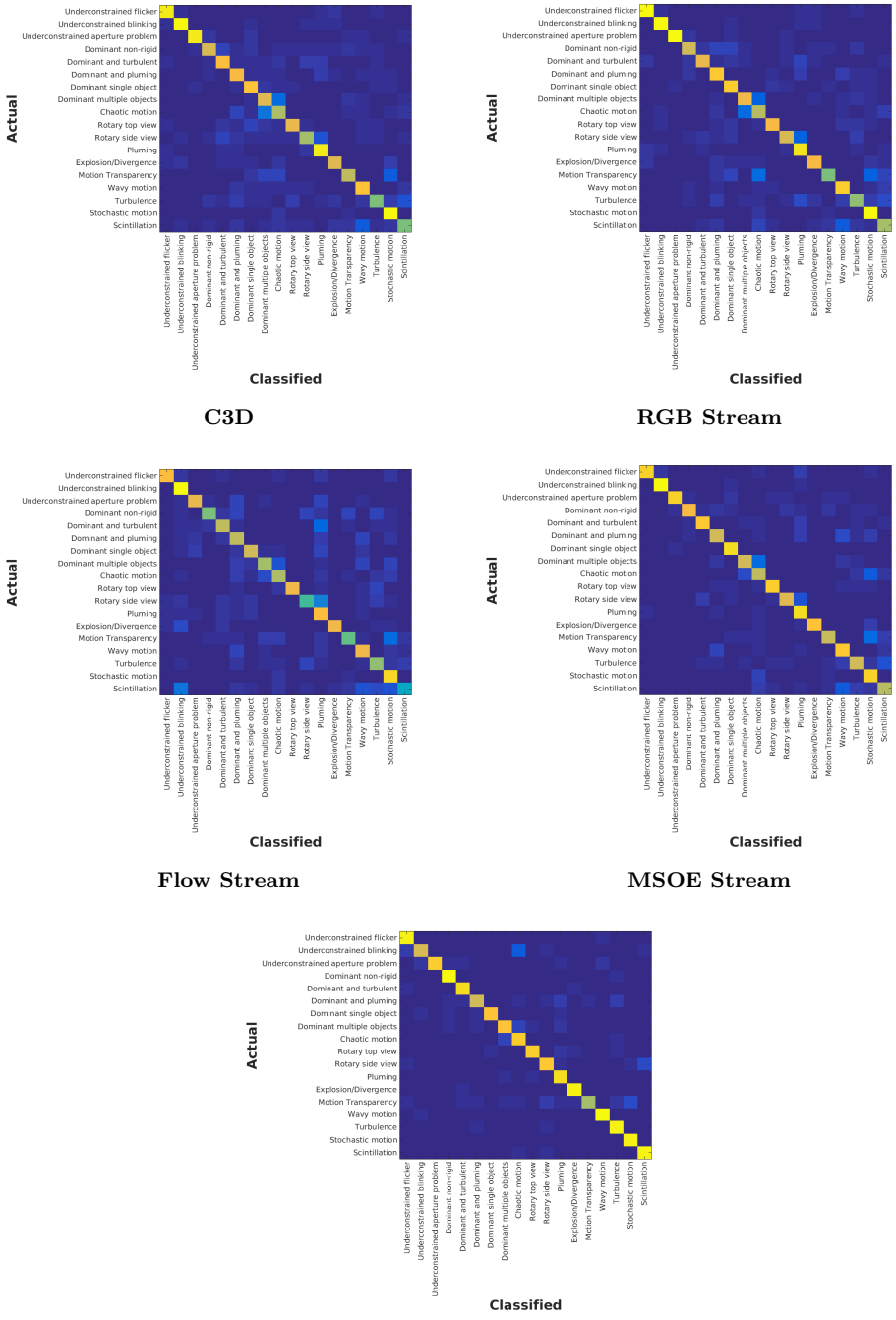


Figure 5.7: Confusion matrices of all the compared ConvNet architectures on the *dynamics* based organization of the new DTDB.

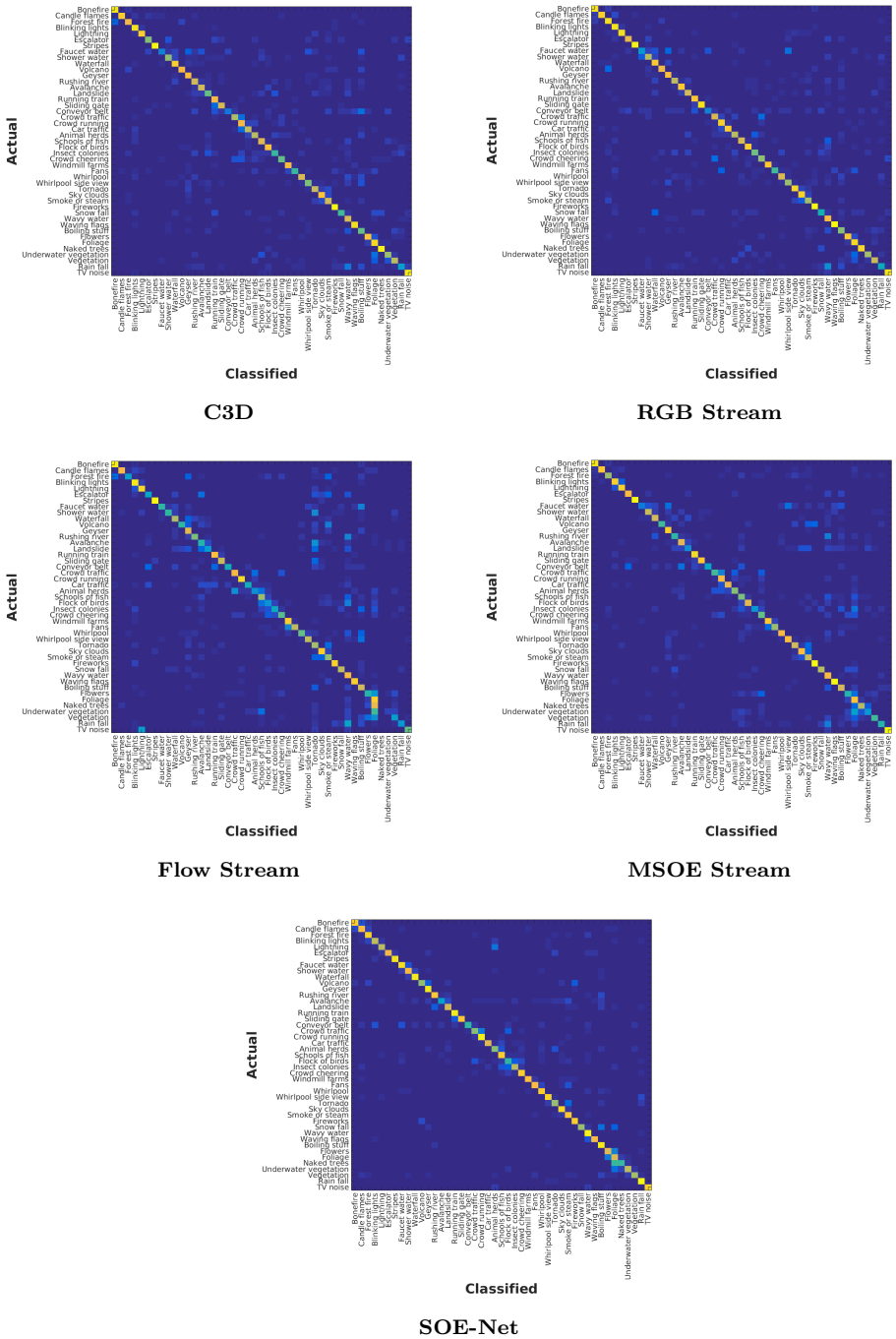


Figure 5.8: Confusion matrices of all compared ConvNet architectures on the *appearance* based organization of the new DTDB.

These points are underlined by noting that MSOE stream has the best performance compared to the other individual streams, with increased performance margin ranging from $\approx 4-8\%$. Based on this result, to judge the two-stream benefit we fuse the appearance (RGB) stream with MSOE stream to yield MSOE-two-stream as the overall top performer among the learning-based approaches. Importantly, recall that the MSOE input representation was defined to overcome the limitations of optical flow as a general purpose input representation for learning dynamics. These results speak decisively in favour of MSOE filtering as a powerful input to dynamics-based learning: It leads to performance that is as good as optical flow for categories that adhere to optical flow assumptions, but extends performance to cases where optical flow fails. Finally, it is interesting to note that the superior performance offered by the learning-free SOE-Net shows that there remains discriminatory information to be otherwise learned from this dataset.

Turning attention to the appearance based results reveals the complementarity between the proposed dynamics and appearance based organizations. In this case, it is found that SOE-Net and MSOE-two-stream are the best overall performers, with the latter slightly better by 1% recognition accuracy. These results attest to the generally strong representational ability of SOE-Net across both pattern dynamics and appearance, while the two-stream approach did not show such flexibility.

Considering the performance of individual learned streams as well as the learned single stream C3D, the dominance of appearance in dataset organization leads to the best performer being the RGB stream that is designed to learn appearance information. Interestingly, C3D's performance, similar to the RGB stream, is on par for the

two organizations although C3D performs slightly better on the appearance organization. This result suggests that C3D's recognition is mainly driven by similarities in appearance in both organizations and it appears relatively weak at capturing dynamics. This limitation may be attributed to the extremely small support of C3D's kernels (*i.e.* $3 \times 3 \times 3$). Also, as expected, the performance of the Flow and MSOE streams degrade on the appearance based organization, as they are designed to capture dynamics-based features. However, even on the appearance based organization, MSOE stream outperforms its Flow counterpart by a sizable margin. Here inspection of the confusion matrices (Figure 5.8), reveals that C3D and the RGB stream tend to make similar confusions, which confirms the tendency of C3D to capitalize on appearance. Also, it is seen that the Flow and MSOE streams tend to confuse categories that exhibit the same dynamics (*e.g.* classes with stochastic motion such as Flower, Foliage and Naked trees), which explains the degraded performance of these two streams.

5.4.1.3 Conclusions

Taken together, the results on the DTDB dynamics and appearance organizations show that the analytically defined SOE-Net provides the best performance in comparison to the considered learning-based alternatives. In particular, it had the top performance on the dynamics organization and was essentially tied for best on the appearance organization. Notably, these results hold for comparison to both single stream (C3D) and the two-stream alternatives.

With respect to distinctions amongst the learning-based approaches, two main

conclusions can be drawn. First, comparison of the different architectures reveal that two-stream networks are better able to disentangle motion from appearance information for the learning-based architectures. This fact is particularly clear from the inversion of performance between the RGB and MSOE streams depending on whether the networks are trained to recognize dynamics or appearance, as well as the degraded performance of both the Flow and MSOE streams when asked to recognize sequences based on their appearance. Second, closer inspection of the confusion matrices show that optical flow fails on most categories where the sequences break the fundamental optical flow assumptions of brightness constancy and local smoothness (*e.g.* Turbulent motion, Transparency and Scintillation). In contrast, the MSOE stream performs well on such categories as well as others that are relatively easy for the Flow stream. The overall superiority of MSOE reflects in its higher performance, compared to flow, on both organizations of the dataset. These results challenge the common practice of using flow as the default representation of input data for motion stream training and should be taken into account in design of future spatiotemporal ConvNets.

5.5 Summary and Discussion

In summary, this chapter introduced a new large scale dynamic texture dataset (DTDB) with two complementary organizations (*i.e.* appearance vs. dynamics based). This dataset allowed for validation of the superiority of the SOE-Net representation in the realm of other popular learning based spatiotemporal ConvNets with applica-

tion to dynamic texture recognition.

In addition, the new DTDB dataset has allowed for a systematic comparison of the learning abilities of broad classes of spatiotemporal ConvNets. In particular, it allowed for an exploration of the abilities of such networks to represent dynamics vs. appearance information. Such a systematic and direct comparison was not possible with previous datasets, as they lacked the necessary complementary organizations. The results especially show the power of two-stream networks that separate appearance and motion at their input for corresponding recognition. Moreover, the introduction of a novel MSOE-based motion stream was shown to improve performance over the traditional optical flow stream. This result has potential for important impact on the field, given the success and popularity of two-stream architectures. Also, it opens up new avenues to explore, *e.g.* using MSOE filtering to design better performing motion streams (and spatiotemporal ConvNets in general) for additional video analysis tasks, *e.g.* action recognition. These observations motivated the exploration of an analytically defined two-stream architecture presented next in Chapter 6.

Finally, the fact that a learning free ConvNet, SOE-Net, yielded best overall performance on DTDB, further underlines the room for further development with learning based approaches. An interesting way forward, explored later in Chapter 7, is to consider a learning enhanced variant of SOE-Net using DTDB and evaluate the potential benefit it can gain from the availability of suitable training data.

Chapter 6

Application 2: Video Object Segmentation

6.1 Introduction

While Chapters 4 and 5 provided an in depth analysis of the performance of the proposed analytic framework on a classification task, this chapter will push the evaluation to a challenging detection task requiring pixel level accuracy. In particular, this chapter evaluates the analytic approach to ConvNet realization presented in this dissertation on the task of Video Object Segmentation (VOS).

6.1.1 Motivation

The goal of video object segmentation is to isolate objects of interest (or foreground) from the background in a given video. Video object segmentation is an important

and challenging task in computer vision. It can be used in several other application such as action recognition, object tracking and video editing and summarization. From a video understanding point of view, video object segmentation is an especially interesting task given the potential importance of both motion and appearance information in its solution. Therefore, it is used in this dissertation to further investigate the relative strengths and weaknesses of the proposed analytical framework to ConvNet realization.

As previously mentioned, one of the main strengths of ConvNets is their hierarchical structure, which allows for extracting features that become progressively more global across the levels of the hierarchy. This progressive information extraction across layers can be very useful for tasks like object segmentation where global cues can help identify the overall location of the foreground object, while lower level cues can help identify finer grained details and hence lead to nicely delineated segments. In the particular case of video object segmentation, motion and appearance information represent two important and complementary cues. Therefore, the work presented in this chapter builds on the above noted two observations and proposes a novel two-stream ConvNet architecture for video object segmentation that is essentially learning free. The proposed architecture builds on the strength of the hierarchical structure of ConvNets, while not requiring prior supervision or training data because all building blocks are designed based on theoretical considerations as outlined in Chapter 3. At the same time, the proposed architecture includes both an appearance and a motion stream to extract complementary features, which are especially useful for the task of video object segmentation.

In addition to the two-stream architecture, the method proposed here also introduces general motion and appearance based priors to complement the overall two-stream approach and better tackle the peculiarities of the video object segmentation task. Also, smooth binary masks are obtained by enforcing temporal consistency throughout the video.

6.1.2 Related Work

Research in the field of video object segmentation is generally classified into three main streams, *i.e.* supervised, semi-supervised and unsupervised, according to the level of human intervention required. In the following, the focus will be on relatively recent approaches; reviews of earlier approaches are available elsewhere, *e.g.* [94].

Supervised approaches in the context of video object segmentation are those methods that require a human in the loop [8, 177, 46]. In those approaches, groundtruth information or guiding key strokes and corrections are interactively introduced by a human to correct for drifting due to mask propagation and errors in the underlying algorithms. While such methods achieve impressive results, they cannot be considered as a complete solution to the VOS problem due to the high level of human supervision required. Instead, these methods can mainly be used for video post-production or to help generate groundtruth.

Semi-supervised methods usually only require groundtruth annotations for the first frame. Some of the methods falling under this category use the initial mask to learn an accurate foreground/background appearance model, that is then used to generate a segmentation for the next frames [136, 126]. More generally however,

semi-supervised methods use the initial foreground mask to treat the VOS problem as a tracking task [171, 187]. Although semi supervised methods require less human intervention, their reliance on an initial foreground mask makes them a sub-optimal solution given the unrealistic nature of the constraint regarding the availability of an initial mask.

Although far more challenging, unsupervised methods for video object segmentation are a more interesting direction for VOS research and the work presented in this dissertation falls under that paradigm. Unsupervised methods do not assume any level of human supervision. However, these methods usually tackle the VOS problem by incorporating certain assumptions about the application scenario. Generally, those methods assume the foreground object to be sufficiently different from the background (via its appearance, its motion or both) and to appear consistently throughout the video [135, 93, 102, 115, 45, 175, 77]. To identify the initial location of objects, unsupervised methods either find motion boundaries using optical flow [135, 93], use a larger set of object proposals that are then ranked based on appearance cues [102, 115] or obtain an initial saliency map to find the particulars of the scene foreground [45, 175, 77]. In this work, the saliency map approach is adopted to directly use the features extracted from the proposed network. Once an initial estimate is obtained, unsupervised methods usually rely on motion [135], appearance [102] or both [45, 77] to propagate the initial mask and enforce temporal smoothness. In the present work, both motion and appearance information are used in synergy to propagate and correct the initial estimate via a diffusion process over a graph in feature space, as was done in other popular approaches (*e.g.* [45, 77]), albeit with

some modifications that will be specified in the remainder of this chapter.

More recently, with the advent of deep learning, ConvNets also gained popularity as a tool to solve the VOS problem *e.g.* [82, 104]. These methods, build on the success of ConvNets in the image segmentation world [109, 68] by posing the problem as a pixelwise classification task trained over a labeled dataset. Interestingly, although such deep learning based methods are highly supervised and require groundtruth annotations during training, they typically are classified with the unsupervised techniques, in the context of video object segmentation, because there is no human in the loop. However, reliance on groundtruth data over a large dataset for training is still a major shortcoming of these methods, especially given the cost of generating labeled pixelwise data for this task. In complement, the ConvNet architecture proposed here is completely learning free and therefore does not require groundtruth labels.

6.1.3 Contributions

In light of previous research, the contributions of this chapter are as follows. **1)** A two-stream variant of the SOE-Net architecture introduced in Chapter 3, is presented with the aim of isolating appearance and motion information to better exploit their complementarity. **2)** To further capitalize on motion and appearance information, a new set of motion and appearance based priors are introduced and tailored to fit the task at hand. **3)** To demonstrate the capabilities of the proposed algorithm, an ablation study evaluating the various steps involved in this work is performed as well as comparisons to state-of-the-art and similar learning based approaches.

6.2 Technical Approach

The developed approach builds on the complementarity between appearance and motion information at various stages. First, features based on motion and appearance information are extracted from a given video using the analytically defined two-stream ConvNet architecture described in Section 6.2.1. Second, the motion and appearance features are used to obtain saliency maps identifying the rough location and shape of the salient objects in the video. Saliency maps are extracted using each feature separately, then the layerwise results are combined using a new feature selection method specifically designed for the task of video object segmentation. Those saliency maps are further refined by introducing priors regarding the general appearance and motion of foreground vs. background regions. Finally, the saliency maps are converted into binary masks by enforcing temporal smoothness and consistency throughout the video. The general framework of the proposed method is depicted in Figure 6.1.

6.2.1 The Two-Stream Spatiotemporal Representation: MSOE-SO-Net

The employed architecture relies on a modified version of SOE-Net described in Chapter 3. The SOE-Net architecture presented an analytically defined 3D spatiotemporal network capable of extracting spacetime features at various levels of abstraction owing to the use of a basis set of 3D kernels in its convolutional layers in a recurrent fashion. Here, SOE-Net is modified such that motion and appearance information

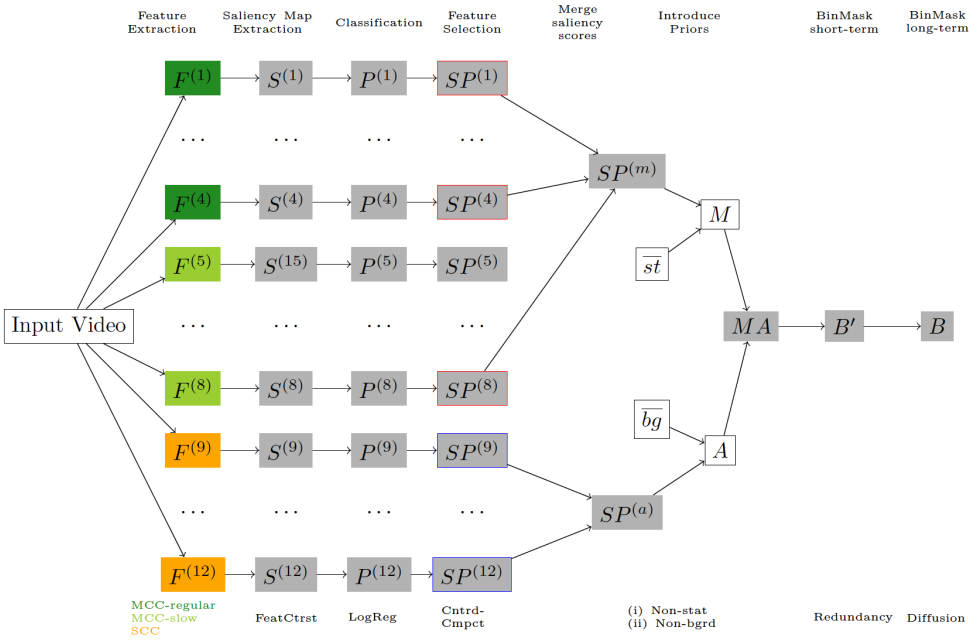


Figure 6.1: Video object segmentation framework. Given an input video, motion $(F^{(1)}, \ldots, F^{(8)})$ and appearance $(F^{(9)}, \ldots, F^{(12)})$ features are extracted from each layer of the two-stream network. Here, the superscripts identify the features extracted from each layer and each stream of the network with the dark and light green boxes representing the medium and slow motion streams respectively while the orange boxes correspond to the appearance stream. A saliency map, $S^{(l)}$, is computed for each feature, $F^{(l)}$ $\forall l$, using feature contrast. Each saliency map, $S^{(l)}$, is used to extract training data for foreground (and background) at each frame, where the amount of training data is selected based on a rough estimate of the object size. This operation allows for each pixel in the video to be considered in a two-class classification problem: $c = 1$ (foreground) and $c = 0$ (background). Using the automatically selected training data, each pixel is assigned a classification score using its corresponding features, $F^{(l)}$, via logistic regression to obtain foreground probability maps $P^{(l)}$. The classification scores are combined with the original saliency maps to obtain the final layerwise saliency maps, $SP^{(l)}$. Reliable maps are selected (red and blue outlines on $SP^{(l)}$ for motion and appearance features, respectively) by comparing spatiotemporal centeredness and compactness of the saliency scores. Priors on background motion, $\overline{st}$, and background appearance, $\overline{bg}$ are used to further refine the motion and appearance saliency maps. The finally saliency map, MA, is converted into a binary mask in two steps by enforcing short term redundancy, B', and long range similarity, B. Notation along the top of the diagram indicates the processing steps used to produce the maps; notation along the bottom indicates specific approach applied for each step, as elaborated in the accompanying text.

are split into two different streams to better exploit the complementarity between the two information sources for the task of video object segmentation. The motion stream relies on a spatially Marginalized Spatiotemporal Oriented Energy representation for capturing dynamics and is therefore dubbed MSOE-Net. The appearance stream relies on a Spatial Orientation representation for capturing spatial appearance and is dubbed SO-Net. The combination of the two-streams is consequently referred to as MSOE-SO-Net.

