DISENTANGLING VISUAL CONCEPTS ACROSS SPACE AND TIME: FROM IMAGE HIERARCHIES TO VIDEO DYNAMICS

Matthew Kowal

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, Canada

August 2025

Abstract

This dissertation advances the interpretability of deep vision models, with a particular focus on disentangling representations across space, layers, and time. As deep learning systems increasingly underpin critical applications, understanding their internal representations and decision-making processes is essential. The dissertation is structured into three parts that address this challenge from complementary perspectives. The first part introduces a framework for quantifying static and dynamic information in spatiotemporal models, offering a principled measure of how such models encode temporal dependencies compared with static counterparts. The second part presents a novel methodology for discovering and localizing semantically meaningful concepts within these spatiotemporal models, facilitating a deeper understanding of the internal features that drive predictions. The third part extends this analysis by identifying interlayer concept circuits, i.e., structured pathways through which concepts propagate across layers, revealing how information flows and transforms within deep image models. Together, these contributions provide a toolkit for interpreting complex neural architectures and lay the groundwork for more transparent and accountable artificial intelligence systems in dynamic visual domains. Overall, this dissertation provides new tools and insights for understanding the multilayered and spatiotemporal characteristics of deep vision models.

Acknowledgements

Completing a PhD has been one of the great privileges of my life that was not only possible, but *enjoyable*, due to the numerous amazing people I had supporting me throughout the journey.

Thank you to my supervisor, Kosta, for taking a chance on me as a fresh undergraduate, who didn't know how to code. I promised him I would be here for the long run and would make him proud (and I hope that I have). You would smile at how frequently I repeat quotes and pieces of advice I have collected from you over the years to pass it down to my own mentees (I promise I will forever be one with the pixels). Having a great supervisor is by far the most important part of staying motivated during ones PhD for so many reasons. So thank you Kosta for teaching me to care about the details, how to anticipate fruitful research directions, and for your effort in creating an amazing lab environment which feels like family. Even when we switched labs partway through my PhD, the culture came too, which is how I know the causal factor is you.

To my lab: Thank you for making coming into the lab so much fun throughout my graduate school years. From roasting our own coffee beans, dancing to techno in the early hours of the morning, Frisbee in the summer evenings, seasonal hikes through Ontario forests, or smashing beignets and listening to jazz music on the streets of New Orleans–you have left me with memories that will last a lifetime and, many of you have become lifelong friends, so I am sure we will make many more.

I also wish to give a special thanks to Richard Wildes. It is a rare treat to work with your grand-supervisor and those projects with you and Kosta were some that I will cherish. Apart from teaching me about the importance of mathematical rigor in computer science, how to create bold research ideas, and esoteric grammar rules, you constantly provided me with clear and thoughtful career advice to a junior researcher beginning their research journey. I hope you are enjoying your retirement and don't get too close to the polar bears!

Of course, I had tremendous support outside of school as well. Thank you to my Brother and Sister (a.k.a The Poops) for being a constant source of level-headed advice and reminding me why we are doing these crazy degrees in the first place. You both inspire me to be the best version of myself and taught me to prioritize my physical and mental health first, before anything else. And I really appreciate you going through a PhD first, Emma, to show me how it's done.

Thank you Mom and Dad for your unconditional support throughout my life and providing me the opportunity to follow my career passion for many years, which seems to be increasingly rare these days. I am forever grateful that we are not only a close family, but fantastic roommates, which not something every child can say about their parents. I'll cherish those morning walks, lunch-time chats, and nightly dinners for the years I was living with you at home during grad school. And an extra thank you for letting Kali, Plato and I take over your house during the pandemic.

Finally, to my wife, Kali (and our dog, Plato). Finding and marrying you was the best thing that could have happened to me, both in and out of school. My GPA immediately jumped a full grade point when I met you during undergrad. You then encouraged my decision to leave my full-time job and switch into the field of AI. You supported me throughout my PhD, offering technical support which helped generate successes (I am a much better writer now, thank you) and emotional support through the tough rejections (there were many). This is your dissertation as much as mine and I could not have done it without you. I love you and I cannot wait to spend our lives together.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Researchers have spent decades handcrafting methods for extracting features from visual data for the purposes of using these features for downstream tasks. Much of this time was spent trying to figure out how to take an intuitive visual abstraction (e.g., motion, edges, shape, etc.) and implement ways to quantify the existence of this abstraction in data. Feature engineering for computer vision is an extremely difficult problem and, unsurprisingly, it was not uncommon for a single feature extractor to be the outcome of an entire graduate degree (e.g., [165]).

Artificial Intelligence (AI), and deep learning in particular, has recently and fundamentally changed how scientists approach computer vision. Instead of handcrafting methods, the deep learning paradigm works as follows: (i) define a hypothesis space of possible solutions in the form of different functions (e.g., convolutional filters, multi-layer perceptron (MLP) connections, attention layers, etc), (ii) define a cost function between the model prediction and some target value that is differentiable with respect to the model parameters, (iii) and optimize the parameters, via gradient descent, such that the cost is minimized. While such a paradigm relieves the user of defining the parameters a priori required to perform the task well (i.e., these are learned by the model), the user must still define the hypothesis space (not to mention the cost function and optimization algorithm) a priori. Far and away the most effective hypothesis spaces take the form of deep neural networks (DNNs), that stack layer upon layer to learn hierarchical features of the data.

The resulting DNNs used in most computer vision applications have dozens to hundreds of layers, and millions or even billions of parameters. It is not surprising that the actual computations performed by such a model are non-trivial to understand. Although it is easy to examine a single neuron or layer to summarize the absolute feature response, quantifying or visualizing responses and simultaneously constraining them to be interpretable by humans remains an open research problem.

Towards a solution to this problem, the field of explainable AI (XAI) integrates mathematical and algorithmic techniques with DNNs to produce interpretable explanations of a model's logic. *Post-hoc interpretability can be defined as methods that provides a human with information about how a pretrained model computes information.* This computation could be identifying the impact of input features on the output (e.g., feature attribution methods), quantifying what concepts are encoded by the model internally (e.g., concept-based interpretability), allow the human to better predict the model output, or help understand model training dynamics (e.g., loss curves).

Concept-based interpretability is a leading paradigm for understanding the latent representations of deep vision models; however, the application of these methods has largely been confined to static images. It remains unclear whether state-of-the-art video architectures, such as video transformers, truly capture and exploit temporal dynamics, or if they primarily rely on static cues, as some studies suggest. Even if these models do encode meaningful spatiotemporal concepts, current concept-based frameworks do not readily extend to uncovering such representations in video data. Furthermore, the notion of "concepts" itself warrants deeper scrutiny: concepts do not materialize in isolation within a single layer, but rather emerge from complex compositions of activations across the network's depth. Without accounting for this compositional hierarchy, any attempt to explain a model's computations risks oversimplification. At the time of writing, there is no robust method to systematically extract a hierarchical concept graph from state-of-the-art vision architectures. These gaps in concept-based interpretability—both in terms of temporal reasoning and hierarchical structure—motivate the investigations presented in this dissertation.

1.2 Contributions

In this dissertation we investigate post-hoc interpretability methods. This exploration results in three main contributory studies across two under-explored axes, time, and layers. For each study, we have made multiple novel technical contributions and improvements compared to existing methods.

1. While evidence suggests that action recognition algorithms are heavily influenced by visual appearance in single frames, no quantitative methodology exists for evaluating such static bias in the latent representation compared to bias toward dynamics. This study [2, 147] tackles this challenge by proposing an approach for quantifying the static and dynamic biases of any spatiotemporal model, and applies the approach to three tasks, action recognition, automatic video object segmentation (AVOS) and video instance segmentation (VIS). Our key findings are: (i) Most examined models are biased toward static information. (ii) Some datasets that are assumed to be biased toward dynamics are actually biased toward static information. (iii) Individual channels in an architecture can be biased toward static, dynamic or a combination of the two. (iv) Most models converge to their culminating biases in the first half of training.

We then explore how these biases affect performance on dynamically biased datasets. For action recognition, we propose StaticDropout, a semantically guided dropout that debiases a model from static information toward dynamics. For AVOS, we design a better combination of fusion and cross connection layers compared with previous architectures. Our code is publicly available at `https://yorkucvil.github.io/Static-Dynamic-Interpretability/`

2. The second study focuses on the problem of concept-based interpretability of transformer representations for videos [146]. Concretely, it seeks to explain the decision-making process of video transformers based on high-level, spatiotemporal concepts that are automatically discovered. Prior research on concept-based interpretability has concentrated solely on image-level tasks. Comparatively, video models deal with the added temporal dimension, increasing complexity and posing challenges in identifying dynamic concepts over time. In this work, we systematically address these challenges by introducing the first Video Transformer Concept Discovery (VTCD) algorithm. To this end, we propose an efficient approach for unsupervised identification of units of video transformer representations - concepts, and ranking their importance to the output of a model. The resulting concepts are highly interpretable, revealing spatio-temporal reasoning mechanisms and object-centric representations in unstructured video models. Performing this analysis jointly over a diverse set of supervised and self-supervised representations, we discover that some of these mechanism are universal in video transformers. Finally, we demonstrate that VTCD can be used to improve model performance for fine-grained tasks. Our code is publicly available at `https://yorkucvil.github.io/VTCD/`

3. The final study presents a new methodology for post-hoc interpretability for vision models, the Visual Concept Connectome (VCC), which discovers human interpretable concepts and their interlayer connections in a fully unsupervised manner [149]. Our approach simultaneously reveals fine-grained concepts at a layer, connection weightings across all layers and is amendable to global analysis of network structure (e.g., branching pattern of hierarchical concept assemblies). Previous work yielded ways to extract interpretable concepts from single layers and examine their impact on classification, but did not afford multilayer concept analysis across an entire network architecture. Quantitative and qualitative empirical results show the effectiveness of VCCs in the domain of image classification. Also, we leverage VCCs for the application of failure mode debugging to reveal where mistakes arise in deep networks. Our code and demo are publicly available at `https://yorkucvil.github.io/VCC/`.

1.3 Outline

We organize the content of this dissertation in the following fashion.

- This first chapter has served to place post-hoc interpretability in the context of deep learning as well as to introduce our major contributions.
- Chapter 2 provides a background and survey of interpretability methods for computer vision models, contrasting *pixel-based* and *concept-based* interpretability methodologies.
- Chapter 3 presents an analysis of static and dynamic information in deep neural networks trained for video understanding tasks.
- Chapter 4 presents spatiotemporal concept discovery in video transformers.
- Chapter 5 presents multilayer concept discovery and decomposition.

In each chapter, we introduce motivations and problem definition, review extant approaches (including discussions of pros and cons), introduce our novel contributions and finally show results of empirical evaluation. In the final chapter, we summarize our technical findings, societal implications and provide outlines of several future directions.

List of Publications

* denotes equal contributions. †denotes publications presented in this dissertation.

- *Title*: Self-supervised compositional feature representation for video understanding†
  Authors: **Matthew Kowal**, Pavel Tokmakov, Rares Ambrus, Adrien Gaidon
  Venue: US Patent App. 18/807,734, 2025

- *Title*: Archetypal SAE: Adaptive and Stable Dictionary Learning for Concept Extraction in Large Vision Models
  Authors: Thomas Fel*, Ekdeep Singh Lubana*, Jacob S. Prince, **Matthew Kowal**, Victor Boutin, Isabel Papadimitriou, Binxu Wang, Martin Wattenberg, Demba Ba, Talia Konkle
  Venue: International Conference on Machine Learning, 2025

- *Title*: Universal Sparse Autoencoders: Interpretable Cross-Model Concept Alignment
  Authors: Harrish Thasarathan, Julian Forsyth, Thomas Fel, **Matthew Kowal**, Konstantinos G. Derpanis
  Venue: International Conference on Machine Learning, 2025

- *Title*: Quantifying and Learning Static vs. Dynamic Information in Deep Spatiotemporal Networks†
  Authors: **Matthew Kowal**, Mennatullah Siam, Md Amirul Islam, Neil Bruce, Richard P. Wildes, Konstantinos G. Derpanis
  Venue: Transactions on Pattern Analysis and Machine Intelligence, 2024

- *Title*: Visual Concept Connectome (VCC): Open World Concept Discovery and their Interlayer Connections in Deep Models†
  Authors: **Matthew Kowal**, Richard P. Wildes, Konstantinos G. Derpanis
  Venue: (Spotlight) Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2024

- *Title*: Multi-modal News Understanding with Professionally Labelled Videos (ReutersViLNews)
  Authors: Shih-Han Chou*, **Matthew Kowal***, Yasmin Niknam*, Diana Moyano, Shayaan Mehdi, Richard Pito, Cheng Zhang, Ian Knopke, Sedef Akinli Kocak, Leonid Sigal, Yalda Mohsenzadeh
  Venue: Canadian Artificial Intelligence Conference, 2024

- *Title*: Understanding Video Transformers via Universal Concept Discovery†
  Authors: **Matthew Kowal**, Achal Dave, Rares Ambrus, Adrien Gaidon, Konstantinos G. Derpanis, Pavel Tokmakov

Venue: (Spotlight) Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2024

- *Title*: A deeper dive into what deep spatiotemporal networks encode: Quantifying static vs. dynamic information†
  Authors: **Matthew Kowal**, Mennatullah Siam, Md Amirul Islam, Neil Bruce, Richard Wildes, Konstantinos G. Derpanis
  Venue: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2022

- *Title*: Maximizing Mutual Shape Information
  Authors: Md Amirul Islam, **Matthew Kowal**, Patrick Esser, Sen Jia, Bjorn Ommer, Konstantinos G Derpanis, Neil Bruce
  Venue: British Machine Vision Conference, 2022

- *Title*: Simpler Does It: Generating Semantic Labels with Objectness Guidance
  Authors: Md Amirul Islam, **Matthew Kowal**, Sen Jia, Konstantinos G. Derpanis, Neil Bruce
  Venue: British Machine Vision Conference, 2021

- *Title*: SegMix: Co-occurrence Driven Mixup for Semantic Segmentation and Adversarial Robustness
  Authors: Md Amirul Islam, **Matthew Kowal**, Konstantinos G. Derpanis, Neil Bruce
  Venue: International Journal of Computer Vision, 2021

- *Title*: Global Pooling, More than Meets the Eye: Position Information is Encoded Channel-Wise in CNNs
  Authors: Md Amirul Islam*, **Matthew Kowal***, Sen Jia, Konstantinos G. Derpanis, Neil Bruce
  Venue: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021

- *Title*: Position, Padding and Predictions: A Deeper Look at Position Information in CNNs
  Authors: Md Amirul Islam, **Matthew Kowal**, Sen Jia, Konstantinos G. Derpanis, Neil Bruce
  Venue: International Journal of Computer Vision, 2021

- *Title*: Shape or texture: Understanding Discriminative Features in CNNs
  Authors: Md Amirul Islam, **Matthew Kowal**, Patrick Esser, Sen Jia, Björn Ommer, Konstantinos G. Derpanis, Neil Bruce
  Venue: International Conference on Learning Representations, 2021

Chapter 2

Background

This Chapter provides a review of interpretability methods for computer vision, separating previous approaches into two categories: *pixel-based* and *concept-based* interpretability methods. Pixel-based methods focus on understanding which pixels influence the model output or search the space of pixels for the most influential possible input. Concept-based interpretability aims to measure what human-interpretable concepts are encoded in a model's intermediate representations.

2.1 Pixel-based interpretability

The ultimate goal of post-hoc interpretability methods are to make transparent the underlying internal functions of a pretrained neural network. The following section surveys the post-hoc interpretability computer vision literature, starting with local explanations which try to explain model characteristics via individual pixel-based visualizations. It then covers pixel attribution maps that provide saliency maps for individual images.

2.1.1 Visualizing learned features

One way to understand what a model has learned is to search for the input that maximally activates some component of a model (e.g., output or neuron). The two methods to do so are activation maximization (Section 2.1.1.1), where the input is optimized via gradient ascent, or by using generative models to produce such inputs end-to-end (Section 2.1.1.2).

2.1.1.1 Activation maximization

Activation maximization refers to methods that optimize the input to a model, usually via gradient ascent, such that it maximally activates an internal unit of a deep network. Activation maximization methods provide visualizations that indicate what types of input visual phenomenon are responsible for

Method	Norm Penalization	Frequency Penalization	Transformation Robustness	Dataset Exemplars	Multi-layer	Transformers	Video
Erhan, et al. [72]							
Szegedy et al. [235]				✓			
Simonyan et al. [224]	✓						
Mahendran et al. [168]		✓					
Nguyen et al. [182]		✓					
Mordvintsev et al. [176]			✓				
Yosinski et al. [273]			✓				
Oygard et al. [187]		✓	✓				
Tyka et al. [245]		✓	✓				
Cammarata et al. [36]		✓	✓	✓	✓		
Ghiasi et al. [95]		✓		✓		✓	
Fel et al. [82]		✓	✓	✓		✓	
Feichtenhofer et al. [80]		✓		✓			✓

Table 2.1: A summary of core activation maximization methods. Explaining the columns from left to right: Norm penalization penalizes high pixel values. Frequency penalization refers to regularizes that penalize high-frequency generations. Transformation robustness methods randomly augment the input at each iteration to learn invariances over those transformations (e.g., color jitter, random cropping). Some methods return real exemplars along with the generated activation maximization or provide visualizations for multiple layers simultaneously. Later work extends these methods to transformer architectures and video understanding networks.

the activation of this internal unit. A broad range of optimization objectives have been explored, such as class probability logits, layers, channels (i.e., filters for CNNs), or neurons. Activation maximization is formulated as the following optimization problem:

$$x^* = \arg\max_x (a(x) - R(x)), \tag{2.1}$$

where x is the input to the model, x^* is the maximally activating input, $a(x)$ is the output of the network (i.e., layer, channel, and neuron), and $R(x)$ is a regularizer constraining the optimization search space.

Initial work in activation maximization [72, 275] performed optimization of the input without any regularization, targeting neurons for all three layers of two popular models at the time, Deep Belief Networks (DBN) [114] and Stacked Denoised Auto Encoders (SDAE) [249], trained on MNIST [60]. Later work [275] visualizes internal features using DeConvNets [276]. DeConvNets contain the same components as standard convolutional neural networks, but apply the operations (i.e., pooling, filtering) in reverse so as to map features to pixels. To map any feature back onto the input space at a given layer, they recursively (i) unpool, (ii) rectify, and (iii) filter the activation at each layer until the input is reached. The unpooling operation is defined as the approximate inverse of the maxpooling operation and works by tracking the maximum locations of preceding layers via 'switch' variables.

Despite the initial successes in producing interpretable feature visualizations using unregularized methods, generating realistic visualizations requires complex regularization of the optimization objective or using generative priors. Several forms of regularization have been proposed to encourage more realistic and interpretable visualizations from activation maximization. A summary of methods organized by the type of regularization used can be found in Table 2.1. The types of regularization can be distilled into two main categories. *Frequency-based regularization* penalizes high frequency infor-

Figure 2.1: Activation maximization visualizations of the most important concept for three different ImageNet [59] classes for a ResNet50 [110]. Results are produced from the state-of-the-art activation maximization method, MACO [82], which optimizes the input in the Fourier domain rather than in pixel space.

mation in the input image. The goal of this regularizer is to suppress uninterpretable high frequency patterns and induce more lower frequency object shape information to appear in the image. The first approach in this direction [168] introduces a total variation regularizer that penalizes high-frequencies in the input. *Transformation-based regularization* encourages inputs to highly activate the objective (e.g., channel or neuron) invariant of geometric or photometric transformations.

Many works have explored these two main types of regularization and have applied their algorithms for different types of applications and analyses. Simonyan et al. [224] use l_2 regularization on the input to penalize high pixel values while Nguyen et al. [182] leverage activation maximization (and evolutionary) algorithms for fooling deep networks. Their main findings show that, when optimized correctly, unrecognizable input images can cause a DNN to predict a seemingly unrelated category. Mordvintsev et al. [176] initialize the image with dataset examples and apply activation maximization iteratively on its own outputs and apply cropping after each generation which they use for image stylization.

A number of later works use different augmentations to promote the generation of objects in activation maximization visualizations. Yosinski et al. [273] were the first to apply Gaussian blurring as a transformation for activation maximization which aids in reducing high-frequency noise during optimization, guiding the model to focus on broader, more significant patterns. Then, Oygard et al. [187] extended the method of Mordvintsev et al. [176] and Yosinski et al. [273] by apply cropping, Gaussian blurring, and frequency penalization. Tyka et al. [245] further optimized the blurring hyperparameters to produce more realistic visualizations.

Many methods provide dataset exemplars, the naturally occurring examples from the dataset that most highly activates the targeted unit (e.g., neuron, class, etc), along with the generated visualiza-

tions [36, 80, 95, 235]. Providing these natural dataset exemplars, along with the generated visualizations, can produce more interpretable results than the generated ones alone.

More recently, activation maximization has been used to analyze DNNs beyond image-based CNNs for a single target class. Cammarata et al. [36] use activation maximization on adjacent layer channels to discovery 'circuits' of information processing (see Section 2.2.1 for a further discussion on the concepts analyzed). Ghiasi et al. [95] extend activation maximization to transformers. A core technical contribution of this work is that they show how naive application of activation maximization to self-attention components (e.g., queries, keys, and values) produce uninterpretable behaviour and discover that the position-wise feedforward layers result in successful visualizations. MACO [82] builds on these methods by optimizing solely an image's phase spectrum while keeping its magnitude constant to ensure that the generated explanations lie in the space of natural images. Example visualizations from MACO for three different classes are shown in Figure 2.1. Finally, Feichtenhofer et al. [80] perform activation maximization for video models (including two stream models) by leveraging novel spatial and temporal frequency regularizations in their regularization function.

2.1.1.2 Generative visualization of features

Another line of work leverages techniques in generative modelling to produce more realistic visualizations for the interpretations. Instead of restricting oneself to visualizing the activations or maximally activating inputs, these methods generate visualizations while matching a prior data distribution to encourage realism in the generations. The main drawback of these methods is, while the outputs certainly look more realistic and are more understandable, the generations may stray away from the true underlying feature distribution. This is due to the regularization encouraging that they lie in a similar distribution to the prior, e.g., by enforcing the visualization to be undetectable by a pretrained Generative Adversarial Network (GAN) discriminator [101, 213].

Inaugural work in this space by Nguyen et al. [181] simply places a GAN generator network [67] at the beginning of a classification model so that the output of the generator is the input to the classification model. Instead of optimizing the RGB pixels of the input to the classification network, they perform activation maximization on the *latent* code of the generator, which then produces the RGB input image, and thus produces a realistic visualization of the image that most highly activates the neuron of interest. Later work [180] extends this method by introducing additional priors on the latent that improves interpretability and achieved state-of-the-art ImageNet [59] generations at 227×227 resolution. These priors include enforcing the image be classified as the target class by a pretrained CNN and added noise during the latent code optimization to find additional modes of the data distribution.

Invertible neural networks have also been used to interpret deep models [73, 208]. Invertible neural

networks are a class of DNNs that are bijective and are usually used to map the data distribution to a known prior distribution (e.g., Gaussian). These networks are advantageous because you can calculate exact likelihoods and use inverse operations to go back and forth between the prior and data distributions. Invertible Interpretation Networks [73] use invertible networks to map a pretrained model's intermediate representation onto a set of known semantic concepts. They can then traverse the intermediate representation in the known concept space and map the concepts back into the model's representation via the inverse operation. Rombach et al. [208] extend this work by also mapping the latent of an autoencoder to the concept space which, by leveraging the decoder, allows for high-fidelity visualizations of the invariances learned by a model after manipulation of the concepts.

StylEx [153] aligns a StyleGAN [139] latent space with a classifier by jointly training an encoder to map to the StyleGAN latent space while constraining the output of the classifier to be consistent between a target image and generated image. After the alignment, one can manipulate the 'styles' (i.e., latent dimensions) in the StyleGAN latent space to control different attributes or generate counterfactual scenarios. Alternatively, Representation-Conditioned Diffusion Models (RCDM) [28] leverages *diffusion* models to interpret self-supervised representations. Since self-supervised learning (SSL) models have no classifier head, they simply train a diffusion model, conditioned on an intermediate representation, to generate images in the ImageNet data distribution. By sampling many images for a given input representation, they can observe the invariances captured by the model at that feature.

DISSECT [94] explains models by generating Concept Traversals (CTs). CTs disentangle and quantify how different concepts influence a classifier's decision. Contrasting StylEx, which uses StyleGAN to disentangle the representation, DISSECT jointly trains a generator, discriminator, and a *concept disentangler*, which encourages distinctness across concepts and similarity within a concept. They validate DISSECT on synthetic and realistic datasets and demonstrate its ability to identify biases (e.g., skin colour) and provide meaningful explanations by disentangling and traversing multiple concepts simultaneously.

2.1.2 Pixel attribution

The methods described in the previous section (Sec. 2.1.1) for interpreting learned features rely mainly on qualitative visualizations and can cause users to succumb to confirmation bias when they try to identify entities in visualizations. A different line of research, similar to that in traditional machine learning, focuses on feature attribution, which quantifies the contribution of each feature to the output of interest. Alternatively, in the space of computer vision, *pixel attribution* quantifies the contribution of each pixel to the model output. The result of these methods are saliency maps where higher values correspond to a larger contribution of the pixel to the output. The main challenge to overcome with respect to pixel attribution with DNNs is dealing with the numerous non-linear operations that occur between the input and output.

2.1.2.1 Weight and gradient-based attribution

Several methods have been proposed towards the goal of measuring the contribution of each input pixel to a DNNs prediction. Layer-wise Relevance Propagation (LRP) [11] was the first proposed pixel attribution algorithm. The intuition behind LRP is that the activation of a given neuron can be decomposed as a weighted sum of neurons from the previous layer. Recursively applying this decomposition from the model output until the input layer results in assigning each input pixel with the amount it affects the model output.

Later work points out that current pixel attribution methods (e.g., LRP) lack desired axiomatic properties: (i) *Sensitivity* and (ii) *Implementation Invariance* [232]. An attribution method satisfies sensitivity if, for all inputs and 'baseline inputs', i.e., inputs that differ in a single feature but have different predictions, then the differing feature should be given a non-zero attribution. Attribution methods satisfy implementation invariance if, for two networks differing in their implementation that produce the same outputs for all inputs, the attributions are always the same between networks. To this end, they propose a method satisfying both constraints, Integrated Gradients [232], effectively a combination of Gradients (Implementation Invariance) and LRP (Sensitivity). Formally, for the i^{th} pixel, the integrated gradient is defined as

$$\text{IntegratedGrads}_i(x) := (x_i - x_i') \times \int_0^1 \frac{\partial f(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha \tag{2.2}$$

where x' is a baseline value (generally 0) and x is the input. Note that integrated gradients are implemented with a discrete sum and $0 < \alpha < 1$ denotes the step size, where a smaller α usually corresponds with better performance in exchange for more computations.

Another seminal work in this space, Class Activation Maps [283], leverages a common architectural choice in most deep vision models: the global average pooling layer. The global average pooling layer reduces a spatial tensor, z, of size $c \times h \times w$ to one of c, where c is the number of channels, h is the feature height, and w is the feature width. To obtain the logit score for a given class, a linear operation, W, is applied, where W_i denotes the weight of channel i to the logit for that class. Thus, one can obtain a localized heatmap of contribution by taking a weighted average of the non-pooled channel activations according to:

$$Y_{CAM} = W_1 z_1 + W_2 z_2 + ... + W_c z_c, \tag{2.3}$$

where z_i denotes the i^{th} channel of feature tensor z.

Due to the simplicity and effectiveness of the method, as well as the prominence of CNNs for computer vision tasks, many works have extended CAM with different versions. While image gradients [275] can be used as a simple method for localizing important pixels, Grad-CAM [218] combines

the gradients of a desired output class with the convolutional feature maps (i.e., the CAM component) to get more faithful and interpretable output heatmaps. Grad-CAM++ [41] extends these approaches by performing a location-sensitive weighting of the gradients before combining them with the feature maps. This results in pixel-attribution heatmaps that better handle global object shape and multiple object images.

2.1.2.2 Occlusion-based attribution

Contrasting gradient-based approaches, another line of work computes the contribution of input pixels to the output based on masking the input and computing the difference in output values. The intuition is that more important regions, when masked, will produce larger negative changes in the output than unimportant regions. These methods are advantageous in the sense that they do not use privileged information about the model and only rely on input-output correlations to produce their explanations.

Local Interpretable Model-agnostic Explanations (LIME) [206] is an occlusion-based feature attribution method used for tabular, textual, and image data. LIME generates a dataset using masked samples of an existing data point and then a linear, i.e., intrinsically interpretable, model is then trained on this dataset to approximate the model's decisions locally (i.e., but is not required to approximate the global function in an interpretable way). While LIME is applied to tabular and textual data by perturbing single data points, perturbing a single pixel would not provide a strong signal for the explanation. To this end, LIME leverages *superpixels* [15], i.e., image sub-regions composed of similar and connected pixels, to generate variations of the input image. Then, the perturbations are produced by setting an entire superpixel to zero (or the image mean value). Formally, LIME calculates the attribution of an input image x via

$$LIME(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g), \tag{2.4}$$

where x is the input to be explained (e.g., an image), g is a local interpretable model (i.e., the weighted average of logit scores for the target class based on each superpixel), L is a loss (e.g., L_2 error) which measures how close the explanation is to the original prediction by model f, G is the hypothesis space of possible explanations (e.g., linear regression weights of the weighted average), Ω measures the model complexity (e.g., preferring fewer superpixels), and π_x is a proximity measure between two instances that is used to heavily penalize predictions in the local neighbourhood more than ones far away.

Randomized Input Sampling for Explanations (RISE) [195] extends LIME to the pixel level and shows that this granularity improves the fidelity of the explanations. The key to the approach is, instead of masking a single input (i.e., pixel) at a time when calculating the drop in performance, they mask a large fraction of pixels for many iterations and then take a weighted average over all sampled

masks, where the weighting is proportional to the drop in performance for that mask. While RISE provides a more faithful explanation of the model, the computation burden is substantial, requiring up to 10,000 masked forward passes through the model to produce an explanation for a single image. To address this, they actually sample pixels to mask from a lower resolution image, and then bilinearly upsample so that regions greater than a single pixel are masked out in the original image space.

Later work [39] points out that masking images with zeros or the mean does not accurately match the true data distribution and proposes to use a conditional (i.e., conditioned on the rest of the image pixels) generative model to fill in plausible values.

2.1.2.3 Transformer attribution

Another line of work focus solely on visualizing pixel attribution maps for vision transformers. Rollout [2] combines the attention maps from all heads at all layers to generate a final heatmap. Partial LRP [251] extends Rollout by weighting each head differently based on LRP [11]. However, these methods are not specific to a given class since the attention weights are class agnostic. Chefer et al. [43] also considers the gradients of the target class with partial LRP to produce class-specific saliency maps while Transition Attention Map (TAM) [274] uses integrated gradients instead.

ViT-CX [266] provides the first occlusion-based method for visualizing transformer attribution maps. Notably, ViT-CX uses feature maps instead of attention maps. They first show a large degree of redundancy in different transformer feature maps, and therefore perform agglomerative clustering over the feature maps to produce a small set of unique semantic masks. Next, the semantic masks are randomly set to zero to calculate the drop in performance when deleting each region, i.e., the importance of each region. The authors note that naive masking creates artifacts at the mask boundary in the saliency map and that simply adding per-pixel random Gaussian noise reduces this issue. The final saliency map is calculated via the weighted average (based on the drop in target class performance) of the semantic masks obtained through the agglomerative clustering.

2.1.3 Multi-layer attribution methods

A small set of approaches design multi-layer attribution maps that produce the attribution that one layer has on another layer. However, contrasting LRP, these methods produce a separate set of attribution maps for each layer, rather than a single input-output attribution. One work examines CNNs by generating an explanatory graph from the pretrained model [278]. The key assumptions in this method are to use individual filters in the CNN as the base unit of analysis and that filters capture solely object parts (i.e., they do not look to extract colors, textures, etc.). The graph is built in a top down manner, where filters at layer $L + 1$ are explained by a set of similar filters at layer L, through the Expectation-Maximization algorithm, where the filters are modelled by Gaussian Mixtures.

Later work, gradient-based Activation Propagation (gAP) [49] directly provides an extension of GradCAM to explain the information flow of multiple layers and again targets the explanation of CNNs on the filter level. gAP decomposes a filter, i at layer $l+1$ by simply calculating calculating the derivative of that filter with respect to a filter, j at layer l, multiplied by the filters feature map (i.e., following GradCAM). Rather than present thousands of CNN filters, they show the top-k filters in the decomposition based on the total sum of the attribution map, which is not normalized.

The previous studies have revealed complex reasoning patterns emerging across multiple layers in deep models. However, there does not exist any direct way of quantifying the contribution of a concept at one layer to a concept at a deeper layer. Previous work yielded ways to extract interpretable concepts from single layers and examine their impact on classification, but did not afford multi-layer concept analysis across an entire network architecture. To this end, Chapter 5 of this dissertations introduces the Visual Concept Connectome (VCC) [149], which discovers human interpretable concepts and their interlayer connections in a fully unsupervised manner. The approach simultaneously reveals fine-grained concepts at a layer, connection weightings across all layers and is amendable to global analysis of network structure (e.g. branching pattern of hierarchical concept assemblies). Quantitative and qualitative empirical results show the effectiveness of VCCs in the domain of image classification. VCCs allow for the finegrained interpretation of deep models by observing individual concept compositions, or global network analysis by aggregating statistics of edge strengths over many layers. When comparing architecture types, CNNs were found to rely heavily on final layer concepts for prediction, while transformers were found to have more diverse connection strengths in the final layer and a more complex assembly of later concepts from earlier ones. We also show how VCCs can be useful for the application of failure mode debugging to reveal where mistakes arise in deep networks. This work is one of the first to show that finding concept-based 'circuits' within vision models is possible and effective for understanding complex model behaviours.

2.1.4 Discussion

The preceding section presents a review of pixel-based interpretability methods for post-hoc understanding of deep vision models. First, feature visualization methods were reviewed (Section 2.1). These methods optimize, along with various regularizations and constraints, the input to a model in the direction of maximal activation of a particular unit of interest. The user then performs a qualitative analysis of the input to determine what concepts are present. Alternatively, Section 2.1.2 reviewed recent methods for pixel-attribution, where a measure of importance is ascribed to each input pixel. These methods use model weights, activations, and gradients as measures of importance as information flows through a network.

However, visualizing features and pixel attribution may not be the ideal way to interpret the representations of deep networks for a variety of reasons. In the case of feature visualization, the

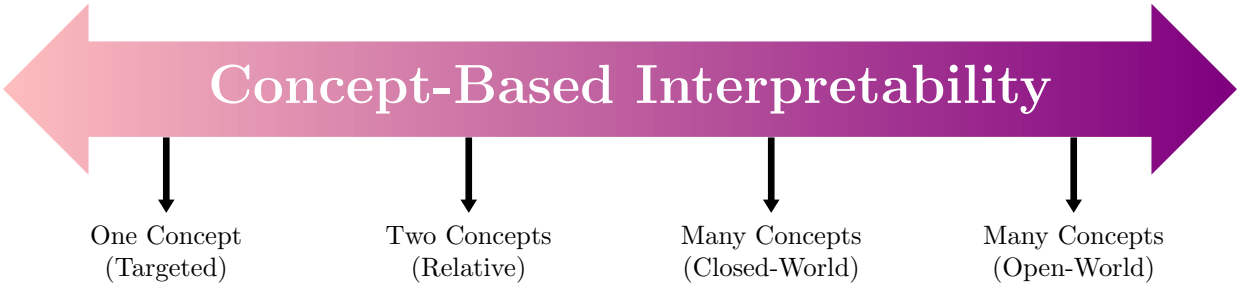


Figure 2.2: The spectrum of concept-based interpretability approaches in terms of the number of concepts used in the analysis. The left side indicates approaches that select one or two concepts to analyze; sacrificing concept breadth for measurement quality. On the right, approaches either use a large labelled concept dataset (Closed-World) to find concepts in the model that also exist in the dataset, or perform unsupervised analysis of model representations (Open-World) to discover concepts without being limited to any apriori labels or concept definitions.

results often do not reside in the true data distribution and are less connected to real use. Additionally, multiple studies have shown that using data exemplars (i.e., real samples from a dataset) provides more interpretable explanations for humans than synthetic ones [29,290]. For pixel attribution, single pixels are not use-friendly in terms of their interpretability to a human and many methods fail randomized baseline tests [6]. In addition, both of these methods have the problem of confirmation bias when humans interpret their results [141].

2.2 Concept-based interpretability

Towards a solution to the aforementioned problems, *concept-based* approaches to interpreting deep vision models explain the representations in terms of human-friendly 'concepts' based on sets of data exemplars. A concept can be defined as *an abstraction which generalizes from a single instance*. For example, colors, parts, objects, groups, or even emotions (e.g., horror) and ideas (e.g., holidays), are concepts. The main goal of concept-based approaches is to (i) identify concepts encoded in a model's representations and (ii) quantify the degree to which the concept affects a DNNs output.

One way to formalize this problem, presented by Kim et al. [141], is that vision models represent concepts in the space, e_v, which can be thought of a set of basis vectors spanning the vector space of the vision model, E_v. Humans represent individual concepts in their brains in an unknown space, e_h. Therefore, the problem of interpreting a vision model based on concepts can be thought of as finding a function $g : e_v \rightarrow e_h$ that can map from the model space of concepts to the human space of concepts. Note that it is expected that this function will not have perfect accuracy nor precision, as e_v is likely to not contain certain human interpretable concepts, or g may fail to quantify some portions of the model domain.

Figure 2.2 shows the spectrum of approaches to concept-based interpretability in terms of the number of concepts used in the analysis. At the left end of the spectrum, one can specify individual

concepts to measure and handcraft methodologies that can quantify the degree to which the model captures this concept. Of course, this approach severely limits the number of concepts that can be measured (e.g., one or two); however, contrasting methods that try to interpret many concepts (i.e., as explored in Sections 2.2.4 and 2.2.5), this direction sacrifices generality for specificity and may often give more accurate and deeper insights into how the model uses a specific concept.

Other approaches (on the right side of the spectrum) place a larger emphasis on the diversity of measured concepts. This can be done by comparing model representations to a large labelled concept dataset, which is denoted as closed-world, since it is not possible to find concepts outside of the labelled dataset. Alternatively, open-world approaches use unsupervised methods (e.g., clustering) to discover concepts and are not limited to any particular set of apriori defined concepts, and are thus denoted as open-world. In summary, studies performing closed-world interpretability usually ask the question "How much does the model use concept X for task Y?", while open-world interpretability asks the question "What concepts are important to model X for task Y?".

We note that there is also a broad range of works in the generative modelling landscape that focus on concepts within generative visual models (e.g., Generative Adversarial Networks [101], Diffusion models [115], etc). While related, this thesis focuses on discriminative architectures because the space of interpretability for generative modelling is vast. Indeed, this space encompasses methods from both the interpretability community [19, 89] and controllable generative models community (e.g., disentangled representation learning [257, 265]). For this reason, we leave the review of the interpretability and controllability of generative modelling to future work.

2.2.1 One concept

2.2.1.1 Single CNN filters concepts

A small, yet impactful, number of studies have been done targeting a single concept in vision models. The Circuits Thread [36] used a number of interpretability methods to isolate specific functions of CNN filters. The initial work [183] studied filters that activate for curves, high-low frequency boundaries, and pose-invariant dog heads. They then provide qualitative evidence of how these concepts are composed through adjacent layers and finally, they discover neurons with the property of *polysemanticity* - neurons that encode more than one semantic concept. This phenomenon of superposition has proven to be a core problem for interpreting all large models and has been studied extensively in the language domain [69]. Further studies contained in the circuits thread provide a more detailed investigation of curve detectors [37], high-low frequency detectors [214], and oriented band-pass filters [194].

Overall, these studies demonstrated using interpretability techniques that there are several reoccurring phenomenon captured by various CNNs. The authors therefore investigate whether any filters are truly 'universal', and discover that all analyzed networks contain both curves and high-low fre-

quency detectors. They further hypothesize that, if artificial networks learn universal features, then maybe biological networks would learn similar features. Indeed, high-low frequency detectors were found three years later in mouse primary visual cortex [64]. In the video domain, Mineault et al. [175] showed that spatiotemporal models produce responses much more similar to neurons in the primate dorsal stream when trained to predict an agent's self motion compared with standard action recognition objectives. Further investigation of human and artificial network visual systems may reveal other fruitful directions of network designs and objectives.

2.2.1.2 Position information

Another concept of focus commonly analyzed in CNNs is absolute position information [9, 127, 130, 131, 140]. CNNs are designed in such a way to be parameter efficient due to the convolutional operator being translation equivariant. In practice, however, this is not the case. Islam et al. [127] began this line of work by showing that CNNs encode absolute position information and produce outputs dependant on the location of input features. Their findings suggested that zero-padding was the main culprit of this phenomenon. An extension of this work [131] further investigated how 'canvases' (i.e., areas surrounding the image region) play a role in the semantic information encoded in CNNs, and further, that position information can act as a feature or a bug depending on the task at hand. Later work by Alsallakh et al. [9] confirmed these findings and demonstrated that encoding position information causes CNNs to have blind spots. If a small object is located in these regions, activations are suppressed and misdetections become more common. Other work showed that using zero-padding of size two injects more position information and showed that this positional information benefits small datasets in particular [140].

Given that the convolutional operator is translation equivariant and a global average pooling layer is generally applied before calculating the logits for a classification network, how could the model encode absolute position information? One study demonstrated that position information is actually encoded within the ordering of the channel dimensions [130]. Moreover, by evaluating performance on both location classification and object recognition, they were able to show that perturbing the channel order removed the positional information but kept the majority of semantic information, demonstrating that disentangling the two is still possible. Subsequent analysis revealed that individual filters learn to detect specific rows, columns or corners containing *padding* values in the input [9, 194]. This is precisely why zero-padding results in a more obvious border region than input-dependant padding types (e.g., circular, reflect, and replicate).

2.2.2 Writing in CLIP

A small subset of works explore an interesting phenomenon found in the CLIP [199] vision-language model. This model, trained to align images with corresponding captions, was observed to be susceptible to typographic attacks: when placing a white sheet of paper with "iPad" in front of an apple, the model classifies the image as an iPad rather than the apple [100]. This was further explored in recent work [170] which aimed to understand why this attack is successful on the CLIP model. Their main contribution is devising a method for separating subspaces within the models activation space that correspond to the spelling capability from the visual processing capability in the CLIP model. After separation, they show applications for reducing text artifacts in downstream image generation tasks and can also defend against typographic attacks.

2.2.3 Two concepts

2.2.3.1 Shape vs. texture information

Prior to attention-based networks becoming the prevailing architecture for most vision tasks [66, 248], CNNs achieved state-of-the-art performance on the majority of computer vision, and specifically image-based, tasks. While visualizing the learned features through activation maximization (Section 2.1.1.1) indicated that CNNs were learning notions of object parts in mid-layers and global shape in deeper layers, Geirhos et al. [30, 93] demonstrated that the concept of global object shape is largely ignored by CNNs when making predictions, and local texture information is used instead. The metric they used was dubbed the 'shape bias' and 'texture bias' of a model. This was measured based on a cue-conflict dataset in which every image contains a shape label and a texture label through the use of stylization [122]. An example image, along with the top-three most likely classes predicted by a ResNet50 [110], is shown in Figure 2.3. They generate a large dataset of these cue conflict images with 16 different classes, i.e., each image is randomly stylized with one of the other possible 15 classes. Then, the 'shape bias' of the model is the percentage of correct classifications made towards the object shape rather than texture, and vice-versa for 'texture bias'. The main findings of the study are, while humans are heavily biased towards predicting the objects shape (95.9%), convolutional neural networks are biased towards texture (77.9% for ResNet50). The authors then propose a method to bias CNNs towards shape, by training the model to predict the shape class on a large generated dataset of stylized images. With the correct training and fine-tuning (on normal images) protocol, they observe moderate improvements in performance and significant improvements in robustness to common out-of-distribution distortions, such as Gaussian noise, motion blur, and weather types (snow, fog and frost).

However, 'shape' is a somewhat ill-defined concept in the above studies, as both global or *local* object shape can contribute to the model decision. One study showed the same 'local shape' issue with

Figure 2.3: Examples from the Shape-Texture Cue Conflict Dataset [93] of (a) a texture image (elephant skin), (b) a normal image of a cat, and (c) a cue conflict image generated by stylizing the cat image with the elephant skin image. The CNN predicts the cue conflict image as the texture class while humans are biased towards the shape category.

the initial shape bias metric [93] and proposed a new metric for shape bias [128]. This metric samples pairs of images containing the same shape or texture information, through the same use of stylization as before [93, 122], and then measures the mutual information between the latent representations at a given layer. This metric ultimately provides a more finegrained metric for shape and texture bias, and also measures this in the *internal* representations of a given model. Further work uses this metric to also maximize a model's bias towards encoding shape information that, as before, results in a more robust model with moderate performance improvements on natural images [129].

2.2.4 Closed-world many concepts

Rather than focus on a small number (i.e., one or two) concepts, Network Dissection [18] leverages a large dataset of visual concepts that enable the user to quantify the degree that a neuron captures a given concept. To this end, they construct the Broadly and Densely Labeled (Broden) Dataset, which consists of five densely labelled datasets listed in Table 2.2 and captures the concepts of objects, scenes, object parts, textures, materials, and colors.

The main idea of Network Dissection is to identify neurons that activate for a given concept. For each image $\mathbf{x}$ in Broden the activation map $A_k(\mathbf{x})$ of every neuron (i.e., convolutional filter) k is obtained. Then the top quantile level T_k is calculated such that $P(a_k > T_k) = 0.005$ for each spatial location across the set of activation maps over the entire dataset. The activation map of unit k is thresholded into a binary segmentation, $M_k(\mathbf{x}) = A_k(\mathbf{x}) \geq T_k$, and then compared against every concept, c, in the dataset by computing the intersection over union (IoU) for all (k, c) pairs as

$$IoU_{k,c} = \frac{\sum |M_k(\mathbf{x}) \cap L_c(\mathbf{x})|}{\sum |M_k(\mathbf{x}) \cup L_c(\mathbf{x})|}, \tag{2.5}$$

Table 2.2: Statistics of each label type in the Broden [18] data set, a compasition of five different datasets capturing a spectrum of visual concepts.

Category	**Sources**	**Classes**	**Avg sample**
Scene	ADE [286]	468	38
Object	ADE [286], Pascal-Context [177]	584	491
Part	ADE [286], Pascal-Part [46]	234	854
Material	OpenSurfaces [20]	32	1703
Texture	DTD [52]	47	140
Color	Generated	11	59 250

where $L_c(\mathbf{x})$ is the annotation masks for some concept c. A neuron is reported as detecting concept c if $IoU_{k,c}$ is above a certain threshold, $IoU_{k,c} > 0.04$.

While Network Dissection identifies concepts in individual neurons, closed-world many-concept based approaches have also been proposed for distributed representations. Interpretable Basis Decomposition (IBD) [285] decomposes the CAM for a given model and target class into a set of finer-grained concepts, each with their own CAM and associated importance to the final prediction. To associate each spatial feature in the model with a concept, they train a separate logistic binary classifier on every concept in the Broden dataset [18] (see Table 2.2) to detect the presence of a concept. They perform the decomposition on the final representation (i.e., before the linear layer) by projecting each feature vector onto the set of learned concept basis vectors. They also add a residual value in the decomposition to explain the contribution not captured by the pre-determined concept basis.

While a common approach for measuring the quality of self-supervised representations is through linear classification performance for downstream tasks, Quantized Reverse Probing [151] aims to interpret self-supervised visual representations by measuring what semantic concepts are encoded in the representations. To do this, they follow a three step approach. First, they evaluate self-supervised models on an image dataset to get features and cluster them to obtain a set of concepts. Second, they label all input images used with expert models trained with full supervision. Finally, given these concept clusters and labelled images, they train a linear model to map the concepts to clusters and measure the mutual information between the representations and concepts.

Another method, Testing with Concept Activation Vectors (TCAV) [141], proposed a method to quantify the model prediction's sensitivity to a single user-chosen concept based on directional derivatives. This method works in the closed-world, supervised setting, where a concept is specified by the user apriori. More specifically, a user first supplies a small dataset ($\sim$50) of positive images, P_C, that represent a concept they are interested in quantifying in a model at a given layer, f_l. Then, a linear classifier is trained to distinguish between layer activations of the positive images, $\{f_l(x) : x \in P_C\}$ and negative random images, $\{f_l(x) : x \in N\}$. We define this classifier as a 'concept activation vector' denoted as $\mathbf{v}_C^l$. Then, for a given object category, k, and set of inputs with that label, X_k, the TCAV

score is calculated via

$$\text{TCAVQ}_{C,k,l} = \frac{|\{\mathbf{x} \in X_k : S_{C,k,l}(\mathbf{x}) > 0\}|}{|X_k|}, \tag{2.6}$$

where $S_{C,k,l}(\mathbf{x})$ is the 'concept sensitivity' of class k to concept C. The concept sensitivity represents the alignment between the CAV, $\mathbf{v}_C^l$, and the positive gradient direction of the class-k logit and a concept at layer l, and is calculated with

$$S_{C,k,l}(\mathbf{x}) = \nabla h_{l,k}(f_l(\mathbf{x})) \cdot \mathbf{v}_C^l, \tag{2.7}$$

where $h_{l,k}$ are the remaining layers from l to the class logit for class k.

Notably, Equation 2.6 only depends on the sign of $S_{C,k,l}$, i.e., the fraction of positive alignments between the gradient and the CAV, and other metrics are possible (e.g., based on the sensitivity magnitude).

2.2.5 Open-world concepts

But what if you do not know apriori the concepts contained in the models of interest or do not have the resources to construct a relevant database of concepts? Concept *discovery* methods identify concepts in the open-world setting and have no prior assumptions on the concepts being looked for. Concept discovery can be formalized as two steps [83]: (i) concept extraction and (ii) concept importance scoring. A small number of papers aim to perform open-world concept discovery on the neuron level, while the majority of work in this space analyzes distributed representations.

2.2.5.1 Single neuron open-world concepts

In terms of neuron-level concept discovery, Neuron Shapley [99] first identifies the most important filters to the model output based on Shapley values [161,166], a game-theoretic principle that calculates the contribution of a player (i.e., a filter) to the game reward (i.e., the model output). Shapley neurons are calculated by computing the output of the network with all possible subsets of neurons zeroed out. Given the computational expense of such a calculation for a network with millions of neurons, they provide multiple approximations based on Monte-Carlo estimation, early truncation, and adaptive sampling. After identifying the most important neurons, they use activation maximization or highest-activating input images to visualize the concepts captured by such neurons. Finally, they show that removing a small number of neurons can provide post-training fixes to models without the access to any training data.

Concept relevance propagation (CRP) [5] combines a local feature attribution methods, LRP [11], with global visualization techniques to visualize what concepts neurons capture at each layer and how they contribute to one another. CRP extends LRP by adding a condition to the relevance

map, e.g., a condition could be an output category. A CRP attribution map computes a heatmap of importance with a set of the top-five most activating image samples, along with a ranking of the neurons importance to a given output class.

Another line of work aims to discover concepts encoded by neurons through the use of natural language descriptions. MILAN [112] (mutual information guided linguistic annotation of neurons) automatically labels neurons with open, compositional, natural language descriptions of the concept being encoded. They first train a vision-language model to caption images conditioned on a highlighted region. Since such a dataset does not exist, they first generate a dataset of image-mask captions through the following steps: (i) for all neurons in seven different models, they recover the highest activating regions (thresholded into a binary mask) in a large dataset of images (composed of ImageNet [59] and Places365 [284] (ii) they present each set of masked images to three humans and ask them to describe what is common to the image region. This results in a dataset of 20k images with 60k total annotations. They then train a simple recurrent image captioning model based on Show-Attend-Tell [267] to caption a set of highlighted images. They then use this trained model to describe new neurons, given their set of highlighted image regions over a held-out dataset. They demonstrate how MILAN can be used to analyze important neurons, audit privacy preserving models, and editing features for improved robustness.

Later work, Latent Visual Semantic Explainer (LaViSE) [271], attempted natural language concept discovery for single CNN filters based on an approach that leverages the contrastive training of a vision and language model. More specifically, they align the pixel-wise representations of a vision model, using a small neural network dubbed the "explainer", with the representations of the text, i.e., the semantic segmentation label, corresponding to that image location. While this uses a labelled dataset for training the explainer, during inference, the explainer can align visual representations to textual ones not seen in the training dataset.

2.2.5.2 Distributed open-world concepts

Concept discovery for distributed representations generally produces a dictionary of concepts by first passing the dataset through the model and obtaining some set of features followed by clustering these features via some dictionary learning algorithm (e.g., K-Means [163], Non-negative Matrix Factorization (NMF) [154]) to produce CAVs. Concept importance scoring involves calculating a set of scores that provide a measure of influence that each concept has to the model output (or class).

Automatic Concept-based Explanations (ACE) [98] proposed the initial concept discovery work focusing on image-based models. For the concept extraction, ACE first partitions a set of input images into superpixels using Simple Linear Iterative Clustering (SLIC) [3]. These superpixels are then passed through a target CNN and the activations are clustered at a given layer, layer "mixed_8"

for the Inception-V3 model [234] is used for all experiments to produce a set of CAVs; CAVs are then pruned to ensure that each concept contains (i) superpixels that span a sufficiently diverse set of the discovery images and (ii) large number of total superpixels. The concept importance for each CAV is done using TCAV [141].

ConceptSHAP [272] extends ACE in multiple ways. First, they define the notion of *completeness* - how sufficient a particular set of concepts are for explaining a model's output. They then introduce a novel concept discovery method that is regularized to be interpretable and complete. Following Supervised Topic Models [172], they discover concepts (i.e., topics) by jointly optimizing a set of concept vectors and a mapping function. Further regularization is added to encourage orthogonality of the discovered concept vectors. They then calculate important scores based on an adaptation of Shapley values [161, 166] to their discovered distributed concepts.

Invertible Concept-based Explanations (ICE) [281] leverages NMF to extract CAVs and, rather than applying clustering on the global average pooled features of superpixels, performs the matrix decomposition on the feature maps themselves. ICE uses TCAV as the concept importance quantification. ICE performs a human study comparing three dictionary learning techniques: (i) PCA, (ii) KMeans, and (iii) NMF. Computationally, they show that PCA is the most faithful method in terms of accuracy and fidelity of the concepts learned, followed by NMF and then KMeans. However, when considering human ability to predict model outputs based on the interpretations, NMF was superior.

Concept Recursive Activation FacTorization (CRAFT) [86] extends previous concept-based interpretability by producing corresponding attribution heatmaps along with the concept exemplars. CRAFT uses random cropping on the final model layer to propose the initial dataset of segments (note they have an optional method to recursively decompose the set of segments in the final layer to an earlier layer). Then, similar to ICE, NMF is applied to result in a set of CAVs. To obtain concept importance scores, they leverage methods from the field of sensitivity analysis and use the *total Sobol indices* [227]. The core of this importance metric is to measure the average variance of a concept on the output when randomly removing channels encoding that concept in the final layer of the CNN. Finally, they combine previous gradient-based approaches (e.g., GradCAM [218]) and introduce a way to implicitely differentiate through the NMF operation. Thus, CRAFT produces concepts with attribution heatmaps within the concept region.

Most recently, dictionary learning has seen a revival for its use for open-world concept discovery in the visual domain. Driven by their promising results in NLP [56, 215, 236], sparse autoencoders (SAEs) have seen wide usage in their ability to discover meaningful directions in feature space for a number of different vision models. SAEs encode activations with a single linear layer with an activation function to project the feature into a sparse (encouraged via regularization), high-dimensional space. Finally, a single linear layer projects back into the activation space and a reconstruction loss is applied to the original input. The dimensionality increase is motivated by the polysemanticity results from recent

work [183] which suggests that more concepts are encoded in an activation than there are channels. Recent work [83] has shown an equivalence between SAEs and other clustering based approaches (e.g., NMF), however, the core benefit of SAEs are their scalability: they are relatively easy to train at large scales on billions of data points with stochastic gradient descent. Optimized GPU-based libraries and frameworks that were developed for training neural networks can now be applied for extracting concepts from neural networks.

Early work compared SAEs to clustering and analyzed early layers of Inception v1 [102, 176], revealing hypothesized but hidden features. SAEs have also been leveraged to construct text-based concept bottleneck models [144] from CLIP representations [23, 188, 199, 203], showcasing their versatility across modalities. One work improves the stability of SAEs across training runs by imposing constraints on the learned dictionary [85]. Universal SAEs [240] train a single SAE jointly across several vision architectures to discover both shared (i.e., found in all models) and unique concepts (found in only one model).

2.2.6 Concept-based interpretability for video understanding tasks

Interpreting deep spatiotemporal models is a largely understudied topic in computer vision despite achieving state-of-the-art performance on video understanding tasks, such as action recognition and video object segmentation. However, these models remain largely opaque and interpreting them has remained a largely under explored area of research [113]. Regarding the research that has been done, one core and frequently asked question regarding deep video understanding models is to what extent they actually reason temporally to make decisions? Original action recognition datasets were simply assumed to require temporal reasoning to solve. However, many different studies have subsequently found that DNNs trained for video are largely relying on static *appearance* concepts over dynamic *motion*-based concepts [50, 160, 252].

2.2.6.1 One concept

An early effort uses three time-aware proxy tasks to understand the temporal capabilities of video models [96]. The first task is predicting the arrow of time and measures the property of temporal asymmetry (i.e., is the distinction between forward and backward time direction encoded in the model). Second, they measure temporal continuity through the auxiliary task of future frame selection. Finally, they evaluate the temporal causality of an encoder by measuring the action classification performance on the temporally challenging Something Something v2 dataset [103], a fine-grained action recognition dataset with actions solely distinguishable based on temporal and motion information (e.g., moving something from left to right vs. moving something from right to left). They observed that recurrent models handle temporal asymmetry better and 3D convolutional encoders perform better on continuity

and causality tasks. Furthermore, they propose a hybrid between recurrent and hierarchical models: Time-Aligned DenseNet, which is densely connected along the temporal dimension (instead of the spatial one like in the original DenseNet [119]). They show this model has the best performance trade-offs between the three tasks.

A related study quantifies the scene representation bias of action recognition datasets [50, 160]. Li et al. [160] measure the scene representation bias of a dataset via

$$B_{scene} = \log \frac{M(D, \phi_{scene})}{M_{rand}}, \tag{2.8}$$

where ϕ_{scene} is a scene representation, $M(D, \phi_{scene})$ is an action classification accuracy with a scene representation ϕ_{scene} on a dataset D, and M_{rand} is a random chance action classification accuracy on the dataset D. The scene representation-based classifier, $M(D, \phi_{scene})$, is typically a network trained solely on scene classification. The intuition of this metric is 'How much can a model encoding scene information predict about the action in a dataset?'. They demonstrate that existing action recognition datasets, such as Kinetics [38] or UCF101 [228], contain a large bias towards the scenes and objects in a video, rather than the action being performed. Towards a solution to this problem, they collect a new dataset, Diving48, consisting of 48 types of dives in competition settings. Thus, the visual stimuli of the scene and objects are identical across videos and accordingly the measured bias is much lower. Choi et al. [50] then proposed a training strategy to debias a model from using scene information, towards using action and motion information, on any dataset. The core of their approach involves using a person-detector to mask out the people doing actions in the scene, and implementing an entropy regularizer to encourage the model to be uncertain about the action being performed.

Recent work also has appeared that introduced an action recognition dataset to completely decouple static and dynamic information [124]. More specifically, given a standard RGB video, this approach generates an 'appearance free' version of a dataset through the following steps: (i) initialize a noisy image frame, (ii) compute the optical flow between the two target adjacent input frames, and (iii) use the optical flow to warp the initial noisy image frame. Steps (ii) and (iii) are repeated for every pair of frames in the video to produce an entire AFD video. Examples of AFD are shown in Figure 2.4.

2.2.6.2 Two concepts

Other work predicate model interpretation on a specialized dynamic texture recognition dataset [106]. Hadji et al. [106] designed a cue-conflict dataset, Dynamic Texture DataBase (DTDB), for texture recognition. Each video can have a *static* and *dynamic* label; multiple examples are shown in Figure 2.5. By training different architectures on both the appearance and dynamics organization of the dataset, one can gain an understanding of the capabilities of the models by comparing the resulting performance between the two. Their main findings include superior capabilities of two-stream mod-

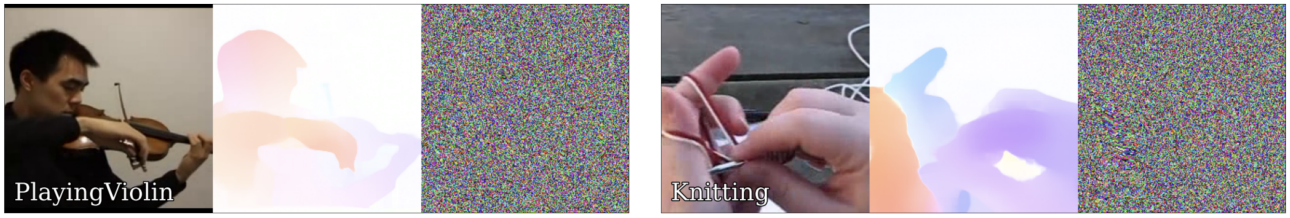


Figure 2.4: Examples from the Appearance Free Dataset (AFD). AFD contains videos that have all solely dynamic information while all appearance information is removed. We show two frames from videos here, where each consists of the RGB frame (left), optical flow in Middlebury color coding [13] (middle), and the corresponding appearance free frame (right). When seen as a video, AFD reveals the motion of the action while each individual frame provides no discriminative information. Figure reproduced with permission from [124].

els to encode both dynamics and appearance features, and they design a novel architecture which outperforms previous state-of-the-art architectures.

While the preceding works focused on interpreting models based on their inputs and outputs, the first major contribution in this dissertation [147, 148], presented in Chapter 3, proposes a method to quantify the static and dynamic bias contained in the *internal* representations of deep spatiotemporal models. This method works by sampling video pairs that share either solely static or dynamic information. More specifically, to sample videos with the same dynamics but perturbed static information, video stylization is applied to a video with two different styles. Alternatively, to sample videos with the same static information but perturbed dynamics, the frames of two copies of the same stylized video are randomly permuted. Then, after passing multiple static and dynamic video pairs through the model, the mutual information is calculated between the static pairs and dynamic pairs, independently. The relative mutual information provides a measure of the biases towards static or dynamic information on a per-layer or per-channel basis. This work disputes the common assumption that the action recognition dataset Diving48 [160], a dataset covering 48 types of dives, was dynamics biased and instead showed that Something-Something-v2 [103] provides a better benchmark for dynamic representations. Later work [148] extended this analysis and introduced methods for increasing the dynamic information learned by models for the tasks of fine-grained action recognition and automatic video object segmentation for camouflaged animals.

2.2.6.3 Closed and Open-world concepts

There are currently no methods exploring closed world concept-based interpretability for $N >> 2$ concepts which is mainly due to the following two reasons [146]. First, it is very burdensome to acquire exhaustive per-pixel labels for videos, i.e., as done in Network Dissection [18]. Second, it is not clear what an exhaustive list of possible concepts for spatiotemporal models would look like, as this would include all appearance-based concepts (e.g., objects, colors, etc), dynamic concepts (e.g., flickering lights, motion, etc), combinations of the two (e.g., objects with directed motion, and changing colors

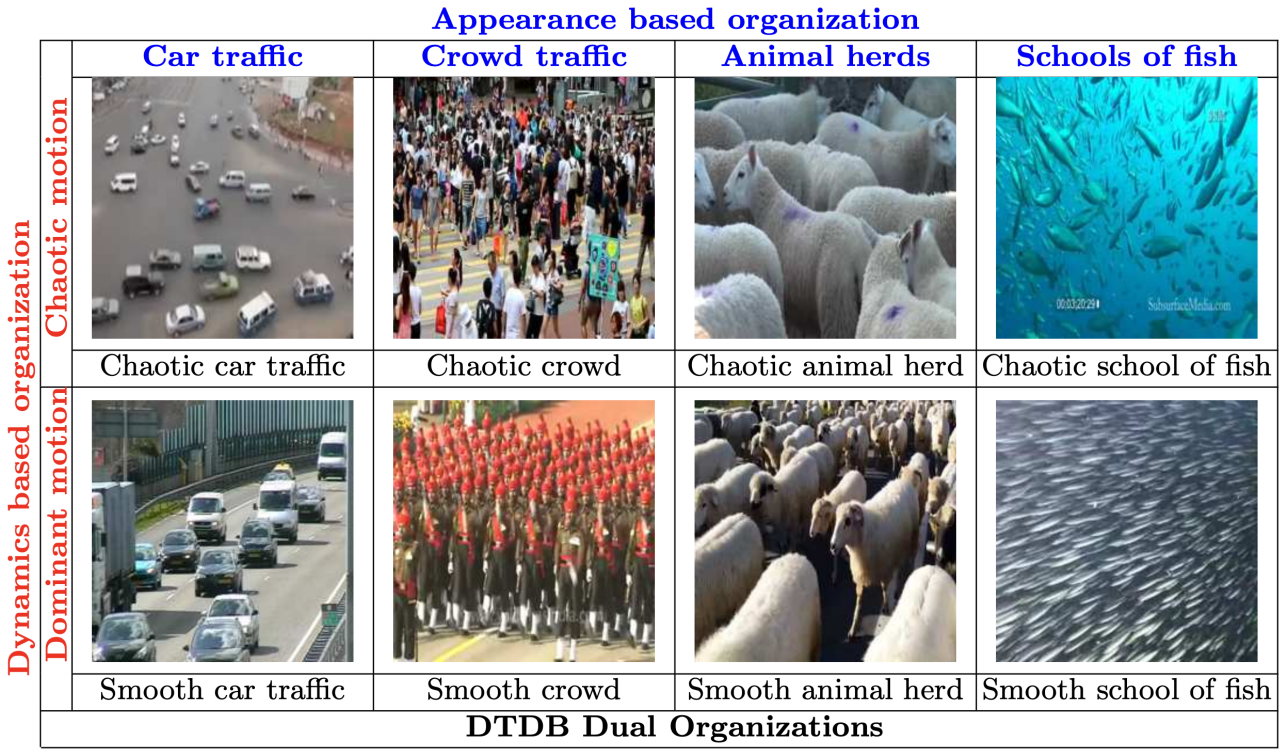


Figure 2.5: Examples from the Dynamic Texture DataBase (DTDB). DTDB contains videos that are labelled with both dynamic and appearance labels. The videos here are organized into dynamic classes along the row axis and appearance labels along the column axis. Figure reproduced with permission from [106].

with flicker), and more (e.g., positional concepts).

To this end, another chapter of this thesis (Chapter 4) proposes the first open-world concept discovery algorithm for video transformers, Video Transformer Concept Discovery (VTCD). The approach follows the three main steps of image-based concept-based interpretability as follows. First, each video is passed through the model and the features are clustered into tubelets (i.e., using SLIC). Then, the resulting dataset of tubelets are globally pooled and then clustered to produce concept centroids. Finally, after showing how existing concept-importance methods fail to robustly rank the importance of concepts within transformer heads, we propose Concept Randomized Importance Sampling (CRIS). CRIS randomly samples a large fraction (50%) of all concepts in the model and then evaluates the drop in performance. Sampling over thousands of iterations while tracking the heads that participate in the largest performance drops results in an accurate ranking of concepts within the heads of all transformer layers. Inspired by recent work on image model neurons [68], we further explore the idea of *Rosetta Concepts* for video, shared concepts across multiple models found by comparing the overlap of concept tubelets found by VTCD. Multiple shared computational processes are found to exist across video transformers trained for different tasks including: temporally and spatially invariant position concepts at lower layers, the emergence of object-centric concepts at middle layers, and spatio-temporal object-centric representations emerge at deeper layers. Finally, we propose two downstream

applications of their VTCD. First, we show that pruning the right heads of action classification model results in a 4.3% increase in accuracy while reducing FLOPs by 33%. Second, we leverage VTCD to perform zero-shot video-object segmentation (VOS) using models trained without per-pixel labels. This chapter lays the groundwork for future open-world video interpretability research.

2.2.7 Discussion

This section reviewed methods for concept-based interpretability, which aims to identify concepts in a model's intermediate representations and quantify the importance of the concept to the model output. This can be done in the closed world setting where a set of labelled concepts are known apriori, or in an open-world setting where concepts are discovered automatically.

These methods were reviewed according to the number of concepts targeted. At the low end of the spectrum, one or two concepts are measured with high fidelity. Several datasets have also been proposed that cover several concepts of interest [18, 106]. At the other end of the spectrum, concepts are discovered in the unsupervised setting using general clustering-based frameworks.

While the goal of concept-based interpretability is to decompose the entire model into concepts that are interpretable to humans, it is unclear whether this is possible. Indeed, there may exist concepts that are influential for the model's decision making process but are non-interpretable to humans. This could be because they are simply too high-bandwidth (humans have a limited number of subjects they can keep in their mind at a given time) or they are just unrecognizable (e.g., a specific set of frequencies or colors that are not distinguishable to the human eye).

While concept-based interpretability seems like the most promising direction for interpretability research, there exist many open problems which resulted in the main contributions of this thesis presented in the following sections.

2.3 Chapter summary

The preceding chapter provides a detailed background on interpretability methods for computer vision, dividing previous work into two main categories: pixel-based and concept-based methods.

The first category, pixel-based interpretability, aims to make a model's functions transparent by examining the influence of individual pixels. One major line of work is feature visualization, which searches for an input that maximally activates a component of the model, like a neuron or class output. These methods often use activation maximization, an optimization process that can require complex regularization to produce realistic and interpretable images. The other primary pixel-based approach is pixel attribution, which quantifies the contribution of each input pixel to the model's decision, typically resulting in a saliency map. This is achieved through several techniques, including gradient-

based methods like Layer-wise Relevance Propagation (LRP) and Class Activation Maps (CAM) , as well as occlusion-based methods like LIME and RISE, which systematically mask parts of an image to measure the corresponding drop in performance. However, the chapter notes that these pixel-level methods have significant limitations; visualizations can be unrealistic, and both visualization and attribution methods can be hard for humans to interpret without succumbing to confirmation bias.

The second category, concept-based interpretability, is presented as a solution to these issues by explaining a model's representations in terms of human-friendly 'concepts' that are grounded in sets of data exemplars. The chapter organizes these approaches along a spectrum based on the number of concepts analyzed. At one end are focused studies on one or two concepts, which sacrifice breadth for a highly specific measurement of how a model uses particular abstractions, such as shape versus texture or the encoding of absolute position information. At the other end are methods that analyze many concepts simultaneously. These can be 'closed-world,' relying on large, pre-labelled datasets of concepts like the Broden dataset to see which ones the model has learned, or 'open-world.' Open-world concept discovery methods operate without prior assumptions or labels, using unsupervised techniques like clustering or dictionary learning to automatically identify the concepts a model has formed on its own. The chapter notes that the application of these techniques to video models is a largely understudied topic, setting the stage for the contributions of this dissertation.

Chapter 3

Quantifying and learning static vs. dynamic information

3.1 Motivation

This chapter focuses on the problem of interpreting the information learned by deep neural networks (DNNs) trained for video understanding tasks. Interpreting deep spatiotemporal models is a largely understudied topic in computer vision despite their achieving state-of-the-art performance on video understanding tasks, such as action recognition [289] and video object segmentation [255]. These models are trained in an end-to-end fashion to learn discriminative static and dynamic features over space and time. Here, we use the term *static* to refer to attributes that can be extracted from a single image (e.g., color and texture) and the term *dynamic* to attributes that arise from consideration of multiple frames (e.g., motion and dynamic texture).

While this learning-based paradigm has led to great success across a wide range of tasks, the internal representations of the learned models remain largely opaque. This lack of explainability is unsatisfying from both scientific and application perspectives. From a scientific perspective, there is limited understanding of what information is driving the decision-making underlying the network output. Elucidating the decision-making process may yield directions to improve models. From an applications perspective, there have been multiple cases showing the ethical and damaging consequences of deploying opaque vision models, e.g., [35, 108]. Currently, however, the explainability of spatiotemporal models is under-explored [113]. Some evidence suggests that these models exhibit considerable bias toward static information, e.g., [50, 111, 124, 252]; therefore, an interesting question to answer about the representations in deep spatiotemporal models is: *How much static and dynamic information is being captured?* While a few video interpretation methods exist, they have various limitations, e.g., being primarily qualitative [81], using a certain dataset that prevents evaluating the effect of the training dataset [106] or using classification accuracy as a metric without quantifying a

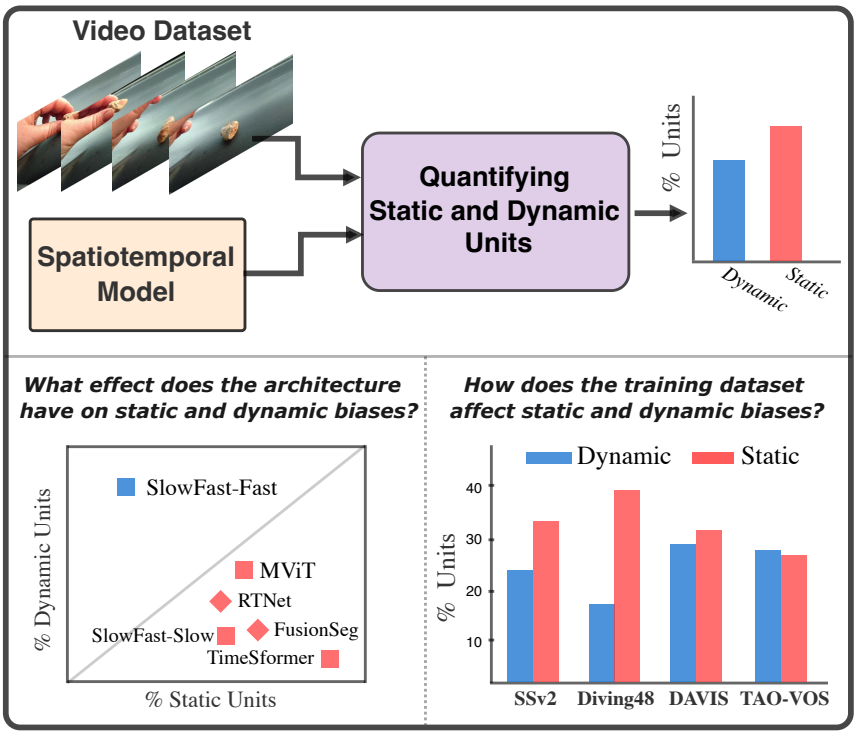


Figure 3.1: We introduce a general technique that, given a trained spatiotemporal model and a video dataset, can quantify the bias in any hidden representation within the model toward encoding static (red) or dynamic (blue) information. We use this technique to study action recognition (squares) and video segmentation (diamonds) and explore the effect of models and training datasets on static and dynamic biases.

model's *internal* representations [106, 124, 220].

In response, we present a quantitative paradigm for evaluating the extent that spatiotemporal models are biased toward static or dynamic information in their internal representations. We define bias toward a certain factor (dynamic or static) as the percentage of units (i.e., channels) within intermediate layers that encode that factor; see Figure 3.1 (top). Inspired by previous work [73,128], we propose a metric to estimate the amount of static vs. dynamic bias based on the *mutual information*, the amount of information one random variable contains about another, between sampled video pairs sharing these factors. We explore three common tasks to show the efficacy of our approach as a general tool for understanding spatiotemporal models, action recognition, automatic video object segmentation (AVOS) and video instance segmentation (VIS). We focus our study on answering the following five questions: (i) How do model architectures affect static and dynamic biases? (ii) How does the training dataset affect these biases? (iii) What role do units that jointly encode static and dynamic information play in relation to the architecture and dataset? (iv) When are statics and dynamics learned during training? (v) What impact do these biases have on performance? In regards to the last question, we demonstrate that controlling the static and dynamic biases in action recognition and AVOS can improve performance on tasks requiring dynamics.

Contributions. We make five main contributions: (i) We introduce a general method for quantifying

the static and dynamic bias contained in spatiotemporal models, including a novel sampling procedure to produce static and dynamic video pairs. (ii) We propose a technique for identifying units that jointly encode static and dynamic factors. (iii) Using the aforementioned techniques, we provide a unified study on three widely researched tasks, action recognition, AVOS and VIS, with a focus on the effect of architecture and training dataset on a model's static and dynamic biases; see Figure 3.1 (bottom). (iv) We propose StaticDropout, a semantically guided dropout technique for debiasing models from statics and toward dynamics that can improve model performance on datasets which require dynamics. (v) We demonstrate how a proper selection of the fusion and cross connection modules in AVOS architectures can guide a model to learn better dynamics and improve performance for the task of segmenting camouflaged entities. This work extends our previous work [147] by including a new task in our analysis (Video Instance Segmentation) 3.4.1.3, 3.4.2.3 and 3.4.3.3, the effect of training on static and dynamic biases 3.4.3, and introducing two methods for controlling the dynamic bias learned by action recognition 3.5.2 and video object segmentation models 3.5.3. Among other findings, we discover that all studied networks are heavily static biased, except for two-stream architectures with cross connections encouraging models to capture dynamics. We show that the majority of models converge to their final static and dynamic biases within the first half of training iterations. Additionally, we confirm that, contrary to previous beliefs [22, 160], the Diving48 [160] dataset is not dynamically biased and Something-Something-v2 (SSv2) [103] is better suited to evaluate a model's ability to capture dynamics.

3.2 Related work

3.2.1 Spatiotemporal models

Deep spatiotemporal models that learn discriminative features across space and time have proven effective for video understanding tasks [1, 289]. Extant models can be broadly categorized (agnostic of the downstream task) into: two-stream approaches that separately model motion and appearance features [38, 77, 132, 205, 288], 3D convolutions that jointly model motion and appearance [38], attention-based models with different forms of spatiotemporal data association [22, 205], models relying on recurrent neural networks [241] and hybrid models that combine elements of the aforementioned models [38, 205, 241]. Our approach to quantifying bias is not limited to the particulars of a model and is applicable to all extant and future models. We empirically demonstrate the flexibility of our approach by evaluating a diverse set of models.

3.2.2 Action recognition

3D convolutional networks are popular for learning spatiotemporal representations of videos for action recognition, e.g., [38, 109, 133, 237, 244]. Other work has considered two-stream architectures, where the dynamics were provided directly to one of the streams as optical flow, e.g., [79, 225]. Representative of the state-of-the-art with convolutional networks is SlowFast [77], which is a two-stream 3D CNN that only takes RGB videos as input. To encourage each stream to specialize in capturing predominately static or dynamic information, the temporal sampling rates of the inputs to each stream differ. Recently, attention based approaches have proven to be suited to both static and time-series visual data, including action recognition, with variants of the transformer architecture [22,75,191,248].

3.2.3 Video segmentation

Deep video segmentation approaches can be categorized into two categories [255]: (i) class agnostic, which are referred to as video object segmentation (VOS) and (ii) video semantic segmentation, which predict different semantic categories. Video semantic segmentation has been investigated with both instance-agnostic [88] and instance-aware [259] approaches, where the latter is referred to as video instance segmentation (VIS) [259]. Video object segmentation methods can be categorized into automatic, semi-automatic and interactive [255]. AVOS tries to segment the visual and motion salient objects in videos without a predefined mask initialization, while semi-automatic VOS requires mask initialization in the first frame to consequently track objects within a video. In this work, we mainly focus on AVOS approaches that segment salient objects in videos. We consider two-stream models that fuse motion and appearance features. We also investigate the effect of no cross connections [132] relative to both motion-to-appearance [288] or bidirectional [205] cross connections. Additionally, we explore video instance segmentation approaches that use raw images as input to demonstrate the versatility of our methodology across different video segmentation tasks.

3.2.4 Interpretability of spatiotemporal models

Limited work has been dedicated to the interpretability of spatiotemporal models. Several efforts predicate model interpretation on proxy tasks, e.g., dynamic texture recognition [106] or future frame selection [96]. These approaches do not interpret the learned representations in the intermediate layers and in some cases require training to be performed on specific datasets [106]. Work also has appeared that introduced a dataset to completely decouple static and dynamic information, but used it only to examine overall architecture performance on action recognition and did not examine intermediate representations [124]. Other work focused on understanding latent representations in spatiotemporal models either mostly concerned qualitative visualization [81] or a specific architecture type [282]. A related task is understanding the scene representation bias of action recognition datasets [158, 160].