6.2.1.1 The Motion Stream: MSOE-Net

For the motion stream, the SOE-Net architecture is augmented with an appearance marginalization block that discounts appearance information to extract features capturing motion alone. The appearance marginalization block follows right after the convolution block of the original SOE-Net architecture and its role is to combine all local spacetime responses that are consistent with a single direction of motion independently of the otherwise underlying structure. Detailed description of this block is available elsewhere [36]. In addition to introducing a marginalization block, the proposed motion stream differs from the original SOE-Net architecture in the following three ways. First, while the new motion stream also relies on oriented Gaussian derivative filters during convolution, here the sampling of the used orientations is modified. Specifically, instead of uniformly sampling orientations along the directions of normals to the faces of an icosahedron as done in SOE-Net, the chosen orientations in this work correspond to: static (or no motion), flickering motion in the horizontal and vertical directions, as well as motions in the directions leftward,

rightward, upward, downward and the four diagonals. These orientations are selected to better capture potential directions of motions that may be exhibited by objects moving in typical video object segmentation settings. Notably, previous work that introduced the marginalization block [36] made use of an additional channel capturing lack of structure (*i.e.* an unstructured channel) in the input signal. In this case, however, preliminary experiments demonstrated sub-optimal results when this channel is considered; therefore, this work does not rely on an unstructured channel and only relies on the 10 orientations mentioned above. In addition, those orientations are sampled at two different speeds corresponding to medium (one pixel/frame movement) and slow (half pixel/frame movement) motions. Sampling for slow motions is introduced to capture subtle object motions that may be otherwise masked by faster camera motions. Second, while SOE-Net relied on features extracted after the normalization block of the last layer in application to dynamic textures, this work relies on features extracted from all layers, l. Here, the goal is to capture both low level features that tend to appear in lower layers of the network as well as more abstract features that become apparent in higher layers. Third, the spatiotemporal pooling block used in SOE-Net is replaced with a spatial (*i.e.* 2D) pooling block. Using spatial pooling is once again motivated by the task of video object segmentation where the goal is to identify the foreground object in all frames of the input video; therefore, pooling information across the temporal dimension cannot be used here since that choice would lead to losing information about some of the frames.

6.2.1.2 The Appearance Stream: SO-Net

For the appearance stream, the original 3D oriented bandpass filters used to extract spatiotemporal information in SOE-Net are replaced with their 2D counterpart. Specifically, 2D Gaussian derivative filters are used in the appearance stream, while leaving the other blocks in the architecture unchanged. Notably, the spatiotemporal pooling block is also replaced with a spatial pooling block. For the filter orientations, consistent with the strategy adopted for the motion stream, ten orientations are uniformly sampled. Also, similar to the motion stream, features from all layers are used in this task.

6.2.2 Saliency

In the task of object segmentation, the goal is to isolate the object of interest from the background, which tends to be different (or *salient*) from the rest of the objects in the video. Thus, a saliency map $S \in \mathbb{R}^{H \times W \times T}$ can be defined to indicate regions that are likely to contain objects of interest. To support the unsupervised nature of this work, saliency detection is performed in two steps. First, initial saliency estimates are obtained based on distance measures in feature space and spacetime. Second, those initial saliency maps are used as training data to refine the salient region detection.

6.2.2.1 Saliency Map Initialization: Feature Contrast

The primary goal here is to define a local measure of saliency based on the features extracted from various layers of the network. A saliency map can be defined using

feature contrast with the following three cases in mind: (i) two elements with small spatial distance and similar features are likely to belong to the same object, (ii) two elements with small spatial distance and very different features are indicative of elements belonging to two different objects, and (iii) two elements with large spatial distance and very different features are inconclusive. To satisfy these conditions when defining the saliency map, a feature contrast based definition [48] that uses feature difference and the spatial distances is used to construct a single saliency map from the feature space. This definition is further improved here by exploiting the fact that humans have the tendency to place objects of interest in the center of an image [118, 14]. While it is possible to impose a center bias for a salient object (or border bias for background regions) throughout the video, it is important to note that objects tend to enter and exit the video near the beginning or end of the video. Thus, we impose a *spatiotemporal center bias* in the proposed saliency map formulation, which accounts for objects that enter and leave the field of view across time.

Let $\mathbf{f}_i \in \mathbb{R}^d$ denote the feature descriptor of dimension d for element i, whose spatial coordinate is $\mathbf{x}_i \in \mathbb{R}^2$. Then, the saliency value for element i at frame t, $S_i(t)$, is defined by calculating the pairwise feature and spatial distances to all other elements, j, and combining it with a 2D Gaussian, $G(t)$, imposing a spatiotemporal center prior:

$$S_i(t) = G(\mathbf{x}_i, t) \sum_j \|\mathbf{f}_j - \mathbf{f}_i\|_2^2 e^{-\|\mathbf{x}_j - \mathbf{x}_i\|_2^2}, \tag{6.1}$$

where $G(\mathbf{x}_i, t)$ is a 2D Gaussian-like function adaptive to its temporal location in the video and $\|\cdot\|_2^2$ is the squared L_2-norm. That is, the squared L_2-norm of the

distance between features $\mathbf{f}_j$ and $\mathbf{f}_i$, defined as

$$\|\mathbf{f}_j - \mathbf{f}_i\|_2^2 = \sum_{k=1}^{d} (f_j^k - f_i^k)^2 \tag{6.2}$$

for feature $\mathbf{f}_i = [f_i^1 \ \cdots \ f_i^d]^\top \in \mathbb{R}^d$. Similarly, the spatial distance between two elements is calculated by measuring the spatial distance between their spatial coordinates $\mathbf{x}_i$ and $\mathbf{x}_j$

$$\|\mathbf{x}_j - \mathbf{x}_i\|_2^2 = (x_j - x_i)^2 + (y_j - y_i)^2, \tag{6.3}$$

where $\mathbf{x}_i = [x_i \ y_i]^\top$.

The 2D Gaussian-like function placed at the center of the frame (*i.e.* $\mathbf{x}_c = (\frac{H}{2}, \frac{W}{2})$) is used to impose the center bias that are prevalent in images. To ensure that objects that enter or exit the frame of view in a video are not penalized by the (spatial) center bias, the width of the Gaussian-like function, $\sigma(t)$, is altered based on its temporal location in the video. Thus, a spatiotemporal bias is defined as

$$G(\mathbf{x}_i, t) = \exp\left(-\frac{1}{2}\left(\frac{\mathbf{x}_i - \mathbf{x}_c}{\sigma(t)}\right)^2\right), \tag{6.4}$$

where $\sigma(t) = \frac{1}{T}\left(t - \frac{T}{2}\right)^2$, such that the smallest width of the Gaussian-like function occurs midway through the video (*i.e.* $\frac{T}{2}$) and the rate of the Gaussian width adjusts according to the length of the video, T.

6.2.2.2 Saliency Map Refinement: Classification

The feature contrast based saliency maps allow for a rough localization of salient regions. In order to refine the detected salient regions, a self-training based saliency estimation, which assigns a probability of belonging to foreground to each pixel, is introduced. In particular, the initial saliency maps are used to isolate confident salient regions to be used as training data for a foreground model. Similarly, confident background regions are also identified to train a background model. Using the features belonging to the regions identified as confident foreground and background regions a classifier is trained for each layer, l, and is subsequently used to assign a probability, $P_i^{(l)}$, of belonging to foreground for each pixel, i. This approach allows for taking advantage of supervised classification models, while maintaining the otherwise completely unsupervised nature of the proposed segmentation framework.

To identify training data without resorting to groundtruth information, the initial saliency maps, $S^{(l)}$, as defined pixelwise, i, in (6.1) are used. The simplest approach to isolate training data is to select a threshold above which pixels are considered confident salient regions. However, determining the perfect threshold that can be applied across video frames is a non-trivial task and choosing such a threshold for the entire dataset is even more daunting. For this reason, an intermediate step that allows us to adaptively identify training data for each frame is introduced. Specifically, using the initial saliency maps, an estimate of the object size is obtained framewise by iteratively thresholding the saliency maps at regularly spaced intervals,

$\tau_k \in [0.1, 0.9]$, and calculating the framewise object size, $os(t)$, as

$$os(t) = \frac{\sum_k w_k(t) \cdot a_k(t)}{\sum_k w_k(t)}, \tag{6.5}$$

where w_k is a weight defined as the average of saliency values greater than τ_k and a_k is the proportion of pixels with saliency values greater than τ_k. That is,

$$w_k(t) = \frac{\sum_i S_i(t) \cdot I_{S_i(t) \geq \tau_k}}{\sum_i I_{S_i(t) \geq \tau_k}} \text{ and } a_k(t) = \frac{\sum_i I_{S_i(t) \geq \tau_k}}{H \cdot W}, \tag{6.6}$$

where $I_{S_i(t) \geq \tau_k}$ is an indicator function such that $I_{S_i(t) \geq \tau_k} = \begin{cases} 1, & \text{if } S_i(t) \geq \tau_k \\ 0, & \text{otherwise} \end{cases}$. In essence, this approach calculates the object size as the salience weighted result summed across all threshold values, rather than relying on a single threshold.

The size of the background, $bs(t)$, can be estimated in a similar manner by considering its complement. Algorithm 6.1 provides pseudo-code for the resulting approach to estimating the object size. Using the estimated object size, training data for the foreground is selected as the most salient pixels up to the estimated object size, $os(t)$, while training data for background is selected from the, $bs(t)$, least salient pixels.

Once training data of the foreground and the background have been obtained, several classification models could be used (*e.g.* GMMs, SVM, Logistic Regression, etc.). Logistic regression is used here for its simplicity and speed. Finally, to account for outliers that can result from the pixelwise classification results, the saliency probabilities obtained from the training results, $P_i^{(l)}$, and the initial saliency maps obtained via feature contrast, $S_i^{(l)}$, are combined by averaging the two, to obtain the

final layerwise saliency maps, $SP^{(l)}$.

Input : Saliency map, $S(t) \in \mathbb{R}^{H \times W}$
Output: Object size, $os(t) \in \mathbb{R}$, and background size, $bs(t) \in \mathbb{R}$

for $\tau_k \in [0.1, 0.2, \ldots, 0.9]$ **do**
  Calculate the object size, $a_k^f(t) = \frac{\sum_i I_{S_i(t) \geq \tau_k}}{H \cdot W}$;
  Calculate the weight, $w_k^f(t) = \frac{\sum_i S_i(t) \cdot I_{S_i(t) \geq \tau_k}}{\sum_i I_{S_i(t) \geq \tau_k}}$;
  Calculate the background size, $a_k^b(t) = \frac{\sum_i I_{S_i(t) < \tau_k}}{H \cdot W}$;
  Calculate the weight, $w_k^b(t) = \frac{\sum_i (1 - S_i(t)) I_{S_i(t) < \tau_k}}{\sum_i I_{S_i(t) < \tau_k}}$;
end
Calculate the weighted sum of the foreground, $os(t) = \frac{\sum_k w_k^f(t) a_k^f(t)}{\sum_k w_k^f(t)}$;
Calculate the weighted sum of the background, $bs(t) = \frac{\sum_k w_k^b(t) a_k^b(t)}{\sum_k w_k^b(t)}$;

Algorithm 6.1: Summary description of the employed approach for object size estimation.

6.2.2.3 Merging Saliency Maps

The proposed approach for saliency estimation uses the features of each layer and each stream of the network independently to yield layerwise saliency maps, $SP^{(l)}$. A simple approach to merge the layerwise saliency maps in each stream is to average the corresponding saliency maps. However, given the analytically defined nature of the proposed architecture, it is easy to imagine that the frequency tunings of the filters in a given layer in the architecture can encode the peculiarities of a given sequence more optimally than any other layer. The layers that best encode a given video will largely vary with the nature of the video, *e.g.* presence or absence of camera motion (requiring more or less feature abstraction), background clutter (usually filtered out in lower layers), object sizes (requiring more or less pooling depending on scale).

Therefore, in this work, a feature selection method is proposed to identify the layers of the network that yield the best saliency maps for each video. To do so, measures to evaluate the quality of a saliency map are introduced. Using these measures the layers in the network that yield the best saliency maps are identified. The saliency map quality measures are built around two general observations about video object segmentation and are dubbed compactness and spatiotemporal centeredness here.

Compactness

It is first noted that foreground objects tend to occupy a small portion of a frame rather than the entire frame (*i.e.* they are compact). Thus, a measure to quantify compactness, $Cm^{(l)}(t)$, of a saliency map, $SP^{(l)}$, is defined as the magnitude of its variance. To obtain the variance the center of mass $(\mu_x^{(l)}(t), \mu_y^{(l)}(t))$ at frame t is first calculated as

$$(\mu_x^{(l)}(t), \mu_y^{(l)}(t)) = \left(\frac{\sum_i S_i^{(l)} x_i}{\sum_i S_i^{(l)}}, \frac{\sum_i S_i^{(l)} y_i}{\sum_i S_i^{(l)}} \right) \tag{6.7}$$

for $1 \leq i \leq H \cdot W$. Next, the variance $(\nu_x^{(l)}(t), \nu_y^{(l)}(t))$ at frame t is calculated according to

$$\begin{aligned} (\nu_x^{(l)}(t), \nu_y^{(l)}(t)) = (&\frac{1}{H}\frac{1}{W}\sum_{i=1}^{W}\sum_{j=1}^{H} S^{(l)}(i,j)(x_i - \mu_x^{(l)})^2, \\ &\frac{1}{H}\frac{1}{W}\sum_{i=1}^{W}\sum_{j=1}^{H} S^{(l)}(i,j)(y_i - \mu_y^{(l)})^2). \end{aligned} \tag{6.8}$$

Now, the magnitude of the variance can be calculated using the L_2-norm

$$\|\nu^{(l)}(t)\| = \sqrt{\left[\nu_x^{(l)}(t)\right]^2 + \left[\nu_y^{(l)}(t)\right]^2}.. \tag{6.9}$$

This magnitude is then normalized and the compactness of a saliency map, $SP^{(l)}$, is defined as

$$Cm^{(l)}(t) = \frac{\|\nu_n^{(l)}(t)\|}{\max \|\nu^{(l)}(t)\|}. \tag{6.10}$$

Using this measure, features yielding saliency maps with highly concentrated salient regions are favored over features with saliency values that are more spatially distributed.

Spatiotemporal Centeredness

The second measure used is inspired from the image foreground segmentation field (*e.g.* [118, 14]), where it is common practice to apply a center bias, which is in turn motivated by the tendency of humans to center objects of interest while capturing images. This observation is adapted to the case of video object segmentation via introduction of a measure of spatiotemporal centeredness. (Note that this same idea was also used in the definition of the spatiotemporal center bias for initial saliency estimation, (6.4)). First a measure of spatial centeredness of a saliency map, $SP^{(l)}$, is defined as the distance between the center of mass (6.7) and the center of the

frame, $(\frac{H}{2}, \frac{W}{2})$, normalized by the largest distance it can take to yield

$$Cn^{(l)}(t) = \frac{\sqrt{\left(\mu_x^{(l)} - \frac{H}{2}\right)^2 + \left(\mu_y^{(l)} - \frac{W}{2}\right)^2}}{\sqrt{(\frac{H}{2})^2 + (\frac{W}{2})^2}}. \tag{6.11}$$

Second, to account for the fact that objects can enter the field of view from one side and exit from the other side as a video sequence progresses (*e.g.* a sequence capturing a car moving from point a to point b), a temporal weight, $\alpha(t)$, is introduced to relax the centeredness assumption near the start or end of the video (*i.e.* far from the temporal center). This weight is defined here as a 1D Gaussian-like function in the temporal direction centered at $T/2$:

$$\alpha(t) = \exp\left\{-\frac{1}{2}\lambda\left(t - \frac{T}{2}\right)^2\right\}, \tag{6.12}$$

with λ being a hyperparameter controlling the width of the Gaussian that is empirically set as $\lambda = 6/T$, such that the Gaussian decays smoothly over the duration of the video, *i.e.* $3\sigma = T/2$. Multiplying this weight with the spatial centeredness measure, (6.11), yields a measure of spatiotemporal centeredness.

Combining Centeredness and Compactness

To finally quantify the quality of a saliency map, the measures of centeredness, $Cn^{(l)}$, and compactness, $Cm^{(l)}$, are combined by relying on centeredness more heavily around the temporal center of the video and relying on compactness more heavily elsewhere. To achieve this result, the temporal weight, $\alpha(t)$, is used to yield a

combined quality measure

$$CC^{(l)}(t) = \alpha(t)\ Cn^{(l)}(t) + (1 - \alpha(t))\ Cm^{(l)}(t). \tag{6.13}$$

Using this combined measure, the best layers in each stream are selected. In particular, features yielding saliency quality scores, $CC^{(l)}(t)$, that are within 10% of the minimum are selected and combined via averaging.

6.2.3 Binarization

Similar to the saliency estimation strategy described above, the proposed binarization strategy is also divided into two main steps. First, the saliency maps together with general domain priors are used to obtain the initial binary estimates. Second, using this initialization and temporal consistency constraints, the initial estimates are refined to obtain the final binary masks.

6.2.3.1 Framewise Initialization

To obtain the initial binary mask, estimates of the saliency maps extracted from both the motion and appearance streams of the analytically defined network are used. In addition, two general priors regarding the motion and shape of objects in any typical video object segmentation setting are introduced.

Non-Static Motion Prior

From the motion perspective, the saliency map obtained from the motion stream, $SP^{(m)}$, is complemented with a map measuring the lack of static motion, $\overline{st}$, *i.e.*

places that are in motion or otherwise dynamic. This measure is motivated by the observation that foreground objects tend to be the main moving part in the video and the background is relatively static, or exhibits slower motion structure, in comparison. For this purpose, the static channel is directly extracted from the first layer of the motion stream and its complement (*i.e.* $\overline{st} = 1 - st$), which highlights the non-static parts of the video, is obtained (Recall that due to normalization in its definition, $0 \leq st \leq 1$). To combine the two motion based maps, the same compactness and spatiotemporal centeredness measure, (6.13), introduced to quantify the quality of a saliency map is employed. Specifically, quality measures $CC^{(m)}$ and $CC^{(\overline{st})}$ are obtained for the motion and non-static maps respectively and the two maps are combined as

$$M(t) = \beta(t)\overline{st}(t) + (1 - \beta(t))SP^{(m)}(t), \tag{6.14}$$

where $\beta(t)$ is a weight balancing the relative difference between the two maps in order to give more weight to the best map according to the quality measure; it is defined as

$$\beta(t) = \frac{1}{1 + exp(-(CC^{(m)}(t) - CC^{(\overline{st})}(t)))}. \tag{6.15}$$

Non-Background Appearance Prior

From the appearance perspective, the saliency map obtained from the appearance stream, $SP^{(a)}$, is complemented with a map measuring lack of background appearance attributes, $\overline{bg}$. Specifically, general purpose appearance features such as color are used together with a classifier trained to recognize typical background features

such as sky, clouds, trees and roads. This classification model is borrowed from previous work precisely targeting the task of recognizing general background attributes [43], where the classifier was independently trained on background scenes from the SUN static scene dataset [183]. Using this classifier, a map, bg, indicating the probability of being background is obtained for each frame and its complement, *i.e.* $\overline{bg} = 1 - bg$, is used. The two appearance maps $SP^{(a)}$ and $\overline{bg}$ are combined to obtain the final appearance saliency map, $A(t)$, following the same procedure outlined above, (6.14), for merging motion maps.

Finally, the initialization, $MA(t)$, is obtained by merging the appearance and motion maps in a similar fashion, as depicted in Figure 6.1. At this stage an initial binary mask, $b^{(0)}(t)$, can be obtained from $MA(t)$ by estimating the object size using (6.5) and isolating the foreground object.

6.2.3.2 Temporal Consistency

Once initial framewise estimates are obtained, it is important to enforce consistency throughout the video such that the final binary masks are smooth and consistently highlight the same foreground objects. Following the same principles adopted throughout this work, the temporal consistency block is based on two general observations that typically are applicable to the task of video object segmentation.

Short Range Redundancy

First, it is noted that in a short time frame objects do not tend to exhibit drastic motions and therefore objects shape does not change drastically in a short temporal

window (even if the object is not rigid). Also, for the same reason, the location of the objects remains approximately the same within a small window.

Based on this observation, a short term redundancy constraint is introduced. This constraint enforces binary mask similarity in shape and approximate location within a small temporal window. In particular, a redundancy score, $rs(t)$, is obtained for each frame, t, by looking at the frames directly before and after it in the initial binary mask, $b^{(0)}(t)$. The redundancy score is defined as

$$rs(t) = \frac{1}{3} \sum_{\tau=t-1}^{\tau=t+1} b^{(0)}(\tau). \tag{6.16}$$

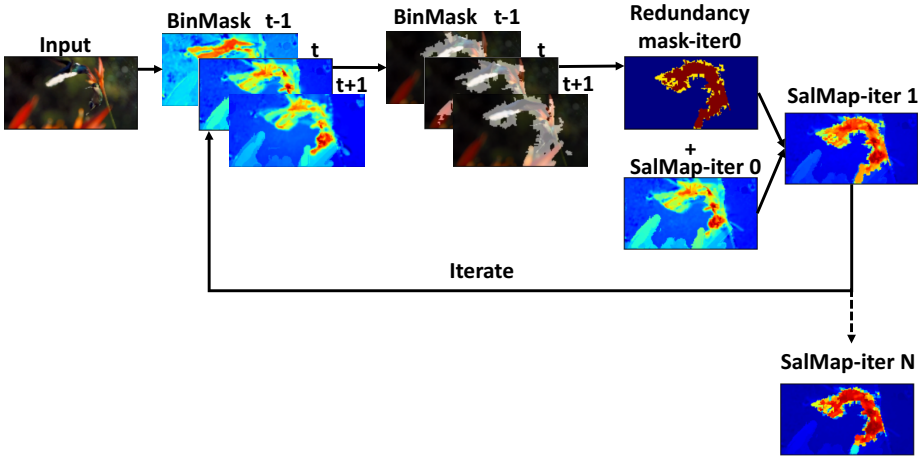


Figure 6.2: Overview of the process used to enforce short term redundancy. Initial binary masks are first obtained from the input saliency maps. Next, a pixelwise redundancy score is obtained by identifying the pixels that appear repeatedly (*i.e.* redundant pixels) within a temporal window of size 3. Finally, this redundancy score is combined with the input saliency maps to clean the saliency maps. The process is repeated in multiple iterations until the binary masks obtained from the saliency maps stop changing.

To correct for short term differences and outliers, the redundancy score, (6.16), is combined with the initial saliency estimate, $MA^{(0)}(t)$, via averaging. The entire process is then repeated in multiple iterations until the estimated salient regions and corresponding binary masks stop changing, yielding locally smooth binary masks, $b^{(r)}(t)$, and saliency maps $MA^{(r)}(t)$. This process is summarized in Algorithm 6.2 and depicted in Figure 6.2. The final result of this processing stage is denoted as B' in Figure 6.1.

Input : Initial saliency map, $MA^{(0)} \in \mathbb{R}^{H \times W \times T}$, initial object size estimate $os^{(0)}$ and stopping threshold τ
Output: post short term redundancy saliency map, $MA^{(r)}$, and binary mask, $b^{(r)} \in \mathbb{R}^{H \times W \times T}$

$it = 0$
$os_{diff} = \inf$
while $os_{diff} > \tau$ **do**
    Obtain initial binary mask from $MA^{(it)}$;
    Calculate the redundancy score $rs^{(it)}$;
    Combine initial saliency estimate with the redundancy score;
    Calculate the new object size $os^{(it+1)}$;
    Check the difference between old and new object size estimates: $os_{diff} = |os^{(it+1)} - os^{(it)}|$;
    it = it + 1
end

Algorithm 6.2: Summary description of the proposed short term redundancy constraint.

Long Range Diffusion

Second, it is also noted that while an object's shape and position can be very different in a longer time frame, their overall appearance features (*e.g.* color) usually do

not change significantly. For this reason, color information is used to propagate binarization results across long temporal windows. In particular, HSV color features are extracted from each frame and the pixelwise colors are converted into region based color features by extracting superpixels [43] and calculating a color histogram **fc** (with 3 channels and 20 bins per channel) for each region (*i.e.* superpixel). Using the color features, a pure appearance based similarity matrix, $D \in \mathbb{R}^{N \times N}$ is calculated as

$$D(i,j) = exp(-\frac{||\boldsymbol{fc}_i - \boldsymbol{fc}_j||_2^2}{\sigma}), \forall (j) \in NN(i) \tag{6.17}$$

with σ being a hyperparameter controlling the degree of similarity and $NN(i)$ being the set of nearest neighbors of superpixel i in the feature space, within a temporal neighborhood of radius r and a spatial neighborhood spanning the entire frame. The similarity matrix, D, creates a graph over the video in feature space where nodes are connected based on their similarity in feature space independently of their spatiotemporal position. Using the similarity matrix, the initial estimates are propagated via diffusion as

$$v^{(i)} = Dv^{(i-1)}, \tag{6.18}$$

where $v^{(0)}$ corresponds to the estimates used to initialize the diffusion process.

The diffusion operation, (6.18), is performed in two phases. In the first phase, the post short term redundancy saliency maps are used for initialization (*i.e.* the results obtained after the short term redundancy stage, $v^{(0)} = MA^{(r)}$) and the radius of the temporal neighborhood used in the nearest neighbor search is set to $r = r_1$. In the

second phase, the binary masks obtained after phase 1 are used for initialization and the temporal radius, r, is set to $r_2 > r_1$. The obtained binary masks, B, after phase 2 represent the final results of the proposed algorithm.

Notably, while previous work also relied on a similar diffusion process for long range temporal consistency, *e.g.* [45, 77], they usually rely on different appearance features entailing significant redundancy (*e.g.* use of both LAB and RGB colors at the same time in addition to HOG features). In addition, while previous work only relied on a saliency map for diffusion, in the proposed approach diffusion is also based on a binary mask, which proved far more beneficial in preliminary experiments.

6.2.4 Implementation Details

Convolution with Gaussian 2^{nd}-order derivatives is realized with separable 5-tap filters, with 10 orientations as specified in Section 6.2.1. (Preliminary experiments showed that 2^{nd}-order derivatives yielded better performance than either 3^{rd} or 4^{th}-order and that complementing the 2^{nd}-order derivatives with their Hilbert transforms yielded no benefit). The number of iterations over the network, *i.e.* the number of layers, is chosen to maintain a large enough support to obtain a reasonable saliency map using the adopted feature contrast definition, even while avoiding undue border effects depending on the size of the input spacetime volume while. These considerations led to using a 4 layer architecture in each stream. Therefore, the final representation relies on 8 motion features coming from the motion stream (*i.e.* 4 from the medium speed motion stream and another 4 from the slow motion stream) and 4 appearance features coming the appearance stream. To impose short term

temporal consistency, the short term redundancy enforcement process described in Algorithm 6.2 is repeated in multiple iterations until the difference in the object size estimated between two iterations is less than 10^{-2}. For long term temporal consistency, the diffusion process relies on a radius $r = r_1 = 15$ for the nearest neighbor search in the first phase and a radius r_2 that spans the entire duration of the video in the second phase.

6.3 Empirical Evaluation

The proposed algorithm is evaluated on the challenging SegTrack-v2 dataset [103], which contains 14 videos presenting the different challenges that can be encountered in the task of video object segmentation. Although 14 videos are available in SegTrack-v2, it is common practice to ignore the *penguin* sequence (*e.g.* [45, 135, 102]) as its groundtruth is misleading (*i.e.* it somewhat arbitrarily covers a couple of penguins only, while the video contains many penguins moving similarly and at the same time). Therefore, this sequence will not be considered in our experiments.

Since the proposed algorithm is built around two main steps (*i.e.* **1)** video to saliency map and **2)** saliency map to binary mask), two metrics are used to evaluate the proposed approach to video object segmentation. In particular, the Area Under the Curve metric (AUC) is used to evaluate the quality of the saliency map and the Jaccard index (*i.e.* Intersection over Union; IoU) is used to evaluate the quality of the binary mask. The following sections present results validating the different

components of the proposed algorithm.

6.3.1 Component-Wise Validation: Raw Video to Saliency Map

The first step in the proposed algorithm is to turn the raw videos into saliency maps (*i.e.* highlight regions likely to contain the object of interest). Therefore, the first set of experiments empirically validate the different steps involved in obtaining the saliency maps.

6.3.1.1 Border-Aware Saliency Estimation and Classification

To obtain layerwise saliency maps, the following three steps are employed: **1)** use the feature contrast definition, (6.1), to identify salient regions across the video frames, **2)** impose a constraint on the location of the objects, (6.4), by attenuating saliency near frame borders (*i.e.* convert saliency maps to Border Aware saliency maps by introducing the spatiotemporal center prior) and **3)** use the layerwise features to classify salient regions and obtain finer grained saliency maps highlighting the probabilities of belonging to foreground/background for each pixel. Table 6.1 documents the benefits of each one of those three steps at each layer of the proposed network. The results show that independently of the layer and modality (*i.e.* motion vs appearance), there is always a benefit in adding each component of the proposed saliency estimation algorithm as evidenced by the results improving from left-to-right in Table 6.1, with the improvement brought by imposing the border-aware location prior being especially striking. Based on these results, all subsequent experiments

	M_CC-Fast			M_CC-Slow			S_CC		
	FC	FC+LP	FC+LP+C	FC	FC+LP	FC+LP+C	FC	FC+LP	FC+LP+C
Layer 1	66.5	69.9	**72.4**	63.6	68.9	**71.4**	66.4	70.1	**72.6**
Layer 2	70.9	73.6	**75.9**	65.2	70.9	**73.0**	68.9	72.3	**73.2**
Layer 3	74.2	76.2	**77.8**	68.1	73.7	**75.3**	72.4	75.1	**75.4**
Layer 4	75.6	76.7	**78.1**	69.2	74.6	**75.8**	73.6	**75.7**	**75.2**

Table 6.1: Benefits of the different saliency estimation stages proposed at multiple layers and speeds. Average AUC(%) on the SegTrack-v2 dataset, with the different saliency estimation stages proposed, at multiple layers and speeds is used for comparison. Here, FC refers to Feature Contrast based saliency estimation without the spatiotemporal center prior applied, FC+LP refers to the introduction of the border aware Location Prior (LP), (6.4), and FC+LP+C refers to the Classification based saliency map refinement introduced in Section 6.2.2.2. Also, M_CC-Fast, M_CC-Slow and S_CC correspond to features extracted after the cross-channel pooling block of each layer in the medium speed motion stream, slow motion stream and the appearance stream, respectively.

presented in this chapter rely on saliency maps obtained using all three components.

Notably, these results also highlight the importance of the proposed multilayer architecture where the quality of the saliency maps improves from one layer to another, with the highest AUC score achieved by the last layer. These results confirm that the features extracted from the proposed network become more abstract (*i.e.* capture progressively more complex features) as we iterate through layers, thereby allowing higher layers to better localize the object of interest. However, given that video object segmentation also requires fine grained localization, lower layers, which focus on lower level features with increased spatial precision, are also important for these purposes.

6.3.1.2 Multiple Layers and Speeds

While Table 6.1 highlighted the benefits of the multilayer architecture, here the complementarity of the various layers and modalities is demonstrated. For this purpose, the saliency maps obtained from the various layers of the network are merged and the combined saliency maps are evaluated. To assess the benefit of the feature selection strategy introduced in Section 6.2.2.3, it is compared to merging the various saliency maps via simple averaging. Table 6.2 confirms the complementarity between the different layers where the results of combining the per modality saliency maps of all layers via simple averaging are better than taking any layer individually. Importantly, the overall best results were obtained by merging the different modalities (*i.e.* motion at various speeds and appearance) using feature selection. These results demonstrate the importance of both motion and appearance in the task of video saliency detection and ultimately segmentation. Notably, while the feature selection strategy brought modest improvement in the quality of the saliency map using the AUC metric, it still plays a key role in the subsequent steps of processing as will be demonstrated in the next Section.

Feature Selection	M_CC-Fast	M_CC-Slow	S_CC	All
No	78.5	76.6	77.3	78.6
Yes	79.2		75.3	**79.6**

Table 6.2: Benefits of merging the various layers and modalities with vs. without feature selection is reported in term of AUC (%) on SegTrack-v2. For the sake of comparison, results of simple averaging across layers and modalities, without selection, is reported in the second row of the final column.

6.3.2 Component-Wise Validation: Saliency Map to Binary Mask

The second step in the proposed algorithm is to convert the saliency maps into a binary mask (*i.e.* obtain a fine grained binary mask based on the highlighted salient regions). Similar to Section 6.3.1 here the different steps involved in obtaining the binary masks are validated empirically.

6.3.2.1 Framewise Initialization

The proposed binarization strategy falls under the diffusion based paradigm, which requires an initial framewise estimation and a similarity based diffusion matrix to impose temporal consistency. Here, the components involved in obtaining the initial binary mask estimation are evaluated first.

The results reported in Section 6.3.1 documented the key role of classification (in improving individual saliency maps; see Table 6.1) and feature selection (in merging the initial saliency maps; see Table 6.2). The importance of those blocks in the first binarization phase is further confirmed here. In particular, Table 6.3 shows the benefits of these blocks on the initial framewise binary mask. Notably, these results specifically highlight the complementarity of the classification block and the feature selection block, with good improvement obtained when the two blocks are taken together. More importantly, Table 6.3 shows the key role and validity of the motion and appearance priors introduced to further refine the initial saliency maps. The validity of those priors is especially clear considering that they always improve the overall IoU score independently of the presence of the other building blocks. Finally,

Classification	Feature Selection	Not-static & Not Bgrd assumption	IoU(%)
-	-	-	19.4
✓	-	-	33.8
-	✓	-	11.2
-	-	✓	29.3
✓	✓	-	37.4
-	✓	✓	24.3
✓	-	✓	37.7
✓	✓	✓	**40.7**

Table 6.3: Performance comparison in IoU(%) highlighting the contributions of the different components of the algorithm used to merge the various saliency maps to obtain the initial binary mask (*i.e.* prior to starting the diffusion process). ✓ indicates that the corresponding processing block is activated.

the best overall results are obtained in the presence of all building blocks, which speaks decisively for the usefulness and complementarity of each block.

6.3.2.2 Temporal Consistency

Once the necessary input to initialize the diffusion process is obtained, the next step in the proposed algorithm is to enforce temporal consistency on the initialization. Temporal consistency is enforced by imposing short term redundancy on shape and location first and enforcing long range appearance similarity second. As previously mentioned in Section 6.2.3.2, long range similarity is imposed by performing diffusion in two phases. The main difference between the two phases is the fact that phase 2 can see all frames in the video as opposed to phase 1 where a small temporal window of radius, r_l, is used. Similar to the previous experiment, the role of the individual components is assesses here by introducing them one block at a time as summarized in Table 6.4. Consistent improvements are observed with the addition of each block.

Short term red	Phase 1	Phase 2 on Sal phase 1	Phase 2 on Bin phase 1	IoU(%)
-	-	-	-	40.7
✓	-	-	-	44.1
-	✓	-	-	57.6
✓	✓	-	-	63.4
✓	✓	✓	-	63.5
✓	✓	-	✓	**69.2**

Table 6.4: Performance comparison in IoU(%) highlighting the contributions of the different components of the algorithm used to propagate the initial binary mask via diffusion. ✓ indicates that the corresponding processing block is activated.

Interestingly, long range diffusion plays a key role with an improvement of ≈17% over the initial binary mask. On the other hand, while short term redundancy alone brings a more modest improvement, *i.e.* ≈4% , it is important to note that using long range diffusion achieves much better results, ≈23%, when short term redundancy is included. Finally, it is interesting to note that once smooth binary masks are obtained, allowing a second round of diffusion using those binary masks as initialization allows for a final boost in the overall performance owing to the improved initialization in the second phase.

6.3.3 Comparison to state-of-the-art

Finally, comparison is now made to all unsupervised methods for video object segmentation reporting results on the SegTrack dataset (KEY [102], FST [135], NLC [45]) including the current state-of-the-art method [77]. In addition, comparison is also made to a state-of-the-art learning based two stream architecture designed for the task of video object segmentation [82]. Following standard protocol when reporting overall video object segmentation results, the IoU metric is used. The results in Table 6.5 show that the proposed approach achieves competitive results in the

Video	KEY [102]	FST [135]	NLC [45]	Hu et al., [77]	FSG [82]	Ours
Learning?	No	No	No	No	Yes	No
birdfall	0.490	0.014	0.565	0.649	0.380	0.605
bird of paradise	0.922	0.837	0.814	0.937	0.699	0.886
bmx	0.630	0.621	0.754	0.847	0.591	0.707
cheetah	0.281	0.396	0.518	0.518	0.596	0.469
drift	0.469	0.811	0.741	0.829	0.876	0.726
frog	0.000	0.629	0.713	0.832	0.570	0.700
girl	0.877	0.441	0.860	0.846	0.667	0.736
hummingbird	0.602	0.335	0.624	0.464	0.652	0.717
monkey	0.790	0.699	0.823	0.739	0.805	0.623
monkeydog	0.396	0.523	0.525	0.381	0.328	0.456
parachute	0.963	0.839	0.859	0.937	0.516	0.900
soldier	0.666	0.453	0.692	0.800	0.698	0.712
worm	0.844	0.705	0.782	0.800	0.506	0.752
Average IoU	0.610	0.562	0.713	**0.737**	0.606	0.692

Table 6.5: Comparison to state-of the art unsupervised methods for video object segmentation on the SegTrack dataset.

majority of cases, while it only falls behind the current state-of-the-art by a small margin. Importantly, it is of particular note that the proposed analytically defined architecture outperforms by a large margin a two-stream learning based architecture specifically designed for the task of video object segmentation [82].

6.3.3.1 Qualitative evaluation

Finally, in addition to the detailed quantitative evaluation presented throughout this empirical evaluation section, Figure 6.3 provides a detailed qualitative evaluation comparing the results obtained using the proposed algorithm (on the left) to the provided groundtruth (on the right) for examples derived from the SegTrack dataset. The middle frame from each video is used for illustration here and a side-by-side comparison for the entire duration of each video is made available at http://www.cse.yorku.ca/~hadjisma/vos.html/.

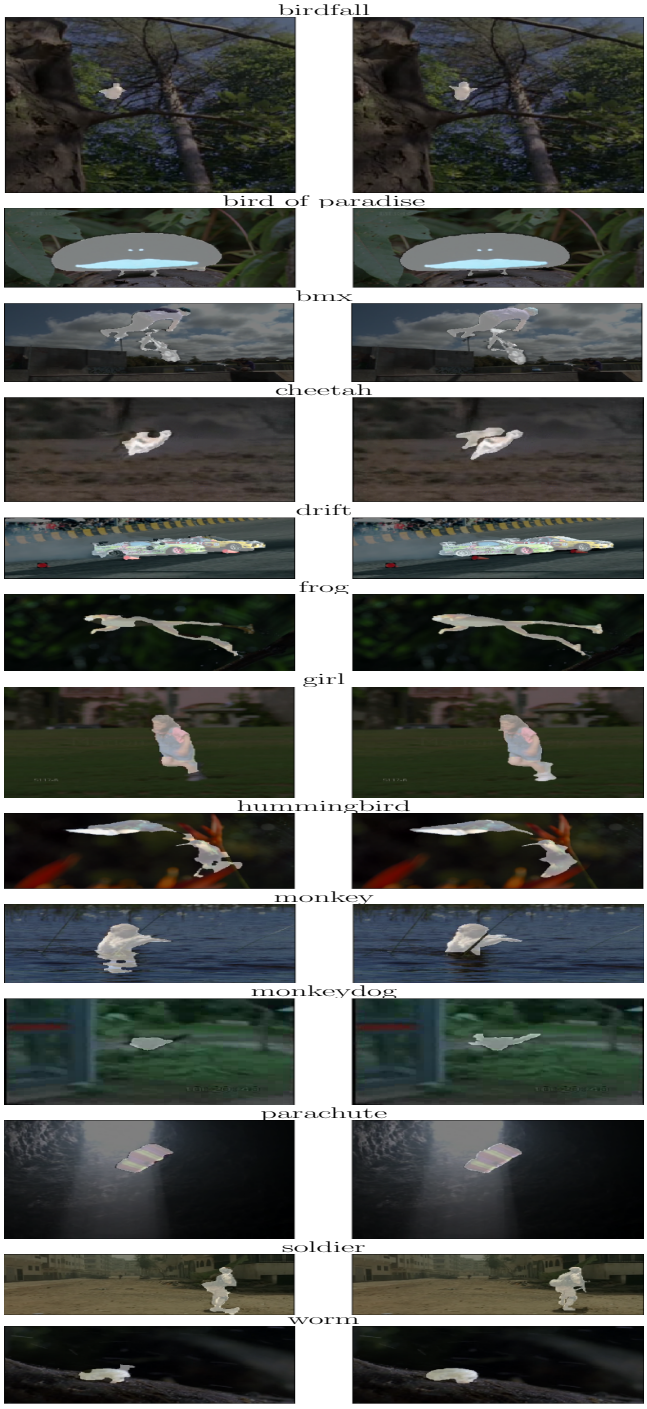


Figure 6.3: Qualitative evaluation of the presented VOS algorithm on the SegTrack dataset with our results shown on the left and the groundtruth shown on the right.

6.4 Summary and Discussion

In summary, this chapter introduced a two-stream variant of the SOE-Net architecture that was dubbed MSOE-SO-Net and its performance was validated on the challenging task of video object segmentation. Similar to Chapter 4, the role of each component of the proposed algorithm was empirically validated. Notably, the overall conclusions previously made in Chapter 5 with respect to the potential complementarity between the motion and appearance stream were judiciously used in this chapter with application to video object segmentation that is especially amenable to showing this complementarity. In comparison to the state-of-the-art, the developed approach showed strong performance.

Chapter 7

On the Role of Learning

7.1 Introduction

Previous chapters in this dissertation focused on the description and evaluation of the analytically defined ConvNet architecture introduced in Chapter 3, under various settings. However, none of the previously presented evaluations involved learned components. Therefore, the goal of this chapter is to introduce such components into the proposed analytic framework to specifically investigate the role of learning.

7.1.1 Motivation

The SOE-Net model introduced in Chapter 3 was designed to be a generic architecture for video understanding. While its efficacy was demonstrated in this dissertation on different tasks, it remains interesting to evaluate the effect that learning can have on such an architecture. This investigation is especially compelling given the simi-

larly generic nature of learned ConvNets, which when trained with a specific dataset are still able to adapt across various tasks and domains. Indeed, one of the unexpected benefits of training multilayer architectures is the surprising adaptability of the features learned in the lower layers across different datasets and even different tasks. Many state-of-the art recognition (*e.g.* [27, 105, 162]) and detection (*e.g.* [61]) systems rely on transfer learning where network parameters are initialized from models trained with a large scale dataset such as ImageNet [151] and later finetuned for the specific task at hand. Systematic exploration of transfer learning strategies even demonstrated that finetuning higher layers only while keeping the lower layers fixed led to systematically better performance when compared to finetuning the entire network [188].

The adaptability of features extracted with multilayer architectures suggests that features extracted at lower levels of the hierarchy tend to be common across different tasks, thereby making multilayer architectures more amenable to transfer learning. Similarly, investigations that visualized the filters learned in the lower layers of most ConvNet architectures (*e.g.* [191, 165, 159, 10]) revealed the presence of structured kernels similar to oriented bandpass filters. These findings suggest that it is interesting to pursue the idea of a hybrid architecture whereby some components are fixed based on analytic principles, while allowing other aspects to be learned and adapt to the specifics of the task at hand. In addition to reducing the number of learned parameters, thereby requiring less training data, this strategy also has the potential to yield more interpretable architectures. Therefore, the work presented in this chapter aims at investigating the role of learning in the realm of the analytic

approach to network realization introduced in Chapter 3 and evaluating the relative benefits of a hybrid architecture involving SOE-Net. The specific targeted task is dynamic texture recognition.

7.1.2 Related Work

Recent research on ConvNet design made use of the observations mentioned above regarding the nature of the learned filters in early layers of a ConvNet architecture. Examples include work learning convolutional filters on networks initialized with predefined bases sets, *e.g.* [81, 30, 182, 112, 198], or work including nonlinearities that model phase information present in such bases, *e.g.* [159]. Other work more explicitly aims at designing a hybrid architecture with fixed handcrafted layers defined over SIFT features, which are locally encoded with Fisher Vectors, and learned layers composed of fully connected layers [137]. More closely related to the work presented in this chapter is the hybrid variant of ScatNet [132, 134], which relies on the ScatNet architecture [18], described earlier in Chapter 2, for lower layers in combination with learned architectures including local encoding layers and ResNets. While the hybrid variant of ScatNet explores the benefit of added learned layers, it takes an ad-hoc approach to selecting the number of ScatNet layers considered as well as the type of learned layers. In contrast, the work proposed in this dissertation pursues a more controlled approach to hybrid network design, thereby shedding more light on the role of learning in a hybrid setting.