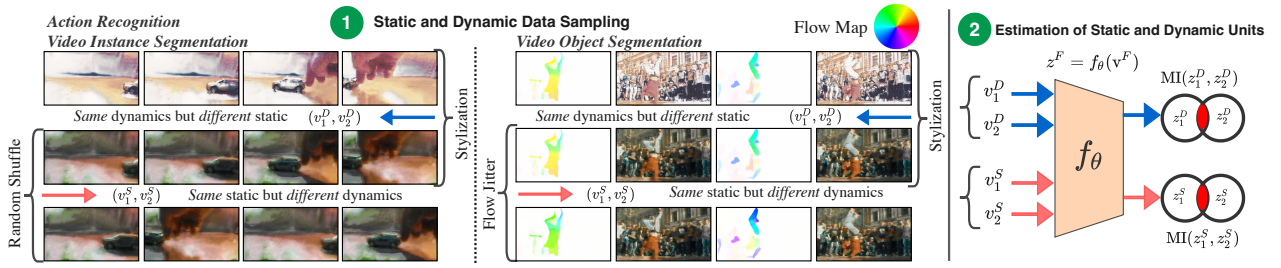


Figure 3.2: Overview of our methodology for analyzing bias toward static or dynamic information. We measure the dynamic and static biases in deep spatiotemporal models for three tasks: action recognition, automatic video object segmentation and video instance segmentation. **(1)** We sample video pairs that share either *static*, (v_1^S, v_2^S), or *dynamic*, (v_1^D, v_2^D), information using video stylization [239] and frame shuffling or optical flow jitter (flow visualized in RGB format). **(2)** Given a pretrained model, f_θ, we compute the mutual information (MI) between intermediate representations of video pairs, z^F, to assess the model's bias toward either factor on a per-layer, l, or per-channel (i.e., unit) basis.

However, these efforts did not focus on the effect of different architectural inductive biases on the learned intermediate representations. Our proposed interpretability technique is the first to *quantify* static and dynamic biases on *intermediate* representations learned in off-the-shelf models for multiple video-based tasks. Most prior efforts focused on a single task, and studied either datasets [160] or architectures [81,169]. In contrast, our unified study covers seven datasets and dozens of architectures on three different tasks, i.e., action recognition, AVOS and VIS.

3.3 Method

We introduce an approach to quantify the number of units encoding static and dynamic information in spatiotemporal models; for an overview, see Figure 3.2. Our approach consists of two main steps. First, given a pretrained spatiotemporal model, we sample static and dynamic video pairs (Section 3.3.1). Second, we use the static and dynamic pairs to estimate the number of units in the model encoding each factor based on the mutual information shared between the pairs (Section 3.3.2).

3.3.1 Sampling static and dynamic pairs

Why static and dynamic? We define static as 'information arising from single frames' and dynamic as 'information arising from the consideration of multiple frames'. The main alternative attribute to dynamics that we considered is 'image motion' (i.e., trackable points or regions), but 'motion' is a subset of dynamic information [62,260] (e.g., stationary flashing lights have dynamics but no motion). Thus, we consider dynamics over motion because it encompasses a wider range of visual phenomena. In complement, we choose the term 'static' over the possible alternative 'appearance', because dynamics also can provide appearance information, e.g., object contours, even if camouflaged in a single frame,

can be revealed through motion. We note that *shape* is a confounder between the static and dynamic factors: It is not feasible to completely disentangle static and dynamic as perceiving the motion of an object necessarily provides localized boundary information of that object. Despite this confounder, video stylization still provides notable differences as there is less mutual appearance information post-stylization (Section 5.4). For our metric, we produce video pairs that contain the same static information and perturbed dynamics, or vice versa, with the end goal of analyzing models trained on large-scale real-world datasets. We now detail our static and dynamic sampling techniques, as visualized in Figure 3.2 (panel 1).

Action recognition. The action recognition models we consider take in multiple frames (four to thirty-two). To construct video pairs with the *same* dynamics but *different* static information (i.e., *dynamic pairs*), we consider the same video but with two *different* video styles. For video stylization, we use a recent video stylization method (with four possible styles) that perturbs static attributes like color, pixel intensity and texture [239], but has less temporal artifacts (e.g., flickering) than stylization methods that consider each image independently [122]. These video pairs will contain objects and scenes that have identical dynamics, but have perturbed static information. To construct pairs with the *same* static information but *different* dynamics (i.e., *static pairs*), we take two videos of the same style, but randomly *shuffle* the frames along the temporal axis; see Figure 3.2 (panel 1, left). In this case, the temporal correlations are altered while the static (i.e., per-frame) information remains identical.

Video object segmentation. The AVOS models considered [132, 205, 288] take a single RGB frame and an optical flow frame as input to the appearance and motion streams, resp.; see Figure 3.2 (panel 1, right). Therefore, we apply an alternative method to frame shuffling to obtain the *static* pairs. For the *static* pair, we use RGB images with the *same* style but alter the dynamics by jittering the optical flow. To do this, we represent flow with a color coding [13] and then randomly perturb the hue and saturation which correspond to the direction and magnitude, respectively. For the *dynamic* pairs, we use the *same* optical flow but a *different* image style. For creating stylized images, we use the same video stylization techniques noted above for action recognition [239], and then sample frames from the generated video.

Video instance segmentation. The inputs to the VIS models considered [259] take single stream, multi-frame RGB inputs. Therefore, we select static and dynamic pairs similar to action recognition models, i.e., shuffling and stylization for static and dynamic pairs, respectively.

3.3.2 Estimating static and dynamic units

We seek to quantify the number of units (i.e., *channels*) in a layer encoding *static* or *dynamic* information and the extent to which individual units perform static, dynamic or joint encodings.

Inspired by recent work that focused on single images [73,128], we use a mutual information estimator to measure the information shared between video pairs.

Layer-wise metric. Given a pre-trained network, f_θ, and a pair of videos, v_1^F and v_2^F, that share the semantic factor F (i.e., *static* or *dynamic*), we compute the features for an intermediate layer l as $z_1^F = f_\theta^l(v_1^F)$ and $z_2^F = f_\theta^l(v_2^F)$ (omitting the l on z to reduce notation). We use $z_1^F(i), z_2^F(i)$ to denote the i^{th} unit (i.e., channel) in N^l dimensional features after a global average pooling layer. Units biased toward the *static* factor will result in a higher correlation among *static* pairs than the *dynamic* pairs and vice versa. Under the assumption that units in the intermediate representation, $z_1^F(i)$ and $z_2^F(i)$, across the dataset are jointly Gaussian, the correlation coefficient can be used as a lower bound on mutual information [150], as used in previous work [73,128]. The bound tightness of this estimator is discussed in detail by Foster et al. [87]. The number of units encoding factor F, N_F, is obtained by computing the correlation coefficient, S_F, over all N^l channels between all video pairs z_1^F, z_2^F, as

$$N_F = \frac{\exp\left(S_F\right)}{\sum_{k=0}^{K} \exp\left(S_k\right)} \cdot N^l, S_F = \sum_{i=1}^{N^l} \frac{\text{Cov}(z_1^F(i), z_2^F(i))}{\sqrt{\text{Var}(z_1^F(i))\ \text{Var}(z_2^F(i))}}, \tag{3.1}$$

where we multiply the Softmax by the number of units in that layer, N^l, to compute the number of units encoding the semantic factor F relative to the other factors considered and $K = \{\text{static}, \text{dynamic}, \text{identical}\}$. In addition to *static* and *dynamic*, we consider a third factor in (3.1), the *identical* factor, where the video pairs have the same static and dynamic factors (i.e., same video, style, frame ordering and optical flow). This baseline factor is the correlation between the model's encoding of the same videos, that gives $S_{\text{Identical}} = 1$ for all layers.

Unit-wise metric. The correlation coefficient, S_F, estimates the relative amount of static and dynamic information over all units in a particular layer; note the pooling done by the summation *before* the Softmax in the layer-wise metric, (3.1). However, it is also desirable to measure static and dynamic information contained in each individual channel. This measurement allows for a more fine-grained analysis of how many channels (i.e., units) encode a factor F above a certain threshold, as well as identify any joint or residual (i.e., non-dynamic or static) units. Thus, we categorize each unit based on how much information (i.e., static vs. dynamic) is encoded, whether any units jointly encode both factors or if there are units that do not correlate with either type of information. We measure the amount of static and dynamic information encoded in each unit $i \in 1, \ldots, N^l$ as

$$s_F^i = \frac{\text{Covariance}(z_1^F(i), z_2^F(i))}{\sqrt{\text{Variance}(z_1^F(i))\text{Variance}(z_2^F(i))}}, \tag{3.2}$$

where each s_F^i is the information of semantic factor F in unit i. Given these individual correlations, we calculate the individual factors by excluding the use of a Softmax and simply threshold the correlation

for each factor with a constant parameter, λ, to yield our unit-wise metrics as

$$N_{\mathrm{F}} = \sum_{i=1}^{N^l} \mathbb{1}[s_F^i > \lambda \wedge s_k^i < \lambda \forall k \in K, k \neq F]$$
$$N_{\mathrm{J}} = \sum_{i=1}^{N^l} \mathbb{1}[s_F^i > \lambda \forall F \in K],\ N_{\mathrm{R}} = \sum_{i=1}^{N^l} \mathbb{1}[s_F^i < \lambda \forall F \in K], \tag{3.3}$$

where $K = \{\text{static}, \text{dynamic}\}$, N_{J} indicates units jointly encoding both and N_{R} are residual units not correlating with these factors under a threshold, λ. Note that we assign units to either joint, dynamic, static or residual and do not allow for an overlap to occur. This approach allows us to investigate the existence of units that jointly encode static and dynamic factors. For all experiments, we set $\lambda = 0.5$ since it is halfway between *no* and *full* positive correlation.

A key consideration in our approach is the distinction between what information is present in a representation versus what information is ultimately used by the model to make a prediction. Our method, being based on correlations, primarily addresses the former by identifying which units covary with static or dynamic factors. This approach, however, may be blind to which of these encoded factors are utilized by downstream layers and ultimately the model decision. For instance, a unit could exhibit equal covariance with both static and dynamic information, yet the model might learn to ignore one of these aspects when computing its final output. While a full causal analysis is beyond the scope of this work, our neuron removal experiments in Section 3.5.1 begin to bridge this gap. By systematically ablating units identified as predominantly static or dynamic and measuring the subsequent drop in performance, we move from a correlational to a more causal understanding, providing evidence that the information encoded in these units is not only present but also functionally critical to the model's decision-making process.

We note that the term 'bias' also has been used to describe the tendancy of a model to make predictions with specific qualities (e.g., shape vs. texture labels [93]). In our work, we define 'bias' towards a specific factor (i.e., static or dynamic) to be the layer (or unit) encoding more relative mutual information of that factor than another (as used in [128]).

3.3.3 Model Biasing

Following an analysis of static and dynamic biases in various architectures, tasks, and datasets, we then aim to control a model's bias to learn dynamic information for two different tasks in Section 3.5. First, we introduce StaticDropout, a new dropout mechanism that encourages action recognition classification models to encode more dynamic information. More specifically, we use Equation 3.2 to calculate the channels in an action recognition model that encode static information, and drop them out every few iterations over the course of training. We show that increasing the number of units

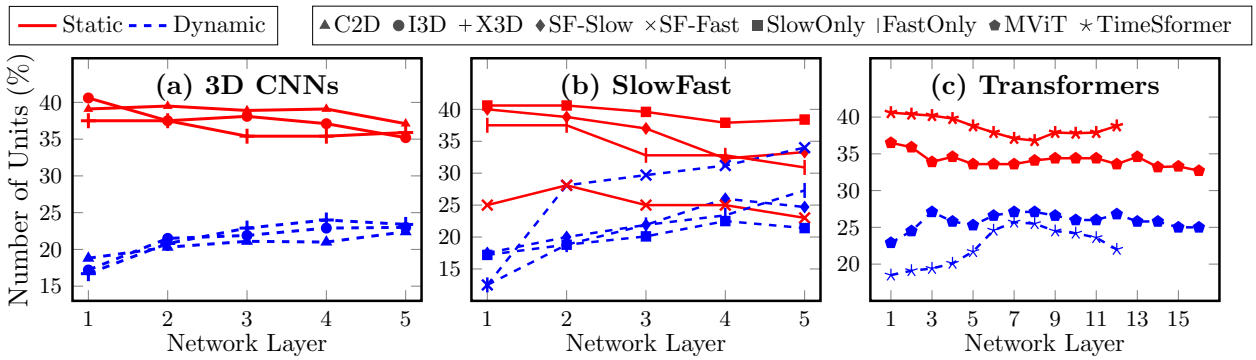


Figure 3.3: Layer-wise analyses using the layer-wise metric, (Equation 3.1), on action recognition networks trained on Kinetics-400 [38] for: (a) single stream 3D CNNs, (b) SlowFast variants and (c) transformer variants. SF-Slow and SF-Fast denote the representation taken before the fusion layer from the slow and fast branches, respectively.

dropped corresponds with more dynamic information being encoded in the model.

For video object segmentation, we provide a formal analysis on the different types of fusion and cross connection layers found in two-stream models. We propose a simple combination of layers that we empirically show to encode more dynamic information, using Equation 3.2. Moreover, we show this simple improvement can increase model performance on dynamic centric tasks (e.g., camouflaged object segmentation [152]) by up to 10% mean Intersection over Union (mIoU).

3.4 Empirical evaluation

We choose the tasks of action recognition, AVOS, and VIS to demonstrate the generality of our approach. They differ in their semantics (i.e., multi-class vs. binary vs. instance-based), labelling (i.e., video-level vs. pixel-level), and input types (multi-frame images vs. single frame and optical flow). We explore five main research questions and show the corresponding results with respect to our quantitative techniques for both tasks: (i) What is the effect of the model architecture on the *static* and *dynamic* biases (Section 3.4.1)? (ii) What effect does the training dataset have on *static* and *dynamic* biases (Section 3.4.2)? (iii) What are the characteristics of jointly encoding units in relation to model architectures and datasets (Section 3.4.1 and 3.4.2)? (iv) When are static and dynamic biases learned during training (Section 3.4.3)? (v) What is the effect of static and dynamic biased units on performance (Section 3.5.1)? Finally, we demonstrate two approaches where we use our insights to improve the performance of action recognition (Section 3.5.2) and AVOS models (Section 3.5.3).

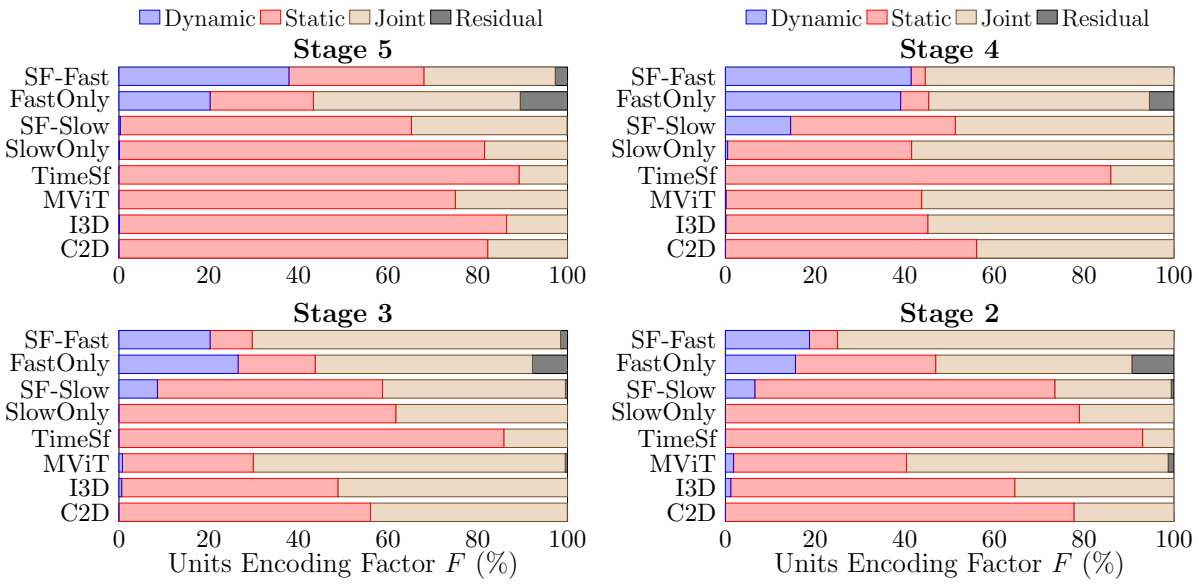


Figure 3.4: Estimates of the dynamic, static, joint and residual units using the unit-wise metric, (Equation 3.3), on action recognition networks trained on Kinetics-400 [38].

3.4.1 Model architectures

3.4.1.1 Action recognition

Architectures. We study three types of models with respect to their static and dynamic biases: (i) single stream 3D CNNs (i.e., C2D [256], I3D [38] and X3D [76] models), (ii) SlowFast [77] variations, where the individual streams are referred to as the SlowOnly and FastOnly models and (iii) transformer-based architectures [22, 75]. For all models, the number of frames and sampling rate is (8×8), except for the FastOnly network (32×2), MViT (16×4), and TimeSformer (8×32). Note, we found through experimentation on the FastOnly model, that ablating the total number of frames, between four to 64, did not have a noticeable impact on the static vs. dynamic biases. To identify the static and dynamic units of all models, we generate the Stylized ActivityNet [74] validation set and use it for sampling *static* and *dynamic* pairs. We choose this dataset since the action distribution is similar to Kinetics-400, yet much smaller in size making it memory efficient when computing our metrics.

Implementation details. All models were taken from the SlowFast [77] repository[1] except the TimeSformer [22], which has its own codebase[2]. All model weights trained on Kinetics-400 [38] and Something-Something-v2 [103] (SSv2) are taken directly from the SlowFast repository, except for the FastOnly model which we train ourselves. All training strategies are chosen to be similar to the original ones found in the SlowFast repository. We use decaying learning rate protocols to ensure that

[1] `https://github.com/facebookresearch/SlowFast`
[2] `https://github.com/facebookresearch/TimeSformer`

models have fully converged. The FastOnly model is trained on Kinetics-400 for 40 epochs with SGD, a weight decay of 1e-4, a batch size of 32 and a base learning rate of 0.03 that decreases by a factor of 10 at epochs 15, 30 and 35. On SSv2, the FastOnly model is trained for 25 epochs with SGD, weight decay of 1e-4, a batch size of 32 and a base learning rate that is decreased by a factor of 10 at epochs 10 and 20. On Diving48 [160] the FastOnly model is trained for 100 epochs with SGD, weight decay of 1e-4, a batch size of 32 and a base learning rate of 0.0375 that decreases by a factor of 10 at epochs 40, 60 and 80. The SlowOnly model is trained on Diving48 for 100 epochs with SGD, weight decay of 1e-4, a batch size of 32 and a base learning rate of 0.00375 that decreases by a factor of 10 at epochs 40, 60 and 80. All models trained with temporal frame shuffling (see Section 3.4.2.1) incur the same hyperparameters as their unshuffled counterparts.

Layer-wise analysis. Figure 3.3 shows the layer-wise analysis, (Equation 3.1), for various action recognition models. The transformers are measured at every layer and the convolutional architectures are measured at five 'stages', corresponding to ResNet blocks [110]. Interestingly, all single stream networks are biased toward static information at all layers even though the video frames of the static pairs are randomly shuffled. Most of the 3D CNNs (e.g., I3D and SlowOnly) have a similar percentage of dynamic units as the C2D network, suggesting that these models do not capture sufficiently complex dynamic representations. While the static and dynamic biases in 3D CNNs do not fluctuate significantly over the different layers, the transformer-based architectures encode an increasing amount of dynamic information up until about halfway through the model, at which point the pattern tapers off and even reverses slightly.

For the SlowFast network, we measure the biases on the representations for the slow and the fast branches before fusion of the features. As shown in Figure 3.3 (b), the fast branch has a nontrivial number of dynamic units. A key component of the SlowFast network is the fusion branches that aim to transfer information from the fast branch to the slow branch. This transfer is accomplished by concatenating the slow and fast features followed by a time-strided convolution. Comparing the dynamic and static between the SlowOnly and SlowFast (slow) branch can reveal whether dynamic information is transferred between the pathways. The addition of the fast branch increases the dynamic units in the slow pathway by 3.3% as early as stage two, showing the ability of the two-stream architecture to capture dynamics.

Unit-wise analysis. We now examine individual units using our unit-wise metric, (Equation 3.3), with $\lambda = 0.5$ and report the results in Figure 3.4. Considering the final representation before the fully connected layer (i.e., stage 5), all single stream models, other than FastOnly, contain mainly *static* and *joint* units. In contrast, the FastOnly model and SlowFast-Fast branch produce a nontrivial number of *dynamic* units. Another finding consistent with the results from Figure 3.3, is that the Fast model extracts more dynamic information *when trained jointly with the Slow branch* than when trained independently. Studying the earlier layers (Figure 3.4) confirms the previous findings that

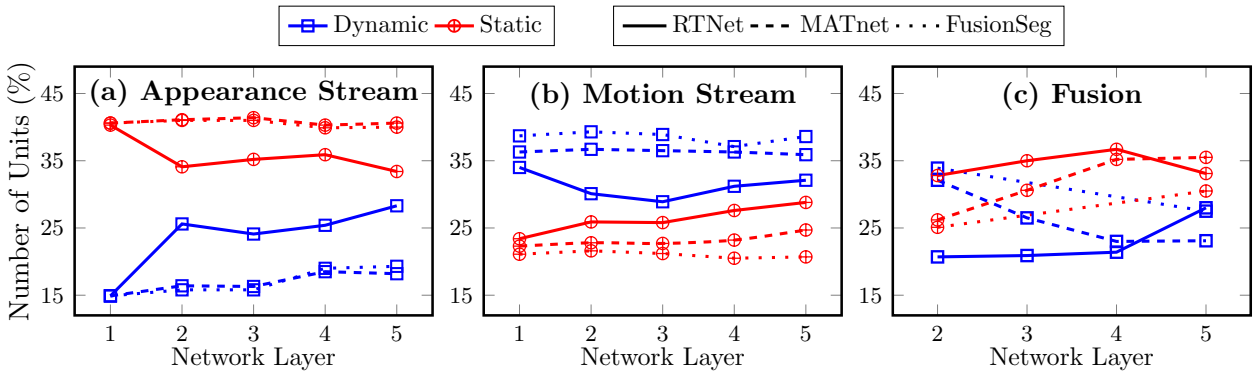


Figure 3.5: Layer-wise analysis on off-the-shelf AVOS networks. Encoding of dynamic and static factors for motion, appearance streams and fusion layers in FusionSeg [132], MATNet [288] and RTNet [205] using the layer-wise metric, (Equation 3.1). Fusion layers are mostly biased toward the static factor.

the SlowFast model's cross connections are successful at transferring dynamic information from the fast branch to the slow branch throughout the network. SF-Slow (i.e., the slow branch) produces a small but notable number of dynamic units early in the network. Interestingly, SF-Fast (i.e., the fast branch) gradually produces more dynamic units as the representation flows deeper through the network. All single stream models other than the FastOnly model produces mainly static and joint units in all layers. The MViT and TimeSformer produce mainly static and joint units; however, the ratio of these units changes through the layers of the MViT model while it remains stable through the layers of the TimeSformer.

3.4.1.2 Automatic video object segmentation

Architectures. We study the dynamic and static biases of two-stream fusion AVOS models that take optical flow and an RGB image as input, with different types of cross connections: (i) FusionSeg [132] with no cross connections, (ii) MATNet [288] with motion-to-appearance cross connections and (iii) RTNet [205] with bidirectional cross connections. We concentrate on two-stream models because they currently are the strongest AVOS performers. For both MATNet [288] and RTNet [205], we use the models provided by the authors without further fine-tuning. We use a stylized version of DAVIS16 in our analysis to evaluate the static and dynamic biases for the previous models, with stylization according to Section 3.3.1. In the case of both motion and appearance streams, we analyse features after cross connections, if present. Similarly, we analyze the features extracted after the fusion layers in all models, if present.

Implementation details. For a fair comparison with MATNet and RTNet, that fuse motion and appearance features in the intermediate representations, we use a modified version of FusionSeg [132] trained on DAVIS16 [192] in our analysis. Our modified model follows an encoder-decoder ap-

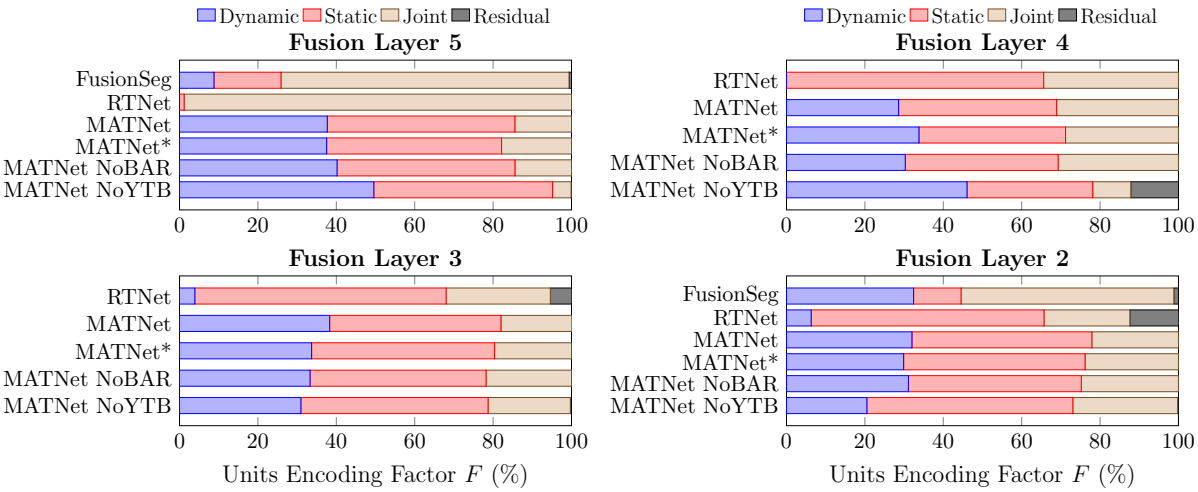


Figure 3.6: Unit-wise analysis on off-the-shelf AVOS networks. Individual units analysis for the three models for fusion layer 5 using the unit-wise metric, (Equation 3.3). MATNet has the largest number of dynamic units.

proach [44], resulting in two fusion layers at stages two and five. Our FusionSeg model is trained with a batch size of eight, using SGD with learning rate 0.001, a momentum of 0.9, a "poly" learning rate policy using a power of 0.9 and weight decay 1×10^{-4}. As we want to isolate the effect of training datasets, we do not perform pretraining with the motion stream, as proposed in the original paper [132]. The MATNet variants are trained with two GPUs in parallel with batch size six (the original MATNet used a batch size of two and a single GPU). For the rest of the hyperparameters and training procedure, we follow the original work [288]. We denote the original model provided by the authors as "MATNet", while our reproduction of MATNet as "MATNet*". We analyze MATNet trained only on DAVIS16 (i.e., without any Youtube-VOS data), which we call "MATNet NoYTB" and also MATNet trained without its boundary-aware refinement module and boundary loss, denoted as "MATNet NoBAR".

Layer-wise analysis. Figure 3.5, shows the layer-wise analysis for the motion and appearance streams as well as the fusion layers according to our layer-wise metric, (Equation 3.1). Similar to our finding with the action recognition models in Section 3.4.1.1, the majority of the video object segmentation models are biased toward the *static* factor in the fusion layers (i.e., fusion layers three, four and five). We observe an increase in the dynamic bias in the appearance stream as we go deeper in the network, especially for RTNet. In contrast, the static and dynamic biases in the motion streams of both FusionSeg and MATNet are somewhat consistent throughout layers. Interestingly, in RTNet, the *static* bias increases as the representation goes deeper in the network. This result likely stems from the bidirectional cross-connections in RTNet.

Unit-wise analysis. The individual unit analysis for these models obtained using our unit-wise metric, (Equation 3.3), with $\lambda = 0.5$ is shown in Figure 3.6. Looking at the final representation of the

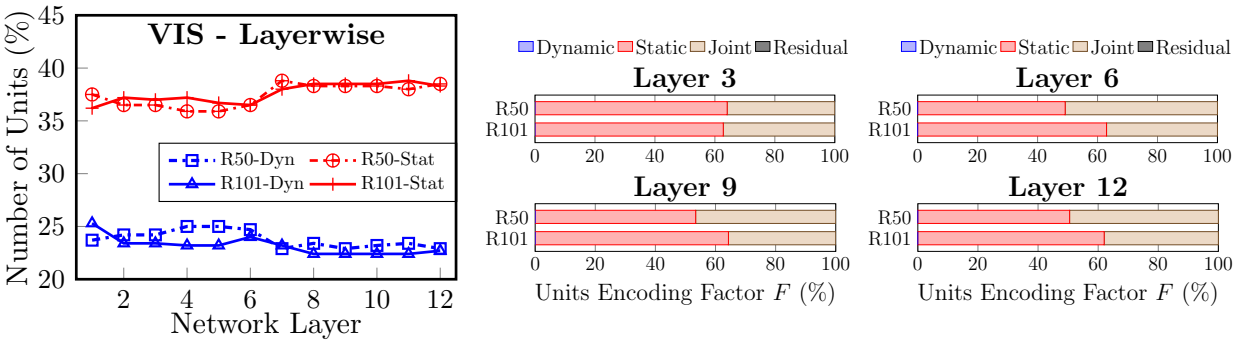


Figure 3.7: Layer and unit-wise analysis on off-the-shelf state-of-the-art VIS models. **Left**: Encoding of dynamic and static factors for motion, appearance streams and fusion layers in VisTR-R50 and VisTR-101 [259] using the layer-wise metric, (Equation 3.1). All layers are biased toward the static factor. **Right**: Individual units analysis for the two VisTR variants using the unit-wise metric, (Equation 3.3).

models (i.e., fusion layer 5), we observe that MATNet has more dynamic and static units compared to RTNet and FusionSeg, which both show a greater number of jointly encoding units. This pattern indicates that cross connections, as present in MATNet, can lead to an increase in the specialized units that encode the static and dynamic factors in the late fusion layers (here we define 'specialized' as simply a unit's dedication to encoding a specific factor). The "MATNet NoBAR" and "MATNet NoYTB" results are consistent, and confirm that the source behind such an increase is not the BAR module or YTB training. The earlier fusion layers in Figure 3.6 also show that MATNet captures the most dynamic units. In fusion layer two, FusionSeg appears on par with MATNet in terms of dynamic units, but has fewer static units and more joint units. In comparison, RTNet tends to have the most unbalanced units of all three models, which become skewed toward joint encoding units in the late fusion layers. These patterns show that over all fusion layers, MATNet generally has a more balanced ratio of dynamic and static units and more dynamic units than other models. It also shows that MATNet trained on solely DAVIS'16 exhibits a similar pattern of capturing more dynamic units than other models, but has less dynamic units in the earlier layers, unlike the original MATNet. This result suggests models with cross connections that are not pretrained on saliency segmentation datasets are beneficial to the model's encoding capabilities when the goal is to capture dynamics, e.g., camouflage object segmentation.

3.4.1.3 Video instance segmentation

Architectures. For the VIS task, we conduct our analysis on two versions of the state-of-the-art VisTR [259] model. VisTR is an end-to-end trained encoder-decoder transformer based architecture with a 2D CNN backbone used for the initial feature extraction. Both the encoder and decoder have six layers for which we compute the static and dynamic biases.

Implementation details. All pre-trained models are retrieved from the original repository [259] with no training done on our end. All VIS models are trained on the YouTube-VIS [268] dataset. We use the same 2D CNN backbones as per the original paper [259], either a ResNet50 or ResNet101. We perform the static and dynamic estimation using the stylized DAVIS16 dataset.

Layer-wise analysis. Figure 3.7 (left) shows the results of our layer-wise analysis on two variants of VisTR [259]. It is observed that both models are strongly biased toward the static factor. This bias toward static information increases later in the network, e.g., at the first transformer layer the ResNet101 variant has static and dynamic biases of 36.2% and 25.3%, respectively, and at the last layer has biases of 38.3% and 22.7%, respectively. This result likely stems due to the nature of the VIS task that requires tracking and object-level matching across frames.

Unit-wise analysis. Figure 3.7 (right) shows the results of our layer-wise and per-unit analysis on two variants of VisTR [259]. Notably, VisTR with either backbone produces solely static and joint units, however, the ratio of these units differ at various layers of the models. The ResNet50 backbone contains about twice the number of static units compared with joint units for the first three layers, and then converges to an even ratio thereafter. Meanwhile, the ResNet101 backbone contains a larger number of static units throughout the entire architecture. This suggests that the 2D backbone plays a role in the type of information captured for spatiotemporal models and that deeper backbones may capture more static information.

3.4.1.4 Summary and shared insights

We showed that most of the examined state-of-the-art models for all tasks are biased toward encoding static information. We also demonstrated the efficacy of two-stream models with motion-to-appearance [288] (fast-to-slow [77]) cross connections to enable greater encoding of dynamic information. For the VIS task, we observed that solely static and joint units are produced in the architectures analyzed, however the 2D backbone can influence the ratio between these units. Finally, we documented that the final layers of dynamic biased models are capable of producing more specialized dynamic units compared to the joint units produced by static biased models.

3.4.2 Training datasets

3.4.2.1 Action recognition

Datasets. With the knowledge that action recognition models often use static context biases in the data to make predictions [50, 61, 124], we consider datasets in the following evaluations which were designed with the goal of benchmarking a model's ability to capture dynamic information. Two popular datasets of this type are Something-Something-v2 [103] (SSv2) and Diving48 [160]. SSv2 is

Dataset	SlowOnly		FastOnly	
	Dyn.(%)	Stat.(%)	Dyn.(%)	Stat.(%)
Kinetics	21.4	38.4	27.3	30.9
Diving48	23.1	34.0	23.8	27.3
SSv2	23.6	32.6	28.5	23.0

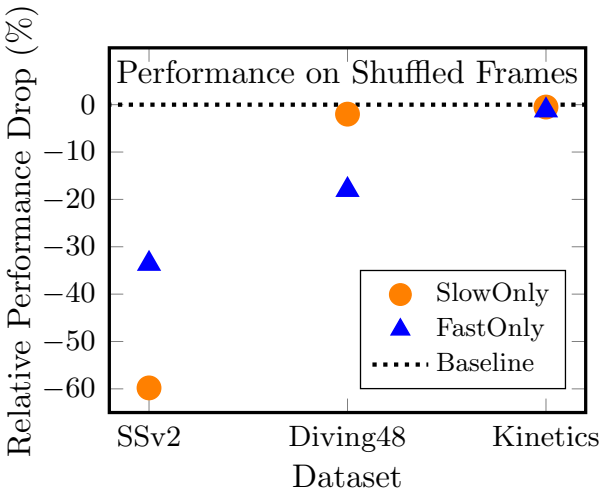


Figure 3.8: Analyses of biases of action recognition datasets. **Left:** *Dynamic* and *static* biases using the layer-wise metric, (Equation 3.1), for models trained on Kinetics-400 [38], Diving48 [160] and SSv2 [103]. **Right:** Relative drop in Top 1 Acc (%) for the SlowOnly and FastOnly models trained with shuffled frames with respect to standard training (i.e., baseline).

a fine-grained ego-centric dataset with 174 classes and over 30,000 unique objects. Notably, different actions in SSv2 include similar appearance but different motions, e.g., the classes 'moving something from right-to-left' and 'moving something from left-to-right'. Diving48 [160] was created to be "a dataset with no significant biases toward static or short-term motion representations, so that the capability of models to capture long-term dynamics information could be evaluated" [159]. All actions are a particular type of dive and differ by only a single rotation or flip. We compare Kinetics-400, Diving48 and SSv2 to determine the extent that each dataset requires dynamics for action recognition.

Dataset bias. We use the layer-wise metric, (Equation 3.1), to estimate the static and dynamic units captured in the last layer of two models trained on the three datasets, as shown in the table of Figure 3.8 (left). We generate Stylized SSv2 and Stylized Diving48 to produce the static and dynamic estimates (and continue using Stylized ActivityNet for Kinetics-400 trained models). We measure the last layer, as the final prediction is made directly from it and thus is most representative of what information the model uses for the final prediction. The SlowOnly and FastOnly architectures follow a similar pattern to that found in Section 3.4.1, with the FastOnly consistently capturing more dynamic information. Surprisingly, models trained on Diving48 capture a similar amount of dynamics compared to Kinetics. These results may seem curious at first, as it seems unlikely that models could perform well on Diving48 without dynamic information.

To further understand this result, we conduct a simple experiment, where the model only has static information to learn from. As discussed in Section 3.3.1, frame-shuffled videos will have the same static information as a non-shuffled input, but the temporal correlations, and hence dynamic information, will be corrupted. This forces the model to focus on static information for classification. We compare the Top-1 validation accuracy of models trained and validated on shuffled frames to that of models

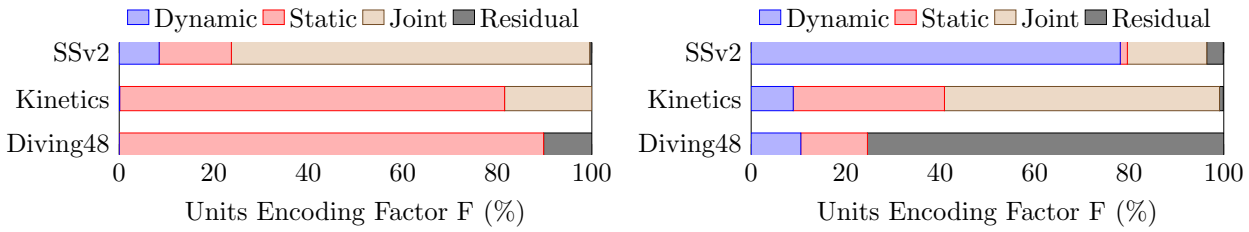


Figure 3.9: Estimating the dynamic, static, joint and residual units using the unit-wise metric, (Equation 3.3), for the SlowOnly (**left**) and FastOnly (**right**) models on Kinetics-400 [38], Diving48 [160] and SSv2 [103]. Dynamic units arise from dynamic-biased models (e.g., FastOnly) and residual units from training on Diving48.

with standard training. Figure 3.8 (right) shows the results of the SlowOnly and FastOnly networks on Diving48, SSv2 and Kinetics-400, in terms of the relative performance on shuffled frames compared to unshuffled. For a fair comparison, we initialize all models from Kinetics-400. Both models show strong relative performance when trained on shuffled videos for Diving48 and Kinetics-400; however, for SSv2 the classification performance is decreased to a greater extent when trained on shuffled frames. These results show that SSv2 is a better alternative for temporally benchmarking networks.

Figure 3.9 shows the individual units (from the last layer) for two models (one static biased, SlowOnly, and one dynamic biased, FastOnly) on Kinetics-400, Diving48 and SSv2. The SlowOnly model trained on Kinetics-400 contains only static and joint units; however, when trained on Diving48 or SSv2, both residual and dynamic units emerge, demonstrating the impact of the training dataset on producing specialized dynamic units. Unlike the SlowOnly model, the FastOnly model contains many dynamic units trained on any dataset, showing the efficacy of the architecture for producing specialized dynamic units. Interestingly, each dataset is unique in the type of units that emerge. Diving48 produces residual units, suggesting there are other factors at play beyond dynamic and static information. On the other hand, SSv2 produces the most dynamic units for both models.

3.4.2.2 Automatic video object segmentation

Datasets. We study the impact of the following four video segmentation datasets on a model's static and dynamic biases: DAVIS16 [192], Weakly Labelled ImageNet VID [132], YouTube-VOS [268] and TAO-VOS [250]. DAVIS16 [192] is the most widely used benchmark for automatic VOS, with 50 short-temporal extent sequences of two to four seconds and 3455 manually annotated frames. ImageNet VID [132] contains 3251 weakly labelled videos and was used in previous work to pretrain a model's motion stream [132]; in contrast, we use it as a general training dataset (i.e., beyond just for motion streams). YouTube-VOS [268] is another widely used AVOS dataset with 3471 videos in the training set, which is usually combined with DAVIS dataset following [288] to end up with 14K training images. We assess the two training datasets separately and evaluate how they affect the static and dynamic biases. Finally, TAO-VOS [250] contains 626 relatively long videos (36 seconds on average) that

Dataset	Fusion Layer 5		Fusion Layer 2	
	Dyn.(%)	Stat.(%)	Dyn.(%)	Stat.(%)
DAVIS	27.8	30.1	34.0	25.9
YouTube-VOS	25.2	34.2	33.5	24.9
ImageNetVID	26.4	33.1	33.0	24.6
TAO-VOS	26.4	25.8	33.7	23.2

Table 3.1: Biases of video object segmentation datasets using the layer-wise metric, (Equation 3.1), for FusionSeg's fusion layers five and two, trained on DAVIS16 [192], YouTube-VOS [268], ImageNetVID [132] and TAO-VOS [250].

are annotated in a hybrid fashion between manually and weakly labelled frames, resulting in 74,187 frames. We convert the annotations to exclude instances and instead consider foreground/background annotations only.