7.1.3 Contributions

In light of previous research, the contributions of this chapter are as follows. **1)** A systematic evaluation of the role of learning with each SOE-Net layer is conducted to identify optimal points for introducing learning. **2)** Consideration is given to various learned architectures in combination with SOE-Net's analytically defined layers. The considered learned architectures include fully connected architectures, depthwise (*i.e.* 1×1) convolutional layers and standard learned 3D convolutional layers as well as their combinations. **3)** An analysis of the learned layers is provided, which sheds light on their benefits and thereby opens new avenues for future improvements to the analytic approach. **4)** An evaluation of the hybrid architecture with training data of various sizes is provided to study its stability in the face of limited training data.

7.2 Technical Approach

7.2.1 Augmenting SOE-Net with Learning

While SOE-Net's building blocks are defined analytically based on theoretical considerations, it can nevertheless be augmented with learning capabilities. Here, the motivation is to add more flexibility to the network, such that it can adapt across various applications, while still minimizing heavy reliance on learning. Learned components can be introduced at various stages throughout the network such as learning dataset biases via introduction of a learned bias term in the convolutional layers, learning various filter parameters such as the orientations of the basis set, or aug-

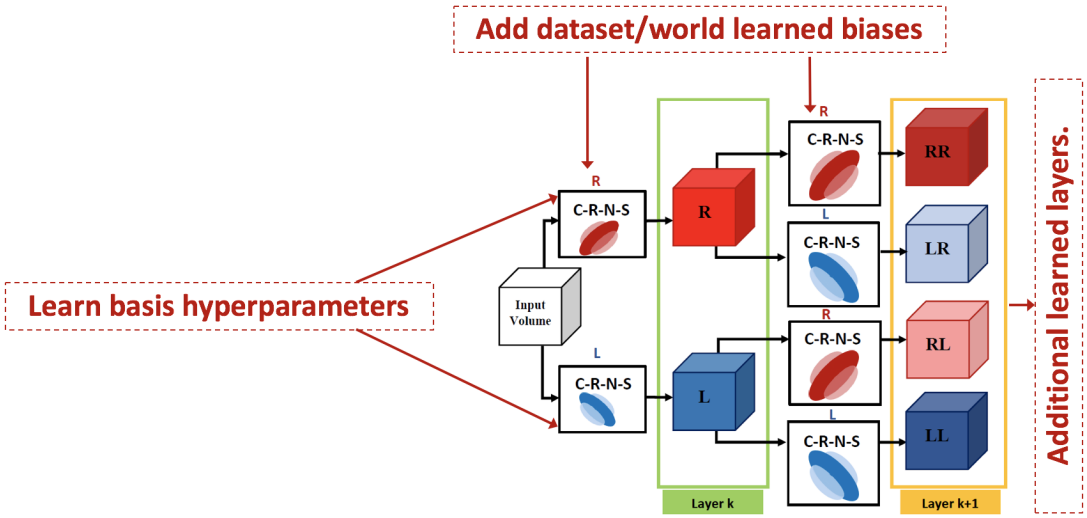


Figure 7.1: Potential places for augmenting SOE-Net with learning capabilities.

menting the network with fully learned layers by building on the analytically defined components, *e.g.* see Figure 7.1. Further, the analytically defined network can be used as the basis for fine-tuning, analogous to fine-tuning a pretrained network (*e.g.* via transfer learning); however, in this case the pretrained network has been analytically defined and therefore did not itself require (pre)training.

In this work, the focus is on training additional layers attached to the otherwise analytically defined SOE-Net layers, as is typically done in transfer learning settings [188], to yield a hybrid network. In particular, after K iterations through SOE-Net's recurrence, the output of the last layer is followed by a learning based architecture, as indicated on the far right side of Figure 7.1. In other words, SOE-Net can be considered as a replacement for the costly pretraining of a typical learned architecture that usually requires a very large dataset for video understanding tasks. A systematic exploration of various learned architectures is considered in this work to identify

good practices for achieving an optimal hybrid architecture. The particular learned architectures that are added to SOE-Net are as follows.

Fully Connected Layers

Fully connected layers provide one-to-one connections between all units of any two consecutive layers and thereby support global computations. They were exemplified in the earliest artificial neural networks and remain standard as the upper layers in most contemporary ConvNet architectures, *e.g.* [95, 166, 71, 163, 162, 169], as they have shown significant success in combination with lower convolutional layers.

Depthwise Convolutional Layers

Depthwise convolutions were first successfully introduced into ConvNets design with the NiN architecture [105] and since became part of several popular ConvNet architectures, *e.g.* [166, 71, 50]. Similar to the cross-channel pooling block used in SOE-Net, their main goal is to achieve dimensionality reduction. Depthwise convolutional layers reduce redundancies in the learned filters by learning combinations along the channel dimension, while otherwise leaving the pixelwise feature maps untouched.

Learned 3D Convolutional Layers

Learned 3D convolutional layers are used here as the natural substitute for the analytically defined convolutional layers of SOE-Net that rely on a predefined 3D oriented basis set.

7.2.2 Training Details

A variety of complementary training options were considered in augmenting SOE-Net with learned layers. Specifically, in terms of training strategies, a non-trivial number of variations were considered to identify the ideal training hyperparameters for providing a fair comparison across experiments, including various settings of the learning rate, batch sizes as well as weight decay. Also, different methods to combat overfitting were considered including variations of dropout ratios, data whitening and learning rate scheduling. Out of the various tested combinations, the results reported below are obtained as follows.

After K iterations through SOE-Net the last layer is followed by a global sum pooling layer in which the extracted features are aggregated over the entire spacetime volume (except for the case of depthwise convolutions, in which case the learned layers are introduced prior to global pooling). Subsequently, one of the learned architectures mentioned above (*i.e.* fully connected, depthwise convolution or learned 3D convolution) is added. Implementation details for these architectural layers are provided in Sections 7.3.1 through 7.3.3. Finally, a softmax layer is added to yield classification.

The network was trained with a mini batch size of 32 for a total of approximately 100 epochs with early stopping in case of overfitting. It was found that using two different learning rates during training yielded the best results across experiments. In particular, the learning rate is set to 10^{-4} for the learned layers, whereas a higher learning rate of 10^{-3} is used for the last softmax layer. The weight decay is set to 5×10^{-4} and the dropout ratio for fully connected layers is set to 0.5 in all experiments.

More details pertaining to the implementation of each learned architecture evaluated here are presented in their corresponding sections below.

7.3 Empirical Evaluation

Given that the SOE-Net architecture was thoroughly investigated under the task of dynamic texture recognition, this application will again be used in this chapter for the sake of completeness and consistency with the rest of this dissertation. In particular, the evaluations presented here will rely on the dynamic organization of the DTDB dataset introduced in Chapter 5. DTDB is chosen here for two main reasons. First, it is the only dynamic texture dataset that is large enough to support training based evaluations. Second, it provides a true dynamics based organization allowing better judgment on the various networks' abilities to capture pattern dynamics.

Notably, all experiments presented here relied on a reduced resolution version of DTDB where all sequences have been resized to a spatial resolution of 128×171. During training, centrally cropped clips of size $112 \times 112 \times 42$ are randomly sampled from each sequence, whereas all clips covering the entire duration of a sequence are considered during testing. Reliance on the reduced resolution version was necessitated by available computational resources to allow for training different networks of various sizes under the same input settings. As a point of comparison, a simple version of training-free SOE-Net is used for a baseline where the extracted features are used with a simple logistic regression classifier. Importantly, because all learning based experiments relied on the reduced resolution dataset, a smaller instantiation

of SOE-Net (*i.e.* using 4 layers and one scale only, as well as smaller 7-tap filters) is used here and therefore all results of the hybrid SOE-Net should be compared to this new baseline.

7.3.1 SOE-Net with Fully Connected Layers

The first natural place to augment SOE-Net with learning capabilities is via the introduction of additional Fully Connected (FC) layers. The results reported here were obtained using 2 fully connected layers of size 2048 each. Also, to identify layers of SOE-Net benefiting the most from introducing learned layers, instantiations of SOE-Net of various depths (*i.e.* layers, K) were used in conjunction with a global sum pooling layer, fully connected layers and a softmax.

	No-FC	Learned Parameters	With-FC	Learned Parameters
SOE-Net-1	29.6	-	53.6	∼4.2M
SOE-Net-2	62.8	-	66.8	∼5.0M
SOE-Net-3	64.1	-	66.9	∼5.8M
SOE-Net-4	**67.0**	-	**68.4**	∼7.5M

Table 7.1: Augmenting SOE-Net with fully connected layers. Two learned fully connected (FC) layers are introduced following a systematically varied number of analytically defined convolutional layers. In the first column, SOE-Net-K corresponds to a K layer SOE-Net. No-FC results are obtained with the learning free SOE-Net, whereas With-FC are obtained with additional learned FC layers. In this and all tables of results in this chapter, the reported performance numbers correspond to dynamic texture recognition accuracy on DTDB's dynamics organization. Also, the number of learned parameters is reported in the order of millions of parameters (M), unless otherwise stated, *e.g.* ∼4.2M corresponds to around 4.2 million parameters.

As can be seen in Table 7.1, the initial strategy for turning the learning free SOE-Net into a hybrid network via addition of learned fully connected layers, shows

very high promise and illustrates the benefit of augmenting SOE-Net with training. However, it is clear that fully connected layers become less useful as we add more convolutional layers to SOE-Net. For example, using 2 layers of SOE-Net and learned FC layers is almost equivalent to using 4 layers of SOE-Net without requiring any learned parameters. Nonetheless, we can see that the system always learns some useful information by attaching FC layers to the last layer of SOE-Net, independently of the number of SOE-Net layers used. It is notable, however, that all improvements offered by augmenting SOE-Net with learning come at a considerable cost of needing to learn many parameters (*e.g.* up to 7.5 million with SOE-Net-4). Still, these results suggest an interesting way forward to further investigate hybrid architectures and identify the balance between analytically defined and learned layers for optimal performance.

7.3.2 SOE-Net with Depthwise Convolutions

Following previous work, *e.g.* [132, 134], this experiment explores the benefits of using Depthwise Convolutions (DC) as a learned local encoding method that learns pixelwise combinations over the features extracted with SOE-Net. This strategy allows for exploring the benefits of learning relationships along the various features dimensions only (*i.e.* without directly altering the features along the spatiotemporal dimensions, as would be the case using standard convolutions). Notably, as opposed to previous work, *i.e.* [132, 134], no fully connected layers are used at this stage to avoid confounding variables in the analysis of depthwise convolutions. In particular, one layer of 1×1 convolutional filters is attached to the K^{th} layer of SOE-Net with

a target reduction in the dimensionality of the extracted features of one order of magnitude. The resulting feature maps are then followed by a global pooling layer and a softmax as in the previous experiment. Notably, this experiment was not performed for a 1 layer instantiation of SOE-Net due to the very small number of feature maps at layer 1, thereby making such an experiment trivial (*i.e.* SOE-Net-1 only has 20 channels, as opposed to SOE-Net-2 for example, which includes 400 channels).

	No-DC	Learned Parameters	With-DC	Learned Parameters
SOE-Net-1	29.6	-	-	-
SOE-Net-2	62.8	-	59.2	∼16K
SOE-Net-3	64.1	-	64.1	∼65K
SOE-Net-4	**67.0**	-	60.0	∼258K

Table 7.2: Augmenting SOE-Net with depthwise convolutional (DC) layers. One layer of depthwise (*i.e.* 1×1) convolutions is introduced on top of a systematically varied number of analytically defined convolutional layers. In the first column, SOE-Net-K corresponds to a K layer SOE-Net. No-DC results are obtained with the learning free SOE-Net, whereas With-DC are obtained with additional learned DC layers. Notably, here the number of learned parameters is reported in the order of thousands of parameters (K), *e.g.* ∼258K corresponds to around 258 thousand parameters.

While the results in Table 7.2 suggest that depthwise convolutional layers are not as beneficial in the pursuit of a hybrid architecture as fully connected layers, those results also reveal a rather interesting behavior for the 3 layer instantiation of SOE-Net. Indeed, while using DC layers lead to a decrease in performance compared to learning-free SOE-Net, with layer 4 suffering the most, it is notable that SOE-Net-3 results are not similarly affected. This makes layer 3 a potential candidate position to introduce learned components.

Based on this observation, it is interesting here to visualize the filter weights

learned at layer 3 for further analysis. To this end the depthwise learned filters are extracted and the weights representing the contributions of each SOE-Net-3 channel in the encoding are summed and summarized in a histogram. Given that the learned filters entail positive and negative weights, two histograms (representing the amount of times a given channel participates in the encoding positively and negatively) are generated, as shown in Figure 7.2. These results show that all channels participate in the encodings to various extents. More importantly, these results show the importance of capturing negative combinations of channels as evidenced by the prevalence of learned negative weights across channels. Notably, such negative weights were also present in the weights learned at other layers. Such negative combinations are not explicitly modeled in the SOE-Net architecture and this suggests an interesting way forward to improving the analytic definition of SOE-Net by capturing such combinations.

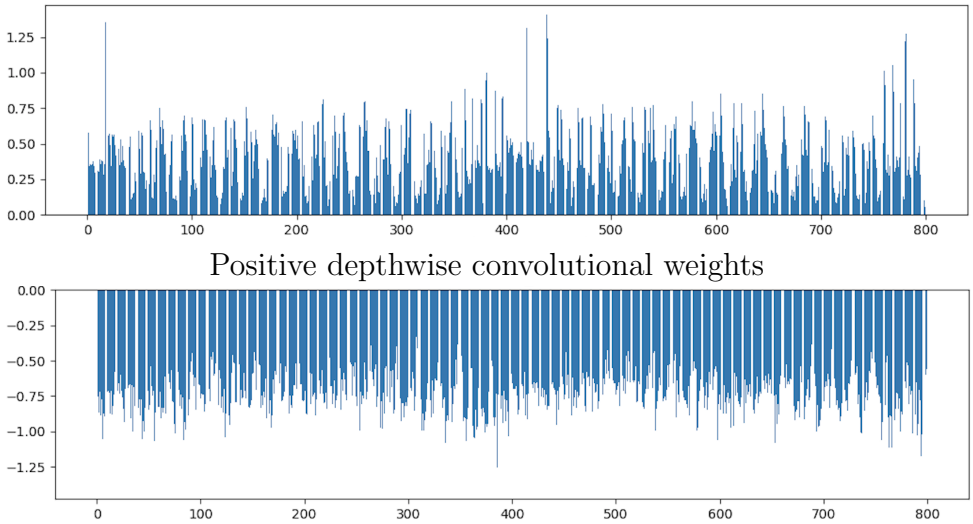


Figure 7.2: Analysis of learned depthwise convolutional filters.

7.3.3 SOE-Net with 3D Convolutional Layers

Next, it is interesting to replace SOE-Net analytically defined layers with Learned 3D Convolutional (L3D) layers. Here, the goal is to identify layers that may benefit the most from learning. To achieve this goal, SOE-Net layers are slowly replaced with learned layers starting from the last layer of SOE-Net, thereby going from a completely learning-free architecture (*i.e.*[SOE-Net-4, L3D-0]) to a fully learned network (*i.e.*[SOE-Net-0, L3D-4]). Each learned layer introduced is set up to match exactly its corresponding SOE-Net layer both in terms of the number of input and output channels as well as the filter sizes (*i.e.* $7 \times 7 \times 7$ kernels are used). Also, following standard practices in defining learned 3D spatiotemporal ConvNets (*e.g.* [169, 23]), each L3D layer is followed by a ReLU rectification and a pooling layer with a stride of 2.

		Accuracy	Learned Parameters
No learned Layers	[SOE-Net-4, L3D-0]	67.0	-
Replacing SOE-Net with L3D	[SOE-Net-3, L3D-1]	**68.6**	∼4.4M
	[SOE-Net-2, L3D-2]	68.0	∼5.5M
	[SOE-Net-1, L3D-3]	64.2	∼5.7M
	[SOE-Net-0, L3D-4]	60.8	∼5.8M

Table 7.3: Replacing SOE-Net layers with learned 3D (L3D) convolutional layers. SOE-Net analytically defined layers are progressively replaced with standard learned 3D convolutional layers. In this case, [SOE-Net-K, L3D-N] corresponds to a K layer SOE-Net attached to a N layer L3D convolutional layers

The results summarized in Table 7.3 demonstrate that learning layers 3 and 4 yield better results in comparison to the learning-free SOE-Net, with the greatest benefit coming from learning layer 4 alone. Reflecting back on the results of the previous experiment, the overall outcome of the current experiment further indicate

the adequacy of introducing learned components after layer 3 of SOE-Net.

Given the superiority of a learning-based layer 4 in comparison to the learning-free layer of SOE-Net, it is compelling to compare the learned filters to the analytic basis set. For this purpose, all filters learned at layer 4 are compared to each one of the 10 oriented 3^{rd} order Gaussian derivative filters used in SOE-Net, using Normalized Cross Correlation (NCC). The NCC scores for all learned filters are averaged for each oriented filter separately, while making sure to treat positive and negative correlations separately (*i.e.* positive NCC scores are averaged together and similarly for negative NCC scores). The results, summarized in the histograms shown in Figure 7.3, reveal two interesting features of the learned filters. First, it is seen that there exists almost as many positively correlated filters as negatively correlated ones. This result suggests the learning of similar filters that are simply out of phase versions of one another. This observation is in line with previously reported research analyzing the shape of learned filters, *e.g.* [159], and speaks decisively in favor of the two-path rectification strategy adopted in SOE-Net. Second, it is seen that the learned filters are relatively correlated with the oriented basis set employed in SOE-Net, to various extents. A closer inspection of the learned filters, as shown in Figure 7.4 depicting the most highly correlated learned filter with one of the oriented basis filters, further confirms the presence of an oriented bandpass structure in the learned filters. This observation further indicates the suitability of using oriented basis sets at higher layers, as advocated in the design of SOE-Net. On the other hand, the relatively lower correlation scores suggests a potential way forward, which may involve learning the specific filter tunings to better adapt to the data.

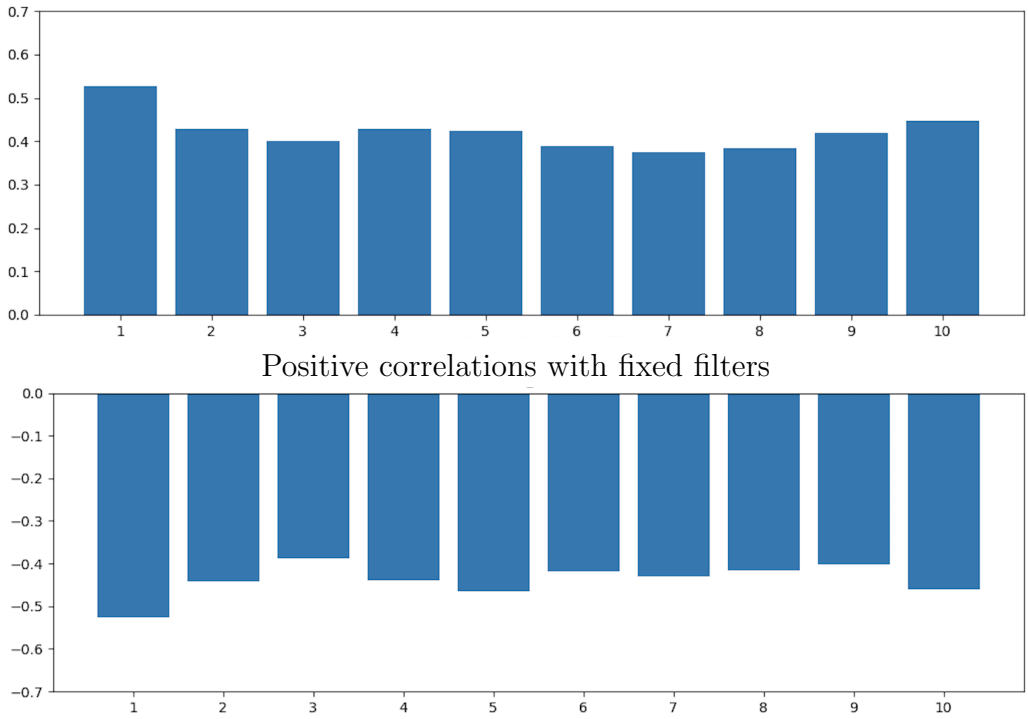


Figure 7.3: Analysis of learned 3D convolutional filters.

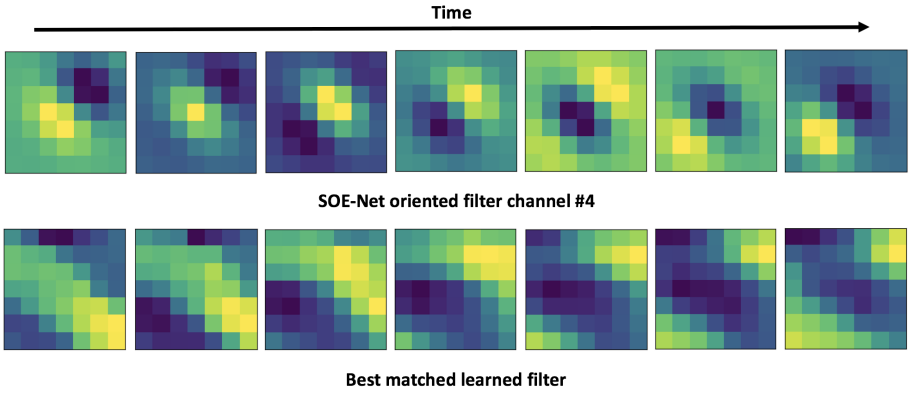


Figure 7.4: Visualization of one of the filters learned at Layer 4 in the employed [SOE-Net-3, L3D-1] architecture. The filters shown depict the most highly correlated pair with the (top) filter being one of the channels in the oriented basis set used in SOE-Net and the (bottom) filter being the learned filter with the highest NCC score when compared to the top filter.

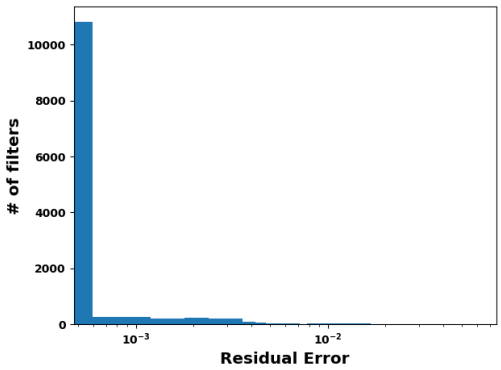


Figure 7.5: Distribution of the residual error resulting from fitting each filter learned at layer 4, in the [SOE-Net-3, L3D-1] setting, to a 3^{rd}-order Gaussian derivative filter.

Recall that the orientations of the analytically defined basis filters used in SOE-Net were set to uniformly sample the set of directions in 3D (Section 3.2.2); however, the specific orientation of that overall sampling was arbitrarily specified. Therefore, it is possible that the learned filters correspond to 3D oriented bandpass filters along different directions and scales than those of the analytically defined basis. To evaluate this possibility, all of the learned filters were least-squares fit to a 3^{rd}-order Gaussian derivative model. The distribution of resulting residuals is shown in Figure 7.5, which strongly suggests that the learned filters are in fact well modeled as oriented bandpass filters. This result suggests that the lower correlations shown in Figure 7.3 arise not because the learned filters are not oriented bandpass in nature, but rather because the learning has selected a different set of orientations and scales from those that were specified for the analytically defined set.

Overall, these results further emphasize the suitability of oriented bandpass filters at higher layers of a ConvNet and show the potential role of learning the specific filter tunings in future instantiations of the SOE-Net model.

7.3.4 Hybrid SOE-Net

The analysis of hybrid models presented so far considered three potential learned architectures in isolation; however, these architectures are usually used in combination in standard learned ConvNets, *e.g.* [105, 166, 176]. Therefore, it is now interesting to study the potential benefit of their combination. Based on the results of the previous experiments, the combined architectures are introduced after layer 3 of SOE-Net.

	FC	L3D-1	DC	Accuracy	Learned Parameters
SOE-Net-3	-	-	-	64.1	-
	✓	-	-	66.9	∼5.8M
	-	✓	-	68.6	∼4.3M
	-	-	✓	64.1	∼65K
	✓	✓	-	**70.0**	∼8.8M
	✓	-	✓	68.3	∼4.4M
	✓	✓	✓	69.8	∼9M

Table 7.4: SOE-Net hybrid architecture. SOE-Net analytically defined layers are combined with various learned layers to identify the optimal hybrid architecture. Here, FC, L3D-1 and DC layers rely on the same settings used in the experiments exploring their use individually as described in Sections 7.3.1, 7.3.2 and 7.3.3, respectively. ✓ indicates active components.

Table 7.4 presents a detailed analysis of the various combinations considered. From these results it appears that using fully connected layers in combination with any other architecture is always beneficial. This observation may be attributed to the fact the FC layers entail the largest number of learned components. Interestingly, using depthwise convolutions in combination with standard 3D convolutions does not bring an added value. Overall, combining the spatiotemporal SOE-Net architecture with a learned spatiotemporal architecture (*i.e.* using 3D convolutions) and fully connected layers results in the best hybrid architecture. Motivated by this

		No-FC	Learned Parameters	With-FC	Learned Parameters
SOE-Net-3	L3D-1	68.6	∼4.3M	70.0	∼8.8M
	L3D-12	73.5	∼10.7M	**74.2**	∼15.6M
	L3D-123	**74.2**	∼16.3M	72.5	∼20.4M

Table 7.5: Deeper hybrid SOE-Net. Here L3D-1, L3D-12 and L3D-123 correspond to adding 1, 2 and 3 Learned 3D convolutional layers following a 3 layer SOE-Net as well as 2 fully connected layers of size 2048 each.

observation, experiments including more L3D layers attached to SOE-Net-3, followed by 2 fully connected layers of size 2048 each, were also considered and the obtained results are summarized in Table 7.5. Here, adding more than one L3D layer (*i.e.* for L3D-12 and L3D-123) necessitated the use of smaller tap filters, to minimize border artifacts, due to the reduction in the resolution of the data caused by the pooling blocks used in each L3D layer; see Section 7.3.3 for the definition of an L3D layer. Specifically, filters of size $5 \times 5 \times 5$ and $3 \times 3 \times 3$ are used in L3D layers 2 and 3 respectively, whereas filters of size $7 \times 7 \times 7$ are used in all SOE-Net layers as well as L3D-1.

The results in Table 7.5 show the benefit of adding more learned layers on top of SOE-Net. First, it is seen that, in the presence of fully connected layers, adding 2 learned convolutional layers brings a larger improvement in performance compared to its 3 layer counterpart. This behavior can largely be attributed to the significant increase in the number of learned parameters in the latter case. Conversely, when no fully connected layers are used, more learned convolutional layers continue to improve the overall performance. These results are in line with the observations reported in Section 7.3.1, which demonstrated that adding more SOE-Net layers makes the fully connected layers less beneficial. Finally, reflecting back on the overall conclusion of

Chapter 5, this experiment confirms the benefits of further developments involving learning based approaches in combination with SOE-Net.

7.3.5 Hybrid SOE-Net with Limited Data

The results of the previous section resulted in the design of a hybrid SOE-Net architecture and demonstrated its benefits in terms of overall performance. It is now interesting to investigate the benefits of this hybrid architecture in terms of its stability in the face of limited training data. For this purpose, a hybrid architecture consisting of a 3 layer SOE-Net, 2 L3D layers and 2 FC layers is used, *i.e.*[SOE-Net-3, L3D-12, FC-2], as it yielded the right balance between performance and number of learned components; see Table 7.5. In addition to the settings used in the previous experiment, where the entire train set is used for training hybrid SOE-Net, here other settings with decreasing amounts of training data are also considered. Specifically, $X\%$ samples are randomly selected from each class in the training set, while still using the entire test set for testing. The variations considered include using 75%, 50% and 25% of the entire training set. For comparison, an equivalent fully learned architecture (*i.e.* corresponding exactly to the hybrid architecture employed here) dubbed L3D-Net, is also trained under the same settings. Notably, the number of training parameters in L3D-Net corresponds to ∼17M, which accounts for an increase of only ∼1.4M parameters compared to its hybrid SOE-Net counterpart.

The results, summarized in Figure 7.6, show that hybrid SOE-Net performance degrades less rapidly compared to L3D-Net. For example, when only 50% of the data is used for training the performance of hybrid SOE-Net drops by ∼9% whereas

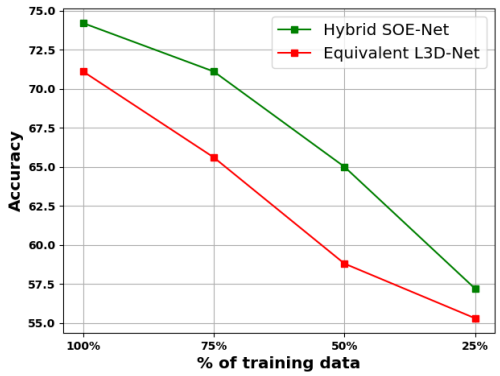
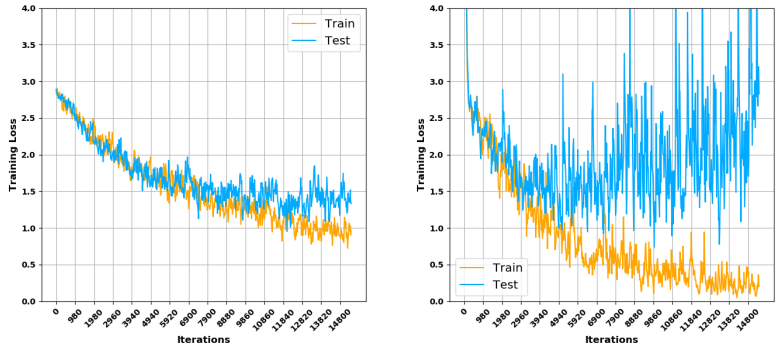


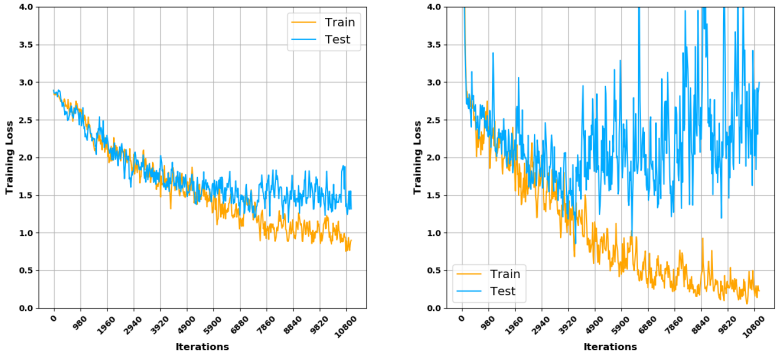
Figure 7.6: Performance of hybrid SOE-Net, using the [SOE-Net-3, L3D-12, FC-2] setting, compared to its fully learned counterpart with limited training data. Here comparison is made in terms of overall classification accuracy when using the entire test set for testing and different amounts of training data presented during training.

L3D-Net performance drops by ∼13% under the same settings. These results speak in favor of hybrid SOE-Net stability compared to its fully learned alternative, even while the number of learned parameters in the latter is not much larger than the former, *i.e.* L3D-Net has only 8% more parameters to learn compared to hybrid SOE-Net.

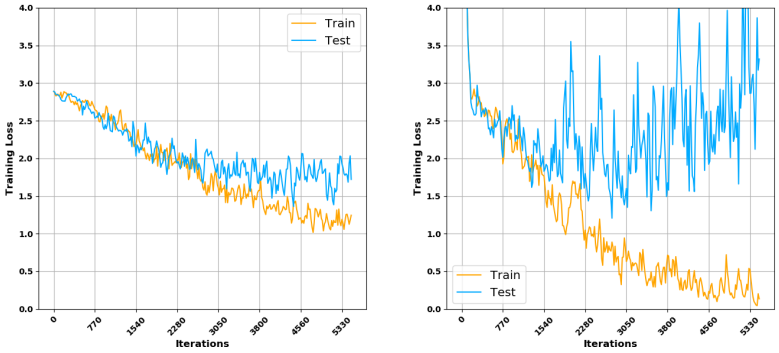
To further evaluate the stability of hybrid SOE-Net compared to its fully learned counterpart, it is important to examine their behaviors during training. Figure 7.7 depicts training versus testing loss curves for the different settings considered with data collected every 20 iterations during the training process. These results further confirm the superiority of hybrid SOE-Net when confronted with limited training data, where it incurs relatively low overfitting independently of the amount of training data used. On the other hand, dramatic overfitting is observed with L3D-Net early on during the training process.



(a) using 75% of the training data



(b) using 50% of the training data



(c) using 25% of the training data

Figure 7.7: Training behavior of hybrid SOE-Net compared to its fully learned counterpart with limited data. Here training stability comparison is made between (left) hybrid SOE-Net using the [SOE-Net-3,L3D-12,FC-2] setting and (right) its equivalent fully learned counterpart. Each plot depicts train (orange) and test (blue) loss curves with data collected every 20 iterations during the training process. (a) Corresponds to training with 75% of the data, (b) corresponds to training with 50% of the data and (c) corresponds to training with 25% of the data.

7.4 Summary and Discussion

In summary, this chapter presented a systematic exploration of the role of learned components in the realm of the analytically defined ConvNet presented in this dissertation. In particular, the focus in this chapter was on combinations of the analytically defined model, *i.e.* SOE-Net, with additional learned architectures. This systematic analysis yielded a hybrid architecture, which not only demonstrated better performance compared to its fully learned counterpart, but also proved more robust when confronted with scarce training data. These results show high promise for future work exploring more hybrid variations, including other learned architectures as well as introducing learning in some components of the SOE-Net model itself.

Chapter 8

Summary and Conclusions

8.1 Summary and Overall Discussion

The obscure nature behind the success of ConvNets has sparked a significant interest in understanding their behavior and their learned components, as was throughly discussed in Chapter 2. Correspondingly, the main goal of this dissertation, introduced in Chapter 1, was to shed light on the operations of ConvNets in an effort to understand this wide success. To achieve this goal, a controlled approach to network design was adopted to minimize the number of unknowns and thereby yield a more interpretable ConvNet. Consequently, this dissertation presented an analytic framework for network realization that is completely learning free and where all components are designed based on theoretical considerations. The resulting representation is therefore readily interpretable in terms of multiorientation, multiscale properties of the input imagery.

Overall, the key results that have been presented are as follows.

- This work is the first to propose an analytically defined ConvNet architecture for spacetime image understanding, SOE-Net. The main features of the proposed architecture, rigorously defined in Chapter 3, can be summarized in five main points as follows: **1)** A small set of theory motivated, analytically defined filtering operations are employed in cascade to yield a hierarchical representation. **2)** The same set of operations are applied repeatedly via a recurrent connection; however, different information is extracted at each layer as the input changes due to the operations of the previous pass. **3)** A two-path rectification is used to explicitly capture phase information that is usually learned in standard ConvNets. **4)** Divisive normalization is applied to minimize sensitivity to multiplicative contrast variations and for practical signal representation. **5)** The signal properties and the knowledge of feature map definitions (*i.e.* as all other aspects of the network are known) are employed to determine pooling parameters precisely, both spatiotemporal and cross-channel.

- The proposed analytical framework to network design was thoroughly evaluated on two different tasks in Chapters 4 through 6, *i.e.* dynamic texture recognition and video object segmentation. The efficacy of the approach was demonstrated where it yielded state-of-the-art performance compared to its alternatives, even while considering other learning based architectures.

- It was demonstrated that analytic principles can be applied with success to more than one architecture design. In particular, a 3D spatiotemporal ar-

chitecture, *i.e.* SOE-Net, was used for a task that relies on capturing pattern dynamics (*i.e.* dynamic texture recognition) and a two-stream architecture, *i.e.* MSOE-SO-Net, for a task relying on the complementarity between appearance and motion information (*i.e.* video object segmentation).

- In addition to the main theoretical contributions, this dissertation also introduced the world's largest dynamic texture database (DTDB) in Chapter 5 and provided two different organizations (*i.e.* appearance and dynamics based). DTDB has been released to the public. This dataset allowed for uniquely insightful experiments regarding the ability of major classes of spatiotemporal ConvNets to capture pattern dynamics versus appearance and participated in better understanding their shortcomings. Also, the availability of such a large scale and complementary dataset highlighted the importance of adapting the data to the end task in transfer learning settings, as shown in Appendix B.

- In conjunction with the new database, a new tack on learning based architectures also was proposed via the introduction of the MSOE-two-stream described in Chapter 5. The use of a new input representation in this network further highlighted the importance of using theoretically motivated considerations in filter specification, even when those considerations are used to modify the first layer representation only.

- While the adopted controlled approach to network design yielded a strong learning-free model, the value of introducing learned components was also considered in Chapter 7. That analysis further highlighted the benefits of the pro-

posed framework, allowing for the design of a robust hybrid architecture taking the best from both worlds. In addition, the systematic analysis of hybrid architectures suggested ways to improve the learning-free SOE-Net. Indeed, these suggestions lead to interesting directions for future research, as discussed in the next section.

8.2 Future Work

In light of the work presented in this dissertation, several directions for future research can be considered, as follows.

- From a model learning perspective, the benefit of introducing learned components to an otherwise analytically defined ConvNet was documented in this dissertation via adding learned layers after the analytically defined layers. In future work, further considerations for learning specific network parameters within the SOE-Net model itself can be examined. Potential places that can benefit from this approach include learning specific filter tunings such as filter orientations and scales as well as additional biases. Also, the analytically defined SOE-Net could itself be fine-tuned to adapt to specific tasks. The results of such an exploration could lead to directions for introducing further analytic definitions into the model design, thereby yielding even more interpretable hybrid architectures.

- From a network design perspective, it should be noted that all architectures considered in this dissertation are feedforward. However, it would be interest-

ing to consider variations that include feedback connections as well. Notably, it is known that such connections are prevalent in the visual cortex [96]. For example, use of feedback connections could be used to produce an even shallower network, while maintaining performance. Further, use of feedback connections has been shown to yield especially good performance with applications involving grouping and attention such as segmentation [89] and detection [160]. Therefore, including such connections is an especially compelling direction for future work involving the use of the analytic framework presented in this dissertation. For example, it would be interesting to examine the benefit of such connections with application to video object segmentation.

- From the dataset perspective, DTDB with its dual organization can be used in other tasks involving dynamic texture description. For example, the dual organization can be used for an attribute based dynamic texture description task (*e.g.* pluming vs. boiling volcano or turbulent vs. wavy water flow), as done in static scene attribute based description [183]. Also, the same dataset can be used in multi-modal description tasks in which the sound produced by the texture in motion can be used as well. (Audio tracks are available for many of the DTDB entries.) In addition, DTDB can be used as a valuable tool for several other applications beyond dynamic texture description alone. Examples include dynamic texture synthesis as well as dynamic scene segmentation and recognition, where dynamic textures constitute the building blocks of such scenes.

Appendix A

SOE-Net Visualizations

Figure 3.2 presented in Chapter 3 (reproduced here as Figure A.1 for convenience) provides visualizations of layerwise processing results of SOE-Net on two different examples. Here a detailed description of the operations illustrated in the figure is provided. Snapshots from the corresponding video[1] as well as additional real life examples are shown in Figures A.2 and A.3.

Notably, for the sake of clarity in illustration, only a small number of orientations (3), scales (1) and layers (2) are used in the visualizations. A larger number is used in practice, as specified in Chapter 3.

[1]The corresponding video as well as corresponding real life examples can be found at: https://www.youtube.com/watch?v=1IR76ADMjJA

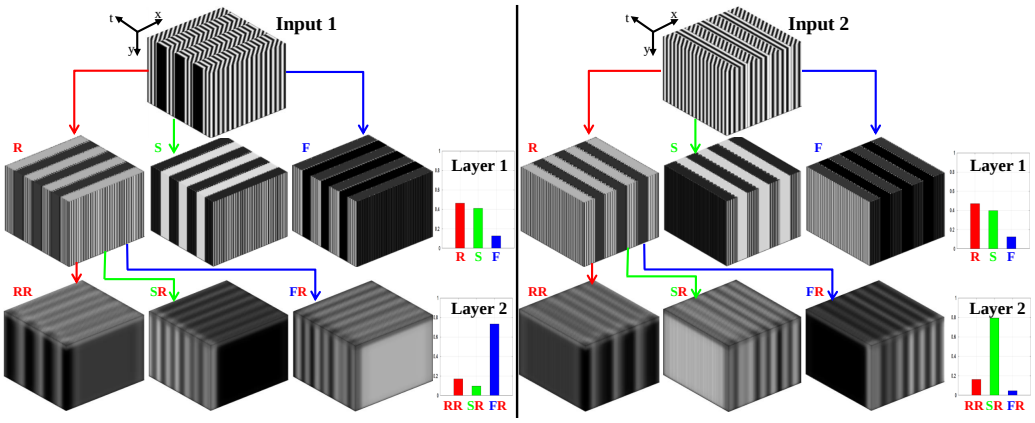


Figure A.1: Emergence of abstract features via repeated filtering. (Left) Input synthetic sinusoidal pattern alternately moves right (orientation along x-t diagonal) and stays static (orientation parallel to time axis). (Right) Same pattern moving right behind a static picket fence with same spatial pattern (*i.e.* background motion viewed between the spaces in a stationary picket fence). **SOE-Net** is used with filters tuned to **R**ightward motion, **S**tatic (no motion) and **F**lickering (pure temporal change). **L**ayer **1** captures the local rightwardly moving and static portions; but not the more abstract alternating temporal move-stop pattern of the left input, nor the fact that the right input maintains the same spatially interleaved moving and static stripes. Indeed, measurements aggregated over the volumes (shown as histograms on the right) are the same at this layer. The difference is revealed at $\boldsymbol{L_2}$ after applying the same filters on the **R** response of $\boldsymbol{L_1}$: The move-stop behavior of the left texture becomes explicit and yields a large **F** response. In contrast, the constant rightward motion and picket fence of the right texture yield a large **S** response. In practice, more directional filters are applied at each layer; see Sec. 3.2.2.

A.1 The Move-Stop Example

The first pair of examples, depicted in Figure A.2, provide visualizations of synthetic and natural image sequences showing a pattern that alternately moves and then stops across time. The synthetic input is the same as illustrated on the left side of Figure A.1. A synthetic example was used in the body of the dissertation for the sake of clarity with regards to the exact input image composition. The natural image analogue that is additionally provided here shows that the pattern of results generalizes to real-world scenarios.

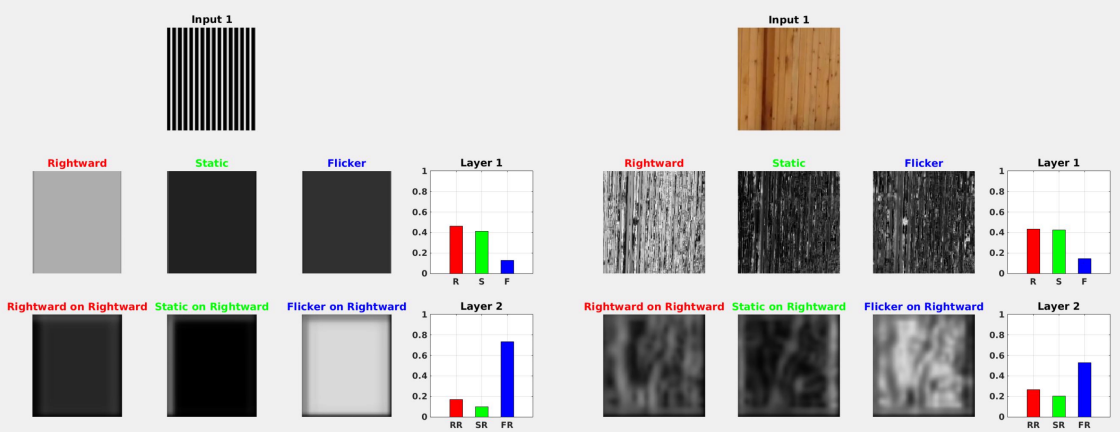


Figure A.2: Snapshots from the synthetic move stop example (left) as well as its corresponding real life example, illustrating a fence alternately moving and stopping (right).

The synthetic example in this pair depicts a vertically oriented spatial sinusoidal grating that alternates moving rightward one pixel/frame and remaining stationary in time. It is seen in Layer 1 filter results that when the pattern is in motion there are relatively large responses in the Rightward channel and small responses in the Static channel, whereas when the pattern is stationary there are relatively small responses

in the Rightward channel and large responses in the Static channel. In contrast, there are only negligible responses in the Flicker channel throughout. (Larger responses are shown as brighter image intensities.) The aggregate responses for each channel over the entire spacetime volume, shown as histograms at the right side of the Layer 1 depiction, reflect these response patterns well. Thus, Layer 1 results accurately capture the local motion of the input pattern. Notably, however, the more global spatiotemporal pattern of move-stop is not made explicit at this layer of processing.