Dataset bias. We train our modified FusionSeg with two fusion layers (layers two and five) on the four datasets. We compute the static and dynamic biases for the training datasets using the layer-wise metric, (3.1), and report the results in Table 3.1. The model trained on TAO-VOS has the least amount of static bias out of all four datasets. However, the datasets do not differ much in their dynamic bias.

We analyse the datasets in terms of the individual unit analysis using the unit-wise metric, (Equation 3.3), with $\lambda = 0.5$. It is seen in Figure 3.10 (left) that models trained on TAO-VOS produce the highest number of specialized dynamic biased units compared to the other datasets. To explore this matter further, we evaluate the center bias for the four datasets by calculating the average number of groundtruth segmentation masks for each pixel over the entire dataset (normalized to 0-1), with results shown in Figure 3.10 (right). It is seen that for both layers, the percentage of specialized dynamic units is greatest for the dataset that has least center bias, i.e., TAO-VOS, as its center bias map is far more diffuse than the others. These results have implications for how the datasets can be used best for different tasks. For example, more general motion segmentation without concern for centering might be better served by training with a dynamic biased dataset (e.g., TAO-VOS).

3.4.2.3 Video instance segmentation

Datasets. We study the impact of the following two datasets on a VIS model's static and dynamic biases: YouTube-VIS 2019 (YTVIS) [270] and Occluded Video Instance Segmentation (OVIS) [196]. YTVIS is a popular VIS benchmark and contains 2,883 videos, 40 categories, 131k total masks, an average of 1.7 instances per video and an average video length of 4.6 seconds. OVIS contains 901 videos and corresponding annotations of *occluded objects* (i.e., masks of occluded objects are also labelled). OVIS contains 901 videos, 25 categories, 296k masks, an average of 5.8 instances per video and an average video length of 10.1 seconds. Both datasets are manually labelled every 5 frames.

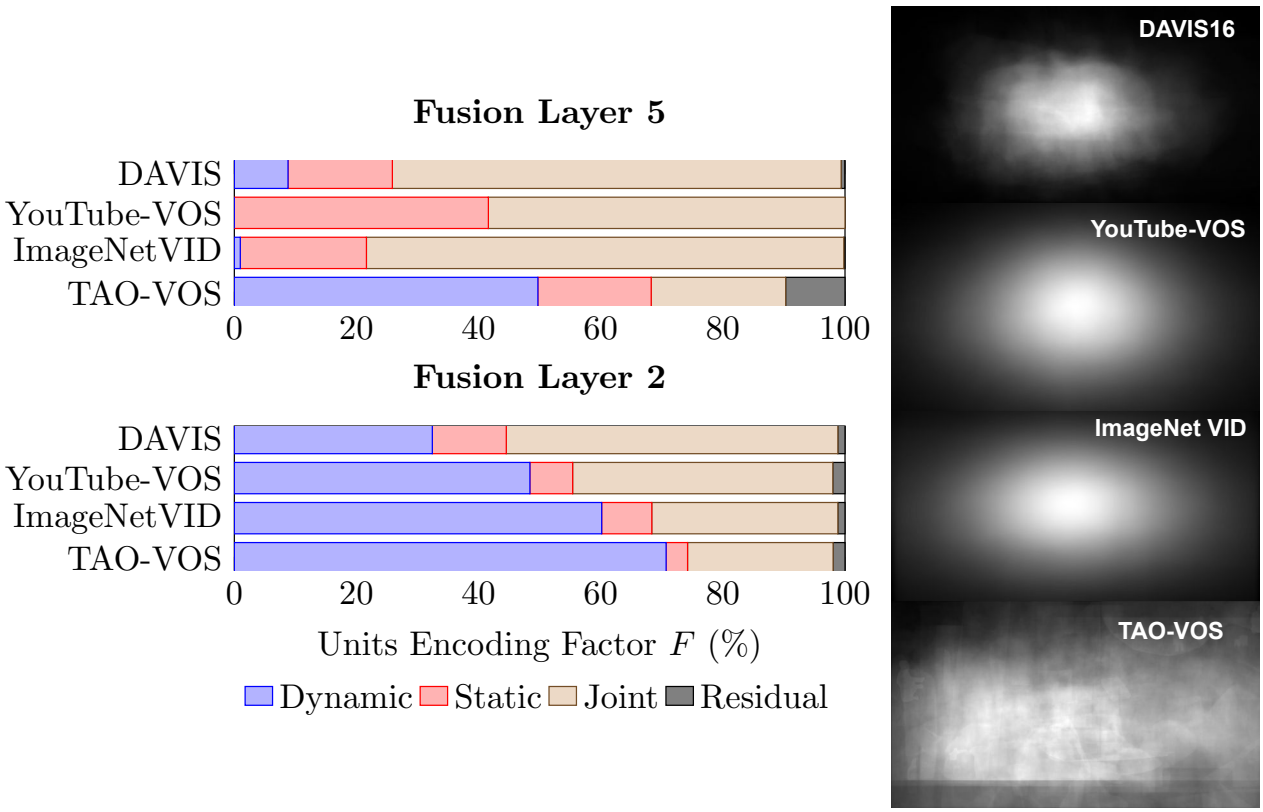


Figure 3.10: Analyses of biases of AVOS datasets. **Left**: Estimating the dynamic, static, joint and residual units using the unit-wise metric, (Equation 3.3), for FusionSeg's fusion layers five and two trained on DAVIS16 [192], YouTube-VOS [268], ImageNetVID [132] and TAO-VOS [250]. **Right**: Center bias plots for the four datasets. The results show the emergence of more dynamic units for both fusion layers when trained on the least center biased dataset (i.e., TAO-VOS).

Dataset	Layer 6		Layer 12	
	Dyn.(%)	Stat.(%)	Dyn.(%)	Stat.(%)
YTVIS [270]	21.9	39.6	22.1	38.5
OVIS [196]	26.3	34.6	21.4	38.3

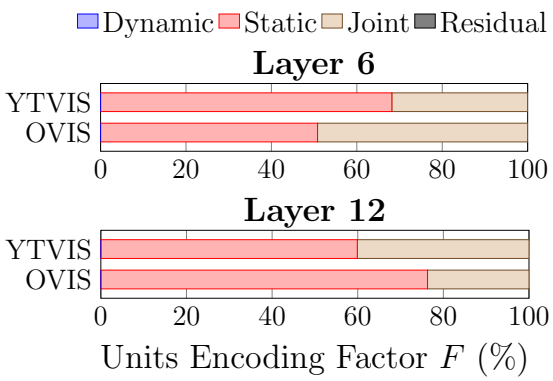


Figure 3.11: Analyses of biases of VIS datasets. **Left:** Results for the static and dynamic biases using the layer-wise metric, (Equation 3.1), for layers 6 and 12 of VisTR-R50 trained on OVIS [196] and YTVIS [270]. **Right:** Unit-wise analysis, (Equation 3.2), of a VisTR-R50 model trained on both datasets for layers 6 and 12. Overall, the static and dynamic biases between the two datasets are generally in agreement with each other.

Dataset bias. Figure 3.11 (left) shows the static and dynamic biases of a VisTR-R50 model trained on both datasets using the layer-wise metric, (Equation 3.1). The model trained on OVIS contains notably more dynamic bias than the YTVIS trained model in the last layer of the encoder (i.e., layer six) while the biases are similar in the final decoder layer (i.e., layer 12). This result is further demonstrated when observing the unit-wise results, (Equation 3.2), shown in Figure 3.11 (right). While both datasets produce solely static and joint units, the model trained on OVIS produces more jointly encoding units in the encoder than the model trained on YTVIS and the opposite is true in the decoder. These results suggest that while dynamics are learned in the encoding layers, the model may lack the ability to decode the dynamics. We observe a similar pattern when comparing to the biases of action recognition transformers (Section 3.4.1.1) where solely static and joint units also are produced. This pattern might be due to the mixing of information that is present in self-attention layers, which could inhibit specialized dynamic units from emerging. An interesting future direction suggested by these results is the design of self-attention layers which can admit specialized dynamic units throughout the model (e.g., by introducing two streams, as in the SlowFast model).

3.4.2.4 Summary and shared insights

For action recognition, our results raise questions about some of the widely adopted datasets. In particular, Diving48 is claimed to be a good benchmark for learning dynamics [160], while our results suggest that SSv2 is better suited for evaluating a model's ability to capture dynamics. In AVOS, we found training on TAO-VOS yields the largest number of specialized dynamic units. Thus, it may be a better training dataset for tasks that rely on capturing dynamics (e.g., motion segmentation). Similarly, for VIS, we demonstrated that OVIS may provide a better signal for learning dynamic information, particularly in the earlier layers of a network.

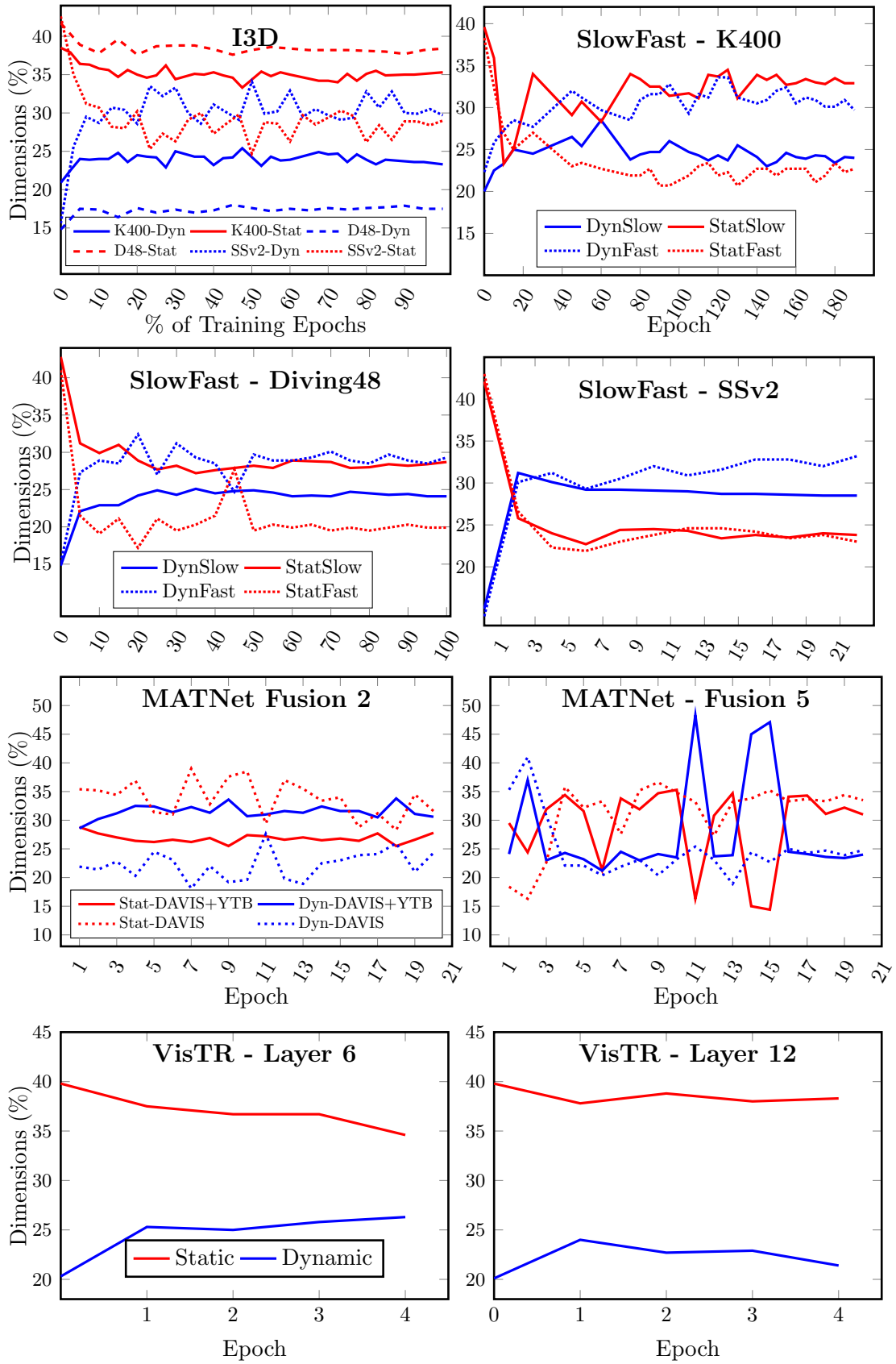


Figure 3.12: **Top and second row:** The encoding of static and dynamic bias for I3D [38] (8x8 R18) and SlowFast [77] (4x16 R18) over the course of training on Kinetics-400. **Third row** The encoding of static and dynamic bias over the course of training for MATNet trained on DAVIS and DAVIS + YouTube-VOS. **Fourth row** The encoding of static and dynamic bias for the VisTR [259] model over the course of training on the OVIS dataset [196].

3.4.3 When are statics and dynamics learned in training?

Based on our previous analysis of models post-training, we now ask: "What are characteristics of learning static and dynamic information over the course of training?". Understanding how the models learn static and dynamic information can potentially inform researchers to build better training protocols for learning specific types of information.

3.4.3.1 Action Recognition

We examine both a single and multi-stream architecture and choose the I3D [38] and SlowFast [77] models. We evaluate these models on Kinetics [38], Diving48 [160] and SSv2 [103] as each dataset differs in the static and dynamic information learned during training (see Section 3.4.2.1). Figure 3.12 (top and second row) shows the layer-wise results for action recognition. Both the static and dynamic information are largely learned during the first half of training in all cases. Comparing both models for Kinetics-400 and Diving48 further confirms our previous finding (see Section 3.4.2.1) that both datasets result in similar amounts of static and dynamic information. Conversely, models trained on SSv2 encode more dynamic information for both the single stream and two-stream models. The I3D model has a balance of static and dynamic information throughout training, while both branches in the SlowFast model quickly converge to be biased toward encoding dynamic information. The latter indicates that, given a model which has the ability to encode either type of information (e.g., within the two branches in the SlowFast model), SSv2 will yield a greater amount of dynamics.

3.4.3.2 Automatic video object segmentation

We train MATNet two datasets: (i) DAVIS16 only (DAVIS) and (ii) DAVIS16 combined with YouTube-VOS (DAVIS+YTB) as originally proposed [288]. Figure 3.12 (third row) shows the epoch-wise biases learned by MATNet on both settings. When trained on DAVIS+YTB, fusion layer 2 learns more dynamics compared to DAVIS alone, which confirms our previous findings (see Figure 3.10 fusion layer 2). Additionally, when trained on DAVIS+YTB, the model quickly converges toward being dynamics biased. On the other hand, fusion layer 5 shows an interesting pattern of alternation between the static and dynamic factors throughout training and culminates to being static biased. These results align with our previous findings that MATNet's early fusion layers converge to being dynamics biased, unlike late fusion layers (see Figure 3.5c). This result may stem from the fact that deeper layers tend to capture more abstract information compared to earlier layers. When trained on DAVIS, the final fusion layer becomes static biased in the early epochs, which aligns with the findings from action recognition.

3.4.3.3 Video instance segmentation

Figure 3.12 (fourth row) presents results for the VisTR model [259] trained on the OVIS dataset [196] for the last layer of the encoder (layer six) and decoder (layer 12). Interestingly, we observe different patterns in each layer. In layer six, the dynamic bias increases monotonically during training while static bias decreases over training. In layer 12, the dynamic bias increases by $> 4\%$ in the first epoch but decreases thereafter. This pattern suggests that the encoder and decoder have different training dynamics as they converge to their resulting biases in alternate ways. These results align with the results from Figure 3.7 that also show differences between the encoder and decoder in terms of the ratio of joint to static units.

3.4.3.4 Summary and shared insights

All models in the three tasks converge to their culminating biases within the first half of training epochs, except for MATNet trained on DAVIS+YTB. We also observed an interaction between models and datasets in which certain combinations of the two produce significantly more stability in their learned biases than others over the course of training.

3.5 Controlling model bias

We have demonstrated the effect to which architectures, datasets and training protocols have on a model's biases. We now ask: (i) *Is it possible to control the static and dynamic biases of a model?* and (ii) *What impact do static and dynamic units have on performance?* We first show that the type of bias encoded by a unit greatly determines the impact it has on final performance through *neuron removal* experiments for action recognition and video segmentation. Motivated by the varying impact different biases have on overall performance, we aim to improve the performance for each task. For action recognition, we propose *StaticDropout*, a novel dropout strategy that uses our estimation technique to dropout static-biased units during training, with the ultimate goal of debiasing models away from static information. For AVOS, we perform a detailed analysis on cross connections and fusion design choices with respect to their static and dynamic biases and show how to encourage a previously static biased model to become more biased toward dynamics. In both domains, we show an improvement in performance for tasks that require dynamics.

3.5.1 Neuron removal

To evaluate the effect of static and dynamic units on overall performance, we conduct perturbation experiments where we remove the top-k units (i.e., channels) that are biased toward the static or dynamic factor during inference and evaluate the performance drop. The removal is done by setting

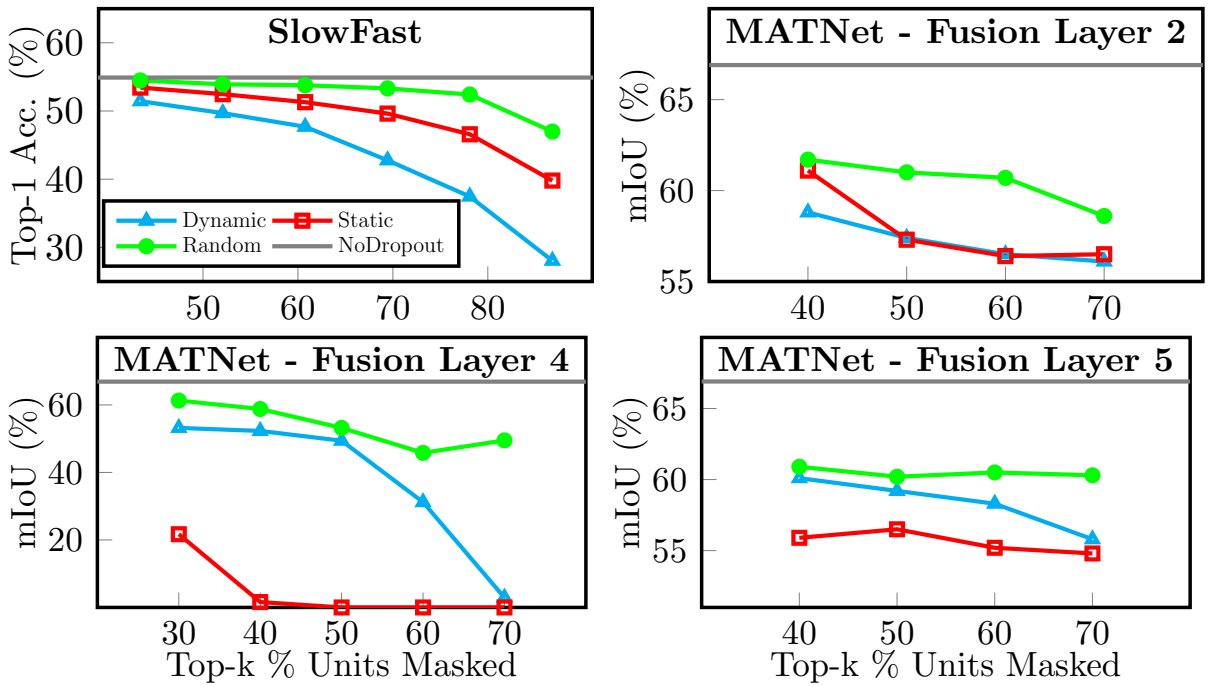


Figure 3.13: Top-k unit removal results for static and dynamic factors with respect to random units on: (action recognition) the final layer of SlowFast model trained on SSv2 dataset, and (video object segmentation) three fusion layers of the MATNet model trained on DAVIS and YouTube-VOS.

all activations to zero in the identified channels. We compare these static or dynamic biased units with respect to randomly selected channels. A lower Area Under Curve (AUC) suggests that the units are more important for model performance.

Figure 3.13 shows the unit attribution curves [83] for action recognition and AVOS. In action recognition we evaluate the final layer in the SlowFast model trained on SSv2 and evaluate on the SSv2 validation set to report the top-1 accuracy. As can be seen in Figure 3.13 (top left), the dynamic factor maximally reduces the model's performance, e.g., removing 70% of the SlowFast model dynamic units decreases the performance by 5% more than static units, which may be because the SlowFast model encodes a significant amount of dynamic information in the fast branch and dynamics are important for the SSv2 dataset.

We conduct similar experiments to AVOS for the early and late fusion layers of the MATNet model trained on DAVIS and YouTube-VOS. Empirically, we found that at intermediate layers there is a higher chance of randomly selecting units that cumulatively have a significant impact on the performance unlike the final layers, which was also found in [48]. Thus, in video object segmentation, as we remove units from the intermediate fusion layers (unlike action recognition where we ablate the final layer), we randomly sample from the units that are least biased towards the dominant factor (i.e., static or dynamic) of that layer to avoid the aforementioned scenario. More specifically, the random baseline first takes the least biased units toward the significant factor of this layer (i.e., dynamics in

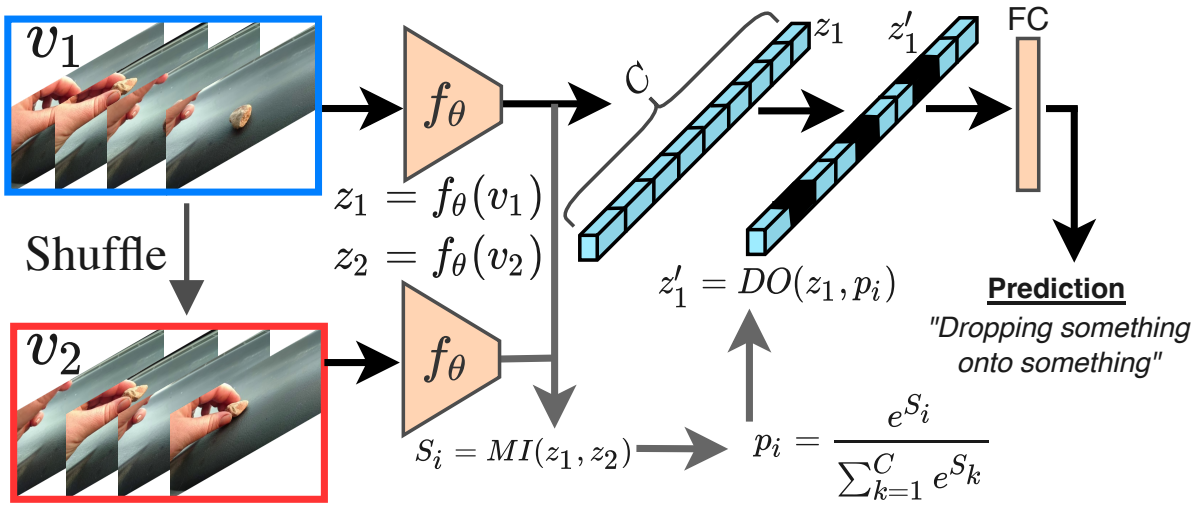


Figure 3.14: An overview of our StaticDropout method. Given a video, v_1, we generate v_2 by randomly shuffling the frames. v_1 and v_2 are passed through the model, f_θ, to produce intermediate representations of the videos, z_1 and z_2. Using these representations, we estimate the mutual information (using Equation 3.1) of each channel, i, between the two representations, S_i. We determine the probability of dropping out channel, p_i, by performing a Softmax over S_i.

fusion layer 2, and static in other fusion layers) and then randomly samples $x\%$ units within $(x+5)\%$ of these units.

We evaluate on the moving camouflaged animals [152] (MoCA) dataset, where the objects of interest are camouflaged animals and hence largely indistinguishable from their backgrounds in the absence of motion, and report the mean intersection over union (mIoU) (see Section 3.5.3 for more details on evaluation). The results in Figure 3.13 (bottom left and right) consistently demonstrate that for every fusion layer the factor with the highest impact on performance is the factor it is most biased toward, as examined earlier (Figure 3.5c and Section 3.4.3.2). More specifically, fusion layer 2 is slightly dynamic biased, while fusion layers 4 and 5 are static biased, e.g., fusion layer 4 achieves 0% when removing only 40% of the most static units.

In both tasks, these experiments document that masking out the top-k channels based on our metric can help remove biased units in the model and consequently affect its accuracy more compared with randomly selected channels. We now aim to use these insights to show possible mechanisms for improving the performance of a model.

3.5.2 StaticDropout

We propose StaticDropout, a semantically guided dropout technique with the goal of static-debiasing a model. Previous work, InfoDropout [222], used a measure of self information to identify neurons in a CNN that encode texture information for image-based models. Alternatively, we perform StaticDropout during training on static-biased channels in action recognition models. In-

foDropout [222] observed that dropping out neurons encoding high frequency caused the model to encode more low-frequency (e.g., shape) information. Accordingly, our intuition is that dropping out units biased towards static information will force the model to rely on *dynamic* information.

Approach. Our overall approach is outlined in Figure 3.14. We determine the channels in the model encoding static information by constructing statically similar video pairs during training. More specifically, given a video v_1, we shuffle the video frames to generate v_2. We pass both representations through the model to obtain intermediate representations z_1 and z_2 from layer l (for fair comparison against standard dropout, we set l to the last layer before the fully connected layer, and hereon omit l for brevity). Using the unitwise metric, (Equation 3.2), we calculate a score, S_i, representing the static information encoded by channel i. We calculate the probability of dropping channel i via a multinomial distribution defined as

$$p_i = \frac{e^{S_i}}{\sum_{k=1}^{C} e^{S_k}}. \tag{3.4}$$

The dropout rate corresponds to the total fraction of channels being dropped out of the layer, with p_i the probability (relative to other channels) that channel i will be included in the dropped channels.

Evaluation Protocol. We first demonstrate the ability of our proposed StaticDropout technique to manipulate static and dynamic biases as measured by two different metrics. For the first metric, we use the unitwise metric, (Equation 3.3), to calculate the ratio of dynamic units relative to dynamic and static units. The second metric is the model's relative validation performance on shuffled vs. unshuffle frames. We apply StaticDropout to two different 3D-Resnet variants (i.e., the SlowOnly architecture) with 18 and 50 layers, on two different datasets, Diving48 and SSv2.

Implementation Details. Models trained on SSv2 are trained with the original SlowFast [77] repository hyperparameters. They are trained for 30 epochs and use a cosine learning rate decay with three warmup epochs, a base learning rate of 0.1 and a warmup starting learning rate of 0.08. Models trained on Diving48 are trained for 100 epochs and use a cosine learning rate decay with 10 warmup epochs, a base learning rate of 0.1 and a warmup starting learning rate of 0.01. We experiment with dropout rates of $r = \{0.1, 0.3, 0.5, 0.7\}$. The static channel scores, S_i, are re-estimated using Equation 3.4 every 30 iterations. Following InfoDropout [222], we finetune our models without any dropout with a learning rate of 1e-5 for two and five epochs on SSv2 and Diving48, respectively. For baselines, we consider the same model architectures with standard dropout [229] and without any dropout applied.

Results. Figure 3.15 shows the debiasing results for models trained on Diving48 and SSv2. StaticDropout is successful in debiasing the model away from static and toward dynamic information. In terms of our metric, (Equation 3.3), both the models see a significant jump in the ratio of dynamic to static units on SSv2 and Diving48 (Figure 3.15 right). The debiasing also has a strong effect in re-

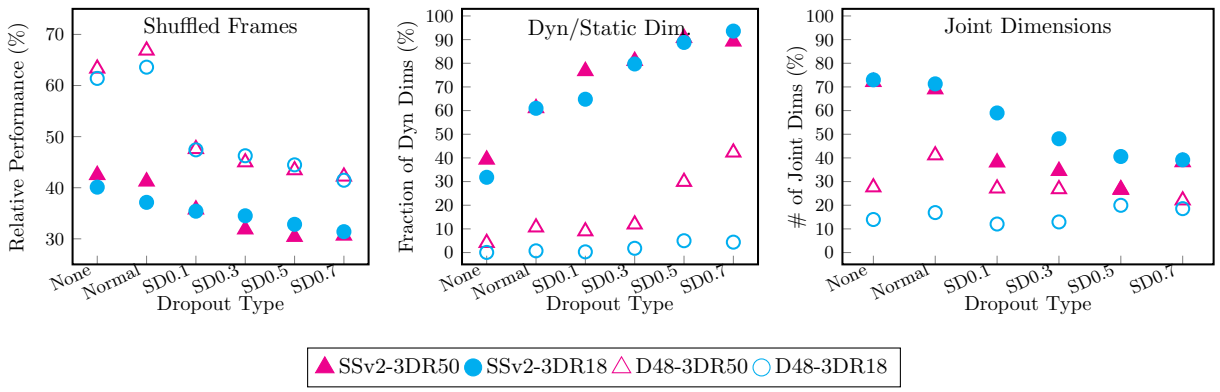


Figure 3.15: StaticDropout influences the static and dynamic information of a model. (Left) We evaluate the relative performance, in terms of top-1 classification accuracy as A_{shuff}/A_{orig}, where A_{orig} and A_{shuff} are the normal and shuffled model performance at validation time, respectively. (Middle) We also show the fraction of channels encoding dynamic information relative to the channels encoding either static or dynamic. Finally, (right) we show the number of joint encoding units for 3D-ResNet50 and 3D-ResNet18 on the SSv2 and Diving48 datasets as a function of StaticDropout rates.

ducing the models' ability to classify shuffled video frames compared with the standard or no-dropout baselines (Figure 3.15 left). These results suggest that StaticDropout has a strong influence on the types of specialized units contained in the model and also the model's bias toward encoding dynamic and static information. Figure 3.15 (bottom) shows the results in terms of joint encoding units. Interestingly, most models contain fewer joint encoding units as the StaticDropout rate is increased, but the 3D ResNet18 trained on Diving48 deviates slightly from this pattern. This suggests that StaticDropout encourages the model to produce specialized units, in the form of static, dynamic or residual units. Notably, the effect of StaticDropout on the model bias is consistently *dose-dependant*, meaning the biases are tunable via the dropout rate, r.

We evaluate the performance of models trained with StaticDropout on the SSv2 [103], as it requires the maximal amount of dynamics compared with other datasets (see Sec 3.4.2.1). We also evaluate on the SomethingElse [171] dataset, an 'object debiased' relabelled version of SSv2 where the set of objects (i.e., "somethings") appearing in the training set for a specific action is disjoint from the set of objects appearing in the validation set for the same action (see [171] for additional details). We experiment on the SomethingElse compositional split with the intuition that our StaticDropout technique may debias the model from focusing on object appearances and more generally toward longer range motions contained in the videos. The results for both dataset splits are shown in Figure 3.16 (top) in terms of Top-1 accuracy and percentage of dynamic units in the model's final layer. StaticDropout outperforms both the baselines in terms of accuracy, and significantly increases the number of dynamic units. However, a dropout rate of $r = 0.7$ does not achieve the best performance, suggesting that simply maximizing the number of dynamic units is not optimal, and that a balance of dynamics and statics should be learned. Figure 3.16 (bottom) shows the average performance difference between

Dropout	SSv2 [103]		SomethingElse [171]	
	Top-1 (%)	Dyn. Units (%)	Top-1 (%)	Dyn. Units (%)
None	54.18	10.7	49.43	4.6
Normal (r=0.5)	54.55	18.2	49.79	17.5
StaticDropout (r=0.1)	**55.19**	56.9	50.38	34.2
StaticDropout (r=0.3)	54.55	53.6	**51.16**	45.1
StaticDropout (r=0.5)	54.01	**80.0**	50.06	**51.4**

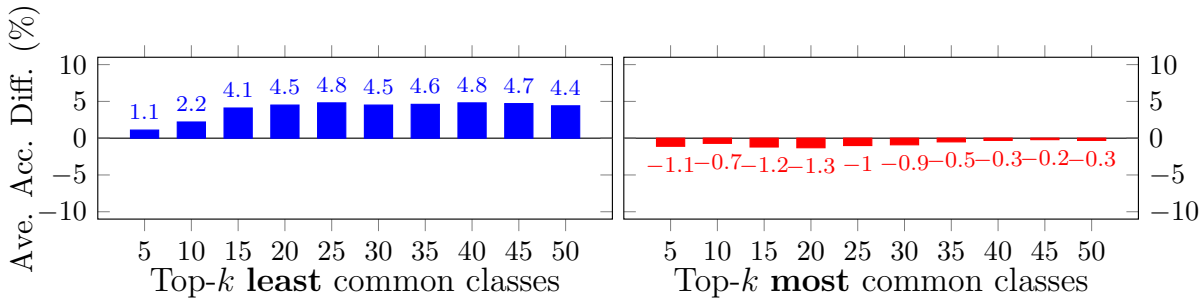


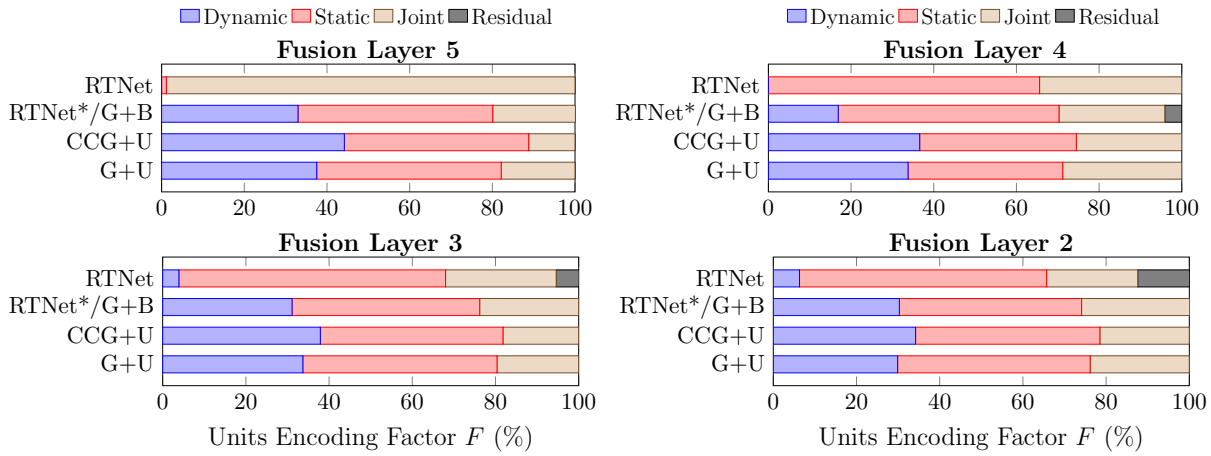
Figure 3.16: **Top:** StaticDropout results for 3D-ResNet50 trained on SSv2 and SomethingElse [171]. **Bottom:** Average performance difference between StaticDropout ($r = 0.3$) and baseline dropout of a 3D-ResNet50 for the top-k most and least common classes in the SomethingElse [171] compositional split. StaticDropout performs better on rare classes, while performing marginally worse on common classes.

a 3D-ResNet50 trained with StaticDropout compared with standard dropout on the top-k most and least common classes. Interestingly, on average, our model significantly outperforms the baseline on the rarest classes in the dataset while slightly under-performing the baseline on the most common classes. This result suggests that our regularization specifically targets classes in the tail end of the data distribution.

3.5.3 Fusion and cross connection study

Our goal in this final section is to encourage a previously static biased AVOS model (i.e., RTNet) to have more dynamic units. Based on our analysis in Sections 3.4.1.2 and 3.4.2.2, we hypothesize that the reason behind AVOS models being static biased stems from the sub-optimal application of bidirectional cross connections. Toward this goal, we conduct a detailed analysis of the fusion and cross connection types in the aforementioned architectures and evaluate how they affect biases and model accuracy.

Approach. We now analyze the different types of *fusion* and *cross connections*. Cross connections can operate *Bidirectionally* (i.e., motion-to-appearance and appearance-to-motion), as in RTNet, or *Unidirectionally* with motion-to-appearance cross connection, as in MATNet. The *Convex Combination Gated Fusion* in RTNet enforces the attention weights to sum to one for both motion and appearance streams. While the *Gated Fusion* in MATNet learns to weigh each stream without this constraint.



Method	Fusion	CC	Flip	mIoU	SR_{Mean}
FusionSeg [132]	-	-	✗	42.3	39.2
RTNet [205]	CCG	B	✗	60.7	50.2
MATNet [288]	G	U	✓	64.2	54.4
MATNet*	G	U	✗	67.3	56.6
MATNet*	CCG	U	✗	64.4	51.8
RTNet*	G	B	✗	**70.3**	**58.1**
MATNet*	G	U	✓	68.5	58.1
RTNet*	G	B	✓	**70.6**	**58.6**

Figure 3.17: **Top:** Estimates of the dynamic, static, joint and residual units using our metric for unit-wise analysis, (3), $\lambda = 0.5$ at different fusion layers of RTNet. **Bottom:** Evaluation of AVOS models on MoCA [152]. * means networks implemented by us (see Section 3.4.1.2). We compare cross connections: unidirectional motion-to-appearance (U) vs. bidirectional (B), and fusion types: gated (G) vs. convex combination gated (CCG). Flip means performing data augmentation with flipping during inference, as originally done in [288].

Both mechanisms can be decomposed into channel attention followed by spatial attention. For the sake of unification we pose the channel attention in the *Convex Combination Gated Fusion* as,

$$A_c = \mathcal{F}_e(\mathcal{F}_s(U_a \oplus U_m)), \quad Z_a = A_c \circ U_a, \tag{3.5a,b}$$

$$Z_m = (1 - A_c) \circ U_m, \tag{3.5c}$$

where U_a, U_m are the appearance and motion features, respectively, $\circ$ is the Hadamard product, $\oplus$ denotes concatenation, $\mathcal{F}_s$ and $\mathcal{F}_e$ are the squeeze and excitation operators, respectively, and $A_c \in \mathbb{R}^c$ are the channel attention weights where c is the number of channels for the input appearance or motion feature maps. A weighted combination is used to attend to motion and appearance features, enforcing a strong constraint on the attention weights during training. On the other hand, the channel attention module in the *Gated Fusion* can be given as

$$U = U_a \oplus U_m, \quad A_c = \mathcal{F}_e(\mathcal{F}_s(U)), \quad Z = A_c \circ U, \tag{3.6a-c}$$

where $A_c \in \mathbb{R}^{2c}$ is the channel-wise attention on the combined motion and appearance features without

enforcing them to to sum to one, unlike in RTNet.

In contrast, the spatial attention in *Convex Combination Gated Fusion* can be given as

$$Z = A_{sp} \circ Z_a \oplus (1 - A_{sp}) \circ Z_m, \tag{3.7}$$

where A_{sp} is the spatial attention maps generated from a convolutional layer and a sigmoid function, while Z_a and Z_m are the average pooled appearance and motion features. Finally, the spatial attention in *Gated Fusion* can be given as

$$Z = A_{sp} \circ Z + Z, \tag{3.8}$$

where Z is the combined motion and appearance features. Again, it is clear that the spatial attention module in the *Convex Combination Gated Fusion* restricts the attention weights between both motion and appearance features, unlike the *Gated Fusion*. Therefore we hypothesis that *Gated Fusion* will yield better results for tasks requiring a complex interaction between motion and appearance features. We now ablate the fusion types along with the two different types of cross connections on such a task.

Evaluation protocol. We show the static and dynamic biases for the different fusion and cross connections, and evaluate accuracy on a dataset which requires dynamics (MoCA). We follow previous work by removing videos that contain no predominant target locomotion, which produces a subset of 88 videos for evaluation [269]. We evaluate using mean intersection over union and success rate mean with varying IoU thresholds ranging from 0.5 to 0.9. The original MATNet used horizontal flipping during the inference and averaged predictions from the original and flipped versions. To ensure fair comparison between RTNet and FusionSeg we show results with and without the flipping augmentation during inference when reporting on MoCA.

Results. Figure 3.17 (top) shows the static and dynamic biases of the different fusion and cross connection types. It is seen that the bidirectional cross connections (G+B) incur a small decrease in the dynamic bias with respect to unidirectional ones (G+U), especially on the final fusion layers (i.e., fusion 4 and 5). The convex combination gated fusion (CCG+U) leads to a decrease in the ratio of joint to dynamic units with respect to gated fusion (G+U) (e.g., the final fusion layer for CCG+U Joint/Dynamic is 25.3 vs. 47.5 for G+U). The performance of all models on the MoCA dataset is shown in Figure 3.17 (bottom). The original MATNet and MATNet* both outperform the off-the-shelf RTNet model and FusionSeg, which is confirmation of our previous findings that both models are heavily static biased (see Figure 3.6). Additionally, it demonstrates that our proposed model (RTNet*) with bidirectional cross connections and Gated Fusion (B+G) trained on YouTube-VOS and DAVIS shows sizable (+9.9%) gain with respect to off-the-shelf RTNet. We also observe from the results of MATNet* (G+U) vs. MATNet* (CCG+U) that the added constraint of convex combination can degrade the performance with respect to gated fusion on a dynamically heavy task (i.e., MoCA).