The illustrated results for Layer 2 are generated by applying the same set of three filters (Rightward, Static and Flicker) to the Rightward output of Layer 1. In contrast to Layer 1 results, at Layer 2 it is seen that throughout the entire video there are large responses only in the Flicker channel. Thus, Layer 2 results capture the more global move-stop pattern in terms of temporal change (*i.e.* Flicker). Again, the aggregate responses shown as histograms at the right side of the Layer 2 depiction reflect these patterns well.

The natural image example of this pair depicts a vertically oriented bamboo fence that alternates moving rightward and remaining stationary in time. The pattern of processing results is exactly analogous to what was found for the corresponding synthetic example: At Layer 1 large responses alternate in time between the rightward and static filterings as the input pattern moves and stops, respectively. At Layer 2 a large response is found only in the Flicker filtered result, capturing the temporal change as the fence moves and stops.

Overall, this pair of examples visualize how Layer 1 results capture local pattern structure (*e.g.* local motion), while Layer 2 captures more global and abstract

structure (*e.g.* spatiotemporal arrangement of local motions).

A.2 The Picket Fence Example

The second pair of examples, shown in Figure A.3, provide visualizations of synthetic and natural image sequences showing background motion as seen through the spaces in a static picket fence. The synthetic input is the same as illustrated on the right side of Figure A.1. As before, a synthetic example was used for the sake of clarity with regards to the exact input image composition. The natural image analogue that is additionally provided here shows that the pattern of results generalizes to real-world scenarios.

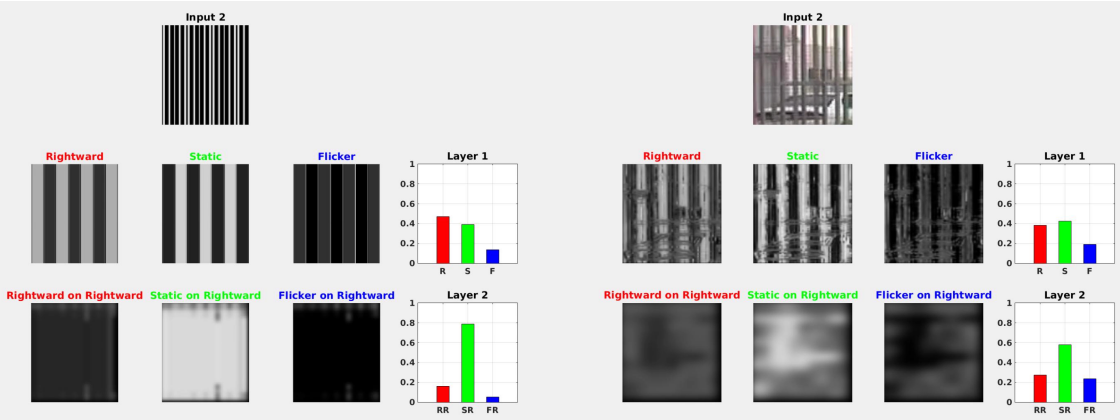


Figure A.3: Snapshots from the synthetic picket example (left) as well as its corresponding real life example, illustrating a picket fence moving in front of a static one (right).

The synthetic example in this pair depicts the same spatial pattern as in the previous synthetic example, but now it moves rightward (1 pixel/frame) behind a static picket fence with the same spatial pattern. At Layer 1, the locally moving vs. static

portions of the pattern show as corresponding spatially alternating stripes of strong responses in the Rightward and Static channels, respectively. In contrast, there are only negligible responses in the Flicker channel throughout. (Larger responses are shown as brighter image intensities.) The aggregate responses for each channel over the entire spacetime volume, shown as histograms at the right side of the Layer 1 depiction, reflect these response patterns well. Thus, Layer 1 results accurately capture the local motion of the input pattern. Notably, however, the more global temporal consistency of the pattern (*i.e.* the same interleaved stripes of constant velocity background motion and stationary foreground fence persists throughout the input) is not made explicit at this layer of processing. Indeed, the aggregate histogram responses at this layer are the same for this pattern and the previous move-stop pattern (*i.e.* the total amount of local rightward vs. static vs. flicker structure is the same).

As in the previous examples, the illustrated results for Layer 2 are generated by applying the same set of three filters (Rightward, Static and Flicker) to the Rightward output of Layer 1. In contrast to Layer 1 results, at Layer 2 the temporal consistency of the pattern (*i.e.* the pattern does not change with time) is made explicit as relatively large responses in the Static filtered results, with all other responses relatively small. Corresponding results are also shown in the histograms at the right side of the Layer 2 depiction.

Notably, Layer 2 histograms capture the differences between the move-stop example (large Flicker response) vs. the picket fence (large Static response).

The natural image example in this pair depicts a fence that is seen moving rightward between the spaces in a similar non-moving fence. The pattern of processing

results is exactly analogous to what was found with the corresponding synthetic example: At Layer 1 large responses alternate in space between the rightward and static filterings according to the spatial locations of the moving and static fence components, respectively. At Layer 2 a large response is found only in the Static filtered result, capturing the temporal consistency of the global pattern.

This pair of examples thereby reaffirm the ability of SOE-Net to progressively abstract from simple, local attributes to more abstract, global attributes across layers.

Appendix B

Additional Details on DTDB Definition and Use

This appendix consists of three main sections. Section B.1 provides an in depth discussion regarding the dynamics and appearance categories specification and thereby complements Section 5.2.1 of the main document. This section also serves as a guide to the visualizations and example sequences provided in Figures B.1 through B.5 as well as the corresponding video[1].

Section B.2 provides detailed description and implementation details of the baseline algorithms used in the paper and thereby complements Section 5.3 of the main document.

Finally, Section B.3.3 provides extensive results and additional comparisons with previously reported results using the baseline spatiotemporal ConvNets evaluated in Chapter 5 and it thereby complements Section 5.4 of the main document.

[1]A corresponding video, which also describes with examples the logic behind the choice of the 18 and 45 dynamic and appearance categories, respectively, can be found at: https://www.youtube.com/watch?v=Gj43-hGiso8&feature=youtu.be

B.1 DTDB Categories

B.1.1 Dynamic Categories Specification

As mentioned in Section 5.2.1, DTDB was created with the main goal of building a true *dynamic* texture dataset where sequences exhibiting similar dynamic behaviors are grouped together irrespective of their appearance. Previous work provided a principled approach to defining five coarse dynamic texture categories based on the number of spatiotemporal orientations present in a sequence [36], as given in the left column of Table 5.2 in the main document, which is reproduced as Table B.1 for convenience.

Original YUVL categories	DTDB categories		
Name/Description		**Name/Description**	**Example sources**
Underconstrained spacetime orientation	↓	**Aperture Problem**	conveyor belt, barber pole
		Blinking	blinking lights, lightning
		Flicker	fire, shimmering steam
Dominant spacetime orientation	↓	**Single Rigid Object**	train, plane
		Multiple Rigid Objects	smooth traffic, smooth crowd
		Smooth Non-Rigid Objects	faucet water, shower water
		Turbulent Non-Rigid Objects	geyser, fountain
		Pluming Non-Rigid Objects	avalanche, landslide
Multi-dominant spacetime orientation	↓	**Rotary Top-View**	fan, whirlpool from top
		Rotary Side-View	tornado, whirlpool from side
		Transparency	translucent surfaces, chain link fence vs. background
		Pluming	smoke, clouds
		Explosion	fireworks, bombs
		Chaotic	swarming insects, chaotic traffic
Heterogeneous spacetime orientation	↓	**Waves**	wavy water, waving flags
		Turbulence	boiling liquid, bubbles
		Stochastic	windblown leaves, flowers
Isotropic	↓	**Scintillation**	TV noise, scintillating water

Table B.1: Dynamics based categories in the DTDB dataset. A total of 18 different categories are defined by making finer distinctions in the spectrum of dynamic textures proposed originally in [36]. Subdivisions of the original categories occur according to increased variance (indicated by arrow directions) about the orientations specified to define the original categories; see text for details. Table reproduced from Table 5.2 for convenience.

The original enumeration of coarse categories [36] is used here as a point of departure, but the original categories are subdivided to yield a much larger set of

18 categories, as given in the middle column of Table B.1. The original categories are subdivided in a way that accounts for increased variance about the prescribed orientation distributions in the original classes, as follows.

1. Sequences with a globally *underconstrained spacetime orientation* (*i.e.* sequences that cannot be assigned an exact direction of motion because the recovery of their corresponding spacetime orientation is ambiguous) were split into three sub-categories starting from: (1) the most simple and typical example of underconstrained spacetime orientation which is the case of *aperture problem* sequences, next are (2) sequences dominated by pure temporal luminance changes which we refer to as *blinking* and finally (3) sequences with temporal luminance changes but tied with some level of spatial structural changes as well, *i.e.* *flicker*. See Figure B.1 for corresponding visual examples.

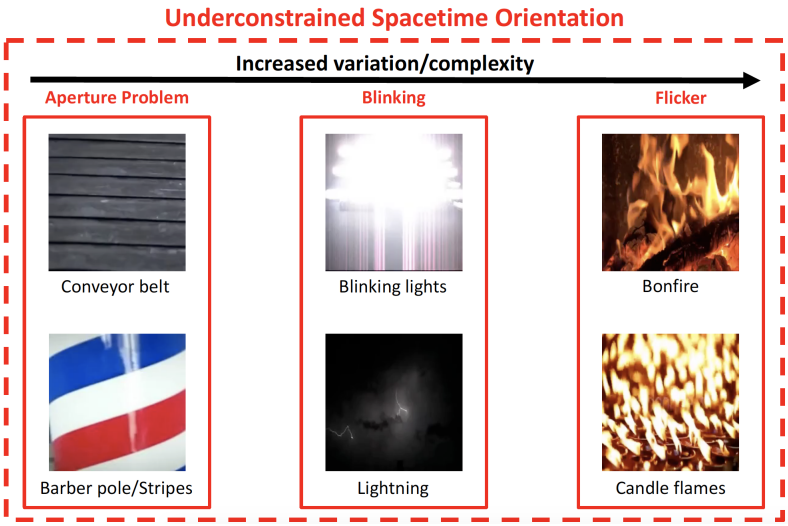


Figure B.1: Sample finer distinctions made within DTDB's underconstrained motion category.

2. Patterns falling under *dominant orientation* (*i.e.* sequences dominated by a single spacetime orientation) were split into five sub-categories containing: (1) *single rigid objects*, (2) *multiple objects*, (3) *non-rigid objects*, (4) *turbulent non-rigid* and (5) *pluming non-rigid* objects, all exhibiting motion along a dominant direction, albeit with increasing variance. See Figure B.2 for corresponding visual examples.

Figure B.2: Sample finer distinctions made within DTDB's dominant motion category.

3. Patterns with *multi-dominant* (but countable/distinguishable) spacetime orientations were split into six sub-categories by making finer grained distinctions according to an increased number of dominant spacetime orientations. In particular, the following distinctions are made: (1) simple *rotary motions seen from the top*, (2) simple *rotary motions seen from the side*, (3) two superimposed motions, *i.e. motion transparency or translucency*, (4) particles expanding in various directions *i.e. pluming*, (5) again particles expanding or diverging in

different directions but with increased variance, *i.e. explosion*, and finally (6) the extreme case of multiple objects all moving in different but distinguishable directions that we call *chaotic motion*. See Figure B.3 for corresponding visual examples.

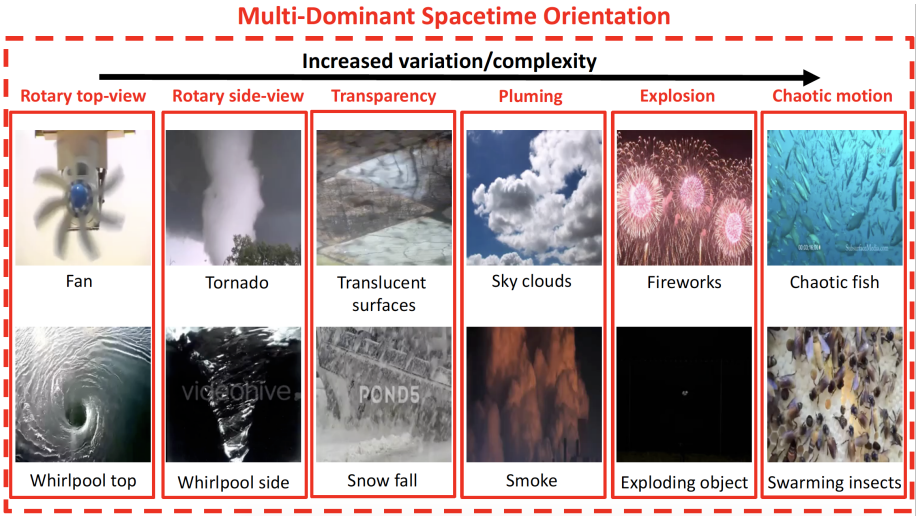


Figure B.3: Sample finer distinctions made within DTDB's multi-dominant motion category.

4. The next set of sequences are characterized by the composition of multi-dominant but *heterogeneous* spacetime orientations. Such patterns are subdivided into three sub-categories according to an increased level of complexity as (1) *wavy motion*, (2) *turbulent fluids* and (3) *stochastic motion*. See Figure B.4 for corresponding visual examples.

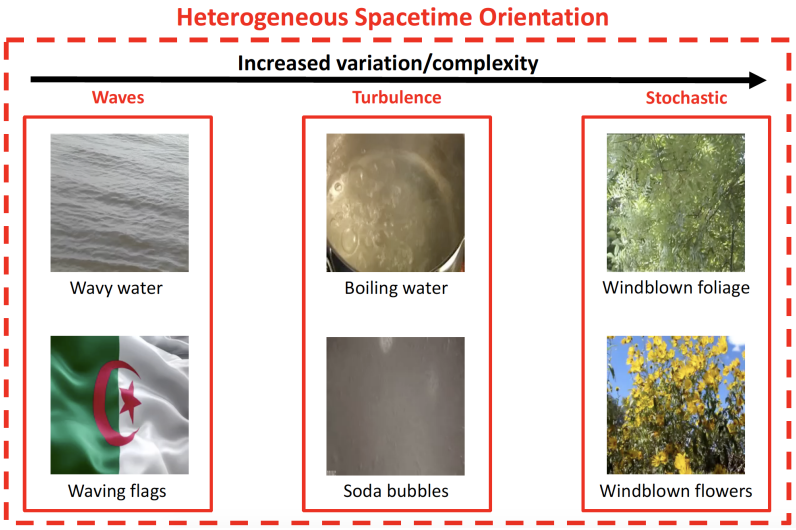


Figure B.4: Sample finer distinctions made within DTDB's heterogeneous motion category.

5. Finally, the extreme case of sequences falling under the original category *isotropic* does not permit further subdivision based on increased variance about its defining orientations, because although it may have significant spatiotemporal contrast, it lacks in discernible orientation(s), *i.e.* it exhibits isotropic pattern structure. See Figure B.5 for corresponding visual examples.

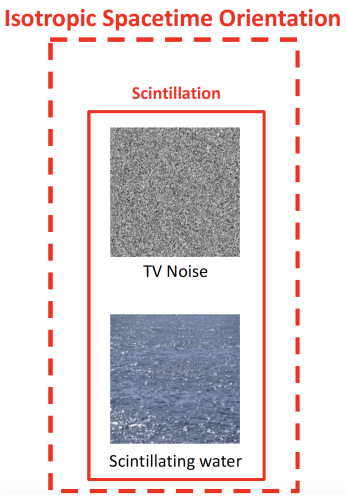


Figure B.5: Samples from DTDB's isotropic motion category.

In the end, 58 different keywords were used (*i.e.* appearance based video tags) to populate the entire dynamics based organization of the DTDB dataset and that resulted in DTDB-dynamics containing $\sim 10,020$ videos divided into 18 different dynamic categories.

B.1.2 Appearance Categories Specification

Because the main focus during data collection was dynamics, it was noticed that not all appearance based video tags (or keywords) generated enough appearance based sequences. For example, sequences with moving airplanes were used as part of populating the single rigid object dynamic category but taken alone they did not yield enough (*i.e.* > 100) appearance based sequences and therefore were not used in the appearance based organization. The same rule applied to sequences of shimmering steam (used in the flicker dynamic category) and translucent surfaces

(used in the transparency dynamic category), among others. Therefore, to keep the dataset balanced in the appearance organization and provide each category with a similar amount of training samples, any category containing less than 100 sequences was ignored in the appearance based organization.

In the end, 45 of the keywords (or appearance based video tags) were selected, which taken together covered all the dynamics categories to specify appearance based categories. The 45 selected keywords allowed for a complete and balanced appearance based organization of the new DTDB dataset with $\sim$ 9206 videos, which is also complementary to the dynamics based DTDB. Labels beneath the individual images in Figures B.1 - B.5 are indicative of the appearance categories.

B.2 Spatiotemporal ConvNets Evaluated: Implementation Details

B.2.1 C3D

The original C3D network with eight convolutional layers (*i.e.* learned convolutional filters, with interspersed nonlinear and pooling layers) followed by two fully connected layers and a softmax layer is used and trained on each organization of DTDB separately. Specifically, C3D with a softmax layer of 18 or 45 outputs is employed depending on whether the training is performed on the dynamics or appearance version of DTBD, respectively. For all experiments, the publicly available code of C3D[2] is used with input videos split into non-overlapping 16-frame clips, as described in

[2]https://github.com/facebook/C3D

the corresponding original paper [169].

Again, following the procedure described in the original paper for C3D, results reported in Table 5.3 are obtained by averaging the softmax scores of all the 16-frame clips of each video in the test set.

Transfer learning results reported in Tables B.2 and B.3 are obtained by applying the C3D network model, trained on either UCF101, DTDB-dynamics or DTDB-appearance, densely to 16-frame clips of the input videos in YUVL and YUP++. In particular, as mentioned in Section B.3.1, *pool5* layer features are extracted from each 16-frame clip and averaged across the entire video. Notably, although it is common practice to use fc6 features when using C3D in transfer learning experiments, preliminary evaluation revealed that for the case under consideration, features extracted from the last pooling layer of C3D yielded systematically better results than fc6 and therefore all transfer learning experiments rely on those features. These feature vectors are then normalized and fed directly to a linear SVM classifier for recognition on YUVL [36] and YUP++ [36], respectively. Training the SVM classifier is achieved using the standard leave-one-out procedure that is usually adopted with the YUVL and YUP++ datasets.

Finetuning results reported in Tables B.4 and B.5 are obtained following the same procedure described above for the transfer learning experiments. However, in this case, the C3D model pretrained on Sports-1M is used and finetuned on either DTDB-dynamics or DTDB-appearance. Additional results obtained by finetuning on UCF101 are also provided in Section B.3.3 of this appendix.

B.2.2 Two-Stream ConvNet

In all the experiments with two stream networks, the original Two-Stream network introduced in [162], where each stream is made of 5 convolutional layers followed by two fully connected layers and a softmax layer, is used. Similar to the training of C3D, each stream is trained separately from scratch on each version of DTDB (*i.e.* a softmax layer with 18 outputs is used when training on DTDB-dynamics and 45 outputs when training on DTDB-appearance). Here, it is important to note that these baseline appearance and motion streams involve a similar number of parameters to be learned to that of C3D (*i.e.* $\sim 88M$ learned parameters for each stream of the Two-Stream network and $\sim 80M$ for C3D), which allows for a straightforward and fair comparison between the two networks. In implementation, the publicly available code[3] for two stream networks is used. The RGB stream is trained with mini batches of randomly selected and cropped 224×224 RGB frames at the input as outlined in the original paper [162]. The same procedure is applied to the flow stream while replacing the RGB frames at the input with stacks of $L = 10$ optical flow channels, yielding an input to the temporal stream of size $224 \times 224 \times 2L$. The optical flow fields are extracted before training using the standard TVL1 optical flow algorithm [190].

At test time, the softmax scores of all frames of each video in the test set are averaged (*i.e.* similar to what was done for C3D) to obtain the final recognition accuracy results for each stream as reported in Tables B.2 and B.3.

Also similar to the procedure described in the previous section for C3D's eval-

[3]https://github.com/yjxiong/caffe

uation, results of transfer learning reported in Tables B.4 and B.5 are obtained by extracting *pool5* feature vectors from the RGB and Flow streams as trained from scratch under UCF101, DTDB-dynamics and DTDB-appearance. These features are similarly normalized and fed to a linear SVM classifier.

B.2.3 MSOE-two-stream

For the proposed MSOE-two-stream network, the same architecture used for the original two stream network [162] is employed. However, the use of stacks of optical flow channels as input to the motion stream is eschewed in favor of using stacks of appearance Marginalized Spatiotemporal Oriented Energy (MSOE) channels [36]. The MSOE representation was originally proposed in [36] to model dynamic textures and it relies on 3D, (x, y, t), oriented filters that are applied to an input video along multiple spacetime orientations to capture motion information along those various directions. In the MSOE-two-stream, the 2 optical flow channels that represent motion in the x and y directions are replaced with the MSOE channels that capture motion in various directions, θ_i, and at various speeds, S_j. In particular, 3D oriented 3^{rd}-order Gaussian derivative filters are used to extract 20 MSOE channels corresponding to: static (or no motion), flickering motion in the horizontal and vertical directions, as well as motions in the directions leftward, rightward, upward, downward and the four diagonals, all at two different speeds, S_j, corresponding to medium (one pixel per frame movement) and fast (two pixels per frame movement) speed. To this set an addition measurement is included that captures lack of structure in the sequences; see [36] for a detailed description and the theory behind the MSOE

representation.

Similar to what is done in the original optical flow motion stream, where the 2 optical flow channels are stacked along $L = 10$ consecutive frames yielding an input to the motion stream with $2 \times L = 20$ channels, stacks of the 20 MSOE channels of $L = 10$ consecutive frames are used as well. This results in an input to the MSOE stream with $20 \times L = 200$ input channels. Replacing optical flow with MSOE channels leads to the newly proposed motion stream dubbed MSOE stream. Combination of the MSOE stream with the RGB stream is performed via late fusion using a linear SVM on the softmax scores of the individual streams as outlined in the original Two-Stream network [162] and this combination via late fusion leads to the newly proposed MSOE-two-stream network for dynamic texture recognition. Code of MSOE-two-stream[4] is built on top of the original caffe implementation.

The exact same evaluation procedure outlined in the previous section for the standard two stream networks is adopted for evaluation of the newly proposed MSOE-two-stream. Notably, with application to the transfer learning experiments, fusion is performed through the concatenation of the *pool5* layers of the RGB and MSOE streams, as it does not make sense in this context to average those features since they no longer represent softmax scores; instead, they are features encoding two different but complementary information sources.