Motivated by the previous experiments. which showed the impact of static and dynamic biases on

performance (Section 3.5.1), these results demonstrate how simply selecting the appropriate combination of fusion and cross connection mechanisms can significantly impact what is encoded in models as well as the downstream performance.

3.6 Chapter summary

This chapter has advanced the interpretability of learned spatiotemporal models for video understanding, especially action recognition, AVOS and VIS. We first introduced a general method for analyzing the extent that various architectures capitalize on static vs. dynamic information. We showed how our method can be applied to investigate the static vs. dynamic biases in datasets. Furthermore, we demonstrated the impact of static and dynamic biases on overall performance through a new type of regularization for action recognition (StaticDropout) and architectural modifications in the fusion and cross connection layers for AVOS. Future work can apply our method to additional video understanding tasks (e.g., action prediction) and use the proposed performance enhancing techniques on different datasets, architectures and tasks which require dynamics.

Chapter 4

Open-world concept discovery in video transformers

4.1 Introduction

Understanding the hidden representations within neural networks is essential for addressing regulatory concerns [53, 116], preventing harms in deployment [35, 108], and can aid innovative model designs [57]. This problem has been studied extensively for images, both for convolutional neural networks (CNNs) [18, 86, 98, 141] and, more recently, vision transformers (ViTs) [201, 253], resulting in multiple key insights. For example, image classification models extract low-level positional and texture cues at early layers and gradually combine them into higher-level, semantic concepts at later layers [18, 95, 183].

However, while video transformers do share their overall architecture with image-level ViTs, the insights obtained in existing works do very little to explain their inner mechanisms. Consider, for example, the recent approach for tracking occluded objects [247] shown in Figure 4.1 (top). To accurately reason about the trajectory of the invisible object inside the pot, texture or semantic cues alone would not suffice. What, then, are the *spatiotemporal* mechanisms used by this approach? Are any of these mechanisms *universal* across video models trained for different tasks?

To answer these questions, in this chapter we present the Video Transformer Concept Discovery algorithm (VTCD), the first concept discovery methodology for interpreting the representations of deep video transformers. We focus on concept-based interpretability [83, 86, 98, 281] due to its capacity to explain the decision-making process of a complex model's distributed representations in high-level, intuitive terms. Our goal is to decompose a representation at any given layer into human-interpretable 'concepts' without any labelled data (i.e., concept discovery) and then rank them in terms of their importance to the model output.

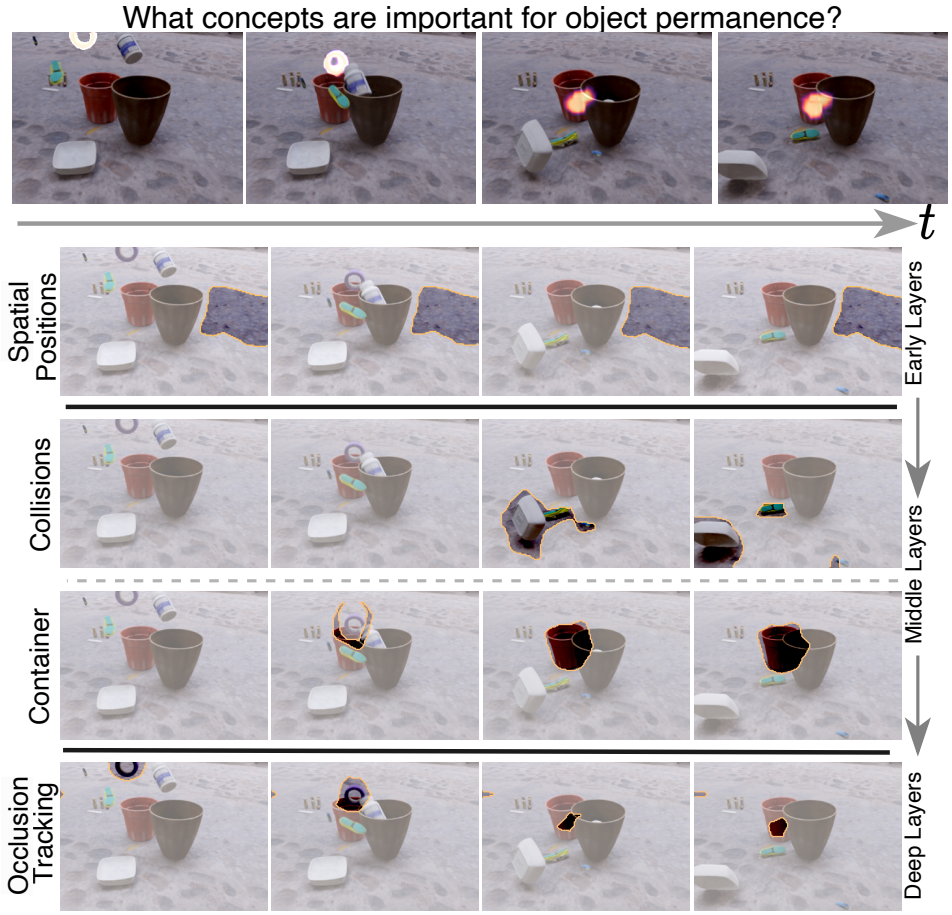


Figure 4.1: Heatmap predictions of the TCOW model [247] for tracking through occlusions (top), together with concepts discovered by VTCD (bottom). We can see that the model encodes positional information in early layers, identifies containers and collision events in mid-layers and tracks through occlusions in late layers. Only one video is shown, but the discovered concepts are shared between many dataset samples (see video for full results).

Concretely, we first group *model features* at a given layer into spatiotemporal tubelets via SLIC clustering [3], which serve as a basis for our analysis (Section 4.3.1.1). Next, we cluster these tubelets across videos to discover high-level concepts [63, 83, 86, 155, 281] (Section 4.3.1.2). The resulting concepts for an occluded object tracking method [247] are shown in Figure 4.1 (bottom) and span a broad range of cues, including spatiotemporal ones that detect events, like collisions, or track the containers.

To better understand the decision-making mechanisms of video transformers, we then quantify the importance of concepts for the model's predictions. Inspired by previous work on saliency maps [195], we propose a novel, noise-robust approach to estimate concept importance (4.3.2). Unlike existing techniques that rely on gradients [141], or concept occlusion [83], our approach effectively handles redundancy in self-attention heads in transformer architectures.

Next, we use VTCD to study whether there are any universal mechanisms in video transformer models, that emerge irrespective of their training objective. To this end, we extend recent work [68] to automatically identify *important* concepts that are shared between several models in Section 4.4.1. We then analyze a diverse set of representations (e.g., supervised, self-supervised, or video-language) and make a number of discoveries: (i) many concepts are indeed shared between models trained for different tasks; (ii) early layers tend to form a spatiotemporal basis that underlines the rest of the information processing; (iii) later layers form object-centric video representations, even in models trained in a self-supervised way.

We also show how VTCD can be applied for downstream tasks. Firstly, pruning the heads of an action classification model according to their estimated importance yields a 4.3% *increase* in accuracy while reducing computation by 33%. Secondly, object-centric concepts discovered by VTCD can be used for video-object segmentation (VOS) and achieve strong performance on the DAVIS'16 benchmark [193] even for self-supervised representations.

4.2 Related work

Our work introduces a novel *concept-based interpretability* algorithm that focuses on *transformer-based representations* for *video understanding*. Below, we review the most relevant works in each of these fields.

4.2.1 Concept-based interpretability

Concept-based interpretability is a family of neural network interpretability methods used to understand, post-hoc, the representations that a model utilizes for a given task. Closed-world interpretability operates under the premise of having a labeled dataset of concepts [18, 141]. However, for videos, it is

unclear what concepts may exist and also difficult to densely label videos even if they were known a priori.

In contrast, unsupervised concept discovery makes no assumptions on the existence of semantic concepts and uses clustering to partition data into interpretable components within the model's feature space. ACE [98] and CRAFT [86] segment input images into superpixels and random crops, respectively, before applying clustering at a given layer. In videos, however, the potential tubelets far outnumber image crops, prompting us to introduce a more efficient method for segmenting videos into proposals in Section 4.3.1.1.

A necessary component of concept-based interpretability is measuring the importance (i.e., fidelity) of the discovered concepts to the model. However, the aforementioned methods [83, 86, 98, 141, 281] were developed for CNNs, and are not readily applicable to transformers. The main challenge of ranking concepts in attention heads is due to the transformers' robustness to minor perturbations in self-attention layers. To address this limitation, we introduce a new algorithm to rank the significance of any architectural unit, covering both heads and intra-head concepts in Section 4.3.2.

Recent work [68] identifies neurons that produce similar activation maps across various image models (including transformers). However, neurons are unable to explain the full extent of a models' distributed [70] representation. In contrast, our method works on features of arbitrary dimension and is applied to video models.

4.2.2 Transformer interpretability

The interpretability of transformers have received significant attention recently, due to its success in a variety of computer vision and natural language processing (NLP) tasks. Early work [201] contrasted vision transformer representations with CNNs (representational differences per layer, receptive fields, localization of information, etc). Other work aims to generate saliency heatmaps based on attention maps of a model [42, 43]. Later works focused on understanding the impact of different training protocols [189, 253] (e.g., self-supervised learning (SSL) vs. supervised) and robustness [197, 287]. The features of a specific SSL vision transformer, DINO [10], were explored in detail and shown to have surprising utility for part-based segmentation tasks. However, none of these works address concept-based interpretability or study video representations.

Independently, studies in NLP analyzed self-attention layers [71,251] and found that heads are often specialized to capture different linguistic or grammatical phenomenon. This is qualitatively seen in vision works that show dissimilar attention maps for different self-attention heads [54,138]. Moreover, other NLP works [174, 251] explore the impact of removing heads and find that only a small number need to be kept to produce similar performance. Our findings agree with evidence from these works and in Section 4.5 we further demonstrate that targeted pruning of unimportant heads from video

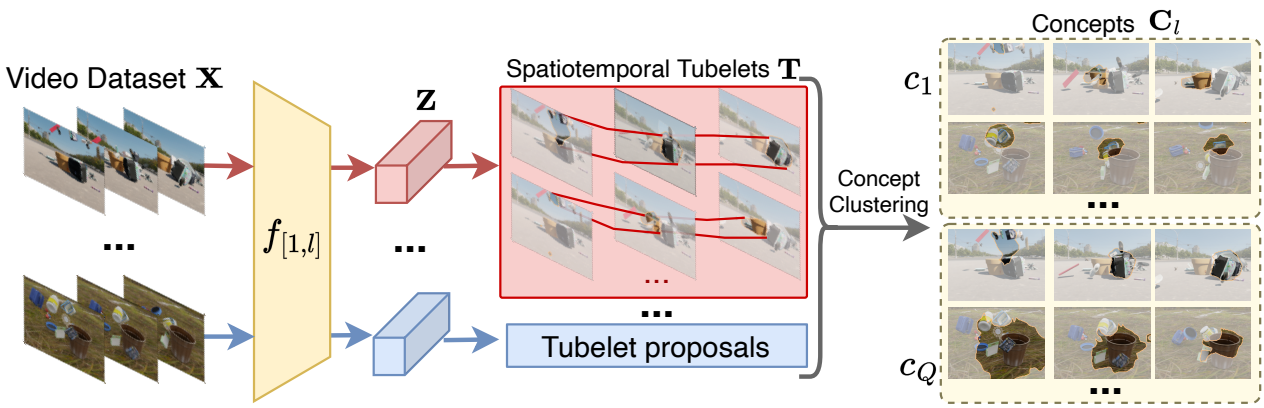


Figure 4.2: Video Transformer Concept Discovery (VTCD) takes a dataset of videos, $\mathbf{X}$, as input and passes them to a model, $f_{[1,l]}$ (shown in yellow). The set of video features, $\mathbf{Z}$, are then parsed into spatiotemporal tubelet proposals, $\mathbf{T}$ (shown in red), via SLIC clustering in the feature space. Finally, tubelets are clustered across the videos to discover high-level units of network representation - concepts, $\mathbf{C}$ (right).

transformers can actually *improve* a model's performance.

4.2.3 Interpretability of video models

Video model interpretability is an under-explored area of research considering the recent successes of deep learning models in action recognition [145], video object segmentation [157, 193, 242, 247], or self-supervised approaches [78,202,231,243,254]. Efforts have used proxy tasks to measure the degree to which models use dynamic information [96,106,124] or scene bias [50,158,160]. One method quantifies the static and dynamic information contained in a video model's intermediate representation [147,148]. However, these methods can only measure one or two predefined concepts (i.e., static, dynamic, or scene information) while our approach is not restricted to a subset of concepts. Another work visualizes videos that activate single neurons (or filters) in 3D CNN's via activation maximization with temporal regularization [81]. While this method has no restrictions on what a neuron can encode, it only applies to 3D CNNs and does not truly capture 'concepts' across distributed representations (i.e., feature space directions that generalize across videos).

4.3 Method

We study the problem of decomposing a video representation into a set of high-level open-world concepts and ranking their importance for the model's predictions. We are given a set of (RGB) videos, $\mathbf{X} \in \mathbb{R}^{N \times 3 \times T \times H \times W}$, where N, T, H, and W denote the dataset size, time, height, and width, respectively, and an L layer pretrained model, f. Let $f_{[r,l]}$ denote the model from layer r to l, with $f_{[1,l]}(\mathbf{X}) = \mathbf{Z}_l \in \mathbb{R}^{N \times C \times T' \times H' \times W'}$ being the intermediate representation at layer l. To decompose $\mathbf{Z}_l$

into a set of human-interpretable concepts, $\mathbf{C}_l = \{c_1, \ldots, c_Q\}$, existing, image-level approaches [86,98] first parse the N feature maps into a set of M proposals, $\mathbf{T} \in \mathbb{R}^{M \times C}$ ($M > N$), where each T_m corresponds to a region of the input image. These proposals are then clustered into $Q << M$ concepts in the feature space of the model to form an assignment matrix $W \in \mathbb{R}^{M \times Q}$. The importance of each concept c_i to the model's prediction is quantified by a score $s_i \in [0, 1]$. Performing this analysis over all layers in f produces the entire set of concepts for a model, $\mathbf{C} = \{\mathbf{C}_1, \ldots, \mathbf{C}_L\}$, together with their corresponding importance scores.

Existing approaches are not immediately applicable to video transformers because they do not scale well and are focused on 2D CNN architectures. In this work, we extend concept-based interpretability to video representations. To this end, we first describe a computationally tractable proposal generation method (Section 4.3.1.1) that operates over space-time feature volumes and outputs spatiotemporal tubelets. Next (Section 4.3.1.2), we adapt existing concept clustering techniques to video transformer representations. Finally, in Section 4.3.2 we propose a novel concept importance estimation approach applicable to any architecture units, including transformer heads.

4.3.1 Concept discovery

4.3.1.1 Tubelet proposals

Previous proposal methods [83,86,98] use superpixels or crops in RGB space to generate segments; however, the number of possible segments is exponentially greater for videos. Moreover, proposals in color space are unrestricted and may not align with the model's encoded information, leading to many irrelevant or noisy segments. To address these drawbacks, we instantiate proposals in *feature space*, which naturally partitions a video based on the information contained within each layer (shown in Figure 4.2, left).

We produce tubelets per-video via Simple Linear Iterative Clustering [3] (SLIC) on the spatiotemporal features as

$$\mathbf{T} = \text{GAP}(\mathbf{B} \odot \mathbf{Z}) = \text{GAP}(\text{SLIC}(\mathbf{Z}) \odot \mathbf{Z}), \tag{4.1}$$

where $\mathbf{T} \in \mathbb{R}^{M \times C}$ is the set of tubelets for the dataset, $\mathbf{B} \in \{0, 1\}^{C \times M \times N \times T' \times H' \times W'}$ are spatiotemporal binary support masks obtained from SLIC, M is the number of tubelets for all N videos ($M >> N$), and GAP represents global average pooling over the space and time dimensions.

SLIC is an extension of the K-Means algorithm [163] that controls a trade-off between cluster support regularity and adaptability, and also constrains cluster support masks to be connected. Together these properties produce non-disjoint tubelets that are easier to interpret for humans because they reduce the need to attend to multiple regions in a video at a time. Further, the pruning step in SLIC makes it more robust to the hyperparameter that controls the desired number of clusters, as it

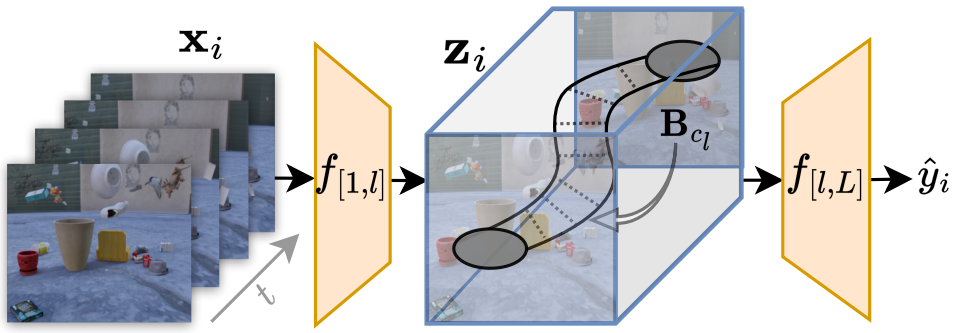


Figure 4.3: A visual representation of concept masking for a single concept. Given a video $\mathbf{x}_i$ and a concept, c_l, we mask the tokens of the intermediate representation $\mathbf{z}_i = f_{[1,l]}(\mathbf{x}_i)$ with the concepts' binary support masks, $\mathbf{B}_{c_l}$, to obtain the perturbed prediction, $\hat{y}_i$.

automatically prunes spurious, disconnected tubelets. Next, we describe our approach for grouping individual tubelets into higher-level concept clusters.

4.3.1.2 Concept clustering

Recent work [83,86,281] has used Non-Negative Matrix Factorization (NMF) [63] to cluster proposals into concepts. Given a non-negative data matrix, $\mathbf{T}^+ \in \mathbb{R}^{M \times C}$, NMF aims to find two non-negative matrices, $\mathbf{W}^+ \in \mathbb{R}^{M \times Q}$ and $\mathbf{C}^+ \in \mathbb{R}^{Q \times C}$, such that $\mathbf{T}^+ = \mathbf{W}^+\mathbf{C}^+$, where $\mathbf{W}^+$ is the cluster assignment matrix. Unfortunately, NMF cannot be applied to transformers as they use GeLU non-linearities, rather than ReLU, resulting in negative activations.

We solve this problem by leveraging Convex Non-negative Matrix Factorization [63] (CNMF). Despite the name, CNMF extends NMF and allows for negative input values. This is achieved by constraining the factorization such that the columns of $\mathbf{W}$ are convex combinations of the columns of $\mathbf{T}$, i.e., each column of $\mathbf{W}$ is a weighted average of the columns of $\mathbf{T}$. This constraint can be written as

$$\mathbf{W} = \mathbf{TG}, \tag{4.2}$$

where $\mathbf{G} \in [0,1]^{C \times Q}$ and $\sum_j \mathbf{G}_{i,j} = 1$. To cluster a set of tubelets, $\mathbf{T}$, into corresponding concepts, we optimize

$$(\mathbf{G}^*, \mathbf{C}^*) = \underset{\mathbf{C}>0, \mathbf{G}>0}{\arg\min} ||\mathbf{T} - \mathbf{TGC}||^2, \tag{4.3}$$

where the final set of concepts are the rows of the matrix $\mathbf{C}$, i.e., concept centroid c_i is the i^{th} row of $\mathbf{C}$ (Figure 4.2, right).

4.3.2 Concept importance

Given a set of discovered concepts, we now aim to quantify their impact on model performance. One approach, shown in Figure 4.3, is to mask out each concept independently and rank the importance based on the drop in performance [83]. Formally, let c_l be a single target concept, and $\mathbf{B}_{c_l} \in \{0,1\}^{C\times M\times N\times T'\times H'\times W'}$ the corresponding binary support masks over $\mathbf{X}$. It can then be masked in layer l via

$$\hat{y} = f_{[l,L]}(\mathbf{Z}_l \odot (1 - \mathbf{B}_{c_l})). \tag{4.4}$$

While this approach works well for CNNs [83], transformers are robust to small perturbations within self-attention layers [174, 251]. Therefore, single concept masking has little effect on performance (shown by results in Figure 4.4). Instead, we mask *a high percentage* of sampled concepts in parallel (across all layers and heads) and then empirically validate in Section 4.5.5 that averaging the results over thousands of samples produces valid concept rankings.

Formally, we propose **C**oncept **R**andomized **I**mportance **S**ampling (CRIS), a robust method to compute importance for any unit of interest. To this end, we first randomly sample K different concept sets, such that each $\mathbf{C}^k \subset \mathbf{C}$. We then define $\mathbf{C}_l^k$ as the set of concepts in $\mathbf{C}^k$ discovered at layer l, with $\mathbf{B}_{\mathbf{C}_l^k}$ denoting the corresponding binary support masks. We mask out every concept at every layer via

$$\hat{y}_k = g(\tilde{\mathbf{B}}_{\mathbf{C}_L^k} \odot f_{[L-1,L]}(\cdots(\tilde{\mathbf{B}}_{\mathbf{C}_1^k} \odot f_{[0,1]}(\mathbf{X})))), \tag{4.5}$$

where $g(\cdot)$ is the prediction head (e.g., an MLP) and $\tilde{\mathbf{B}}$ denotes the inverse mask (i.e., $1-\mathbf{B}$). Finally, we calculate the importance of each concept, c_i, via

$$s_i = \frac{1}{K}\sum_{k}^{K}(\mathbb{D}(\tilde{y}, y) - \mathbb{D}(\hat{y}_k, y))\mathbb{1}_{c_i\in\mathbf{C}^k}, \tag{4.6}$$

where $\tilde{y}$ is the original prediction without any masking and $\mathbb{D}$ is a metric quantifying performance (e.g., accuracy).

4.4 Understanding transformers with VTCD

Our algorithm facilitates the identification of concepts within any unit of a model and quantifying their significance in the final predictions; however, this is insufficient to fully represent the computations performed by a video transformer. It is also crucial to understand how these concepts are employed in the model's information flow.

As several recent works have shown [71, 186], the residual stream of a transformer serves as the backbone of the information flow. Each self-attention block then reads information from the residual

stream with a linear projection, performs self-attention operations to process it, and finally writes the results back into the residual stream. Crucially, self-attention processing is performed individually for each head with several studies showing, both in vision [10, 66, 138] and NLP [251], that different self-attention heads capture distinct information. In other word, heads form the basis of the transformer representation.

A closer analysis of the concepts found in the heads of the TCOW model [247] with VTCD allows us to identify several information processing patterns in that model. In particular, Figure 4.1 shows that the heads in early layers group input tokens based on their spatiotemporal positions. This information is then used to track objects and identify events in mid-layers, and later layers utilize mid-layer representations to reason about occlusions. Next, we study which of these mechanisms are *universal* across video transformers trained on different datasets and objectives (*cf.* [68]).

4.4.1 Rosetta concepts

Inspired by previous work [68], we mine for *Rosetta concepts* that are shared between models and represent the same information. The key to identifying Rosetta units is a robust metric, R, where a higher R-score corresponds to the two units having a larger amount of shared information. This previous work [68] focused on finding such neurons in image models based on correlating their activation maps. We instead measure the similarity between concepts (i.e., distributed representations) via the mean Intersection over Union (mIoU) of the concepts' support.

Formally, we mine Rosetta concepts by first applying VTCD to a set of D models $\{f^1, \ldots, f^D\}$, resulting in discovered concepts, $\mathbf{C}^j = \{c_1^j, \ldots, c_i^j\}$, and importance scores, $\mathbf{S}^j = \{s_1^j, \ldots, s_i^j\}$, for each model f^j. We then aim to measure the similarity between all concept D-tuples from the models. Given a set of D concepts, $\{c_i^1, \ldots, c_i^D\}$, and corresponding binary support masks, $\{\mathbf{B}_i^1, \ldots, \mathbf{B}_i^D\}$, we define the similarity score of these concepts as

$$R_i^D = \frac{|\mathbf{B}_i^1 \cap \cdots \cap \mathbf{B}_i^D|}{|\mathbf{B}_i^1 \cup \cdots \cup \mathbf{B}_i^D|}. \tag{4.7}$$

Naively computing the similarity between all D-tuples results in an exponential number of computations and is intractable for even small D. To mitigate these issues, we exclude two types of concepts: (i) unimportant ones and (ii) those with a low R-score among d-tuples, where $d < D$. More specifically, we only consider the most important $\epsilon\%$ of concepts from each model. We then iterate over $d \in \{2, \ldots, D\}$ and filter out concepts that have R-scores less than δ for all d-tuples in which it participates. Formally, the filtered Rosetta d-concept scores are defined as

$$\mathbf{R}_{\epsilon,\delta}^d = \{R_i^d \,|\, R_i^d > \delta \,\forall R_i^d \in \mathbf{R}_\epsilon^d\}, \tag{4.8}$$

where $\mathbf{R}_{\epsilon}^{d}$ is the set of all R-scores among d concepts after the ϵ importance filtering. This results in a significantly smaller pool of candidates for the next stage, $d+1$, reducing the overall computational complexity of the algorithm. Finally, as some concepts may reside in a subset of the models but are still interesting to study, we examine the union of all important and confident Rosetta d-concepts corresponding to R-scores $\mathbf{R}_{\epsilon,\delta}^{2} \cup \cdots \cup \mathbf{R}_{\epsilon,\delta}^{D}$.

4.5 Experiments

We evaluate our concept discovery algorithm quantitatively and qualitatively across a variety of models and tasks.

4.5.1 Datasets

We primarily use two datasets in our experiments: TCOW Kubric [247] and Something-Something-v2 (SSv2) [103]. The former is a synthetic, photo-realistic dataset of 4,000 videos with randomized object location and motion, based on the Kubric synthetic video generator [104]. This dataset is intended for semi-supervised VOS (semi-VOS) through occlusions. SSv2 contains 220,847 real videos intended for fine-grained action recognition. Each sample is a crowdsourced video of a person-object interaction (i.e., doing something to something). Unlike other video classification benchmarks [38,228], temporal reasoning is fundamental for distinguishing SSv2 actions, making it an ideal choice for analyzing spatiotemporal mechanisms.

4.5.2 Models

We evaluate four models with public pretrained checkpoints: (i) TCOW [247] trained on Kubric for semi-VOS, (ii) VideoMAE [243] trained on SSv2 for action classification (Supervised VideoMAE), (iii) VideoMAE self-supervised on SSv2 (SSL VideoMAE), and (iv) InternVideo [258], a video-text model trained contrastively on 12M clips from eight video datasets and 100M image-text pairs from LAION-400M [215]. As TCOW requires a segmentation as input, when applying it to SSv2, we manually label the most salient object in the initial frame.We focus our analysis on the first two models, and use the last two to validate the universality of our Rosetta concepts.

4.5.3 Implementation details

For all experiments, we run VTCD on 30 randomly sampled videos and discover concepts from all heads and layers. We focus on keys in the self-attention heads, as they produce the most meaningful clusters per prior work [10] and present video results with queries and values on the project page.

	TCOW		VideoMAE	
Model	Positive ↓	Negative ↑	Positive ↓	Negative ↑
Baseline + Occ	0.174	0.274	0.240	0.300
Baseline + CRIS	0.166	0.284	0.157	0.607
VTCD (Ours)	**0.102**	**0.288**	**0.094**	**0.625**

Table 4.1: Comparison of our tubelet proposal approach to the widely used random crop tubelets [83, 86, 272] with occlusion-based importance [83] (Baseline + Occ) for both TCOW [247] and VideoMAE [243]. Our tubelets result in concepts that are more faithful to the model's representations even when the baseline is equipped with our concept scoring algorithm (Baseline + CRIS).

Concept discovery When generating tubelets (Section 4.3.1.1), we use 12 segments and set all other hyperparameters to the Scikit-Image [246] defaults, except for the compactness parameter, which is tuned on a held-out set for each model to the following values: TCOW - 0.01, VideoMAE - 0.1, SSL-VideoMAE - 0.1, InternVideo - 0.15. When clustering concepts using CNMF (Section 4.3.1.2) we follow the same protocol as [10] and use the Elbow method, with the Silhouette metric [209] as the distance, to select the number of clusters with a threshold of 0.9.

Concept importance For all importance rankings using CRIS, we use the original loss the models were trained with. For InternVideo, we use logit value for the target class by encoding the class name with the text encoder, and then taking the dot product between the text and video features. We use 4,000 masking iterations for all models, except for TCOW [247], where we empirically observe longer convergence times and use 8,000 masks.

4.5.4 VTCD target metrics

To discover and rank concepts, VTCD requires a target evaluation metric. For TCOW Kubric, we use the intersection-over-union (IoU) between the predicted and the groundtruth masks. For SSv2, we use classification accuracy for a target class.

4.5.5 Quantitative evaluation

This section presents a quantitative validation of VTCD's key components compared to various baselines. We first assess our tubelet proposal methodology, followed by a comparison of CRIS with other concept importance methods. We follow the standard evaluation protocol, and measure the *fidelity* of the discovered concepts [86, 98, 281]. To this end, we calculate attribution curves [83, 86, 98], where concepts are removed from a model in either most-to-least order (*positive* perturbation), or least-to-most (*negative* perturbation). The intuition is that concepts, and corresponding importance scores, with higher fidelity will have a steeper performance decrease when removing the most important concepts, and vice-versa for the reverse order.

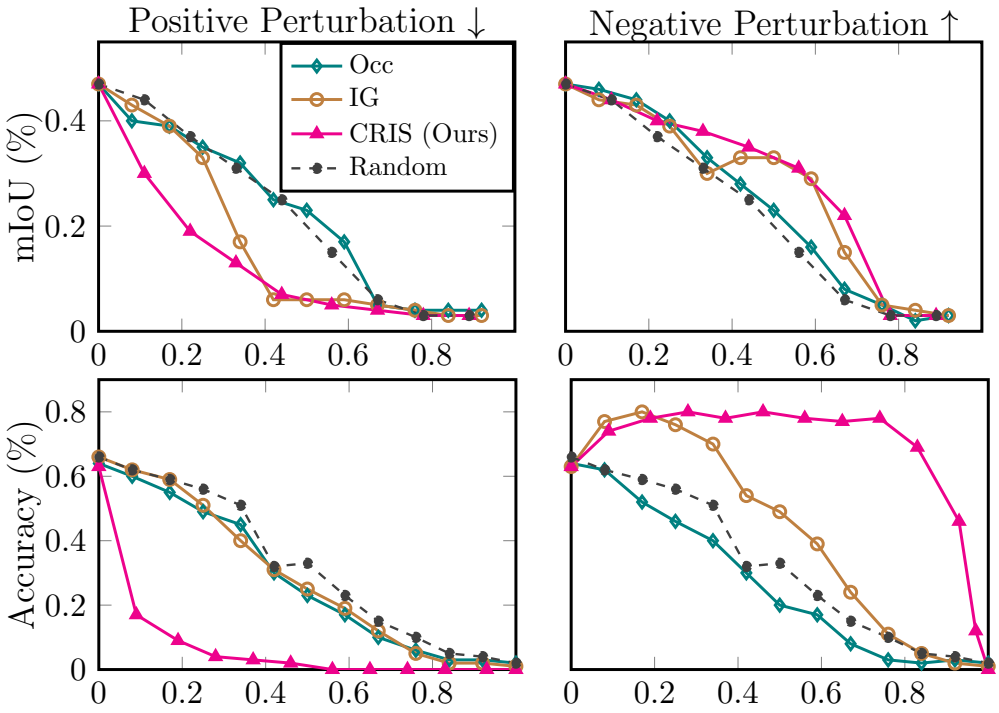


Figure 4.4: Attribution curves for every layer of TCOW trained on Kubric (top) and VideoMAE trained on SSv2 (bottom). We remove concepts from most-to-least (left) or least-to-most important (right). CRIS produces better concept importance than methods based on single concept occlusions (Occ) or gradients (IG).

Model	Accuracy ↑	GFLOPs ↓
Baseline	37.1	180.5
VTCD 33% Pruned	**41.4**	121.5
VTCD 50% Pruned	37.8	**91.1**

Table 4.2: Pruning unimportant heads in VideoMAE results in improved efficiency and accuracy when targeting a subset of classes. Here, we target the six SSv2 classes containing types of spills.

4.5.5.1 Tubelet validation

Unlike previous methods that partition the inputs into proposals in the pixel space, VTCD generates the proposals via SLIC [3] clustering in the model's feature space. We ablate this design choice by comparing VTCD with a strong *random cropping* baseline used by recent image-based concept discovery methods (ConceptSHAP [272], CRAFT [86], Lens [83]), shown as 'Baseline + Occ' in Table 4.1. In all cases, VTCD yields concepts that are more faithful to the model's representation. To further isolate the effect of proposals on the performance, we equip the baseline with our concept importance estimation approach (shown as 'Baseline + CRIS'). We observe that VTCD still outperforms this strong baseline in all settings, validating that generating tubelet proposals in the feature space of the transformer indeed yields concepts that are more faithful to the model's representation.

Method	Target	Occluders	Containers
Baseline	3.0	31.5	43.3
VTCD (Ours)	**19.2**	**69.7**	**73.8**
TCOW (supervised)	36.8	76.8	78.2

Table 4.3: Evaluating the accuracy of object tracking concepts found in the TCOW model by VTCD and the random crop baseline used in other recent methods [83, 86, 272].

4.5.5.2 Concept important evaluation

In Figure 4.4, we plot concept attribution curves of our method for TCOW for Kubric (top) and supervised VideoMAE for SSv2, targeting 10 randomly sampled classes and averaging results (bottom). In addition, we report several baselines: (i) concept removal in a random order, (ii) standard, occlusion-based concept importance estimation [83], and (iii) a gradient based approach [83, 141]. In all cases, CRIS produces a more viable importance ranking, dramatically outperforming both random ordering and the occlusion baseline. The integrated gradients method performs similarly to ours for TCOW, but is much worse for the action classification VideoMAE.

Notably, we observe that the performance actually *increases* for VideoMAE when up to 70% of the least important concepts are removed. Recall that SSv2 is a fine-grained action classification dataset. VTCD removes concepts that are irrelevant for the given class, hence increasing the robustness of the model's predictions. We further quantify this effect in Table 4.2, where we focus on the six classes where a 'spill' happens. We then use CRIS to rank the *heads* in VideoMAE according to their effect on performance using the training set and report the results for pruning the least important ones on the validation set. Pruning 33% of the heads actually *improves* the accuracy by 4.3% while reducing FLOPS from 180.5 to 121.5. Further removing 50% of the heads retains the original performance (+0.7%) and reduces FLOPs to 91.1. These results show that one can tune the trade-off between performance and computation using VTCD for pruning.

4.5.5.3 Concept validation

Directly measuring concept accuracy is not possible in an open world approach as we don't know a priori what concepts should be present in a model. There are, however, special cases where we can directly measure the accuracy of *some* of the concepts. For example, TCOW is trained to track through occlusions, so we can expect to find concepts that correspond to the target object and containers/occluders in it. We perform this evaluation for VTCD and the random crop baseline [83, 86, 272] and report the mIoU between the best found concepts and the groundtruth masks in Table 4.3 (top). These results validate the accuracy of VTCD, which is able to reach up to 94% of the performance of the fully-supervised TCOW by discovering concepts in its intermediate representations.

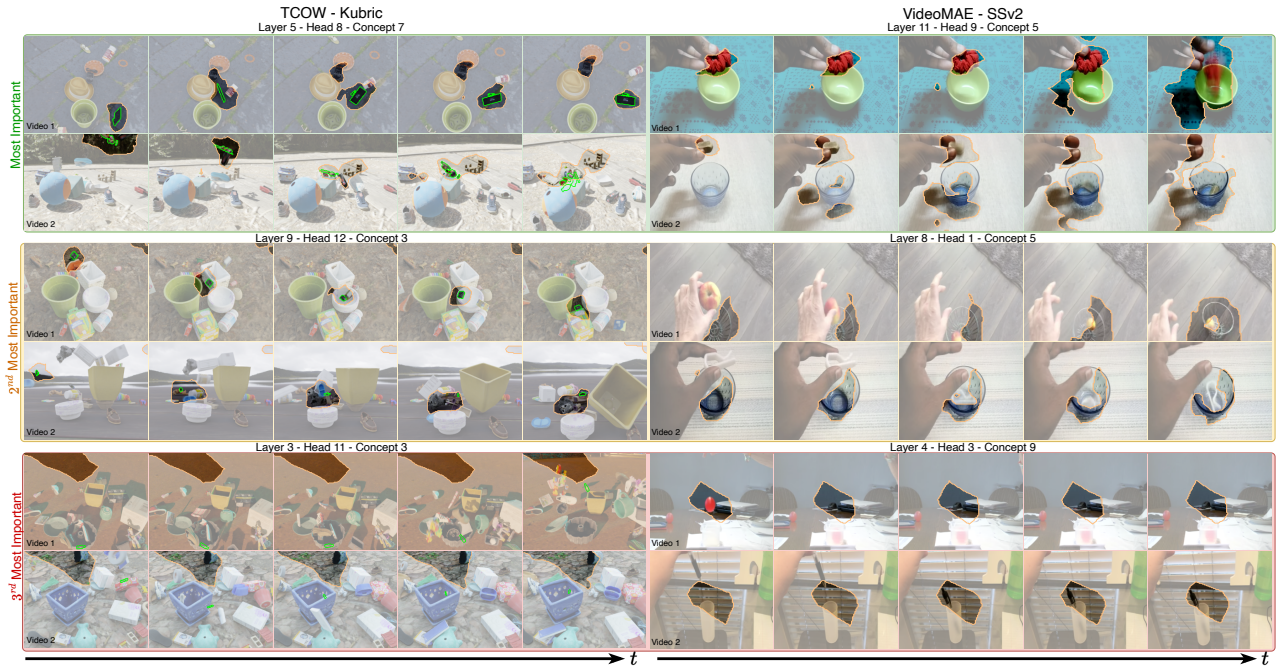


Figure 4.5: The top-3 most important concepts for the TCOW model trained on Kubric (left) and VideoMAE trained on SSv2 for the target class *dropping something into something* (right). Two videos are shown for each concept and the query object is denoted with a green border in Kubric. For TCOW, the 1^{st} and 2^{nd} (top-left, middle-left) most important concepts track multiple objects including the target and the distractors. For VideoMAE, the top concept (top-right) captures the object and dropping event (i.e. hand, object and container) while the 2^{nd} most important concept (middle-right) captures solely the container. Interestingly, for both models and tasks, the third most important concept (bottom) is a temporally invariant tubelet. See Section 4.5.7 for further discussion (and the video for full results).

4.5.6 Qualitative analysis

We have seen that the importance assigned to concepts discovered by VTCD aligns with the accuracy of the model. We now assess the concepts qualitatively. To this end, Figure 4.5 shows two representative videos for the top three most important concepts for the TCOW and VideoMAE models for the class *dropping something into something*.

For TCOW, the most important concept occurs in layer five and tracks the target object. Interestingly, the same concept highlights objects with similar appearance and 2D position to the target. This suggests that the model solves the disambiguation problem by first identifying possible distractors in mid-layers (i.e., five) and then using this information to more accurately track the target in final layers. In fact, the second most important concept, occurring in layer nine, tracks the target object throughout the video.

For VideoMAE, the most important concept highlights the object until it is dropped, after which the object and its container are highlighted. The second concept clearly captures the container, but notably not the object itself, making a ring-like shape. These concepts identify an important mechanism for differentiating similar classes (e.g., *dropping something into/behind/in-front of something*).

The third concept for each model captures similar information, occurring in early layers: a temporally invariant, spatial support. This corroborates research [10, 95] suggesting that positional information processing occurs early in the model, acting as a reference frame between semantic information and the tokens themselves. We now turn to Rosetta concepts: concepts *shared* across multiple models.

4.5.7 Rosetta concepts

We begin by applying VTCD to the four models, targeting two classes chosen due to their dynamic nature: *rolling something across a flat surface* and *dropping something behind something*. We then mine Rosetta concepts following Section 4.4.1, with $\delta = 0.15$ and $\epsilon = 15\%$ in all experiments. The resulting Rosetta 4-concepts contains 40 tuples with an average R-score (Equation 4.7) of 17.1. Note the average R-score between all possible 4-concepts is 0.6, indicating the significance of the selected matches. Figure 4.6 visualizes one mined Rosetta 4-concept, which captures an object tracking representation in the late layers. This demonstrates that universal representations indeed exist between all four models. Our project website shows more shared concepts as videos.

Next, we qualitatively analyze all Rosetta d-concept with $d \in \{2, 3, 4\}$ at various layers. Figure 4.7 shows a representative sample. In early layers, we find that the models learn spatiotemporal basis representations (Figure 4.7, left): they decompose the video space-time volume into connected regions, facilitating higher-level reasoning in later layers. This is consistent with prior works that show spatial position is encoded in early layers of image transformers [10, 95].

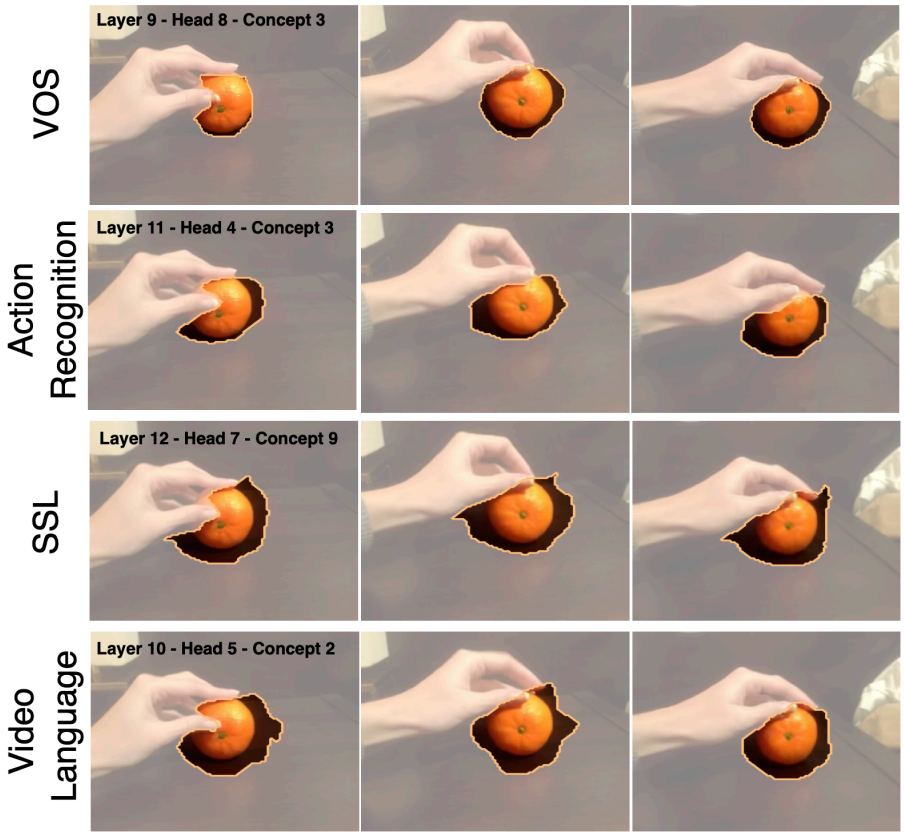


Figure 4.6: A sample Rosetta concept that is universal across four models trained for different tasks. Interestingly, we observe that all four models contain object centric representations. See Figure 4.7 for more.

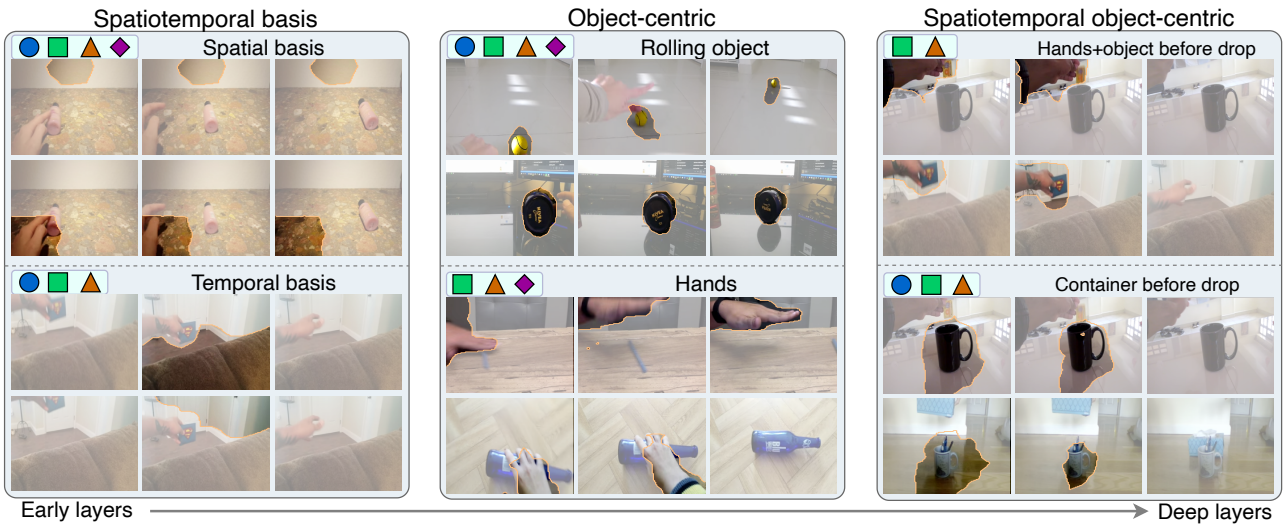


Figure 4.7: Universal concepts emerge in video transformers despite being trained for different tasks. Early layers encode spatiotemporal positional information. Middle layers track various objects. Deep layers capture fine-grained spatiotemporal concepts, e.g.,related to occlusion reasoning (see video for full results). Legend: ● TCOW, ■ Supervised VideoMAE, ▲ SSL VideoMAE, ◆ InternVideo.

Features	VTCD	VTCD + SAM [143]
VideoMAE-SSL	45.0	68.1
VideoMAE	43.1	66.6
InternVideo	45.8	68.0

Table 4.4: We apply VTCD to discover object tracking concepts and evaluate them with mIoU on the DAVIS'16 validation set.

In the mid-layers (Figure 4.7, middle), we find that, among other things, all the models learn to localize and track individual objects. This result introduces a new angle to the recently developed field of object-centric representation learning [14, 91, 164, 217]: it invites us to explore how specialized approaches contribute, given that object concepts naturally emerge in video transformers. In addition, all models, except for synthetically trained TCOW, develop hand tracking concepts, confirming the importance of hands for action recognition from a bottom-up perspective [230, 277].

We validate the accuracy of the discovered object-centric concepts on the DAVIS'16 video-object segmentation (VOS) benchmark [193]. To this end, we run VTCD on the training set and select concepts that have the highest overlap with the groundtruth tracks. We then use them to track new objects on the validation set and report results in Table 4.4. In this analysis, we solely evaluate representations that are not directly trained for VOS. While solid performance is achieved by all representations, segmentation accuracy is limited by the low resolution of the concept masks. To mitigate this, we introduce a simple refinement step that samples random points inside the masks to prompt SAM [143]. The resulting approach, shown as 'VTCD + SAM', significantly improves performance. We anticipate that future developments in video representation learning will automatically lead to better VOS methods with the help of VTCD.

Finally, deeper layers contain concepts that build on object-centric representations to capture spatiotemporal events. For example, three models learn to identify containers before an object is dropped into them and two models track the object in the hand until it is dropped. One notable exception is InternVideo [258], which is primarily trained on images with limited spatiotemporal modeling. Perhaps surprisingly, these concepts are also found in the self-supervised VideoMAE [243], which was never trained to reason about object-container relationships. This raises the question: Can intuitive physics models [40, 264] be learned via large-scale video representation training?

4.6 Conclusion

In this work, we introduced VTCD, the first algorithm for concept discovery in video transformers. We empirically demonstrated that it is capable of extracting human-interpretable concepts from video understanding models and quantifying their importance for the final predictions. Using VTCD, we discovered shared concepts among several models with varying objectives, revealing common processing patterns, such as a spatiotemporal basis in early layers. In later layers, useful, higher-level representations universally emerge, such as those responsible for object tracking. Large-scale video representation learning is an active area of research at the moment [16, 26, 33] and our approach can serve as a key to unlocking its full potential.

Chapter 5

Visual Concept Connectomes: Multi-layer concept discovery and their interlayer connections

5.1 Introduction

This chapter focuses on interpreting the intermediate representations of deep networks for computer vision. The goal is understanding how various encoded concepts impact a model's prediction as well as concepts in other layers. We define concepts as abstractions that generalize from particular instances, including those defined locally (e.g., color and orientation), regionally (e.g., texture and shading) and from higher-level considerations (e.g., object parts, wholes and groupings). Human-interpretable concepts are of particular interest in increasing human understanding of models; however, extracting these concepts encoded in deep networks remains an open challenge in computer vision due to the complexity and opaque nature of these models.

Understanding what concepts are learned by deep models and how they are encoded is important for both science and applications. For science, understanding what information drives a model's encoding of different concepts may indicate directions to advance performance. For applications, several negative consequences of deploying opaque vision models have been documented, e.g. [35, 108].

Previous work has focused on interpreting models via feature attribution, which measures the contribution of individual inputs to a model's output [43, 218, 283]; so, explanations are of single pixels and may be difficult to interpret. Other work generates images that maximize activation of a model's features [81, 184, 275]. Like feature attribution, these approaches are qualitative and place most of the burden on the user to determine the concepts revealed. Concept-based interpretability, which identifies human-interpretable abstractions in a model's latent space [86, 98, 141, 146], yield

Figure 5.1: A Visual Concept Connectome (VCC). At each layer the visual concepts learned by a deep model for a given class are revealed as are the learned interlayer concept connections. For each concept, up to four exemplars are shown as unmasked regions in a 2×2 image. Interlayer concept connections are shown as lines with darker lines indicating larger contributions. Shown is a VCC for every convolutional layer of a VGG16 model [233] trained on ImageNet [59] targeting recognition of class "Tow Truck". A closer visualization of VCC subgraphs reveals interesting compositions occurring at different levels of abstraction corresponding at different depths of the model. At early layers (bottom left), we observe oriented patterns ($C_{1,1}$) and brown color ($C_{1,2}$) composing the concept of green and brown orientation ($C_{2,1}$). Middle layers (right) show the concept of 'wheel on the road' ($C_{9,1}$) being composed of wheels ($C_{8,1}$) and regions of asphalt ($C_{8,2}$). The final layer concepts (top left) show that both foreground objects, e.g., tow trucks ($C_{13,1}$), and background regions, e.g., road, trees, humans, or car being towed ($C_{13,2}$), concepts highly influence the final category ($C_{TowTruck}$).

quantitative contributions of a concept to the model's output and explanations on the class-level (vs. pixel-level). These approaches are easier to understand and validate; however, they have not been used to explore *interlayer* relationships.

As it stands, no approaches can quantify the interlayer affect of a given distributed concept at one layer, l, to another concept at a different layer, l' (rather than the model *output*). Even though it is well established that deep networks learn to build concepts hierarchically as information flows through the network [185, 275], understanding the hierarchical representations has been under-researched. Indeed, little is known about the characteristics of this concept hierarchy for today's models. Questions abound: *How many concepts exist in a network? What are the connections and weights between concepts? Does the model architecture impact the hierarchical structure of concept abstractions?*

In response to these questions, we take inspiration from the biological notion of a *connectome* [219], defined as "a comprehensive structural description of the network of elements and connections forming a brain." Analogously, we present the *Visual Concept Connectome* (VCC), a comprehensive structural description of a deep pretrained network model in terms of human-interpretable concepts and their relationships that form the internal representation maintained by the model; Figure 5.1 shows an example. Notably, VCC generation works in the *open-world* setting, i.e., the concepts and interlayer connections are discovered without the need for any predefined concept dictionary.

5.2 Related work

While a number of intrinsically interpretable networks have appeared (e.g. [27,47,144,212,279,280]), we focus on work that, like ours, endeavours to interpret black-box models.

Concept-based interpretability. Closed-world concept interpretability considers cases where a labelled dataset defines the concepts of interest for post-hoc analysis; approaches include Network Dissection [18] and Testing with Concept Activation Vectors (TCAV) [141]. However, a desirable property for concept-based interpretability is not to be restricted to a set of predefined classes (i.e., closed-world), but to support discovery of new and previously unlabelled concepts (i.e., open-world). Approaches to open-world concept interpretabilty include Automatic Concept Explanations (ACE) [98], SegDiscover [121], ConceptSHAP [272], Invertible Concept Explanations [281] and CRAFT [86]. These approaches are limited to single layer analysis and do not explain interlayer relationships.

Activation maximization [81, 168, 184, 275] visualizes the input that most activates a model component (e.g., filter, layer or logit) or how same layer units combine information [183]. These approaches do not capture interlayer relations and place a heavy burden on the user to interpret what concepts appear in their output where there may be no clear resemblance to natural concepts. Class Activation Maps (CAMs) and extensions visualize an input image's local region that contributes to a model's output [41, 218, 283]. Layer-wise Relevance Propagation [24] (LRP) generates pixel contribu-

tion heatmaps by assuming conservation of information is propagated through each neuron. These approaches interpret single images (not classes), are purely qualitative, and are sensitive to small input perturbations [97, 142], yet insensitive to model changes [6]. Less related are directions interpreting generative models, e.g. [17, 162].

Interpretability via decomposition. Most closely related to our work are those that decompose a model's prediction into interpretable components. CRAFT [86] extracts concepts from the last layer of a convolutional neural network (CNN), but also searches earlier layers for sub-concepts for the maximally activating images of a given concept, but do not provide any way to quantify the contribution of sub-concepts to concepts. Another approach combined CAMs of multiple concepts to explain the final prediction [285]; however, it is limited to labelled data (so cannot discover concepts outside training) and only explains a final output, not intermediate relations. Other work interprets CNNs by distilling them into a graph [278]; however, it applies only to individual filters conceived as capturing object parts (i.e., not concepts other than objects and their parts, e.g., textures, colors or grouping), does not consider the fact that representations are encoded via superposition [31, 70, 183] (i.e., more than one concept can be captured per-channel), does not yield meaningful edge weights and is only for CNNs. Yet other work extends Grad-CAM or LRP to perform intra-layer visualizations [4, 49]. While these approaches capture concepts other than object parts, they still ignore the core problem of superposition (and distributed representations) and do not yield interpretation for an entire class, just single images.

Contributions. In the light of previous work, our contributions are fourfold. (i) We introduce and formalize the notion of a Visual Concept Connectome (VCC) for deep network models. The VCC reveals concepts represented at any given layer of a network as well as their interlayer connectivity. (ii) We present a method for extracting a VCC from any pretrained deep network in an unsupervised, open world setting, with a focus on classification. (iii) We validate our approach with quantitative and qualitative experiments wherein we examine various standard models to yield insights into model architectures and training tasks. (iv) We apply VCCs to explaining failure modes in deep models.

5.3 Visual Concept Connectomes (VCCs)

A VCC is a directed acyclic graph, $\mathbf{G}(\mathbf{Q}, \mathbf{E})$, created by distilling a pretrained deep network. The graph is topologically sorted based on the layer order found in the original network and has $n+1$ layers, consisting of n network layers to be analyzed (i.e., any subset of layers that a user selects, including all layers at the extreme) plus the final prediction (e.g., the object category for object recognition). The graph's nodes, $\mathbf{Q}$, are vectors (cluster centroids) representing interpretable concepts and its edges, $\mathbf{E}$, are scalars representing the contribution of one concept to the existence of another.

To construct a VCC, three inputs are required: (i) a set of I images representative of a given task

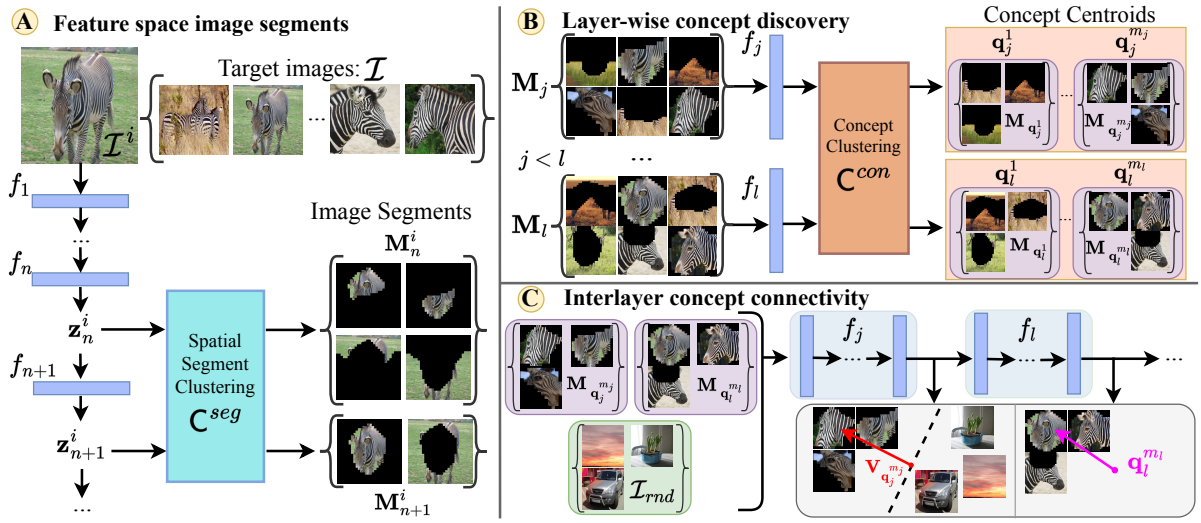


Figure 5.2: The three steps in building a Visual Concept Connectome (VCC). (**A**) For a given image, $\mathcal{I}^i \in \mathcal{I}$, model, F, and layer, n, we produce a set of image segments, $\mathbf{M}_n^i \in \mathbf{M}_n$, based on a recursive spatial clustering, C^{seg} (5.2), of the features $\mathbf{z}_n^i$ conditioned on the clusters from the layer above, $n+1$. We then use (5.3) to generate a set of masked RGB image segments for each layer, $\mathbf{M}_j$. (**B**) For a given layer, j, we pass the image segments from all images, $\mathbf{M}_j$, through f_j and cluster, C^{con} (5.4), these features across the dataset to produce m_j concept centroids, $\{\mathbf{q}_j^1, \ldots, \mathbf{q}_j^{m_j}\}$. (**C**) To measure the contribution of an earlier layer concept, $\mathbf{q}_j^{m_j}$, to a later layer concept, $\mathbf{q}_l^{m_l}$, we employ our Interlayer Testing with CAV (ITCAV) approach (Section 5.3.3), which uses the Concept Activation Vector (CAV) [141] of the earlier concept, $\mathbf{V}_{\mathbf{q}_j^{m_j}}$ (that points away from random examples, $\mathcal{I}_{rnd}$, but toward concept exemplars, $\mathbf{M}_{\mathbf{q}_j^{m_j}}$), and the deeper layer concept, $\mathbf{q}_l^{m_l}$.

(e.g., exemplar images of an object category for object recognition), $\boldsymbol{\mathcal{I}} = \{\mathcal{I}^1, \ldots, \mathcal{I}^I\}, \mathcal{I}^i \in \mathbb{R}^{h \times w \times c}$, with h, w and c the image height, width and channel dimension (e.g., three for RGB), respectively, (ii) an N layer network, $F(\cdot) = \{f_1(\cdot), \ldots, f_N(\cdot)\}$, with f_j denoting feature extraction at layer j, and (iii) a set of n selected layers (a subset of F, possibly improper), which are to be studied.

A VCC is constructed in three main steps. (i) Image segments are extracted in the model's feature space via divisive clustering to produce semantically meaningful image regions for each selected layer (Section 5.3.1). (ii) Layer-wise concepts, i.e., the nodes of the graph, are discovered in an open world fashion (i.e., no labelled data is required) via a second round of clustering over the dataset of image regions, independently for each layer (Section 5.3.2). (iii) Edges are calculated that indicate the contribution of concepts from earlier to deeper layers via an approach we introduce, Interlayer Testing with Concept Activation Vectors (ITCAVs) (Section 5.3.3). We use object recognition for the explanation of the approach; nevertheless, it could be extended to other tasks in a straightforward way (e.g., semantic segmentation). An overview of these steps is provided in Figure 5.2.

5.3.1 Feature space image segments

For each selected layer, we produce a set of image regions that plausibly belong to a concept encoded by the model at that layer. Previous work produces concept segments in RGB space, i.e., superpixels or random crops [86, 98]; however, segmentation based on features different from those over which concepts are to be discovered, i.e., the model's deep features, results in the support for the discovery process being divorced from the process that generated the support. Therefore, our segmentation uses the same deep features to be used subsequently for concept discovery. Our divisive clustering approach is presented visually in Figure 5.2 (**A**). We pass the entire set of I input images, $\boldsymbol{\mathcal{I}}$, for a given class through the model, F, to get features, $\mathbf{z}_n^i \in \mathbb{R}^{h_n \times w_n \times c_n}$, at each selected layer, n according to

$$\mathbf{Z}_n = \{f_n(\mathcal{I}^1), ..., f_n(\mathcal{I}^I)\} = \{\mathbf{z}_n^1, ..., \mathbf{z}_n^I\}. \tag{5.1}$$

To extract image segments at layer n for concept discovery (Section 5.3.2), we cluster the image activations, $\mathbf{Z}_n$, conditioned on the clusters (i.e., masks) generated at the next higher layer, $n+1$. Let $\mathbf{p} = (x, y)$ index spatial coordinates and $\mathbf{B}_n^i(\mathbf{p}; \gamma)$ be a binary mask for cluster γ such that $\mathbf{B}_n^i(\mathbf{p}; \gamma) = 1 \iff \mathbf{z}_n^i(\mathbf{p}) \in \gamma$; otherwise, $\mathbf{B}_n^i(\mathbf{p}; \gamma) = 0$.

We calculate a set of such masks by applying a clustering algorithm, $\mathsf{C}_\Gamma^{seg}(\cdot)$, that returns binary support masks for Γ clusters on its argument. Let Γ_{n+1} be the number of segments at layer $n+1$ and $1 \leq g \leq \Gamma_{n+1}$ index a particular segment, g. For each g we calculate segments at layer n as

$$\left\{\mathbf{B}_n^i(\mathbf{p}; \gamma)\right\}_{\gamma=1}^{\Gamma} = \mathsf{C}_\Gamma^{seg}\big(\mathbf{z}_n^i(\mathbf{p}) \odot \widetilde{\mathbf{B}}_{n+1}^i(\mathbf{p}; g)\big), \tag{5.2}$$

where $\widetilde{\mathbf{B}}_{n+1}^i$ is $\mathbf{B}_{n+1}^i$ upsampled to the resolution of layer n and $\odot$ indicates spatial element-wise masking applied to the individual image activations. The element-wise masking ensures clustering is done on the masked area of $\mathbf{z}_n^i$. Thus, the support of the concept discovery at layer n comes from the support of concepts at layer $n+1$. Notably, we let Γ vary with g; so, different segments, g, at layer $n+1$ can yield a different number of segments, Γ, at layer n.

The top-down conditional clustering, (5.2), is repeated recursively for all image segments in all selected layers. To initiate the recursion, (5.2), at the top layer, n_{top}, all the features are used by setting the number of clusters to be one, and having $\widetilde{\mathbf{B}}_{n_{top}+1}^i(\mathbf{p}; 1) = 1$ for all images.

At each layer, n, we construct a set of RGB segmentations, $\mathbf{M}$, for use in concept discovery (Section 5.3.2) by applying upsampled masks, $\mathbf{B}_n^i(\mathbf{p}; \gamma)$, to a given image, $\mathcal{I}^i$, via

$$\mathbf{M}_n^i(\mathbf{p}; \gamma) = \mathcal{I}^i(\mathbf{p}) \odot \mathbf{B}_n^i(\mathbf{p}; \gamma), \tag{5.3}$$

where $\mathbf{M}_n^i(\mathbf{p}; \gamma) \in \{0, ..., 255\}^{h_n \times w_n}$, with $\{0, ..., 255\}$ specifying RGB value. We follow by defining

$\mathbf{M}_n$ as the set of all RGB image segments at layer n (with each segment given in terms of (5.3)), and letting $\mathbf{M} = \{\mathbf{M}_1, ..., \mathbf{M}_n\}$.

We instantiate the clustering, C^{seg}, via maskSLIC [126], an extension of k-means, with an l_2 norm, that clusters features while respecting a mask and automate selection of the number of clusters via the silhouette method [209].

5.3.2 Layer-wise concept discovery

Given the dataset of masked image segments for each layer, $\mathbf{M} = \{\mathbf{M}_1, \ldots, \mathbf{M}_n\}$, we follow [98] and bilinearly interpolate the segments to the input image size and then pass the segments through the model to get pre-segmented activations $\mathbf{Z}_{\mathbf{M}_j} = f_j(\mathbf{M}_j)$ from the layer where the segments were found in the divisive clustering step (Section 5.3.1). We pass the image segments (as opposed to using the image activations) through the model to remove any contextual information before we discover concepts (the segments are set to zero everywhere outside of the segment boundary). Otherwise, $\mathbf{M}_j$ could mix information from outside the segment.

The layer-wise concept discovery step is presented visually in Figure 5.2 (**B**). With the pre-segmented activations, $\mathbf{Z}_{\mathbf{M}_j}$, calculated for all layers, j, we cluster over the dataset of segment features for each individual layer to produce cluster centroids (i.e., concepts) according to

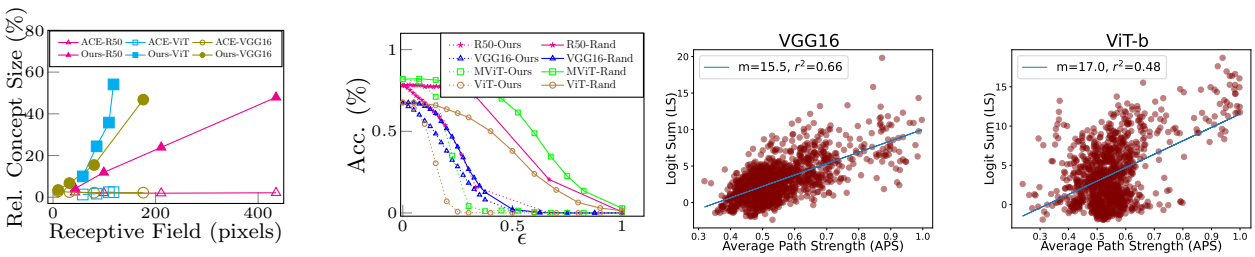
$$\mathbf{Q}_j = \{\mathbf{q}_j^1, \ldots, \mathbf{q}_j^{m_j}\} = \mathsf{C}_{m_j}^{con}(\mathsf{GAP}(\mathbf{Z}_{\mathbf{M}_j})), \tag{5.4}$$

where m_j is the number of concepts discovered at layer j, GAP is spatial global average pooling and $\mathsf{C}_{m_j}^{con}$ is a clustering algorithm that returns m_j cluster centroids. Following previous work [98], we instantiate $\mathsf{C}_{m_j}^{con}$ in terms of standard k-means, using the l_2 norm, and then use a large number of clusters with subsequent pruning to remove noisy clusters. Specifically, we follow previous work [98] and choose the number of clusters to be $k_m = 25$ in the concept discovery step. However, as VCCs target the discovery of concepts at potentially all layers, we select a different pruning protocol [98], where they prune based on a single minimum value. Instead, we prune clusters that have less than Y images, via the generalized logistic sigmoid

$$Y = A + \frac{K - A}{(C + Qe^{-Bt})^{1/\nu}}, \tag{5.5}$$

where $A = -102$, $K = 115$, $C = 1$, $Q = 1$, $B = 0.0004$ and $v = 1$. Pruning based on a sigmoid shaped function enables different levels of leniency when considering what constitutes a concept for each layer. This is crucial as different layers contain a different number of segments from the top-down segmentation algorithm (see Section 5.3.1).

The output of this stage results in discovered concepts (i.e., cluster centroids and associated seg-



(a) (b) (c) (d)

Figure 5.3: Validation of the three VCC method components. (a) Validation of segment proposals. The relative (Rel.) concept segment size compared to entire image for a given layer is plotted against the receptive field (RF) width/height of the same layer. (b) Validation of discovered concepts. For 50 randomly selected ImageNet classes, we discover concepts in four layers of the model. During inference, one randomly selected concept at each layer is suppressed by a factor of ϵ. (c, d) Validation of interlayer concept weights. The unnormalized logit sum (LS) scores, (5.9), for the target class are plotted against the average path strength (APS) scores, (5.8). A positive correlation implies that the ITCAV edge weights connecting a concept to the class are predictive of the model output having a higher probability for that class.

ments) for all layers as $\mathbf{Q} = \{\mathbf{Q}_1, \ldots, \mathbf{Q}_n\}$ for the centroids and $\mathbf{M}_{\mathbf{q}_j^{m_j}}$ their associated RGB segments.

5.3.3 Interlayer concept connectivity

Our approach to quantifying the contribution of discovered concepts between two layers, Interlayer Testing with Concept Activation Vectors (ITCAV), generalizes TCAV [141]. Specifically, while TCAV calculates the contribution of a single mid-layer concept to the class logit using a dataset of labelled concept images, we generalize to operate between any two layers and without a dataset of labelled concept images. Without loss of generality, we consider a single pair of concepts at two layers, which need not be adjacent.

The ITCAV method is presented visually in Figure 5.2 (**C**). We seek to measure the degree to which a concept at an earlier layer contributes to a deeper layer. To do so, we construct two vectors: (i) a vector that represents the concept at the earlier layer, j, and (ii) a vector that represents a positive contribution (i.e., gradient) of a concept at a later layer, l. The closer the alignment of these two vectors, the larger the contribution of the concept at layer j to the concept at layer l. Let $\mathbf{q}_j^{m_j}$ and $\mathbf{M}_{\mathbf{q}_j^{m_j}}$ be the cluster centroid and associated RGB image segments in layer j, and let $\mathbf{q}_l^{m_l}$ and $\mathbf{M}_{\mathbf{q}_l^{m_l}}$ be the cluster centroid and associated RGB image segments in a deeper layer l, i.e., $l > j$.

We proceed by constructing the vector that represents the concept in the feature space of the earlier layer, j. To do so, we employ the Concept Activation Vector (CAV) [141] for the earlier concept, $\mathbf{q}_j^{m_j}$, by training a linear classifier, $h(\cdot)$, on the features of layer j between positive concept inputs, $f_j(\mathbf{M}_{\mathbf{q}_j^{m_j}})$, and random images, $f_j(\boldsymbol{\mathcal{I}}_{rnd})$. The orthogonal vector to the hyperplane defined by $h(\cdot)$, denoted as $\mathbf{V}_{\mathbf{q}_j^{m_j}}$, is the CAV for concept $\mathbf{q}_j^{m_j}$ and points in the direction of concept $\mathbf{q}_j^{m_j}$ in the feature space of layer j.

Next, we construct the vector in the feature space of layer j that points in the direction of positive contribution of concept $\mathbf{q}_l^{m_l}$. To do so, we calculate $\nabla f_l(f_j(\cdot))$, i.e., the gradient of the deeper concept, $\mathbf{q}_l^{m_l}$, at layer l, with respect to the earlier layer, j. Our approach to measuring the sensitivity between the two concepts is then

$$S_{\mathbf{q}_j^{m_j},\mathbf{q}_l^{m_l}}(x) = \nabla\Big(||f_l(f_j(x)) - \mathbf{q}_l^{m_l}||_2\Big) \cdot \mathbf{V}_{\mathbf{q}_j^{m_j}}, \tag{5.6}$$

where $x \in \mathbf{M}_{\mathbf{q}_l^{m_l}}$. The sensitivity score should be positive because when the feature at layer j is perturbed in the direction of concept $\mathbf{q}_j^{m_j}$, i.e., in the direction $\mathbf{V}_{\mathbf{q}_j^{m_j}}$, then the feature vector in deeper layer, l, will be pushed closer to concept $\mathbf{q}_l^{m_l}$. Following TCAV [141], the final ITCAV edge weight, $e_{\mathbf{q}_j^{m_j},\mathbf{q}_l^{m_l}} \in \mathbf{E}$, is the ratio of positive sensitivities,

$$e_{\mathbf{q}_j^{m_j},\mathbf{q}_l^{m_l}} = \big|x \in \mathbf{M}_{\mathbf{q}_l^{m_l}} : S_{\mathbf{q}_j^{m_j},\mathbf{q}_l^{m_l}}(x) > 0\big| / \big|\mathbf{M}_{\mathbf{q}_l^{m_l}}\big|. \tag{5.7}$$

Key to our sensitivity score, (5.6), is the l_2 distance between the output, $f_l(f_j(x))$, and cluster centroid, $\mathbf{q}_l^{m_l}$, *before* taking the gradient, *cf.* CAV [141]. This choice serves two goals: (i) The subset of dimensions of individual image features, $f_j(x)$, that are not well aligned with the cluster centroid, $\mathbf{q}_l^{m_l}$, will be penalized. This penalty will suppress the impact of noisy dimensions from individual samples on ITCAV. We use the l_2 norm because the employed clustering approach used during concept discovery, k-means, uses l_2; see Section 5.3.2. (ii) Collapsing the tensor to a scalar (i.e., a concept similarity score) makes the gradient calculation computationally feasible compared to the full Jacobian.

5.4 Experiments

5.4.1 Implementation details

For evaluation we use ResNet50 [110], VGG16 [226], MobileNetv3 [118], ViT-b [248] and MViT [75] to sample a broad range of popular architectures. The layer names used (according to the PyTorch [190] module nomenclature) for each model when generating all four-layer VCCs are as follows:

1. ResNet18 [110], ResNet50 [110]: *Layer1, Layer2, Layer3, Layer4*
2. VGG16 [226]: *8, 15, 22, 29*
3. MobileNetv3 [211]: *0, 2, 4, 6*
4. MViT [75]: *1, 3, 9, 15*
5. ViT-b [248]: *2, 5, 8, 10*

The layer names used (according to the PyTorch [190] module nomenclature) for each model when generating the all-layer VCCs are as follows:

1. ResNet50 [199]: *layer1.0, layer1.1, layer1.2, layer2.0, layer2.1, layer2.2, layer2.3, layer3.0, layer3.1, layer3.2, layer3.3, layer3.4, layer3.5, layer4.0, layer4.1, layer4.2*
2. VGG16 [226]: *1, 3, 6, 8, 11, 13, 15, 18, 20, 22, 25, 27, 29*
3. MobileNetv3 [211]: *0.0, 1.0, 1.1, 2.0, 2.1, 2.2, 3.0, 3.1, 3.2, 3.3, 4.0, 4.1, 5.0, 5.1, 5.2, 6.0*
4. MViT [75]: *0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15*
5. ViT-b [248]: *0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10*

Following previous work [98,141], we perform a two-sided t-test on ITCAV scores for multiple runs of the same concept vs. different random sets of images to remove statistically insignificant scores and randomly sample from the Broden dataset [18] to get the images, $\boldsymbol{\mathcal{I}}_{rnd}$ (Section 5.3.3), used in statistical testing and CAV training. On average, four and all-layer VCCs take 15 minutes and 36 hours, resp., to generate on an NVIDIA Quadro RTX 6000 GPU.

5.4.2 VCC component validation

Segment proposal validation. Our approach segments concepts based on the feature space of a given layer to allow us to capture concepts across multiple layers (Section 5.3.1). So, the spatial support of valid segments at a layer should follow the receptive field (RF) of filters at that layer, as the segments are constrained by the filters from which they are derived. For comparison, we consider ACE [98], which does segmentation with a layer independent feature (color), limited to the input image to discover concepts at a single layer for a single model. Being directly related to the features at a layer, our approach should produce concepts that better respect the RF at that layer compared to the baseline, whose features are layer independent (e.g., at CNN early layers the RF is small; so, intuitively the patch proposals should be smaller as well and conversely for the later layers).

Figure 5.3 (a) shows the average concept size (in terms of the ratio of segment pixels to the entire image) vs. the RF at the corresponding layer for three different models: (i) ResNet50 [110], (ii) VGG16 [226] and (iii) ViT-b [66] (note we use mean attention distance instead of receptive field for ViT-b). While the baseline [98] produces segments with sizes invariant to the layer analyzed, our approach produces segments with image sizes that scale with the RF. Overall, it is expected by design and we find that our segment proposals follow RFs at each layer.

Concept fidelity. Concept fidelity measures the meaningfulness of discovered concepts with respect to the target model. While other work focuses on single layer concept fidelity [86, 98, 141], we

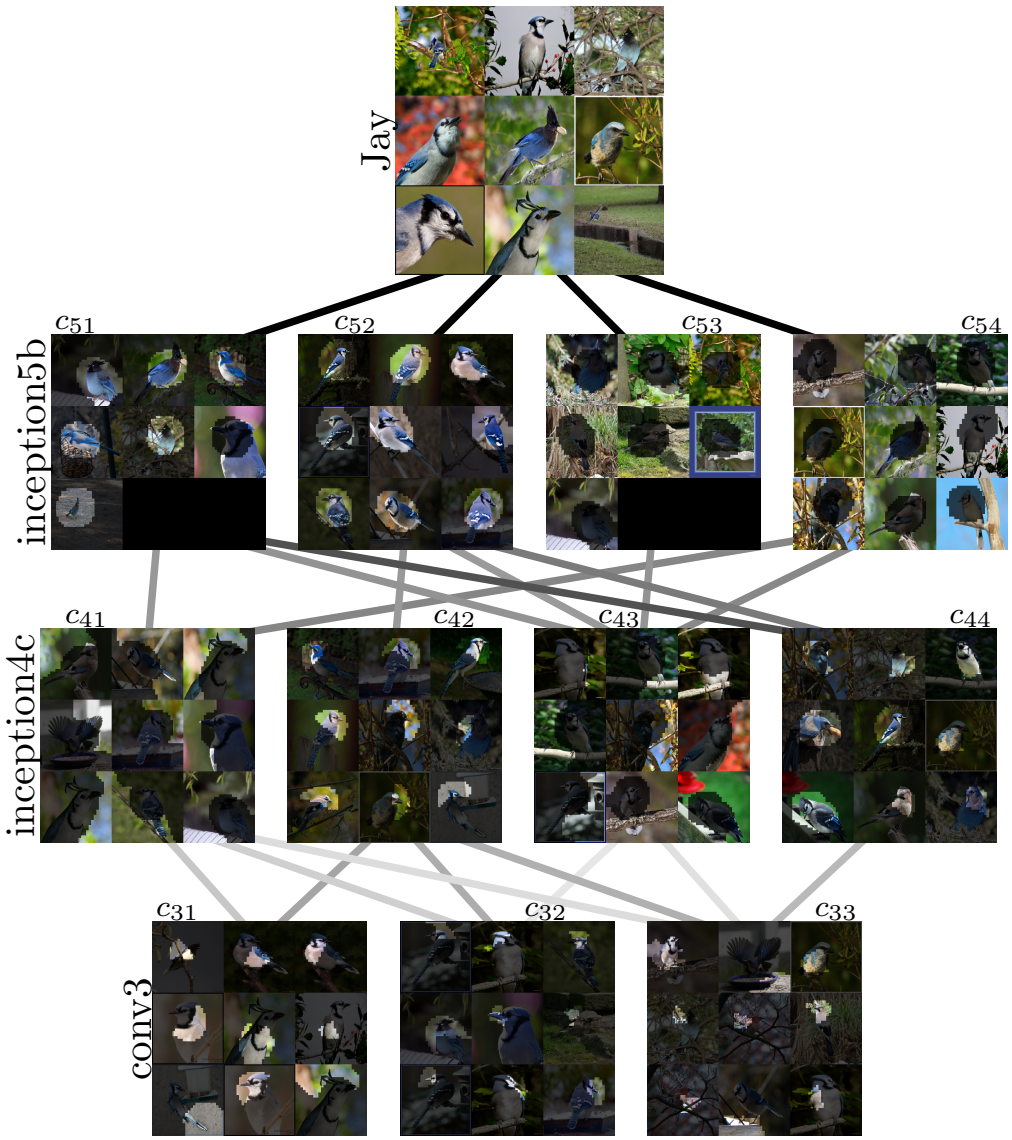


Figure 5.4: A VCC for three selected layers of a GoogLeNet model [233] targeting recognition of class 'Jay'. Darker lines denote stronger connection weights.