Finetuning results are obtained following the same procedure described for transfer learning experiments, while using an RGB stream model pretrained on ImageNet. This pretrained stream is then finetuned on DTDB-dynamics and DTDB-appearance

[4]https://github.com/hadjisma/

to obtain the results in Tables B.4 and B.5. Here, it is important to remember that in the original Two-Stream network [162] only the RGB stream was pretrained on ImageNet, while the Flow stream was always trained from scratch on UCF101. Hence, the same procedure is adopted in the finetuning experiment, whereby only the RGB stream is finetuned from ImageNet pretraining while using MSOE stream trained from scratch on either DTDB-dynamics or DTDB-appearance. Finally, similarly to what was done for C3D, results of finetuning the RGB stream on UCF101 are provided here.

B.2.4 Training

All baseline learning-based networks' (*i.e.* C3D, Two-Stream and MSOE-two-stream) hyper-parameters (*e.g.* momentum and dropout) are fixed as released in their respective original papers. Because the goal of Sections 5.4.1 and B.3.1 was to provide a fair evaluation of the networks under consideration without obscuring the results via use of a major confounding variable (*e.g.* use of the C3D model pretrained on Sports-1M vs two-stream models pretrained on ImageNet), all the networks were first trained from scratch on each dataset separately to obtain the results reported in Tables 5.3, B.2 and B.3. For fair comparison, all networks are trained using the same mini-batch size (set to 16 to fit GPU memory when using the large input size of the MSOE stream, *i.e.* $224 \times 224 \times 200$). Also, all networks are trained for the same number of iterations (set to 100,000 iterations) and using the same learning rate schedule starting from a learning rate of 10^{-2} and lowering it two times after 50K and 70K iterations. Importantly, following this training strategy also resulted

in no network incurring overfitting, as evidenced by only a minor difference between training and testing errors in all cases.

B.2.5 Finetuning

In all experiments involving finetuning of the learning-based networks the publicly available pretrained models were used following the same finetuning procedures outlined in each network's original paper. In particular, for finetuning experiments on C3D the available model as pretrained on Sports-1M is used and finetuned for 20K iterations starting from a learning rate of 10^{-3} that is lowered every 5K iterations. For the RGB stream, the available model pretrained on ImageNet is used and the finetuning procedure described in the original paper [162] is followed. Specifically, this network is finetuned for 20K iterations starting from a learning a rate of 10^{-2} that is lowered after 14K iterations.

B.2.6 SOE-Net

Following the original SOE-Net implementation presented in Chapter 4, a basis set of 3D third-order Gaussian derivative filters with 10 different orientations and 2 scales is used. Also, given the relatively large spatiotemporal support of all the sequences in the DTDB dataset, an instantiation of SOE-Net with 5 layers is used. All SOE-Net results are based on feature vectors formed through the concatenation of the last layer's output at all scales, again, as performed in the original paper [64]. These feature vectors are then normalized and fed directly to the classifier for recognition, as outlined in the original paper.

B.3 DTDB as a Training Substrate

In addition to using DTDB in its two organizations as an analysis tool to understand spatiotemporal ConvNets' strengths and weaknesses, it can also serve as a training substrate to advance research on dynamic texture recognition and other related tasks. Therefore, in complement to the analysis provided in Chapter 5, additional evaluations addressing questions from the dataset's perspective are presented here. In particular, the analysis presented here addresses the following questions: **1)** Does the new dataset provide sufficient challenges to drive future developments in spatiotemporal image analysis? **2)** Can the dataset be beneficial for transfer learning to related tasks? And if so, **3)** What organization of the dataset is more suitable in transfer learning? **4)** Can finetuning on the proposed dataset boost the state-of-the-art on related tasks even while using standard spatiotemporal ConvNet architectures?

B.3.1 Which Organization of DTDB Is Suitable in Transfer Learning?

B.3.1.1 Experimental Protocol

Transfer learning is considered with respect to a different dynamic texture dataset and a different task, dynamic scene recognition. The YUVL dataset [36] is used for the dynamic texture experiments and it is chosen as a representative of a dataset with categories mostly dominated by the dynamics of its sequences. For the dynamic scene experiment, the YUP++ dataset [51] is used. YUP++ is the largest dynamic

scenes dataset with 1200 sequences in total divided into 20 classes; however, in this case the categories are mostly dominated by differences in appearance. Notably, YUP++ provides a balanced distribution of sequences with and without camera motion, which allows for an evaluation of the various trained networks in terms of their ability to abstract scene dynamics from camera motion. Once again, for fair comparison, the various architectures trained from scratch on DTDB are used in this experiment because the goal is not to establish new state-of-the-art on either YUVL or YUP++. Instead, the goal is to show the value of the two organizations of the dataset and highlight the importance of adapting the training data to the application. The conclusions of this experiment are used next, in Section B.3.2, as a basis to finetune the architectures under considerations using the appropriate version of DTDB.

For both the dynamic texture and dynamic scenes cases, the relative benefits of training on the appearance vs. dynamics organizations of DTDB are considered. Also, comparison to training using UCF101 is considered as a representative of a similar scale dataset, but one that is designed for the rather different task of action recognition. Since the evaluation datasets (*i.e.* YUVL and YUP++) are too small to support finetuning, features are extracted from the last layers of the networks as trained under DTDB or UCF101 and used for recognition (as done previously under similar constraints of small target datasets, *e.g.* [169]). A preliminary evaluation comparing the features extracted from the last pooling layer, fc6 and fc7, of the various networks used, showed that there is always a decrement in performance going from fc6 to fc7 on both datasets and out of 48 comparison points the performance

of features extracted from the last pooling layer was better 75% of the time. Hence, results reported in the following rely on features extracted from the last pool layer of all used networks.

For recognition, extracted features are used with a linear SVM classifier using the standard leave-one-out protocol usually used with these datasets [36, 142, 64].

B.3.1.2 Results

To begin, results of transfer learning applied to the YUVL dataset, summarized in Table B.2, are considered. Here, it is important to emphasize that YUVL categories are defined in terms of texture dynamics, rather than appearance. Correspondingly, it is found that for every architecture the best performance is attained via pretraining on the DTDB dynamics-based organization as opposed to the appearance-based organization or UCF101 pretraining. These results clearly support the importance of training for a dynamics-based task on dynamics-based data. Notably, MSOE stream, and its complementary MSOE-two-stream approach, with dynamics training show the strongest performance on this task, which provides further support for MSOE filtering as the basis for input to the motion stream of a two-stream architecture.

Comparison is now made on the closely related task of dynamic scene recognition. As previously mentioned, although YUP++ is a dynamic scenes datasets its various classes are still largely dominated by differences in appearance. This dominance of appearance is well reflected in the results shown in Table B.3. As opposed to the observations made on the previous task, here networks benefited more from an appearance-based training to various extents with the advantage over UCF101

		YUVL-1	YUVL-2	YUVL-3
UCF101 based training	C3D	61.4	65.4	55.7
	RGB Stream	63.6	72.8	60.0
	Flow Stream	84.8	87.3	81.7
	MSOE Stream	80.0	80.2	74.4
	MSOE-two-stream	80.8	84.5	78.8
Dynamics based training	C3D	83.3	86.4	83.4
	RGB Stream	68.1	75.4	65.0
	Flow Stream	87.7	86.9	83.1
	MSOE Stream	**89.2**	**89.3**	**84.8**
	MSOE-two-stream	**90.7**	**91.4**	**87.6**
Appearance based training	C3D	82.2	85.4	80.9
	RGB Stream	67.6	72.8	64.3
	Flow Stream	86.7	85.7	81.3
	MSOE Stream	87.7	87.3	83.6
	MSOE-two-stream	89.8	90.2	86.7

Table B.2: Performance of spatiotemporal ConvNets *trained* using both organizations of DTDB on the various breakdowns of the YUVL dataset [36].

pretraining being particularly striking. In agreement with findings on the YUVL dataset and in Section 5.4.1, the RGB stream trained on appearance is the overall best performing individual stream on this appearance dominated dataset. Comparatively, MSOE stream performed surprisingly well on the static camera portion of the dataset, where it even outperformed RGB stream. This result suggests that the MSOE stream is able to capitalize on both dynamics and appearance information in absence of distracting camera motion. In complement, MSOE-two-stream trained on appearance gives the overall best performance and even outperforms previous state-of-the-art on YUP++ [51].

Notably, all networks incur a non-negligible performance decrement in the presence of camera motion, with RGB being strongest in the presence of camera motion and Flow suffering the most. Apparently, the image dynamics resulting from cam-

		YUP++(S)	YUP++(M)	YUP++
UCF101 based training	C3D	62.5	55.8	58.3
	RGB Stream	64.9	54.4	63.5
	Flow Stream	83.6	51.9	68.9
	MSOE Stream	74.3	52.7	62.0
	MSOE-two-stream	80.1	66.6	74.6
Dynamics based training	C3D	84.3	71.8	76.5
	RGB Stream	81.8	73.7	78.3
	Flow Stream	89.3	64.7	76.8
	MSOE Stream	90.0	67.5	78.4
	MSOE-two-stream	93.3	81.5	87.7
Appearance based training	C3D	85.0	73.7	78.1
	RGB Stream	82.0	**76.2**	**79.9**
	Flow Stream	90.6	65.8	77.0
	MSOE Stream	**91.0**	69.5	79.1
	MSOE-two-stream	**94.7**	**83.2**	**89.6**

Table B.3: Performance of spatiotemporal ConvNets *trained* using both organizations of DTDB on the **S**tatic and **M**oving camera portions of YUP++ and the entire YUP++ [51].

era motion dominate those from the scene intrinsics and in such cases it is best to concentrate the representation on the appearance.

B.3.1.3 Conclusions

This evaluation proved the expected benefits of the proposed dataset over reliance on other comparably sized datasets that are not necessarily related to the end application (*e.g.* use of action recognition datasets, *i.e.* UCF101 [164] for pretraining, when the target task is dynamic scene recognition, as done in [51]). More importantly, the benefits and complementarity of the proposed two organizations were clearly demonstrated. Reflecting back on the question posed in the beginning of this section, the results shown here suggest that neither of the organizations is overall better than another in considerations of transfer learning. Instead, they are complementary and can be used judiciously depending on the specifics of the end application.

B.3.2 Finetuning on DTDB to Establish New State-of-the-art

B.3.2.1 Experimental Protocol

This experiment evaluates the ability of the architectures considered in this study to compete with the state-of-the-art on YUVL for dynamic textures and YUP++ for dynamic scenes when finetuned on DTDB. The goal here is to further emphasize the benefits of DTDB when used to improve on pretrained models. In particular, we use the C3D and two-stream models that were previously pretrained on Sports-1M [90] and ImageNet [151], respectively, then finetune those models using both versions of DTDB. Finetuning details are provided in Section B.2.5.

B.3.2.2 Results

To begin, consider the results on the YUVL dataset, shown in Table B.4. Here, it is seen that finetuning the pretrained models using either the dynamics or appearance organizations of DTDB improves the results of both C3D and MSOE-two-stream compared to the results in Table B.2. Notably, the boost in performance is especially significant for C3D. This can be largely attributed to the fact that C3D is pretrained on a large video dataset (*i.e.* Sports-1M), while in the original two-stream architecture only the RGB stream is pretrained on ImageNet and the motion stream is trained from scratch. Notably, MSOE-two-stream finetuned on DTDB-dynamics still outperforms C3D and either exceeds or is on-par with previous results on YUVL using SOE-Net.

		YUVL-1	YUVL-2	YUVL-3
State-of-the-art	SOE-Net [64]	**95.6**	91.7	**91.0**
Dynamics based finetuning	C3D	89.1	90.0	89.5
	MSOE-two-stream	91.1	**92.7**	90.0
Apperance based finetuning	C3D	88.8	87.4	85.4
	MSOE-two-stream	90.2	91.2	87.8

Table B.4: Performance of spatiotemporal ConvNets *finetuned* using both organizations of DTDB on the various breakdowns of the YUVL dataset [36].

		YUP++(S)	YUP++(M)	YUP++
State-of-the-art	T-ResNet [51]	92.4	81.5	89.0
Dynamics based finetuning	C3D	89.4	80.8	85.5
	MSOE-two-stream	95.9	84.5	90.4
Apperance based finetuning	C3D	90.0	82.7	86.3
	MSOE-two-stream	**97.0**	**87.0**	**91.8**

Table B.5: Performance of spatiotemporal ConvNets *finetuned* using both organizations of DTDB on the **S**tatic and **M**oving camera portions of YUP++ and the entire YUP++ [51].

Turning attention to results obtained on YUP++, summarized in Table B.5, further emphasizes the benefits of finetuning on the proper data. Similar to observations made on YUVL, the boost in performance is once again especially notable on C3D. Importantly, finetuning MSOE-two-stream on DTDB-appearance yields the overall best results and considerably outperforms previous state-of-the-art, which relied on a more complex architecture [51].

Interestingly, results of finetuning using either version of DTDB also outperform previously reported results using C3D or two-stream architectures, on both YUVL and YUP++, with sizable margins [64, 51]. Additional one-to-one comparisons are provided in Section B.3.3.

B.3.2.3 Conclusions

The experiments in this section further highlighted the added value of the proposed dual organization of DTDB in two ways. First, on YUVL, finetuning standard architectures led to a notable boost in performance, competitive with or exceeding previous results obtained using the proposed SOE-Net. Hence, an interesting way forward, would be to finetune SOE-Net on DTDB to further benefit this network from the availability of a large scale dynamic texture dataset. Second, on YUP++, it was shown that standard spatiotemporal architectures, trained on the right data, could yield new state-of-the-art results, even while compared to more complex architectures (*e.g.* T-ResNet [51]). Once again, the availability of a dataset like DTDB could allow for even greater improvements using more complex architectures provided with data adapted to the target application.

B.3.3 Further experiments with DTDB

This section provides additional one-to-one comparisons to previously reported results on both the YUVL and YUP++ datasets using the ConvNet architectures evaluated in this research. Additional experiments were also performed to compare finetuning using DTDB to finetuning on the similarly sized UCF101 dataset and results are reported here. Given that the results in Chapter 5 established the overall superiority of the MSOE stream over the flow stream for the tasks at hand, the following results concentrate on the MSOE-two-stream. These comparisons further highlight the benefit of DTDB's dual organizations as finetuning substrates.

Tables B.6 and B.7 of this appendix summarize all finetuning results obtained

on the variations of the YUVL and YUP++ datasets, respectively, and thereby complement Tables B.4 and B.5. In line with observations made in Section B.3.1, it is clear here that overall the C3D architecture benefited more from finetuning compared to the two-stream architecture and this can be attributed to reliance of C3D on a video dataset in pretraining. In addition, for the two-stream case, only one of the two streams (*i.e.* the RGB stream) benefited from a pretrained model, while an MSOE stream trained from scratch is used, as done in the original two-stream paper [162]. These results highlight the relevance of the proposed experimental procedure adopted in this work (*i.e.* comparisons using architectures trained from scratch), as it allowed for direct and fair comparisons of the evaluated architectures without being affected by such confounding variables.

More importantly, comparing finetuning results of YUVL versus YUP++, reveals a more notable gap in performance between C3D and MSOE-two-stream for the YUP++ dataset compared to YUVL. Comparison of the performance of the finetuned RGB streams on these two datasets sheds light on this situation. In particular, we see that finetuning the RGB stream was much more beneficial for the YUP++ dataset. Here it is important to remember the major difference between these two datasets. While in YUVL classes are determined by differences in dynamics, YUP++ on the other hand distinguishes classes mainly based on differences in appearance. Therefore, pretraining the RGB stream on ImageNet (*i.e.* an appearance based dataset), was much more beneficial when the target application was dominated by differences in appearance.

Notably, the results presented here also compare finetuning on DTDB versus

finetuning on UCF101. Congruent with transfer learning results reported in Section B.3.1, finetuning on either versions of DTDB was more beneficial than finetuning on UCF101. Once again, this further highlights the benefit of DTDB as a finetuning substrate when the target applications differ significantly from the training data (*e.g.* using an action recognition dataset for training when the target task is dynamic scene recognition).

Finally, comparison to previously reported results on both YUVL and YUP++, further emphasize the value of DTDB, that helped extend state-of-the-art.

		YUVL-1	YUVL-2	YUVL-3
Previously reported results [64]	C3D	88.0	89.8	85.5
	RGB Stream	-	-	-
	MSOE Stream	-	-	-
UCF101	C3D-scratch	61.4	65.4	55.7
	C3D-finetuned	87.0	88.4	85.2
	RGB Stream-scratch	63.6	72.8	60.0
	RGB Stream-finetuned	64.8	73.4	63.0
	MSOE Stream	80.0	80.2	74.4
	MSOE-two-stream-scratch	80.8	84.5	78.8
	MSOE-two-stream-finetuned	81.8	83.6	79.7
DTDB-Dynamics	C3D-scratch	83.3	86.4	83.4
	C3D-finetuned	89.1	90.0	89.5
	RGB Stream-scratch	68.1	75.4	65.0
	RGB Stream-finetuned	70.3	76.6	69.9
	MSOE Stream	89.2	89.3	84.8
	MSOE-two-stream-scratch	89.2	89.3	84.8
	MSOE-two-stream-finetuned	91.1	92.7	90.0
DTDB-Apperance	C3D-scratch	82.2	85.4	80.9
	C3D-finetuned	88.8	87.4	85.4
	RGB Stream-scratch	67.6	72.8	64.3
	RGB Stream-finetuned	68.6	75.2	68.1
	MSOE Stream	87.7	87.3	83.6
	MSOE-two-stream-scratch	89.8	90.2	86.7

Table B.6: Detailed performance comparison of spatiotemporal ConvNets, *finetuned* using UCF101 and both organizations of DTDB on the various breakdowns of the YUVL dataset [36].

		YUP++(S)	YUP++(M)	YUP++
Previously reported results [51]	C3D	85.4	76.3	84.0
	RGB Stream	84.3	68.1	82.0
	MSOE Stream	-	-	-
UCF101	C3D-scratch	62.5	55.8	58.3
	C3D-finetuned	86.1	78.0	82.3
	RGB Stream-scratch	64.9	54.4	63.5
	RGB Stream-finetuned	78.8	70.0	76.5
	MSOE Stream	74.3	52.7	62.0
	MSOE-two-stream-scratch	80.1	66.6	74.6
	MSOE-two-stream-finetuned	83.8	75.8	82.4
DTDB-Dynamics	C3D-scratch	84.3	71.8	76.5
	C3D-finetuned	89.4	80.8	85.5
	RGB Stream-scratch	81.8	73.7	78.3
	RGB Stream-finetuned	88.9	82.1	85.8
	MSOE Stream	90.0	67.5	78.4
	MSOE-two-stream-scratch	93.3	81.5	87.7
	MSOE-two-stream-finetuned	95.9	84.5	90.4
DTDB-Apperance	C3D-scratch	85.0	73.7	78.1
	C3D-finetuned	90.0	82.7	86.3
	RGB Stream-scratch	82.0	76.2	79.9
	RGB Stream-finetuned	90.6	83.8	88.1
	MSOE Stream	91.0	69.5	79.1
	MSOE-two-stream-scratch	94.7	83.2	89.6
	MSOE-two-stream-finetuned	97.0	87.0	91.8

Table B.7: Detailed performance comparison of spatiotemporal ConvNets, *finetuned* using UCF101 and both organizations of DTDB on the **S**tatic and **M**oving camera portions of YUP++ and the entire YUP++ [51].

B.4 Summary and Discussion

While Chapter 5 demonstrated the utility of DTDB as an analysis tool from an algorithm comparison perspective, this appendix further highlighted the shear benefits of the dataset itself.

From the dataset perspective, DTDB not only has supported experiments that tease apart appearance vs. dynamics, but also shown adequate size and diversity to support transfer learning to related tasks, thereby reaching or exceeding state-of-the-art even while using standard spatiotemporal ConvNets. Moving forward, DTDB

can be a valuable tool to further research on spacetime image analysis. For example, training additional state-of-the-art spatiotemporal ConvNets using DTDB can be used to further boost performance on both dynamic texture and scene recognition. Also, the complementarity between the two organizations can be further exploited for attribute-based dynamic scene and texture description. For example, the various categories proposed here can be used as attributes to provide more complete dynamic texture and scene descriptions beyond traditional categorical labels (*e.g.* pluming vs. boiling volcano or turbulent vs. wavy water flow). Finally, DTDB can be used to explore other related areas, including dynamic texture synthesis, dynamic scene segmentation as well as development of video-based recognition algorithms beyond ConvNets.