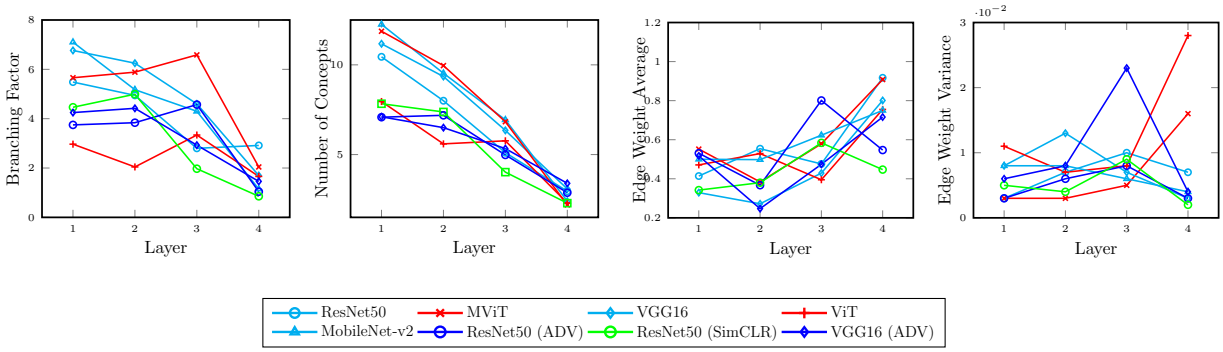


Figure 5.5: Graph metrics on four layer VCCs comparing CNN vs. transformer architectures and training objectives.

measure the effect on a model's output when suppressing the encoding of concepts as the information propagates through the model. The expectation is that suppressing the discovered concepts should result in a quicker decrease in performance compared to a non-concept direction (e.g., a randomly selected direction) as the amount of suppression gets larger. For a given model and category, we compute the concepts at various layers using our method (Section 5.3.2). We perform concept suppression for a given image, $\boldsymbol{\mathcal{I}}^i$, at a given layer, j, for concept $\mathbf{q}_j$, according to $\mathbf{z}_j^i = \mathbf{z}_j^i - \epsilon \cdot \mathbf{q}_j / ||\mathbf{q}_j||_2$, where ϵ controls the degree of perturbation. If the concepts discovered are meaningful, then the accuracy for the model's target class should decrease faster compared to random perturbations, $\mathbf{q}_{rnd}$, as ϵ increases.

Figure 5.3 (b) has concept suppression results averaged over 50 randomly selected ImageNet [59] classes, where a concept is randomly chosen to be suppressed at each layer. Note that the ϵ values required to reduce the model accuracy to zero differs by model; so, we scale the ϵ values to $[0, 1]$ for visualization purposes. For all models, it is seen that perturbing in the opposite direction of a recovered concept more severely impacts the accuracy than a random perturbation.

ITCAV validation. If the interlayer concept connection weights are meaningful, then the accumulated weights from earlier to later layer concepts should be correlated with their probability of predicting the target class. We use this intuition to validate our ITCAV approach to recovering interlayer concept weights as follows. We define the average path strength (APS), a scalar value between zero and one representing the average strength of the connection between a concept and the class logit. Given concept $\mathbf{q}_j$ at layer j, let there be L layers in the VCC between the concept and the logit layer; thus, each path from the concept to the logit consists of L edge weights. We consider all possible paths in the VCC from this concept to the class logit. Let $e_{\mathbf{q}_j}(l, p)$ be the edge weight (i.e., ITCAV score) of the p^{th} path (from concept $\mathbf{q}_j$ to the class concept) for VCC layer l. To calculate the APS score for a given concept, we average over all paths, P, and edge weights in each path to get

$$\text{APS}(\mathbf{q}_j) = \frac{1}{P} \sum_{p=1}^{P} \Big[\frac{1}{L} \sum_{l=1}^{L} e_{\mathbf{q}_j}(l, p) \Big]. \tag{5.8}$$

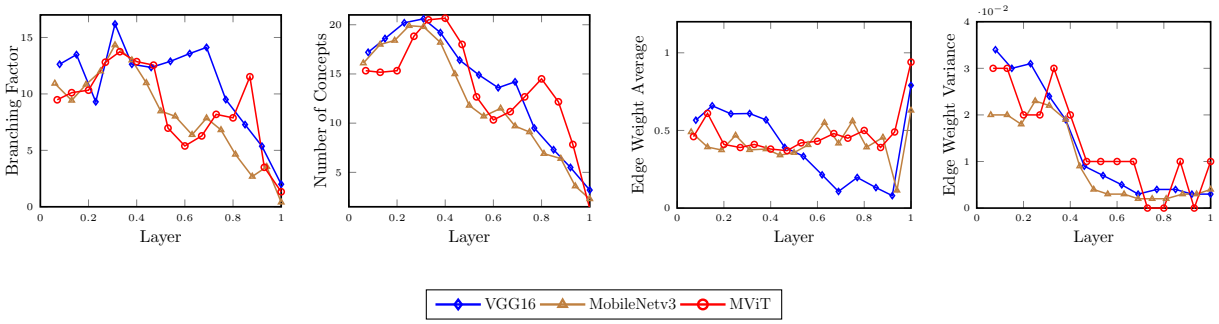


Figure 5.6: Graph metrics of all layer VCCs for three diverse architectures. Layer number normalized to allow for comparison of models with different numbers of layers.

Next, we calculate the logit sum (LS) score for a given concept, $\mathbf{q}_j$, by passing all of the corresponding masked segments, $\mathbf{M}_{\mathbf{q}_j}$, through the model, $F(\cdot)$. We then sum up the logit scores for the target class, c, to yield

$$\text{LS}(\mathbf{q}_j) = \sum_{i=1}^{|\mathbf{M}_{\mathbf{q}_j}|} F(\mathbf{M}^i_{\mathbf{q}_j})|_c, \tag{5.9}$$

where $\mathbf{M}^i_{\mathbf{q}_j}$ denotes the i^{th} segment mask, and $F(\cdot)|_c$ denotes the c^{th} logit score. If the ITCAV edge weights are meaningful, then there should be positive correlation between the APS and LS scores, i.e., $\text{APS}(\mathbf{q}) \propto \text{LS}(\mathbf{q})$.

Figure 5.3 (c, d) shows the LS scores plotted vs. the APS scores for VGG16 [110] and ViT-b [248]. A positive correlation is found between the APS and LS scores for both architectures. These results suggest that the combination of ITCAV scores is predictive of whether a concept is representative of the target class.

5.4.3 Understanding models with VCCs

To demonstrate VCC's unique ability to interpret models at different resolutions, we present model analyses in three parts: (i) qualitative analyses of VCCs generated for a subset of layers as well as at all layers, (ii) quantitative analysis of VCCs generated for a subset of four layers, (iii) a broadened quantitative analysis to encompass *all* model layers.

Visualizing concept hierarchies. Figure 5.4 shows a three layer VCC for GoogLeNet [233] targeting the class "Jay". Notice how the inception5b 'bird' concepts (c_{51}, c_{52}) form as selective weighting of inception4c concepts background (c_{41}), bird part (c_{42}, c_{44}) and tree branch (c_{43}), while the inception5b background concepts (c_{53}, c_{54}) form differently from weighting of solely inception4c background part concepts (e.g., tree branch (c_{43}) and green leaves (c_{41})). Notably, the network separates subspecies of Jay in the final layer (e.g., Blue Jay (c_{52}) and other types (c_{51})). The concepts found in inception4c are composed from varying combinations of colors and parts found in conv3 (e.g., various bird parts (c_{31}, c_{33}) contribute to the bird concepts at inception4c). In the end, both scene

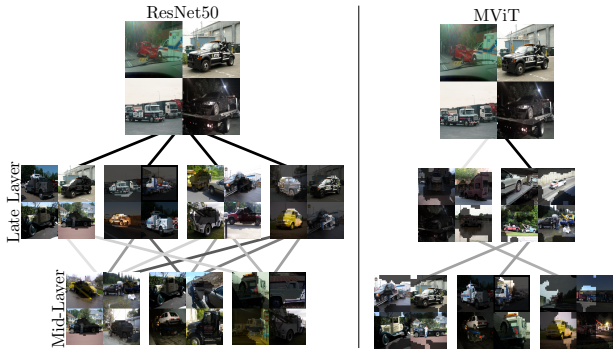


Figure 5.7: VCCs for a CNN (ResNet50) and transformer (MViT) for the class 'Tow Truck'. Shown are only the last two layers with the class output.

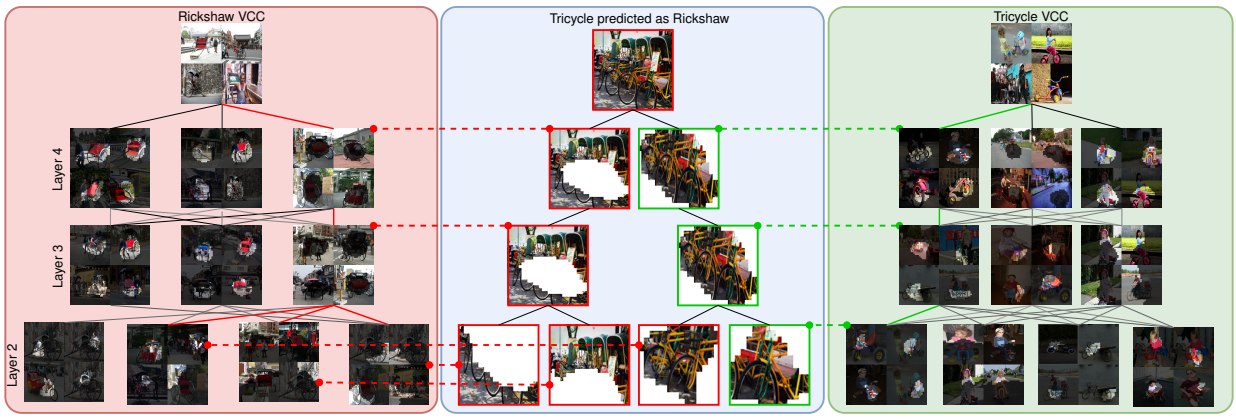


Figure 5.8: Debugging model failure modes with VCCs. We show an image of a tricycle incorrectly predicted by a ResNet50 as a rickshaw (middle) as well as the top-down segmentation of the image (Section 5.3.1). We also show the incorrect (left) and correct (right) VCCs. Following the hierarchy of concepts reveals that the model incorrectly focused heavily on the collapsible hood, starting at Layer 2.

and object contribute with strong weights to the final category.

An all layer VGG-16 VCC visualization was presented in Figure 5.1. As discussed in the caption, hierarchical concept assemblies again are revealed, with both target object as well as its background contributing to the final classification. While both the few layer and all layer visualizations reveal the concept representations of the models under analysis, they afford different levels of granularity: Few layer visualization provides a concise summary with a focus on specific layers, while all layer provides very detailed study.

Subset VCC analysis. We begin our quantitative analysis on four-layer VCCs generated for the CNNs and transformers listed in Section 5.4.1. Layers are selected approximately uniformly across a model to capture concepts at different abstractions. We use standard metrics for analyzing tree-like data structures in a per-layer fashion: branching factor (extent to which the representation is distributed), number of nodes (how many concepts the model uses to encode a particular class), edge weights (strength of connection between concepts) and edge weight variance (variability of connection strength). We calculate averages over 50 VCCs for 50 randomly selected ImageNet classes. Results

are shown in Figure 5.5.

Patterns are apparent in concept numbers and edge weights: more, but weaker, concepts in early layers vs. fewer, but stronger, concepts in later layers. These patterns reflect the shared low-level features (e.g., colors, textures) across classes and more specific features near the end, which yield larger ITCAV values. Also, CNNs show a decreasing branching factor, while transformers maintain a consistent number until the final layer, where all models converge to about two concepts, typically of an ImageNet class's foreground-background structure. Transformers have higher final layer edge weight variance compared to CNNs, indicating their ability to better differentiate earlier concepts' importance in forming later concepts, potentially explaining their superior classification performance (i.e., all information is not equally valuable). We also compare models (ResNet50 and VGG16) when trained with self-supervision [45] (SimCLR) or for adversarial robustness, i.e., on Stylized ImageNet [93] (ADV). We observe that robust and self-supervised models have fewer low-level concepts and compositionality than the originals, likely as their training yields less reliance on texture (stylization perturbs texture) and color (SimCLR training jitters color).

To examine these patterns further, VCC visualizations for a CNN and transformer are shown in Figure 5.7. Here, we limit to two (diverse) models and two later layers with class output for space. The connection diversity in the last layer is indeed observed to be larger for the transformer vs. the CNN. Notably, over half of the concepts in the CNN capture background centric concepts, while the transformer has only a single background centric concept.

All layer VCCs. We present quantitative analyses on all layer VCCs for three diverse models: (i) a standard CNN, VGG-16, (ii) an efficient model, MobileNetv3 and (iii) a transformer, MViT. Averages are taken over 10 VCCs for 10 random ImageNet classes. Results are shown in Figure 5.6. Common trends appear in all models. Concept composition is non-linear across layers, with branching factor ranging from 5-15 and converging to approximately two near the last layers. The peak number of concepts, around 20, is consistently at 30-40% of network depth and, as in the four-layer analysis, also converges to two in the final layer.

Edge weights and variances are in accord with the main findings of the four-layer analysis, but also reveal other insights into the compositional nature of the models. At fine grained, all layer analysis, each model more readily displays unique edge weight characteristics: VGG16's average weights decrease in later layers, MobileNetv3's drop greatly before the final layer and MViT maintains consistent values. Still, overall these results indicate that penultimate concepts differ between CNNs vs. transformers, as supported by our four-layer VCC analysis; see Figure 5.5. Higher variances in initial layers suggest a diverse combination of concepts, whereas deeper layers indicate a more uniform composition. Transformers, however, show a variance increase in the final layer, indicating greater compositionality.

In summary, the four and all-layer VCC analyses consistently highlight key aspects of deep networks: (i) Early and mid-layers use more concepts compared to deeper layers. (ii) Concept interactions,

e.g., ITCAV values and variances, differ greatly between adjacent layers, as opposed to across multiple layers. (iii) CNNs and transformers compose concepts differently, especially at the final layers.

5.5 Application: Diagnosing failure predictions

To show the VCC's practical utility, we consider application to model failure analysis. Contrasting failure analysis methods that do not explain how errors are made compositionally [218] or only offer explanations of single neuron heatmaps [49], VCCs provide insights on compositional concepts across layers and distributed representations.

Figure 5.8 shows a 'Tricycle' incorrectly classified as a 'Rickshaw' by a ResNet50 model, and the corresponding incorrect VCC ('Rickshaw', left) and correct VCC ('Tricycle', right). As the image is decomposed using our top-down segmentation (Section 5.3.1), it is revealed that the majority of pooled segments are closer, in terms of l_2 distance, to concepts in the Rickshaw VCC (red outlines) than the tricycle VCC (green outlines). While the model correctly encoded the wheel and handlebar portions of the images as tricycle concepts, the background and collapsible hood concepts are composed from layers two through four as rickshaw concepts, which may cause the error. We also note the lack of other tricycle-specific concepts (e.g. children).

On the other hand, we do find cases of classification errors in which VCCs do *not* help with debugging the failure. This can be due to several issues, such as the concepts in the correct and incorrect VCC not being fully interpretable, the closest concept to the target segment are not from the incorrect VCC, or the path strengths are noisy and do not follow a meaningful concept trace from the early layers to the target prediction. We believe the performance of this debugging procedure will improve as the core methodological components of VCC discovery improves (i.e., segment decomposition, concept discovery, and multi-layer concept connections) and is an interesting avenue of future work.

5.6 Discussion and conclusion

The Visual Concept Connectome (VCC) is a method for discovering human interpretable concepts and their interlayer connections in a fully unsupervised way. VCCs allow for the interpretation of deep models at multiple resolutions, revealing interesting properties via application to a range of networks. When comparing architecture types, CNNs were found to rely heavily on final layer concepts for prediction, while transformers were found to have more diverse connection strengths in the final layer and a more complex assembly of later concepts from earlier ones. We also applied VCCs to model debugging and showed how the hierarchical concept representation can indicate how mistakes arise. Given VCCs generality, our methodology should be applicable to additional networks, tasks and applications.

Chapter 6

Conclusion and future work

6.1 Summary and discussion

This dissertation focused on quantifying spatial and temporal information in deep video models as well as multi-layer concept-based interpretability. To this end, the dissertation made the following main contributions.

- Investigated and quantified the static and dynamic information contained in deep video models (Chapter 3).
- Designed the first open-world concept discovery algorithm applicable to the analysis of spatiotemporal concepts in video transformers (Chapter 4).
- Extended concept-based interpretability methods to the multi-layer setting; enabling the quantification of interlayer concept contributions (Chapter 5).

As mentioned previously, there is limited but growing understanding of the information captured by deep spatiotemporal models in their intermediate representations. While evidence suggests that video models (e.g., action recognition) are heavily influenced by visual appearance in single frames [34, 50, 160], no quantitative methodology exists for evaluating such static bias in the latent representation compared to bias toward dynamics. We tackled this challenge by proposing an approach for quantifying the static and dynamic biases of any spatiotemporal model [147, 148]. The design of our approach is inspired by recent work quantifying the relative shape and texture information in deep convolutional neural networks (CNNs) [128]. We construct static and dynamic pairs of videos via random temporal frame shuffling and video stylization, respectively. We then measure the relative mutual information between these pairs for a given model layer to quantify the static vs. dynamic bias. We applied our approach to three tasks, action recognition, automatic video object segmentation (AVOS) and video instance segmentation (VIS).

The key findings are: (i) Most examined models are biased toward static information. (ii) Some datasets that are assumed to be biased toward dynamics are actually biased toward static information. (iii) Individual channels in an architecture can be biased toward static, dynamic or a combination of the two. (iv) Most models converge to their culminating biases in the first half of training. We then explored how these biases affect performance on dynamically biased datasets. For action recognition, we proposed StaticDropout, a semantically guided dropout that debiases a model from static information toward dynamics. For AVOS, we designed a better combination of fusion and cross connection layers compared with previous architectures. Indeed, we observed that complex video tasks do require temporal reasoning, and that increasing a model's bias towards the proper encoding of dynamic information can improve performance on such tasks.

Given these findings, i.e., that deep models require both spatial *and* temporal reasoning to solve complex video tasks, we then focused on the problem of spatiotemporal concept-based interpretability of transformer representations for videos [146]. Concretely, we proposed a method to explain the decision-making process of video transformers based on high-level, spatiotemporal concepts that are automatically discovered. Prior research on concept-based interpretability has concentrated solely on image-level tasks. Comparatively, video models deal with the added temporal dimension, increasing complexity and posing challenges in identifying dynamic concepts over time. We then systematically addressed these challenges by introducing the first Video Transformer Concept Discovery (VTCD) algorithm [146]. Similar to previous concept discovery methods for image models [86,98,281], we follow a two step approach: (i) concept discovery and (ii) concept importance. To this end, we first proposed an efficient approach for unsupervised identification of units of video transformer representations (tubelets). We then rank their importance to the output of a model using a novel approach, Concept Randomized Importance Sampling (CRIS), that masks out many concepts in parallel to produce meaningful perturbations in transformer self-attention layers. We followed the standard approach in the literature [86, 98] and validate the importance rankings calculating *concept attribution curves*; removing the concepts in most-to-least important order (or vice versa) and measuring the Area-Under-Curve (AUC) ratio.

The resulting discovered important concepts are highly interpretable, revealing spatio-temporal reasoning mechanisms and object-centric representations in unstructured video models. For example, we found that the most important concepts in a semi-VOS model captured the model tracking the target objects and likely distractors, while the most important concepts for an action recognition model captured spatiotemporal object-centric concepts, like the objects being dropped and the containers after dropping. Performing this analysis jointly over a diverse set of supervised and self-supervised representations, we discovered that some of these mechanism are universal in video transformers. Indeed, all models contain some form of positional processing at early layers while object-centric concepts emerge at later layers. Finally, we demonstrated how VTCD can be used for different

downstream tasks. Specifically, we prune transformer heads to boost fine-grained action recognition while improving efficiency. Next, we find object-tracking concepts and use them for zero-shot semi-supervised video object segmentation.

The previous studies revealed complex reasoning patterns emerging across multiple layers in deep spatiotemporal models, however, there was no direct way of quantifying the contribution of a concept at one layer to a concept at a deeper layer. To this end, we presented a new methodology, the Visual Concept Connectome (VCC) [149], which discovers human interpretable concepts and their interlayer connections in a fully unsupervised manner. Our approach simultaneously reveals fine-grained concepts at a layer, connection weightings across all layers and is amendable to global analysis of network structure (e.g., branching pattern of hierarchical concept assemblies). While our ultimate goal is to apply VCCs to video-based tasks, for simplicity we first developed the method for image classification.

Our method consists of three main components. (i) Image segments are extracted in the model's feature space via divisive clustering to produce semantically meaningful image regions for each selected layer. (ii) Layer-wise concepts, i.e., the nodes of the graph, are discovered in an open world fashion (i.e., no labelled data is required) via a second round of clustering over the dataset of image regions, independently for each layer. (iii) Edges are calculated that indicate the contribution of concepts from earlier to deeper layers via an approach we introduce dubbed ITCAVs. Quantitative and qualitative empirical results showed the effectiveness of VCCs.

VCCs allow for the finegrained interpretation of deep models by observing individual concept compositions, or global network analysis by aggregating statistics of edge strengths over many layers. When comparing architecture types, CNNs were found to rely heavily on final layer concepts for prediction, while transformers were found to have more diverse connection strengths in the final layer and a more complex assembly of later concepts from earlier ones. Finally, we leveraged VCCs for the application of failure mode debugging to reveal where mistakes arise in deep networks.

While VCCs present a proof-of-concept for circuit discovery in vision models, there are several avenues of future work to be explored to improve upon them. First, robust causal validation should be done to better understand if the concept circuits discovered are valid, e.g., following those from the NLP literature such as the Mechanistic Interpretability Benchmark [178]. Second, it is important to quantify the *interpretability* of discovered concepts. This would require human experiments but is crucial in ascertaining whether the method aligns with human notions of concepts for image recognition.

In summary, we showed that many video understanding neural networks are indeed heavily biased towards appearance information, likely due to biases inherent in the data they are trained on. However, there are tasks, such as finegrained action recognition [103] and complex object tracking problems [247], that require temporal reasoning. To this end, the VTCD was designed to better understand this reasoning through the discovery of unsupervised spatiotemporal concepts. We found

universal processing patterns in video transformers across many layers, including positional bases at early layers, object-centric concepts at mid-layers, and spatiotemporal object-centric concepts at deeper layers. These concepts are robust and enable the tackling of different downstream tasks. Finally, we performed a cross-layer quantitative analysis using VCCs. By first discovering concepts at each layer and then quantifying their contribution to adjacent layer concepts, we were able to perform an analysis of concept compositionality for many architectures and training objectives. We observed that transformers are more compositional than CNNs. We also showed that data augmentation used for increasing robustness or reducing reliance on labels can reduce the number of low-level concept circuits by removing color and texture information.

6.2 Societal impact

Being able to precisely interpret neural network representations has several meaningful positive impacts for society. However, we note that the application of these methods in real world deployment settings could have several drawbacks. For example, they may not always be trivial to perform, could demand substantial computing power, or require halting the deployed algorithm for a period during which the interpretation would take place. Alternatively, there are multiple applications of AI that do not require transparency at all [65]: (i) situations where there are no significant consequences to incorrect outputs or (ii) the problem and outputs are studied so rigorously that the system, even if imperfect, is trusted. Therefore, it is important to consider whether a given scenario calls for the use of interpretability methods. We argue that the following scenarios benefit from the use of interpretability: (i) applications with ethical considerations or significant consequences due to the outputs (ii) to enhance model understanding and design (iii) scientific discovery, and (iv) legal and regulatory issues.

6.2.1 Applications and ethical considerations

DNNs are currently being deployed for many applications in several different domains. As one might expect, there have been numerous instances of *harms due to incorrect outputs of these systems* and AI continues to amplifies existing societal biases. In 2015, Google's object recognition repeatedly misclassified dark skinned individuals as *gorilla*, and "fixed" the issue by simply removing the gorilla class from the possible outputs. Part of the problem here lies in the datasets that DNNs are trained on. For example, the Labeled Faces in the Wild (LFW) dataset [120] was found to have more than double the pictures of George Bush than all dark skinned women combined [107] and training on such data will produce models that perform better on subgroups with more representation. But much of this bias lies beyond the data too; even the hardware we use to capture photos are optimized for lighter skin tones. In the 1950's, camera colour-balance benchmarks used a 'Shirley card', a photo of

a white woman named Shirley Page, to calibrate lighting and colour for their cameras [156]. While multiracial Shirley cards started being used in the 1990's, modern day vision systems still have issues classifying underrepresented subgroups. Commercial gender classification perform significantly worse (34.7% vs. 0.8% error) on darker skinned females vs. white males [35].

Other applications of AI can result in immediate physical danger to a person. A prominent example is in the medical setting, where model outputs directly and immediately impacts the quality of care that a patient receives. DNNs are infamous for making decisions based on 'shortcuts' [92] (i.e. spurious correlations) in the data and the medical domain is no different. For example, it was recently found that DNNs trained to detect COVID-19 from chest X-Rays rely on confounding factors rather than medical pathology [58], e.g. since intensive care unit (ICU) beds have larger metal handles used for mobility the type of hospital bed is visible in the X-ray. Similarly, a model that successfully detected pneumonia from X-ray scans failed to detect pneumonia in new hospitals, as it was solely focused on the hospital-specific metal token placed in each scan. Applications outside of medicine also have immediate dangers, such as self-driving vehicles, have the promise of making roads safer but still get into accidents that humans would never make, such as hitting and dragging a person for hundreds of feet before stopping [55].

6.2.2 Model understanding and improvement

There is little formal understanding of deep networks despite a tremendous amount of efforts to do so. Examples of successes are limited; scaling laws [137], Kernel Regimes [262], influence functions [12], and other topics have been able to predict some characteristics of deep models using formal theory, but overall the theoretical understanding of how deep models work is moving much slower than the efforts on the empirical side. Interpretability may be able to provide benefits to both the empirical and the theoretical aspects of deep learning simultaneously. Several works have uncovered several algorithms learned and deployed by models to do computation; such as modulo addition being accomplished by rotational Fourier analysis [179], superposition of concepts [69], transformer induction heads [186] (small circuits used by the model for identifying repeated sequences), and neuron-neuron circuits composing visual phenomenon [183]. In a broader sense, as biologists use empirical methods to propose theories of biology, a true and deep ability to interpret what a model is doing may be the way to turn deep learning from an empirical science into formal theories of deep models.

Numerous studies have examined various biases, unwanted concepts, or model dynamics, and as a result, were able to design *protocols to avoid such behaviours*. Several works have shown that standard pretrained Convolutional Neural Networks (CNNs) are substantially more biased towards encoding texture information than shape information [93, 128] and suggested different training datasets [93] or regularization schemes [129, 222] to bias the model toward encoding more shape information. Similar research has been done in the video domain for static vs. dynamic information [50, 147, 160] and de-

biasing video models away from appearance-centric information [50, 148]. Investigating self-attention maps of vision transformers resulted in improving architecture performance and interpretability with register tokens [57] (i.e. non-contextualized auxiliary tokens used by the model for internal computations). Other work quantified the importance of neurons in transformers [210], and used this importance score to initialize weights of a smaller transformer to improve training times for smaller models compared to random initialization. These examples clearly demonstrate how understanding and interpreting models can provide immediate feedback enabling significant improvements in the engineering of models.

6.2.3 Scientific understanding

Interpretability can also enable humans to improve *their own knowledge* based on understanding the algorithms learned by deep networks. The core idea is that a deep network, when trained on a certain task, may learn important concepts that are understandable, but missed, by humans. By leveraging the correct interpretability methods, humans may extract this information and enhance their own abilities in that domain. DNNs have surpassed human performance on numerous tasks ranging from board games [223] (e.g. chess and Go), to disease identification [7], to predicting the structure of proteins from RNA strings [135]. Recent efforts have successfully mined novel and important information from these models. Advanced chess concepts from AlphaZero [223] (i.e. a system trained solely via self-play) were discovered [216] and then taught to chess grand-masters via concept prototype chess puzzles. Performance of all grand-masters improved on new puzzles after learning AlphaZero's concepts. More recently, a novel structural class of antibiotics was discovered by using interpretability techniques [261]. First, a graph neural network was trained to predict whether or not a new compound will inhibit bacterial growth based on its chemical structure. Then, the authors interpreted the representations using clustering algorithms to find clusters represented novel antibiotic classes. Interpretability for scientific discovery is a new and exciting field, and it is reasonable to expect similar applications in fields such as math and physics in the near future.

6.2.4 Legal and regulatory issues

Many regulatory bodies around the world have ruled that an essential component of automated decision making systems (e.g. DNNs) is to be able to explain why one made a certain decision. Most notably, in 2021, the European Union (EU) passed the General Data Protection Regulation (GDPR) act, which mandates that corporations and governmental organizations must "facilitate the interpretation of the outputs of AI systems by the users" [167]. Any company that offers products within the EU must comply with the GDPR and thus must make interpretability a feature of AI-based products. While the EU was the first large body to sign interpretability requirements into law,

other international bodies are making similar regulations. In 2023, the Canadian parliament proposed the Artificial Intelligence and Data Act, which requires that organizations using AI systems must "include a level of interpretability appropriate to the context" and "information provided should be sufficient to allow the public to understand the capabilities, limitations, and potential impacts of the systems" [125].

Meanwhile, in the United States of America (USA), the Biden-Harris Administration signed an Executive Order requiring new standards for AI safety and security [117]. While interpretability is not explicitly mentioned, requirements involving interpretability include "develop standards, tools, and tests to help ensure that AI systems are safe, secure, and trustworthy", "address algorithmic discrimination", and "accelerate development and implementation of vital AI standards with international partners and in standards organizations, ensuring that the technology is safe, secure, trustworthy, and interoperable".

Regulatory concerns are not solely theoretical; many legal battles are being fought in the US and abroad surrounding different uses of AI. On an individual level, people have sued AI companies when their models have generated false information that could be viewed as defamatory [204]. Another key debate topic in the courts right now is over data privacy and ownership. Large models (large language models (LLMs), foundation models, frontier models) are trained on vast amounts of web-scraped data, including many web pages that have copyright restrictions. It is no surprise that, after GPT-2 was released [200], all papers releasing models with large training runs did not describe in detail the source of their training data, i.e. to avoid inevitable lawsuits. Regardless, there are numerous ongoing lawsuits where the plaintiff has proprietary data and the defendant are companies who are accused of training AI models on this data and leveraging it for profit. This includes a trade group for U.S. authors (e.g. George Saunders, Jodi Picoult and George R.R. Martin, etc.) vs. OpenAI [8], computer programmers vs. Microsoft Copilot [173], visual artists vs. Stability AI [32], and the New York Times vs. OpenAI [105]. Many key questions in these lawsuits are also fundamental problems in interpretability research: (i) To what extent does a single training example effect a model output at deployment time? (ii) Are model's actually generating new information or just interpolating between training examples?

The legal landscape of the EU and North America requires the transparency of AI tools and ethical dilemmas within legal battles would be clarified with more interpretable models. Therefore, it is clear that the interpretability of AI-based systems will continue to grow as an important part of AI regulatory frameworks.

6.2.5 AI existential risk

The previous subsections presented numerous types of immediate risks and harms occurring from the current deployment of AI systems. However, there is a growing faction of researchers who are concerned about the threats of 'artificial super intelligence' (ASI): AI systems that greatly surpass human performance on a large number of tasks. But these are not fringe folk who have a limited understanding of DNNs either; e.g. Turing award winners Geoff Hinton and Yoshua Bengio both consider this a serious issue that requires attention [21]. Alternatively, other prominent figures with a deep understanding of the field, like Yann LeCun, and Jurgen Schmidhuber, think that the chances of an ASI having a real threat to humanity in the near future is essentially zero [134, 207]. Given the diametrically opposing views of leaders in the field of AI research, what should someone think of this possibility?

Proponents of the existential risk phenomenon point out that nobody knows for certain whether the current boom in AI capabilities will reach the level of ASI. Given that developing ASI is a non-zero probability, and that we are not certain that we could control ASI with current capabilities if developed, we should therefore invest a large amount of resources in understanding how to deal with ASI if it arises. This has led to large labs (e.g. Anthropic and OpenAI) introducing AI Alignment teams focusing on developing methods to that can better control current and future AI systems in accordance with human values.

On the other hand, many people disagree with this line of reasoning. Yann LeCunn and other proponents of developing model capabilities faster do not believe that AI will have a realistic existential threat to humans in the near future [134, 207]. One argument says that there will be good ASIs that, with the assistance of humans, will form a more powerful team compared with a handful of bad-acting AIs. Alternatively, many researchers argue that we are nowhere close to general intelligence (i.e. systems that perform on-par to humans at most tasks), let alone super-intelligence, and therefore investing billions of dollars into ASI-defences is non-sensible.

This dissertation takes no stance on existential risk due to AI. Providing evidence to guide these likelihoods, or rigorously defining agreed upon notions of different levels of intelligence [51], would be useful but is out of the scope of this dissertation. Importantly, however, interpretability will be beneficial for *all AI risks*; current and future. As discussed in Sections 6.2.1, 6.2.2, 6.2.3, and 6.2.4, interpreting the model significantly improves our current abilities to address application and regulatory issues as well as improve our understanding of science and the models themselves. In the case of future risks, key proponents of the existential risk hypothesis consistently highlight interpretability and transparency as crucial properties of aligning ASI systems. For all scenarios, interpretability will play an important role in the safe deployment of AI systems.

6.3 Future work

6.3.1 Evaluation of concept-based interpretability techniques

The development of sparse autoencoders as a scalable approach for concept discovery has opened new avenues for interpretability in both vision and language models [83, 84, 203]. However, their relationship to traditional clustering-based methods remains relatively unexplored [83]. Sparse autoencoders inherently focus on disentangling and compressing representations into independent latent factors, whereas clustering methods aim to group similar instances based on shared features. The underlying assumptions, computational trade-offs, and suitability of these approaches for different model architectures, tasks, and datasets warrant deeper investigation. For example, do sparse autoencoders currently work better for large models simply because researchers and engineers are better at scaling DNNs compared to clustering algorithms [90, 238]? Studying these methods systematically across diverse hyperparameter settings, tasks, and architectures could illuminate their relative strengths, weaknesses, and potential hybrid strategies.

Furthermore, evaluating these techniques under a unified framework, including quantitative metrics like concept coherency, interpretability, and task-specific utility, would provide a comprehensive understanding of their trade-offs and guide practitioners in selecting the most effective approach for their specific use case. Currently, it is unclear whether new interpretability methods are achieving real improvements over previous approaches and new evaluation standards are needed. Some non-vision benchmarks, like the recent Mechanistic Interpretability Benchmark [178], provide such a standardized benchmark to compare circuit discovery methods for four standard tasks (e.g., Indirect Object Identification, Arithmetic). A similar benchmark for vision models, images and videos, would be extremely beneficial for the field to advance.

6.3.2 Interpretability beyond compositional concepts

While compositionality has provided a new framework for interpreting model representations [4, 49, 86, 149, 285], the next frontier lies in exploring what lies beyond this paradigm. These methods often rely on breaking down complex concepts into simpler, hierarchical components. However, this approach may miss higher-order "meta-concepts" that are not easily decomposable. Meta-concepts could represent relationships between multiple distinct concepts or encode abstract reasoning mechanisms, such as causal dependencies or cross-modal interactions.

Additionally, understanding the universality of concepts across models trained on diverse objectives remains an open challenge. Previous work, including the work presented in Chapter 4, have performed a limited quantitative analysis of concept universality across different models [68, 146]. Investigating the existence and utility of these meta and universal concepts could not only deepen our

theoretical understanding of deep learning models but also enable more generalizable and transferable interpretability frameworks. Interesting future research directions could be measuring concept universality across models trained for different modalities [123]. Additionally, current methods use post-hoc metrics to approximate the differences between model feature spaces; however, learning a joint interpretable space directly avoids these issues and could enable model-model translation through this disentangled space.

6.3.3 Expanding video interpretability methods

As mentioned previously, interpretability in video models still lags significantly behind its image counterparts. While this dissertation tackles some open challenges in video interpretability (quantifying static vs. dynamic information, Chapter 3, and spatiotemporal concept discovery, Chapter 4), many areas of video understanding have little or no corresponding interpretability research, such as long-form [263] and generative video tasks [198]. A key limitation, and a key reason why the VTCD approach presented in Chapter 4 was unsupervised, is the lack of robust labelled datasets to validate proposed methods for interpreting video models. Current labelled video datasets have limited segmentation labels appropriate for interpretability validation and often focus on atomic actions or simple object interactions, which fall short of representing the complexity of many spatiotemporal concepts encoded by video transformers, as shown in Chapter 4. Addressing this limitation requires curating datasets enriched with temporally aligned, hierarchically labelled concepts that span static, dynamic, and compositional elements. Additionally, current interpretability techniques for video models often focus on short clips, leaving long-duration temporal reasoning largely unexplored [34,146,148]. Future work should extend these methods to handle long-term dependencies, such as tracking objects and events across extended sequences. Techniques that can reveal the dynamic evolution of concepts, such as temporal concept graphs or causal concept chains, would be particularly valuable for understanding long-term video analysis models.

6.3.4 Theoretical foundations of interpretability

Despite the aforementioned empirical advancements to understanding models, the theoretical underpinnings of interpretability remain poorly understood. One work argues that causality-based analysis [25] is a key component to obtaining interpretable models. They argue that truly understanding a model's mechanisms requires ablating components to identify cause-effect relationships within these mechanisms. However, many important questions remain unanswered. Can we establish formal bounds on the extent to which a model can be interpreted? For instance, is there an information-theoretic limit to how disentangled a model's representations can become, particularly for high-bandwidth concepts that require encoding numerous interdependent features? Certain tasks, such as learning complex spa-

tiotemporal patterns or abstract reasoning, may inherently require representations that are too dense or high-dimensional for humans to comprehend fully. Future research could focus on quantifying these limits using principles from information theory, such as Shannon entropy [136, 221], to evaluate the trade-offs between model accuracy and interpretability. Understanding these theoretical constraints could also guide the design of architectures and learning objectives that balance performance with interpretability, ensuring practical utility while maintaining transparency.

References

[1] Nayyer Aafaq, Ajmal Mian, Wei Liu, Syed Zulqarnain Gilani, and Mubarak Shah. Video description: A survey of methods, datasets, and evaluation metrics. ACM Computing Surveys, 52(6):1–37, 2019.

[2] Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 4190–4197, 2020.

[3] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(11):2274–2282, 2012.

[4] Reduan Achtibat, Maximilian Dreyer, Ilona Eisenbraun, Sebastian Bosse, Thomas Wiegand, Wojciech Samek, and Sebastian Lapuschkin. From "where" to "what": Towards human-understandable explanations through concept relevance propagation. arXiv preprint arXiv:2206.03208, 2022.

[5] Reduan Achtibat, Maximilian Dreyer, Ilona Eisenbraun, Sebastian Bosse, Thomas Wiegand, Wojciech Samek, and Sebastian Lapuschkin. From attribution maps to human-understandable explanations through concept relevance propagation. Nature Machine Intelligence, 5:1006–1019, 2023.

[6] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In Advances in Neural Information Processing Systems, volume 31, 2018.

[7] Md Manjurul Ahsan, Shahana Akter Luna, and Zahed Siddique. Machine-learning-based disease diagnosis: A comprehensive review. In Healthcare, volume 10, page 541. MDPI, 2022.

[8] Raluca Albu. The authors guild, John Grisham, Jodi Picoult, David Baldacci, George R.R. Martin, and 13 Other Authors File Class-Action Suit Against OpenAI. Artificial Intelligence, 2023.

[9] Bilal Alsallakh, Narine Kokhlikyan, Vivek Miglani, Jun Yuan, and Orion Reblitz-Richardson. Mind the pad–CNNs can develop blind spots. In International Conference on Learning Representations, 2020.

[10] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep VIT features as dense visual descriptors. In European Conference on Computer Vision Workshops, 2022.

[11] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one, 10(7):e0130140, 2015.

[12] Juhan Bae, Nathan Ng, Alston Lo, Marzyeh Ghassemi, and Roger B Grosse. If influence functions are the answer, then what is the question? Advances in Neural Information Processing Systems, 35:17953–17967, 2022.

[13] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. International journal of computer vision, 92:1–31, 2011.

[14] Zhipeng Bao, Pavel Tokmakov, Allan Jabri, Yu-Xiong Wang, Adrien Gaidon, and Martial Hebert. Discovering objects that can move. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022.

[15] Isabela Borlido Barcelos, Felipe De Castro Belém, Leonardo De Melo João, Zenilton KG Do Patrocínio Jr, Alexandre Xavier Falcão, and Silvio Jamil Ferzoli Guimarães. A comprehensive review and new taxonomy on superpixel segmentation. ACM Computing Surveys, 56(8):1–39, 2024.

[16] Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. V-JEPA: Latent video prediction for visual representation learning, 2024.

[17] David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. Rewriting a deep generative model. In European Conference on Computer Vision, pages 351–369, 2020.

[18] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6541–6549, 2017.

[19] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B Tenenbaum, William T Freeman, and Antonio Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. International Conference of Learning Representations, 2019.

[20] Sean Bell, Kavita Bala, and Noah Snavely. Intrinsic images in the wild. ACM Trans. on Graphics (SIGGRAPH), 33(4), 2014.

[21] Yoshua Bengio, Geoffrey Hinton, Andrew Yao, Dawn Song, Pieter Abbeel, Trevor Darrell, Yuval Noah Harari, Ya-Qin Zhang, Lan Xue, Shai Shalev-Shwartz, Gillian Hadfield, Jeff Clune, Tegan Maharaj, Frank Hutter, Atilim Gunes Baydin, Sheila McIlraith, Qiqi Gao, Ashwin Acharya, David Krueger, Anca Dragan, Philip Torr, Stuart Russell, Daniel Kahneman, Jan Brauner, and Soren Mindermann. Managing extreme AI risks amid rapid progress. Science, 384(6698):842–845, 2024.

[22] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In International Conference on Machine Learning, 2021.

[23] Usha Bhalla, Suraj Srinivas, Asma Ghandeharioun, and Himabindu Lakkaraju. Towards unifying interpretability and control: Evaluation via intervention. arXiv preprint arXiv:2411.04430, 2024.

[24] Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. Layer-wise relevance propagation for neural networks with local renormalization layers. In Artificial Neural Networks and Machine Learning, pages 63–71, 2016.

[25] Ruta Binkyte, Ivaxi Sheth, Zhijing Jin, Muhammad Havaei, Bernhardt Schölkopf, and Mario Fritz. Causality is key to understand and balance multiple goals in trustworthy ml and foundation models. arXiv preprint arXiv:2502.21123, 2025.

[26] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, and Adam Letts. Stable video diffusion: Scaling latent video diffusion models to large datasets. arXiv preprint arXiv:2311.15127, 2023.

[27] Moritz Böhle, Mario Fritz, and Bernt Schiele. B-Cos networks: Alignment is all we need for interpretability. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10329–10338, 2022.

[28] Florian Bordes, Randall Balestriero, and Pascal Vincent. High fidelity visualization of what your self-supervised representation knows about. arXiv preprint arXiv:2112.09164, 2021.

[29] Judy Borowski, Roland Simon Zimmermann, Judith Schepers, Robert Geirhos, Thomas SA Wallis, Matthias Bethge, and Wieland Brendel. Exemplary natural images explain cnn activations better than state-of-the-art feature visualization. In International Conference on Learning Representations, 2020.

[30] Wieland Brendel and Matthias Bethge. Approximating CNNs with bag-of-local-features models works surprisingly well on imagenet. International Conference on Learning Representations, 2019.

[31] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. Transformer Circuits Thread, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[32] Blake Brittain. Artists take new shot at stability, midjourney in updated copyright lawsuit. Legal, 2023.

[33] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. OpenAI Blog, 2024.

[34] Shyamal Buch, Cristobal Eyzaguirre, Adrien Gaidon, Jiajun Wu, Li Fei-Fei, and Juan Carlos Niebles. Revisiting the "Video" in Video-Language Understanding. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022.

[35] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In Conference on Fairness, Accountability and Transparency, pages 77–91, 2018.

[36] Nick Cammarata, Shan Carter, Gabriel Goh, Chris Olah, Michael Petrov, Ludwig Schubert, Chelsea Voss, Ben Egan, and Swee Kiat Lim. Thread: Circuits. Distill, 2020. https://distill.pub/2020/circuits.

[37] Nick Cammarata, Gabriel Goh, Shan Carter, Ludwig Schubert, Michael Petrov, and Chris Olah. Curve detectors. Distill, 2020. https://distill.pub/2020/circuits/curve-detectors.

[38] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6299–6308, 2017.

[39] Chun-Hao Chang, Elliot Creager, Anna Goldenberg, and David Duvenaud. Explaining image classifiers by counterfactual generation. In International Conference on Learning Representations, 2018.

[40] Michael Chang, Tomer Ullman, Antonio Torralba, and Joshua Tenenbaum. A compositional object-based approach to learning physical dynamics. In International Conference on Learning Representations, 2016.

[41] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks. In IEEE Winter Conference on Applications of Computer vision, pages 839–847. IEEE, 2018.

[42] Hila Chefer, Shir Gur, and Lior Wolf. Generic attention-model explainability for interpreting bi-modal and encoder-decoder transformers. In International Conference on Computer Vision, 2021.

[43] Hila Chefer, Shir Gur, and Lior Wolf. Transformer interpretability beyond attention visualization. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 782–791, 2021.

[44] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In European Conference on Computer Vision, pages 801–818, 2018.

[45] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In International Conference on Machine Learning, pages 1597–1607. PMLR, 2020.

[46] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1971–1978, 2014.

[47] Zhi Chen, Yijie Bei, and Cynthia Rudin. Concept whitening for interpretable image recognition. Nature Machine Intelligence, 2(12):772–782, 2020.

[48] Ming-Ming Cheng, Peng-Tao Jiang, Ling-Hao Han, Liang Wang, and Philip Torr. Deeply explain cnn via hierarchical decomposition. arXiv preprint arXiv:2201.09205, 2022.

[49] Ming-Ming Cheng, Peng-Tao Jiang, Ling-Hao Han, Liang Wang, and Philip Torr. Deeply explain CNN via hierarchical decomposition. International Journal of Computer Vision, pages 1–15, 2023.

[50] Jinwoo Choi, Chen Gao, C. E. Joseph Messou, and Jia-Bin Huang. Why can't I dance in the mall? Learning to mitigate scene bias in action recognition. In Advances in Neural Information Processing Systems, 2019.

[51] François Chollet. On the measure of intelligence. arXiv preprint arXiv:1911.01547, 2019.

[52] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3606–3613, 2014.

[53] European Commision. Laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts. European Commision, 2021.

[54] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. In International Conference on Learning Representations, 2020.

[55] John Crace. Cruise recalls all self-driving cars after grisly accident and california ban. News, 2023.

[56] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. arXiv preprint arXiv:2309.08600, 2023.

[57] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. arXiv preprint arXiv:2309.16588, 2023.

[58] Alex J DeGrave, Joseph D Janizek, and Su-In Lee. Ai for radiographic covid-19 detection selects shortcuts over signal. Nature Machine Intelligence, 3(7):610–619, 2021.

[59] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 248–255. IEEE, 2009.

[60] Li Deng. The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine, 29(6):141–142, 2012.

[61] Konstantinos G. Derpanis, Mikhail Sizintsev, Kevin J. Cannons, and Richard P. Wildes. Action spotting and recognition based on a spatiotemporal orientation analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(3):527–540, 2012.

[62] Konstantinos G. Derpanis and Richard P. Wildes. Spacetime texture representation and recognition based on a spatiotemporal orientation analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(6):1193–1205, 2011.

[63] Chris HQ Ding, Tao Li, and Michael I Jordan. Convex and semi-nonnegative matrix factorizations. TPAMI, 32(1):45–55, 2008.

[64] Zhiwei Ding, Dat T Tran, Kayla Ponder, Erick Cobos, Zhuokun Ding, Paul G Fahey, Eric Wang, Taliah Muhammad, Jiakun Fu, Santiago A Cadena, et al. Bipartite invariance in mouse primary visual cortex. bioRxiv, 2023.

[65] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608, 2017.

[66] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations, 2020.

[67] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. Advances in Neural Information Processing Systems, 29, 2016.

[68] Amil Dravid, Yossi Gandelsman, Alexei A Efros, and Assaf Shocher. Rosetta neurons: Mining the common units in a model zoo. In International Conference on Computer Vision, 2023.

[69] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. arXiv preprint arXiv:2209.10652, 2022.

[70] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. Transformer Circuits Thread, 2022. https://transformer-circuits.pub/2022/toy_model/index.html.

[71] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. Transformer Circuits Thread, 2021. https://transformer-circuits.pub/2021/framework/index.html.

[72] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. Technical Report, 2009.

[73] Patrick Esser, Robin Rombach, and Bjorn Ommer. A disentangling invertible interpretation network for explaining latent representations. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9223–9232, 2020.

[74] Bernard Ghanem Fabian Caba Heilbron, Victor Escorcia and Juan Carlos Niebles. ActivityNet: A large-scale video benchmark for human activity understanding. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 961–970, 2015.

[75] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. IEEE International Conference on Computer Vision, 2021.

[76] Christoph Feichtenhofer. X3D: Expanding architectures for efficient video recognition. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 203–213, 2020.

[77] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. SlowFast networks for video recognition. In IEEE International Conference on Computer Vision, pages 6202–6211, 2019.

[78] Christoph Feichtenhofer, Yanghao Li, and Kaiming He. Masked autoencoders as spatiotemporal learners. Advances in Neural Information Processing Systems, 2022.

[79] Christoph Feichtenhofer, Axel Pinz, and Richard P Wildes. Spatiotemporal multiplier networks for video action recognition. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4768–4777, 2017.

[80] Christoph Feichtenhofer, Axel Pinz, Richard P. Wildes, and Andrew Zisserman. What have we learned from deep representations for action recognition? In Proceedings of the Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition), June 2018.

[81] Christoph Feichtenhofer, Axel Pinz, Richard P Wildes, and Andrew Zisserman. Deep insights into convolutional networks for video recognition. International Journal of Computer Vision, 128:420–437, 2020.

[82] Thomas Fel, Thibaut Boissin, Victor Boutin, Agustin Picard, Paul Novello, Julien Colin, Drew Linsley, Tom Rousseau, Rémi Cadène, Lore Goetschalckx, et al. Unlocking feature visualization for deep network with magnitude constrained optimization. Advances in Neural Information Processing Systems, 36:37813–37826, 2023.

[83] Thomas Fel, Victor Boutin, Mazda Moayeri, Rémi Cadène, Louis Bethune, Mathieu Chalvidal, and Thomas Serre. A holistic approach to unifying automatic concept extraction and concept importance estimation. Conference on Neural Information Processing Systems, 2023.

[84] Thomas Fel, Ekdeep Singh Lubana, Jacob S Prince, Matthew Kowal, Victor Boutin, Isabel Papadimitriou, Binxu Wang, Martin Wattenberg, Demba Ba, and Talia Konkle. Archetypal SAE: Adaptive and stable dictionary learning for concept extraction in large vision models. arXiv preprint arXiv:2502.12892, 2025.

[85] Thomas Fel, Ekdeep Singh Lubana, Jacob S Prince, Matthew Kowal, Victor Boutin, Isabel Papadimitriou, Binxu Wang, Martin Wattenberg, Demba E Ba, and Talia Konkle. Archetypal sae: Adaptive and stable dictionary learning for concept extraction in large vision models. In International Conference on Machine Learning, 2025.

[86] Thomas Fel, Agustin Picard, Louis Bethune, Thibaut Boissin, David Vigouroux, Julien Colin, Rémi Cadène, and Thomas Serre. CRAFT: Concept recursive activation factorization for explainability. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2711–2721, 2023.

[87] David V Foster and Peter Grassberger. Lower bounds on mutual information. Physical Review E, 83(1):010101, 2011.

[88] Raghudeep Gadde, Varun Jampani, and Peter V Gehler. Semantic video CNNs through representation warping. In International Conference on Computer Vision, 2017.

[89] Rohit Gandikota, Zongze Wu, Richard Zhang, David Bau, Eli Shechtman, and Nicholas I Kolkin. Sliderspace: Decomposing the visual capabilities of diffusion models. CoRR, 2025.

[90] Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. arXiv preprint arXiv:2406.04093, 2024.

[91] Ruohan Gao, Dinesh Jayaraman, and Kristen Grauman. Object-centric representation learning from unlabeled videos. In ACCV, 2017.

[92] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. Nature Machine Intelligence, 2(11):665–673, 2020.

[93] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. International Conference on Learning Representations, 2018.

[94] Asma Ghandeharioun, Been Kim, Chun-Liang Li, Brendan Jou, Brian Eoff, and Rosalind W Picard. Dissect: Disentangled simultaneous explanations via concept traversals. arXiv preprint arXiv:2105.15164, 2021.

[95] Amin Ghiasi, Hamid Kazemi, Eitan Borgnia, Steven Reich, Manli Shu, Micah Goldblum, Andrew Gordon Wilson, and Tom Goldstein. What do vision transformers learn? a visual exploration. arXiv preprint arXiv:2212.06727, 2022.

[96] Amir Ghodrati, Efstratios Gavves, and Cees G. M. Snoek. Video time: Properties, encoders and evaluation. In British Machine Vision Conference, 2018.

[97] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. In AAAI Conference on Artificial Intelligence, volume 33, pages 3681–3688, 2019.

[98] Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. Conference on Neural Information Processing Systems, 32, 2019.

[99] Amirata Ghorbani and James Y Zou. Neuron shapley: Discovering the responsible neurons. Advances in Neural Information Processing Systems, 33:5922–5932, 2020.

[100] Gabriel Goh, Nick Cammarata †, Chelsea Voss †, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. Distill, 2021. https://distill.pub/2021/multimodal-neurons.

[101] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. Advances in Neural Information Processing Systems, 27, 2014.

[102] Liv Gorton. The missing curve detectors of inceptionv1: Applying sparse autoencoders to inceptionv1 early vision. arXiv preprint arXiv:2406.03662, 2024.

[103] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. In IEEE International Conference on Computer Vision, pages 5842–5850, 2017.

[104] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, and Charles Herrmann. Kubric: A scalable dataset generator. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022.

[105] Michael M. Grynbaum and Ryan Mac. The times sues openai and microsoft over a.i. use of copyrighted work. Business, 2023.

[106] Isma Hadji and Richard P Wildes. A new large scale dynamic texture dataset with application to convnet understanding. In European Conference on Computer Vision, pages 320–335, 2018.

[107] Hu Han, Charles Otto, Xiaoming Liu, and Anil K Jain. Demographic estimation from face images: Human vs. machine performance. IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(6):1148–1161, 2014.

[108] Sven Ove Hansson, Matts-Åke Belin, and Björn Lundgren. Self-driving vehicles-An ethical overview. Philosophy & Technology, pages 1–26, 2021.

[109] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Learning spatio-temporal features with 3D residual networks for action recognition. In IEEE International Conference on Computer Vision Workshops, pages 3154–3160, 2017.

[110] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016.

[111] Yun He, Soma Shirakabe, Yutaka Satoh, and Hirokatsu Kataoka. Human action recognition without human. In European Conference on Computer Vision, pages 11–17, 2016.

[112] Evan Hernandez, Sarah Schwettmann, David Bau, Teona Bagashvili, Antonio Torralba, and Jacob Andreas. Natural language descriptions of deep visual features. In International Conference on Learning Representations, 2021.

[113] Liam Hiley, Alun Preece, and Yulia Hicks. Explainable deep learning for video recognition tasks: A framework & recommendations. arXiv preprint arXiv:1909.05667, 2019.

[114] Geoffrey E Hinton. Deep belief networks. Scholarpedia, 4(5):5947, 2009.

[115] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. Advances in neural information processing systems, 33:6840–6851, 2020.

[116] The White House. President biden issues executive order on safe, secure, and trustworthy artificial intelligence. The White House, 2023.

[117] United States White House. Executive order on the safe, secure, and trustworthy development and use of artificial intelligence. Presidential Actions, 2023.

[118] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, and Vijay Vasudevan. Searching for MobilNetv3. In International Conference on Computer Vision, pages 1314–1324, 2019.

[119] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4700–4708, 2017.

[120] Gary B. Huang, Marwan Mattar, Honglak Lee, and Erik Learned-Miller. Learning to align from scratch. In Advances in Neural Information Processing Systems, 2012.

[121] Haiyang Huang, Zhi Chen, and Cynthia Rudin. SegDiscover: Visual concept discovery via unsupervised semantic segmentation. arXiv preprint arXiv:2204.10926, 2022.

[122] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In IEEE International Conference on Computer Vision, pages 1501–1510, 2017.

[123] Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. The platonic representation hypothesis. arXiv preprint arXiv:2405.07987, 2024.

[124] Filip Ilic, Thomas Pock, and Richard P Wildes. Is appearance free action recognition possible? In European Conference on Computer Vision, pages 156–173, 2022.

[125] Science Innovation and Economic Development Canada. Artificial intelligence and data act. Innovation, Science and Economic Development Canada, 2023.

[126] Benjamin Irving. MaskSLIC: Regional superpixel generation with application to local pathology characterisation in medical images. arXiv preprint arXiv:1606.09518, 2016.

[127] Md Amirul Islam, Sen Jia, and Neil DB Bruce. How much position information do convolutional neural networks encode? In International Conference on Learning Representations, 2019.

[128] Md Amirul Islam, Matthew Kowal, Patrick Esser, Sen Jia, Björn Ommer, Konstantinos G. Derpanis, and Neil D. B. Bruce. Shape or texture: Understanding discriminative features in CNNs. In International Conference on Learning Representations, 2021.

[129] Md Amirul Islam, Matthew Kowal, Patrick Esser, Bjorn Ommer, Konstantinos G Derpanis, and Neil Bruce. Maximize mutual shape information. British Machine Vision Conference, 2022.

[130] Md Amirul Islam, Matthew Kowal, Sen Jia, Konstantinos G Derpanis, and Neil DB Bruce. Global pooling, more than meets the eye: Position information is encoded channel-wise in CNNs. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 793–801, 2021.

[131] Md Amirul Islam, Matthew Kowal, Sen Jia, Konstantinos G Derpanis, and Neil DB Bruce. Position, padding and predictions: A deeper look at position information in CNNs. International Journal of Computer Vision, 2021.

[132] Suyog Dutt Jain, Bo Xiong, and Kristen Grauman. FusionSeg: Learning to combine motion and appearance for fully automatic segmentation of generic objects in videos. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2117–2126. IEEE, 2017.

[133] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3D convolutional neural networks for human action recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(1):221–231, 2012.

[134] Hessie Jones. Juergen schmidhuber, renowned 'father of modern AI,' says his life's work won't lead to dystopia. Forbes, 2023.

[135] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino

Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. Nature, 596(7873):583–589, 2021.

[136] Alexander Jung and Pedro HJ Nardelli. An information-theoretic approach to personalized explainable machine learning. IEEE Signal Processing Letters, 27:825–829, 2020.

[137] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. arXiv preprint arXiv:2001.08361, 2020.

[138] Rezaul Karim, He Zhao, Richard P. Wildes, and Mennatullah Siam. MED-VT: Multiscale encoder-decoder video transformer with application to object segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023.

[139] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4401–4410, 2019.

[140] Osman Semih Kayhan and Jan C van Gemert. On translation invariance in CNNs: Convolutional layers can exploit absolute spatial location. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14274–14285, 2020.

[141] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, and Fernanda Viegas. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In International Conference on Machine Learning, pages 2668–2677, 2018.

[142] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. Explainable AI: Interpreting, Explaining and Visualizing Deep Learning, pages 267–280, 2019.

[143] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In International Conference on Computer Vision, pages 4015–4026, 2023.

[144] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In International Conference on Machine Learning, pages 5338–5348, 2020.

[145] Yu Kong and Yun Fu. Human action recognition and prediction: A survey. International Journal on Computer Vision, 130(5):1366–1401, 2022.

[146] Matthew Kowal, Achal Dave, Rares Ambrus, Adrien Gaidon, Konstantinos G Derpanis, and Pavel Tokmakov. Understanding video transformers via universal concept discovery. arXiv preprint arXiv:2401.10831, 2024.

[147] Matthew Kowal, Mennatullah Siam, Md Amirul Islam, Neil DB Bruce, Richard P Wildes, and Konstantinos G Derpanis. A deeper dive into what deep spatiotemporal networks encode: Quantifying static vs. dynamic information. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 13999–14009, 2022.

[148] Matthew Kowal, Mennatullah Siam, Md Amirul Islam, Neil DB Bruce, Richard P Wildes, and Konstantinos G Derpanis. Quantifying and learning static vs. dynamic information in deep spatiotemporal networks. arXiv preprint arXiv:2211.01783, 2022.

[149] Matthew Kowal, Richard P Wildes, and Konstantinos G Derpanis. Visual concept connectome (VCC): Open world concept discovery and their interlayer connections in deep models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10895–10905, 2024.

[150] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. Physical review E, 69(6):066138, 2004.

[151] Iro Laina, Yuki M Asano, and Andrea Vedaldi. Measuring the interpretability of unsupervised representations via quantized reversed probing. In International Conference on Learning Representations, 2021.

[152] Hala Lamdouar, Charig Yang, Weidi Xie, and Andrew Zisserman. Betrayed by motion: Camouflaged object discovery via motion segmentation. In Asian Conference on Computer Vision, 2020.

[153] Oran Lang, Yossi Gandelsman, Michal Yarom, Yoav Wald, Gal Elidan, Avinatan Hassidim, William T Freeman, Phillip Isola, Amir Globerson, Michal Irani, et al. Explaining in style: Training a gan to explain a classifier in stylespace. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 693–702, 2021.

[154] Daniel Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In T. Leen, T. Dietterich, and V. Tresp, editors, Advances in Neural Information Processing Systems, volume 13. MIT Press, 2000.

[155] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. Nature, 401(6755):788–791, 1999.

[156] Sarah Lewis. The racial bias built into photography. Lens, 2019.

[157] Fuxin Li, Taeyoung Kim, Ahmad Humayun, David Tsai, and James M Rehg. Video segmentation by tracking many figure-ground segments. In International Conference on Computer Vision, 2013.

[158] Yi Li and Nuno Vasconcelos. Repair: Removing representation bias by dataset resampling. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9572–9581, 2019.

[159] Yingwei Li, Yi Li, and Nuno Vasconcelos. Diving48 dataset. `http://www.svcl.ucsd.edu/projects/resound/dataset.html`. Accessed: 2021-11-13.

[160] Yingwei Li, Yi Li, and Nuno Vasconcelos. Resound: Towards action recognition without representation bias. In European Conference on Computer Vision, pages 513–528, 2018.

[161] Stan Lipovetsky and Michael Conklin. Analysis of regression in game theory approach. Applied Stochastic Models in Business and Industry, 17(4):319–330, 2001.

[162] Mengchen Liu, Jiaxin Shi, Kelei Cao, Jun Zhu, and Shixia Liu. Analyzing the training processes of deep generative models. Transactions on Visualization and Computer Graphics, 24(1):77–87, 2017.

[163] S. Lloyd. Least squares quantization in PCM. IEEE Transactions on Information Theory, 28(2):129–137, 1982.

[164] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. Advances in Neural Information Processing Systems, 2020.

[165] Bruce David Lucas. Generalized image matching by the method of differences. Carnegie Mellon University, 1985.

[166] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems, 30, 2017.

[167] Tambiama Madiega. Eu guidelines on ethics in artificial intelligence: Context and implementation. European Parliamentary Research Service, 2019.

[168] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5188–5196, 2015.

[169] Joonatan Manttari, Sofia Broomé, John Folkesson, and Hedvig Kjellstrom. Interpreting video features: A comparison of 3D convolutional networks and convolutional LSTM networks. In Asian Conference on Computer Vision, 2020.

[170] Joanna Materzyńska, Antonio Torralba, and David Bau. Disentangling visual and written concepts in clip. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 16410–16419, 2022.

[171] Joanna Materzynska, Tete Xiao, Roei Herzig, Huijuan Xu, Xiaolong Wang, and Trevor Darrell. Something-else: Compositional action recognition with spatial-temporal interaction networks. In Conference on Computer Vision and Pattern Recognition, 2020.

[172] Jon Mcauliffe and David Blei. Supervised topic models. Advances in Neural Information Processing Systems, 20, 2007.

[173] Cade Metz. Lawsuit takes aim at the way a.i. is built. Technology, 2022.

[174] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? Advances in Neural Information Processing Systems, 2019.

[175] Patrick Mineault, Shahab Bakhtiari, Blake Richards, and Christopher Pack. Your head is there to move you around: Goal-driven models of the primate dorsal pathway. Advances in Neural Information Processing Systems, 34:28757–28771, 2021.

[176] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. Google research blog, 2015.

[177] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 891–898, 2014.

[178] Aaron Mueller, Atticus Geiger, Sarah Wiegreffe, Dana Arad, Iván Arcuschin, Adam Belfki, Yik Siu Chan, Jaden Fried Fiotto-Kaufman, Tal Haklay, Michael Hanna, et al. Mib: A mechanistic interpretability benchmark. In Forty-second International Conference on Machine Learning, 2025.

[179] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. arXiv preprint arXiv:2301.05217, 2023.

[180] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4467–4477, 2017.

[181] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. Advances in Neural Information Processing Systems, 29, 2016.

[182] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 427–436, 2015.

[183] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. Distill, 2020. https://distill.pub/2020/circuits/zoom-in.

[184] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. Distill, 2017. https://distill.pub/2017/feature-visualization.

[185] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. Distill, 2018. https://distill.pub/2018/building-blocks.

[186] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. Transformer Circuits Thread, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

[187] Audun M. Oygard. Visualizing googlenet classes. Auduno blog, 2015.

[188] Jayneel Parekh, Pegah Khayatan, Mustafa Shukor, Alasdair Newson, and Matthieu Cord. A concept-based explainability framework for large multimodal models. 2024.

[189] Namuk Park, Wonjae Kim, Byeongho Heo, Taekyung Kim, and Sangdoo Yun. What do self-supervised vision transformers learn? In International Conference on Learning Representations, 2023.

[190] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems, volume 32, 2019.

[191] Mandela Patrick, Dylan Campbell, Yuki M. Asano, Ishan Misra Florian Metze, Christoph Feichtenhofer, Andrea Vedaldi, and Joao F. Henriques. Keeping your eye on the ball: Trajectory attention in video transformers. In Advances in Neural Information Processing Systems, 2021.

[192] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2016.

[193] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2016.

[194] Michael Petrov, Chelsea Voss, Ludwig Schubert, Nick Cammarata, Gabriel Goh, and Chris Olah. Weight banding. Distill, 2021. https://distill.pub/2020/circuits/weight-banding.

[195] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. arXiv preprint arXiv:1806.07421, 2018.

[196] Jiyang Qi, Yan Gao, Yao Hu, Xinggang Wang, Xiaoyu Liu, Xiang Bai, Serge Belongie, Alan Yuille, Philip Torr, and Song Bai. Occluded video instance segmentation: A benchmark. International Journal of Computer Vision, 2022.

[197] Yao Qin, Chiyuan Zhang, Ting Chen, Balaji Lakshminarayanan, Alex Beutel, and Xuezhi Wang. Understanding and improving robustness of vision transformers through patch-based negative augmentation. Advances in Neural Information Processing Systems, 2022.

[198] Yiran Qin, Zhelun Shi, Jiwen Yu, Xijun Wang, Enshen Zhou, Lijun Li, Zhenfei Yin, Xihui Liu, Lu Sheng, Jing Shao, et al. Worldsimbench: Towards video generation models as world simulators. arXiv preprint arXiv:2410.18072, 2024.

[199] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In International Conference on Machine Learning, pages 8748–8763, 2021.

[200] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. OpenAI blog, 1(8):9, 2019.

[201] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? Advances in Neural Information Processing Systems, 2021.

[202] Kanchana Ranasinghe, Muzammal Naseer, Salman Khan, Fahad Shahbaz Khan, and Michael S Ryoo. Self-supervised video transformer. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022.

[203] Sukrut Rao, Sweta Mahajan, Moritz Böhle, and Bernt Schiele. Discover-then-name: Task-agnostic concept bottlenecks via automated concept discovery. In European Conference on Computer Vision, pages 444–461, 2024.

[204] Siladitya Ray. Openai sued for defamation after chatgpt generates fake complaint accusing man of embezzlement. Business, 2023.

[205] Sucheng Ren, Wenxi Liu, Yongtuo Liu, Haoxin Chen, Guoqiang Han, and Shengfeng He. Reciprocal transformations for unsupervised video object segmentation. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 15455–15464, 2021.

[206] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should i trust you?" explaining the predictions of any classifier. In Conference on Knowledge Discovery and Data Mining, pages 1135–1144, 2016.

[207] Andrea Rizzi. Yann lecun, chief AI scientist at meta: 'human-level artificial intelligence is going to take a long time'. El Pais, 2024.

[208] Robin Rombach, Patrick Esser, and Björn Ommer. Making sense of CNNs: Interpreting deep representations and their invariances with inns. In European Conference on Computer Vision, pages 647–664, 2020.

[209] Peter J Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics, 20:53–65, 1987.

[210] Mohammad Samragh, Mehrdad Farajtabar, Sachin Mehta, Raviteja Vemulapalli, Fartash Faghri, Devang Naik, Oncel Tuzel, and Mohammad Rastegari. Weight subcloning: direct initialization of transformers using larger pretrained ones. arXiv preprint arXiv:2312.09299, 2023.

[211] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetv2: Inverted residuals and linear bottlenecks. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4510–4520, 2018.

[212] Anirban Sarkar, Deepak Vijaykeerthy, Anindya Sarkar, and Vineeth N Balasubramanian. A framework for learning ante-hoc explainable models via concepts. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10286–10295, 2022.

[213] Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In Proceedings of the First International Conference on Simulation of Adaptive Behavior on From Animals to Animats, page 222–227, Cambridge, MA, USA, 1991. MIT Press.

[214] Ludwig Schubert, Chelsea Voss, Nick Cammarata, Gabriel Goh, and Chris Olah. High-low frequency detectors. Distill, 2021. https://distill.pub/2020/circuits/frequency-edges.

[215] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. LAION-400M:

Open dataset of clip-filtered 400 million image-text pairs. arXiv preprint arXiv:2111.02114, 2021.

[216] Lisa Schut, Nenad Tomasev, Tom McGrath, Demis Hassabis, Ulrich Paquet, and Been Kim. Bridging the human-ai knowledge gap: Concept discovery and transfer in alphazero. arXiv preprint arXiv:2310.16410, 2023.

[217] Maximilian Seitzer, Max Horn, Andrii Zadaianchuk, Dominik Zietlow, Tianjun Xiao, Carl-Johann Simon-Gabriel, Tong He, Zheng Zhang, Bernhard Schölkopf, and Thomas Brox. Bridging the gap to real-world object-centric learning. In International Conference on Learning Representations, 2023.

[218] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In IEEE International Conference on Computer Vision, pages 618–626, 2017.

[219] S. Seung. Connectome: How the Brain's Wiring Makes Us Who We Are. Mariner Book. Houghton Mifflin Harcourt, 2012.

[220] Laura Sevilla-Lara, Shengxin Zha, Zhicheng Yan, Vedanuj Goswami, Matt Feiszli, and Lorenzo Torresani. Only time can tell: Discovering temporal data for temporal modeling. In IEEE Winter Conference on Applications of Computer Vision, pages 535–544, 2021.

[221] Claude Elwood Shannon. A mathematical theory of communication. The Bell system technical journal, 27(3):379–423, 1948.

[222] Baifeng Shi, Dinghuai Zhang, Qi Dai, Zhanxing Zhu, Yadong Mu, and Jingdong Wang. Informative dropout for robust representation learning: A shape-bias perspective. In International Conference on Machine Learning, pages 8828–8839. PMLR, 2020.

[223] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. Nature, 529(7587):484–489, 2016.

[224] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034, 2013.

[225] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In Advances in Neural Information Processing Systems, volume 27, 2014.

[226] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. International Conference on Learning Representations, 2014.

[227] Ilya M Sobol. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. Mathematics and computers in simulation, 55(1-3):271–280, 2001.

[228] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402, 2012.

[229] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 2014.

[230] Chen Sun, Abhinav Shrivastava, Carl Vondrick, Kevin Murphy, Rahul Sukthankar, and Cordelia Schmid. Actor-centric relation network. In ECCV, 2018.

[231] Xinyu Sun, Peihao Chen, Liangwei Chen, Changhao Li, Thomas H Li, Mingkui Tan, and Chuang Gan. Masked motion encoding for self-supervised video representation learning. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023.

[232] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In International Conference on Machine Learning, pages 3319–3328. PMLR, 2017.

[233] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1–9, 2015.

[234] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2818–2826, 2016.

[235] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199, 2013.

[236] Alex Tamkin, Mohammad Taufeeque, and Noah D Goodman. Codebook features: Sparse and discrete interpretability for neural networks. arXiv preprint arXiv:2310.17230, 2023.

[237] Graham W Taylor, Rob Fergus, Yann LeCun, and Christoph Bregler. Convolutional learning of spatio-temporal features. In European Conference on Computer Vision, pages 140–153, 2010.

[238] Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling

monosemanticity: Extracting interpretable features from claude 3 sonnet. Transformer Circuits Thread, 2024.

[239] Ondřej Texler, David Futschik, Michal Kučera, Ondřej Jamriška, Šárka Sochorová, Menclei Chai, Sergey Tulyakov, and Daniel Sỳkora. Interactive video stylization using few-shot patch-based training. ACM Transactions on Graphics (TOG), 39(4):73–1, 2020.

[240] Harrish Thasarathan, Julian Forsyth, Thomas Fel, Matthew Kowal, and Konstantinos Derpanis. Universal sparse autoencoders: Interpretable cross-model concept alignment. arXiv preprint arXiv:2502.03714, 2025.

[241] Pavel Tokmakov, Karteek Alahari, and Cordelia Schmid. Learning video object segmentation with visual memory. In IEEE International Conference on Computer Vision, pages 4481–4490, 2017.

[242] Pavel Tokmakov, Jie Li, and Adrien Gaidon. Breaking the "object" in video object segmentation. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023.

[243] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. VideoMAE: Masked autoencoders are data-efficient learners for self-supervised video pre-training. Advances in Neural Information Processing Systems, 2022.

[244] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3D convolutional networks. In IEEE International Conference on Computer Vision, pages 4489–4497, 2015.

[245] Mike Tyka. Class visualization with bilateral filters. mtyka.github.io, 2016.

[246] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. PeerJ, 2:e453, 2014.

[247] Basile Van Hoorick, Pavel Tokmakov, Simon Stent, Jie Li, and Carl Vondrick. Tracking through containers and occluders in the wild. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023.

[248] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems, pages 5998–6008, 2017.

[249] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research, 11(12), 2010.

[250] Paul Voigtlaender, Lishu Luo, Chun Yuan, Yong Jiang, and Bastian Leibe. Reducing the annotation effort for video object segmentation datasets. In IEEE Winter Conference on Computer Vision Applications, 2021.

[251] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5797–5808, 2019.

[252] Tuan-Hung Vu, Catherine Olsson, Ivan Laptev, Aude Oliva, and Josef Sivic. Predicting actions from static scenes. In European Conference on Computer Vision, pages 421–436, 2014.

[253] Matthew Walmer, Saksham Suri, Kamal Gupta, and Abhinav Shrivastava. Teaching matters: Investigating the role of supervision in vision transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023.

[254] Limin Wang, Bingkun Huang, Zhiyu Zhao, Zhan Tong, Yinan He, Yi Wang, Yali Wang, and Yu Qiao. VideoMAE v2: Scaling video masked autoencoders with dual masking. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023.

[255] Wenguan Wang, Tianfei Zhou, Fatih Porikli, David Crandall, and Luc Van Gool. A survey on deep learning technique for video segmentation. arXiv preprint arXiv:2107.01153, 2021.

[256] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7794–7803, 2018.

[257] Xin Wang, Hong Chen, Si'ao Tang, Zihao Wu, and Wenwu Zhu. Disentangled representation learning. IEEE Transactions on Pattern Analysis and Machine Intelligence, 46(12):9677–9696, 2024.

[258] Yi Wang, Kunchang Li, Yizhuo Li, Yinan He, Bingkun Huang, Zhiyu Zhao, Hongjie Zhang, Jilan Xu, Yi Liu, and Zun Wang. InternVideo: General video foundation models via generative and discriminative learning. arXiv preprint arXiv:2212.03191, 2022.

[259] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. In Conference on Computer Vision and Pattern Recognition, 2021.

[260] Richard P Wildes and James R Bergen. Qualitative spatiotemporal analysis using an oriented energy representation. In European Conference on Computer Vision, pages 768–784, 2000.

[261] Felix Wong, Erica J Zheng, Jacqueline A Valeri, Nina M Donghia, Melis N Anahtar, Satotaka Omori, Alicia Li, Andres Cubillos-Ruiz, Aarti Krishnan, Wengong Jin, et al. Discovery of a structural class of antibiotics with explainable deep learning. Nature, pages 1–9, 2023.

[262] Blake Woodworth, Suriya Gunasekar, Jason D Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In Conference on Learning Theory, pages 3635–3673. PMLR, 2020.

[263] Chao-Yuan Wu and Philipp Krahenbuhl. Towards long-form video understanding. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1884–1894, 2021.

[264] Jiajun Wu, Ilker Yildirim, Joseph J Lim, Bill Freeman, and Josh Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. Advances in Neural Information Processing Systems, 2015.

[265] Zongze Wu, Dani Lischinski, and Eli Shechtman. Stylespace analysis: Disentangled controls for stylegan image generation. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 12863–12872, 2021.

[266] Weiyan Xie, Xiao-Hui Li, Caleb Chen Cao, and Nevin L Zhang. Vit-cx: Causal explanation of vision transformers. International Joint Conference on Artificial Intelligence, 2022.

[267] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In International Conference on Machine Learning, pages 2048–2057. PMLR, 2015.

[268] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas Huang. YouTube-VOS: A large-scale video object segmentation benchmark. arXiv preprint arXiv:1809.03327, 2018.

[269] Charig Yang, Hala Lamdouar, Erika Lu, Andrew Zisserman, and Weidi Xie. Self-supervised video object segmentation by motion grouping. In International Conference on Computer Vision, 2021.

[270] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In International Conference on Computer Vision, 2019.

[271] Yu Yang, Seungbae Kim, and Jungseock Joo. Explaining deep convolutional neural networks via latent visual-semantic filter attention. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8333–8343, 2022.

[272] Chih-Kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. On completeness-aware concept-based explanations in deep neural networks. In Advances in Neural Information Processing Systems, volume 33, pages 20554–20565, 2020.

[273] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. International Conference on Machine Learning Deep Learning Workshop, 2015.

[274] Tingyi Yuan, Xuhong Li, Haoyi Xiong, Hui Cao, and Dejing Dou. Explaining information flow inside vision transformers using markov chain. In eXplainable AI Approaches for Debugging and Diagnosis., 2021.

[275] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In IEEE European Conference on Computer Vision, pages 818–833, 2014.

[276] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In International Conference on Computer Vision, pages 2018–2025. IEEE, 2011.

[277] Chuhan Zhang, Ankush Gupta, and Andrew Zisserman. Helping hands: An object-aware egocentric video recognition model. In International Conference on Computer Vision, 2023.

[278] Quanshi Zhang, Ruiming Cao, Feng Shi, Ying Nian Wu, and Song-Chun Zhu. Interpreting CNN knowledge via an explanatory graph. In AAAI Conference on Artificial Intelligence, volume 32, 2018.

[279] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8827–8836, 2018.

[280] Quanshi Zhang, Yu Yang, Haotian Ma, and Ying Nian Wu. Interpreting CNNs via decision trees. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6261–6270, 2019.

[281] Ruihan Zhang, Prashan Madumal, Tim Miller, Krista A Ehinger, and Benjamin IP Rubinstein. Invertible concept-based explanations for cnn models with non-negative concept activation vectors. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 11682–11690, 2021.

[282] He Zhao and Richard P Wildes. Interpretable deep feature propagation for early action recognition. arXiv preprint arXiv:2107.05122, 2021.

[283] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2921–2929, 2016.

[284] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(6):1452–1464, 2017.

[285] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Interpretable basis decomposition for visual explanation. In European Conference on Computer Vision, pages 119–134, 2018.

[286] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 633–641, 2017.

[287] Daquan Zhou, Zhiding Yu, Enze Xie, Chaowei Xiao, Animashree Anandkumar, Jiashi Feng, and Jose M Alvarez. Understanding the robustness in vision transformers. In International Conference on Machine Learning, 2022.

[288] Tianfei Zhou, Shunzhou Wang, Yi Zhou, Yazhou Yao, Jianwu Li, and Ling Shao. Motion-attentive transition for zero-shot video object segmentation. In AAAI Conference on Artificial Intelligence, pages 13066–13073, 2020.

[289] Yi Zhu, Xinyu Li, Chunhui Liu, Mohammadreza Zolfaghari, Yuanjun Xiong, Chongruo Wu, Zhi Zhang, Joseph Tighe, R Manmatha, and Mu Li. A comprehensive study of deep video action recognition. arXiv preprint arXiv:2012.06567, 2020.

[290] Roland S Zimmermann, Thomas Klein, and Wieland Brendel. Scale alone does not improve mechanistic interpretability in vision models. In Thirty-seventh Conference on Neural Information Processing Systems, 2023.