References

[1] Amazon Mechanical Turk. www.mturk.com. 93

[2] Beautiful word clouds. www.wordle.net. 91

[3] Pond5. www.pond5.com. 92

[4] VideoHive. www.videohive.net. 92

[5] YouTube. www.youtube.com. 92

[6] E. Adelson and J. Bergen. The plenoptic function and the elements of early vision. In M. Landy and J. A. Movshon, editors, *Computational Models of Visual Processing*, pages 3–20. MIT Press, Cambridge, 1991. 55

[7] P. Agrawal, R. Girshick, and J. Malik. Analyzing the performance of multilayer neural networks for object recognition. In *European Conference on Computer Vision (ECCV)*, 2014. 16

[8] X. Bai, J. Wang, D. Simons, and G. Sapiro. Video SnapCut: Robust video object cutout using localized classifiers. In *SIGGRAPH*, 2009. 110

[9] C. L. Baker and I. Mareschal. Processing of second-order stimuli in the visual cortex. *Progress in Brain Research*, 134:171–91, 2001. 20, 21, 24, 54

[10] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 9, 16, 44, 144

[11] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(8):1798–1828, 2013. 7

[12] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Conference on Neural Information Processing Systems (NIPS)*, 2007. 1, 8

[13] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. 8

[14] A. Borji, M.M. Cheng, H. Jiang, and J. Li. Salient object detection: A benchmark. *IEEE Transactions on Image Processing (TIP)*, 24:5706–5722, 2015. 118, 124

[15] Y. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in visual recognition. In *International Conference on Machine Learning (ICML)*, 2010. 40

[16] R. M. Boynton and D. N. Whitten. Visual adaptation in monkey cones: Recordings of late receptor potentials. *Science*, 170(3965):1423–1426, 1970. 31

[17] G. J. Brouwer and D. J. Heeger. Cross-orientation supression in human visual cortex. *Journal of Neurophysiology*, (106):2108–2119, 2011. 31, 32, 37

[18] J. Bruna and S. Mallat. Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(8):1872–1886, 2013. 18, 23, 41, 45, 49, 72, 73, 145

[19] A. E. Bryson, W. F Denham, and S. E. Dreyfus. Optimal programming problems with inequality constraints. *American Institute of Aeronautics and Astronautics (AIAA)*, 11:2544–2550, 1963. 10

[20] P. J. Burt. Smart sensing within a pyramid vision machine. *Proceedings of the IEEE*, 76(8):1006–1015, 1988. 39

[21] M. Carandini. What simple and complex cells compute. *The Journal of Physiology*, 577:463–466, 2006. 38

[22] M. Carandini and D. J. Heeger. Normalization as a canonical neural computation. *Nature reviews. Neuroscience*, 13:51–62, 2011. 31, 37

[23] J. Carreira and A. Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 84, 155

[24] T. H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma. PCANet: A simple deep learning baseline for image classification? *IEEE Transactions on Image Processing (TIP)*, 24:5017–5032, 2015. 49

[25] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference (BMVC)*, 2014. 2, 15, 16, 45

[26] D. Chetverikov and R. Peteri. A brief survey of dynamic texture description and recognition. In *Conference on Computer Recognition Systems (CORES)*, 2005. 97

[27] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 66, 83, 144

[28] M. Cimpoi, S. Maji, and A. Vedaldi. Deep filter banks for texture recognition and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 66

[29] D-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units. In *International Conference on Learning Representations (ICLR)*, 2016. 30

[30] T. S. Cohen and M. Welling. Steerable CNNs. In *International Conference on Learning Representations (ICLR)*, 2017. 18, 22, 23, 145

[31] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. 7

[32] D. Dai, H. Riemenschneider, and L.V Gool. The synthesizability of texture examples. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 83

[33] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 15

[34] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 8, 39

[35] P. Dayan and L. F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press, 2005. 26

[36] K. Derpanis and R. P. Wildes. Spacetime texture representation and recognition based on spatiotemporal orientation analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34:1193–1205, 2012. 3, 55, 59, 67, 69, 70, 71, 73, 74, 83, 84, 88, 90, 94, 97, 115, 116, 178, 185, 187, 191, 193, 194, 197, 200

[37] K. G. Derpanis and R. P. Wildes. Dynamic texture recognition based on distributions of spacetime oriented structure. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 191–198, June 2010. 84

[38] S. DiZenzo. A note on the gradient of a multi-image. *Computer Vision, Graphics, and Image Processing (CVGIP)*, 33:116–125, 1986. 62

[39] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 39(4):677–691, 2017. 11

[40] G. Doretto, A. Chiuso, Y. Wu, and S. Soatto. Dynamic textures. *International Journal of Computer Vision (IJCV)*, 51:91–109, 2003. 97

[41] S. Dubois, R. Peteri, and M. Michel. Characterization and recognition of dynamic textures based on the 2D+T curvelet. *Signal Image & Video Processing*, 9:819–830, 2013. 67, 71, 79, 80, 84

[42] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common muli-scale convolutional architecture. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 2, 48

[43] I. Endres and D. Hoiem. Category independent object proposals. In *European Conference on Computer Vision (ECCV)*, 2010. 128, 131

[44] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. Technical Report 1341, University of Montreal, 2009. 14

[45] A. Faktor and M. Irani. Video segmentation by non-local consensus voting. In *British Machine Vision Conference (BMVC)*, 2014. 111, 132, 133, 139, 140

[46] Q. Fan, F. Zhong, D. Lischinski, D. Cohen-Or, and B. Chen. JumpCut: Non-successive mask transfer and interpolation for video cutout. *ACM Transactions on Graphics*, 34(6):195:1–195:10, 2015. 110

[47] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Bags of spacetime energies for dynamic scene recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 7

[48] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Dynamically encoded actions based on spacetime saliency. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 7, 118

[49] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal residual networks for video action recognition. In *Conference on Neural Information Processing Systems (NIPS)*, 2016. 86, 95

[50] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal multiplier networks for video action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 15, 148

[51] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Temporal residual networks for dynamic scene recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 191, 194, 195, 197, 198, 201

[52] C. Feichtenhofer and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 11

[53] J. Feng and T. Darrell. Learning the structure of deep convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 49

[54] S. Fidler, G. Berginc, and A. Leonardis. Hierarchical statistical learning of generic parts of object structure. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. 21, 22

[55] S. Fidler, M. Boben, and A. Leonardis. Similarity-based cross-layered hierarchical representation for object categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 21

[56] S. Fidler and A. Leonardis. Towards scalable representations of object categories: Learning a hierarchy of parts. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 21, 48

[57] J. Freeman, C. M. Ziemba, D. J. Heeger, E. P. Simoncelli, and A. J. Movshon. A functional and perceptual signature of the second visual area in primates. *Nature Neurosience*, 16:974–981, 2013. 20, 21

[58] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 13:891–906, 1991. 56

[59] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition. *Biological Cybernetics*, 36:193–202, 1980. 2, 8, 9, 38, 39, 40

[60] B. Ghanem and A. Narendra. Max margin distance learning for dynamic texture. In *European Conference on Computer Vision (ECCV)*, 2010. 67, 71, 79, 80, 84

[61] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 144

[62] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011. 28

[63] I. J. Goodfelow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *Conference on Neural Information Processing Systems (NIPS)*, 2013. 41

[64] I. Hadji and R. P. Wildes. A spatiotemporal oriented energy network for dynamic texture recognition. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 18, 33, 52, 66, 84, 85, 101, 190, 193, 197, 200

[65] I. Hadji and R. P. Wildes. A new large scale dynamic texture dataset with application to convnet understanding. In *European Conference on Computer Vision (ECCV)*, 2018. 84, 88

[66] I. Hadji and R. P. Wildes. What do we understand about convolutional networks? *arXiv*, 1803.08834, 2018. 6

[67] M. Harandi, C. Sanderson, C. Shen, and B. Lovell. Dictionary learning and sparse coding on Grassmann manifolds: An extrinsic solution. In *IEEE International Conference on Computer Vision (ICCV)*, 2013. 67, 79, 80

[68] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 112

[69] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision (ECCV)*, 2014. 42, 43

[70] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 29, 30

[71] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2, 10, 84, 148

[72] D. J. Heeger. Nonlinear model of neural responses in cat visual cortex. In M. Landy and J.A. Movshon, editors, *Computational Models of Visual Processing*, chapter 9, pages 119–134. MIT Press, Cambridge, 1991. 26, 27, 30, 31, 32, 37, 57, 59

[73] D. J. Heeger. Normalization of cell responses in cat striate cortex. *Visual Neuroscience*, 9(2):181–197, 1992. 27, 31, 32, 37, 38

[74] G. E. Hinton, S. Osindero, and Y-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006. 1, 8

[75] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 8, 11

[76] T. Hong, N. Kingsbury, and M. D. Furman. Biologically-inspired object recognition system with features from complex wavelets. In *IEEE International Conference on Image Processing (ICIP)*, 2011. 48

[77] Y.T. Hu, J.B. Huang, and A.G. Schwing. Unsupervised Video Object Segmentation using Motion Saliency-Guided Spatio-Temporal Propagation. In *European Conference on Computer Vision (ECCV)*, 2018. 111, 132, 139, 140

[78] G. Huang, Z. Liu, L. Van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 10

[79] Y. Huang, K. Huang, L. Wang, D. Tao, T. Tan, and X. Li. Enhanced biologically inspired model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 48, 49

[80] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160:106–154, 1962. 19, 24, 26, 31, 38

[81] J. H. Jacobsen, J. V. Gemert, Z. Lou, and A. W.M. Smeulders. Structured receptive fields in CNNs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 18, 22, 45, 49, 145

[82] S. Jain, B. Xiong, and K. Grauman. Fusionseg: Learning to combine motion and appearance for fully automatic segmentation of generic objects in videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 112, 139, 140

[83] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *IEEE International Conference on Computer Vision (ICCV)*, 2009. 2, 9, 16, 27, 33, 35, 40, 45, 75

[84] Y. Jeon and J. Kim. Active convolution: Learning the shape of convolution for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 15, 45

[85] H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A biologically inspired system for action recognition. In *IEEE International Conference on Computer Vision (ICCV)*, 2007. 20, 22, 38, 48

[86] H. Ji, X. Yang, H. Ling, and Y. Xu. Wavelet domain multifractal analysis for static and dynamic texture classification. *Transactions on Image Processing (TIP)*, 22:286–299, 2013. 67, 79, 80

[87] S. Ji, W. Xu, M. Yang, and K. Yu. 3D convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35:1915–1929, 2013. 95

[88] X. Jin, C. Xu, J. Feng, Y. Wei, J. Xiong, and S. Yan. Deep learning with S-shaped rectified linear activation units. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2016. 30

[89] R. Karim, M. A. Islam, and N. D. B. Bruce. Distributed iterative gating networks for semantic segmentation. In *Winter Conference On Applications of Computer Vision (WACV)*, 2019. 169

[90] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2, 11, 84, 95, 196

[91] J. Kim, D. Linsley, K., and T. Serre. Disentangling neural mechanisms for perceptual grouping. *arXiv*, 1906.01558, 2019. 39

[92] J. Koenderink and A. J. Van Doorn. The structure of locally orderless images. *International Journal of Computer Vision (IJCV)*, 31:159–168, 1999. 39

[93] Y.J. Koh and C.S. Kim. Primary object segmentation in videos based on region augmentation and reduction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 111

[94] I. Koprinska and S. Carrato. Temporal video segmentation: A survey. *Signal Processing: Image Communication*, 16(5):477 – 500, 2001. 110

[95] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Conference on Neural Information Processing Systems (NIPS)*, 2012. 1, 2, 10, 28, 33, 42, 48, 84, 148

[96] V.A. Lamme and P. R. Roelfsema. The distinct modes of vision offered by feedforward and recurrent processing. *Trends in Neuroscience*, 23(11):571–579, 2000. 169

[97] M. Langer and R. Mann. Optical snow. *International Journal of Computer Vision (IJCV)*, 55:55–71, 2003. 89

[98] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. 8, 39

[99] Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. 7

[100] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of IEEE*, 86:2278–2324, 1998. 1, 2, 8, 9, 27, 33, 35, 38, 39, 40, 49

[101] Y. LeCun, K. Kavukcuoglu, and C. Farabet. Convolutional networks and applications in vision. In *International Symposium on Circuits and Systems (ISCAS)*, 2010. 9

[102] Y.J. Lee, J. Kim, and K. Grauman. Key-segments for video object segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2011. 111, 133, 139, 140

[103] F. Li, T. Kim, A. Humayun, D. Tsai, and J.M. Rehg. Video segmentation by tracking many figure-ground segments. In *IEEE International Conference on Computer Vision (ICCV)*, 2013. 133

[104] X. Li and C.C. Loy. Video object segmentation with joint re-identification and attention-aware mask propagation. In *European Conference on Computer Vision (ECCV)*, 2018. 112

[105] M. Lin, Q. Chen, and S. Yan. Network in network. In *International Conference on Learning Representations (ICLR)*, 2014. 41, 42, 63, 144, 148, 159

[106] T-Y. Lin and S. Maji. Visualizing and understanding deep texture representations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 66

[107] D. Linsley, J. Kim, and T. Serre. Sample-efficient image segmentation through recurrence. *arXiv*, 1811.11356, 2018. 39

[108] F. Liu, C. Shen, and G. Lin. Deep convolutional neural fields for depth estimation fom a single image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[109] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 112

[110] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004. 1, 8, 39

[111] Z. Lu, W. Xie, J. Pei, and J. Huang. Dynamic texture recognition by spatio-temporal multiresolution histograms. In *Winter Conference On Applications of Computer Vision (WACV)*, 2005. 67

[112] S. Luan, b. Zhang, C. Chen, X. Cao, J. Han, and J. Liu. Gabor convolutional networks. *arXiv*, 1705.01450, 2017. 18, 22, 23, 145

[113] S. Lyu. Divisive normalization: Justification and effectiveness as efficient coding transform. In *Conference on Neural Information Processing Systems (NIPS)*, 2010. 32

[114] S. Lyu and E. P. Simoncelli. Nonlinear image representation using divisive normalization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 32, 37, 59

[115] T. Ma and L. J. Latecki. Maximum weight cliques with mutex constraints for video object segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 111

[116] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning (ICML)*, 2013. 28, 29, 30

[117] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2, 14, 15

[118] R. Margolin, A. Tal, and L. Zelnik-Manor. What Makes a Patch Distinct? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 118, 124

[119] D. Mély, D. Linsley, and T. Serre. Complementary surrounds explain diverse contextual phenomena across visual modalities. *Psychological Review*, 125(5):769–784, 2016. 39

[120] P. Mishra and B. K. Jenkins. Hierarchical model for object recognition based on natural-stimuli adapted filters. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2010. 49

[121] A. J. Movshon and E. P. Simoncelli. Representation of naturalistic image structure in the primate visual cortex. *Cold Spring Harbor Symposia on Quantitative Biology*, 79:115–122, 2014. 21

[122] A. J. Movshon, I. D. Thompson, and D. J. Tolhurst. Understanding locally competitive networks. *The Journal of Physiology*, 283:53–77, 1978. 26, 38

[123] A. Mumtaz, E. Coviello, G. Lanckriet, and A. B. Chan. Clustering dynamic textures with the hierarchical EM algorithm for modeling video. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35:1606–1621, 2013. 67, 79, 80

[124] A. Mumtaz, E. Coviello, G. Lanckriet, and A. B. Chan. A scalable and accurate descriptor for dynamic textures using bag of system trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 37:697–712, 2015. 67

[125] J. Mutch and D. G. Lowe. Multiclass object recognition with sparse, localized features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. 38, 39, 48

[126] N. Märki, F. Perazzi, O. Wang, and A. Sorkine-Hornung. Bilateral space video segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 110

[127] V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *International Conference on Machine Learning (ICML)*, 2010. 27

[128] J. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 11, 95

[129] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space.

In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 14, 15

[130] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Conference on Neural Information Processing Systems (NIPS)*, 2016. 14, 15

[131] G. Oxholm, P. Bariya, and K. Nishino. The scale of geometric texture. In *European Conference on Computer Vision (ECCV)*, 2012. 83

[132] E. Oyallon, E. Belilovsky, and S. Zagoruyko. Scaling the scattering transform: Deep hybrid networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 145, 152

[133] E. Oyallon and S. Mallat. Deep roto-translation scattering for object classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 18, 23, 41, 45

[134] E. Oyallon, S. Zagoruyko, G. Huang, N. Komodakis, S. Lacoste-Julien, M. Blaschko, and E. Belilovsky. Scattering networks for hybrid representation learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 41(9):2208–2221, 2019. 145, 152

[135] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *IEEE International Conference on Computer Vision (ICCV)*, 2013. 111, 133, 139, 140

[136] F. Perazzi, O. Wang, M. Gross, and A. Sorkine-Hornung. Fully connected object proposals for video segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 110

[137] F. Perronnin and D. Larlus. Fisher vectors meet neural networks: A hybrid classification architecture. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 145

[138] F. Perronnin, J. Sanchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *European Conference on Computer Vision (ECCV)*, 2010. 7

[139] R. Peteri and D. Chetverikov. Dynamic texture recognition using normal flow and texture regularity. In *Iberian Conference on Pattern Recognition and Image Analysis (ibPRIA)*, 2005. 67

[140] R. Peteri, F. Sandor, and M. Huiskes. DynTex: A comprehensive database of dynamic textures. *Pattern Recognition Letters (PRL)*, 31:1627–1632, 2010. 70, 83, 84, 94

[141] Y. Quan, C. Bao, and H. Ji. Equiangular kernel dicitionary learning with applications to dynamic textures analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 66

[142] Y. Quan, Y. Huang, and H. Ji. Dynamic texture recognition via orthogonal tensor dictionary learning. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 66, 67, 71, 79, 80, 193

[143] A. Ravichandran, R. Chaudhry, and Rene R. Vidal. View-invariant dynamic texture recognition using a bag of dynamical systems. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 67, 84

[144] J. Ren, X. Jiang, and J. Yuan. Dynamic texture recognition using enhanced LBP features. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013. 67

[145] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2:1019–1025, 1999. 20, 22, 38, 48

[146] A. Rodriguez-Sanchez, M. Fallah, and A. Leonardis. Hierarchical object representation in the visual cortex and computer vision. *Frontiers in Computational Neuroscience*, 9:142–144, 2015. 1

[147] A. J. Rodriguez-Sanchez and J. K. Tsotsos. The importance of intermediate representations for the modeling of 2D shape detection: Endstopping and curvature tuned computations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 20

[148] A. J. Rodriguez-Sanchez and J. K. Tsotsos. The roles of endstopped and curvature tuned computations in a hierarchical representation of 2D shape. *PLOS ONE*, 7(8):1–13, 08 2012. 20

[149] Y. Rubner and C. Tomasi. Coallescing texture descriptors. In *ARPA Image Understanding Workshop*, 1996. 62

[150] D. E. Rumelhart, G .E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. 9

[151] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 10, 84, 144, 196

[152] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. In *Conference on Neural Information Processing Systems (NIPS)*, 2017. 43

[153] P. Saisan, G. Doretto, Y. Wu, and S. Soatto. Dynamic texture recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001. 67, 71, 83, 84, 94

[154] G. Sapiro and D. Ringach. Anisotropic diffusion of multivalued images with applications to color filtering. *IEEE Transactions on Image Processing (TIP)*, 5:1582–1586, 1996. 62

[155] D. Scherer, A. Müller, and S. Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *International Conference on Artificial Neural Networks (ICANN)*, 2010. 40

[156] I. Sergey and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015. 35, 37

[157] T. Serre, M. Kouh, C. Cadieu, U. Knoblich, G. Kreiman, and T. Poggio. A theory of object recognition: Computations and circuits in the feedforward path of the ventral stream in primate visual cortex. Technical Report 2005-036, Massachusetts Institute of Technology, 2005. 20, 38

[158] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29:411–426, 2007. 20, 22, 38

[159] W. Shang, K. Sohn, D. Almeida, and H. Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *International Conference on Machine Learning (ICML)*, 2016. 29, 30, 45, 57, 144, 145, 156

[160] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond skip connections: Top-down modulation for object detection. *arXiv*, 1612.06851, 2016. 169

[161] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *International Conference on Learning Representations (ICLR)*, 2014. 2, 14, 15

[162] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Conference on Neural Information Processing Systems (NIPS)*, 2014. 1, 3, 11, 15, 86, 95, 96, 99, 101, 144, 148, 186, 187, 188, 189, 190, 199

[163] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 1, 10, 16, 84, 148

[164] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. Technical Report CRCV-TR-12-01, University of Central Florida, 2012. 84, 85, 195

[165] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *International Conference on Learning Representations Workshops (ICLRW)*, 2015. 144

[166] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 10, 148, 159

[167] R. Szeliski. *Computer vision algorithms and applications*. Springer, London; New York, 2011. 19

[168] M. Szummer and R. W. Picard. Temporal texture modeling. In *IEEE International Conference on Image Processing (ICIP)*, 1996. 66

[169] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 1, 2, 11, 48, 76, 77, 85, 95, 96, 99, 101, 148, 155, 185, 192

[170] L. Uhr. Layered "recognition cone" networks that preprocess, classify, and describe. *IEEE Transactions on Computers*, C-21(7):758–768, 1972. 39

[171] D. Varas and F. Marques. Region-based particle filter for video object segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 111

[172] M. Varma and A. Zisserman. Texture classification: Are filter banks necessary? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003. 66

[173] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision (IJCV)*, 62:61–81, 2005. 66

[174] P. Vincent, H. Larochelle, Y. Bengio, and P-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning (ICML)*, 2008. 1, 8

[175] W. Wang, J. Shen, and F. Porikli. Saliency-aware geodesic video object segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 111

[176] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for very deep two-stream convnets. In *European Conference on Computer Vision (ECCV)*, 2016. 15, 159

[177] T. Wang, B. Han, and J.P. Collomosse. TouchCut: Fast image and video segmentation using single-touch interaction. *Computer Vision and Image Understanding (CVIU)*, 120(6):14–30, 2014. 110

[178] H. Wei and Z. Dong. V4 neural network model for visual saliency and discriminative local representation of shapes. In *International Joint Conference on Neural Networks (IJCNN)*, 2014. 48

[179] P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974. 10

[180] R. P. Wildes and J. R. Bergen. Qualitative spatiotemporal analysis using an oriented energy representation. In *European Conference on Computer Vision (ECCV)*, 2000. 52, 55, 57, 59

[181] K. Woodbeck, G. Roth, and H. Chen. Visual cortex on the GPU: Biologically inspired classifier and feature descriptor for rapid recognition. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2008. 48, 49

[182] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow. Harmonic networks: Deep translation and rotation equivariance. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 18, 22, 23, 145

[183] J. Xiao, K. A. Ehinger, J. Hays, A. Torralba, and A. Oliva. SUN Database: Exploring a large collection of scene categories. *International Journal of Computer Vision (IJCV)*, 119(1):3–22, 2014. 128, 169

[184] Y. Xu, S. Huang, H. Ji, and C. Fermuller. Scale-space texture description on sift-like textons. *Computer Vision and Image Understanding (CVIU)*, 116:999 – 1013, 2012. 67, 79, 80

[185] Y. Xu, Y. Quan, H. Ling, and H. Ji. Dynamic texture classification from fractal analysis. In *IEEE International Conference on Computer Vision (ICCV)*, 2011. 67, 79, 80

[186] F. Yang, G. Xia, G. Liu, L. Zhang, and X. Huang. Dynamic texture recognition by aggregating spatial and temporal features via SVMs. *Neurocomputing*, 173:1310 – 1321, 2016. 66, 67

[187] Y. Yang, G. Sundaramoorthi, and S. Soatto. Self-occlusions and disocclusions in causal video object segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 111

[188] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Conference on Neural Information Processing Systems (NIPS)*, 2014. 144, 147

[189] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. In *International Conference on Machine Learning Workshops (ICMLW)*, 2015. 12, 14

[190] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L1 optical flow. In *DAGM Symposium on Pattern Recognition*, 2007. 186

[191] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, 2014. 2, 9, 12, 13, 15, 144

[192] J. Zhang, Y. Barhomi, and T. Serre. A new biologically inspired color image descriptor. In *European Conference on Computer Vision (ECCV)*, 2012. 20

[193] G. Zhao and M. Pietikainen. Dynamic texture recognition using volume local binary patterns. In *European Conference on Computer Vision (ECCV)*, 2006. 71, 84

[194] G. Zhao and M. Pietikainen. Dynamic texture recognition using local binary patterns with an application to facial expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29:915–928, 2007. 67, 79, 80

[195] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. In *International Conference on Learning Representations (ICLR)*, 2014. 9, 13

[196] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 13

[197] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Conference on Neural Information Processing Systems (NIPS)*, 2014. 84

[198] Y. Zhou, Q. Ye, Q. Qiu, and J. Jiao. Oriented response networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 18, 22, 23, 145

[199] C. Zzegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In *Conference on Neural Information Processing Systems (NIPS)*, 2013. 2, 48