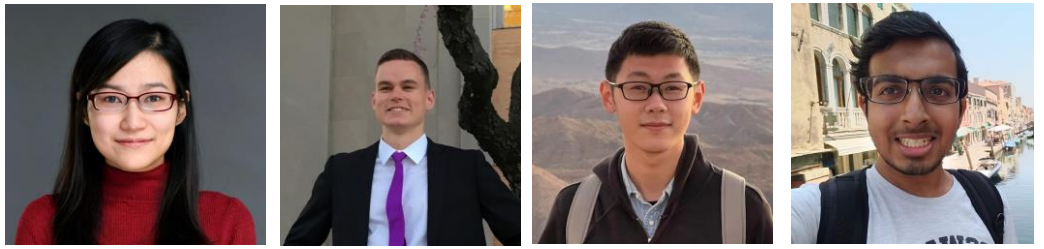
CVPR JUNE 3-7, 2026 DENVER COLORADO

UC San Diego
HALICIOĞLU DATA SCIENCE INSTITUTE

Lily Tuomas Ge Akshay



Principled Interpretability in Vision Models

CVPR 2026 Tutorial

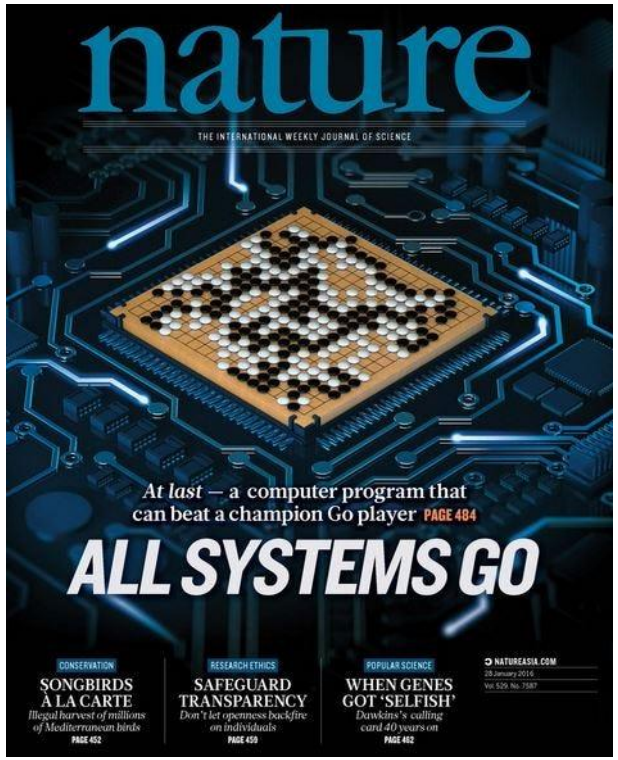
Lily Weng, Tuomas Oikarinen, Ge Yan, Akshay Kulkarni

University of California, San Diego

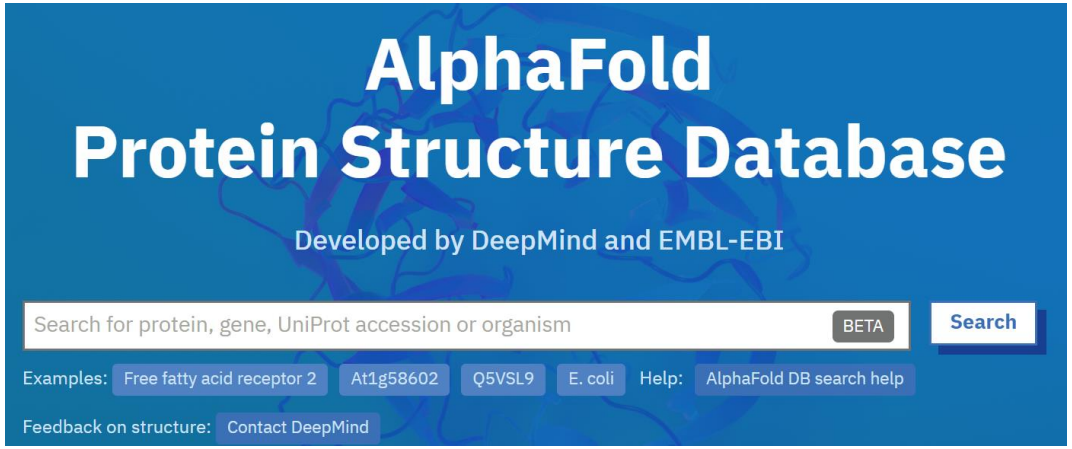
https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

lweng@ucsd.edu

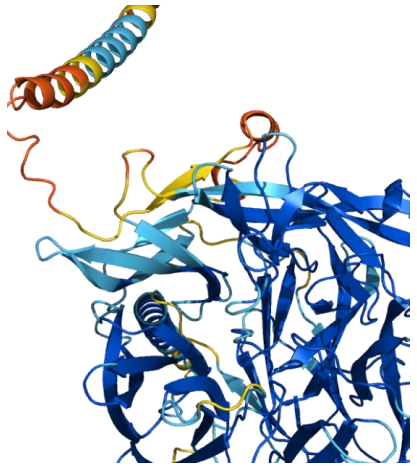
Deep Neural Networks (DNNs) are everywhere



https://becominghuman.ai/summary-of-the-alphago-paper-b55ce24d8a7c



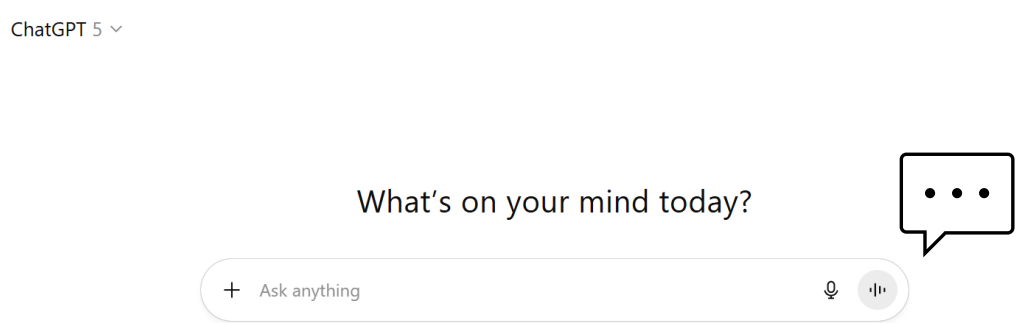
https://alphafold.ebi.ac.uk/



Chat GPT

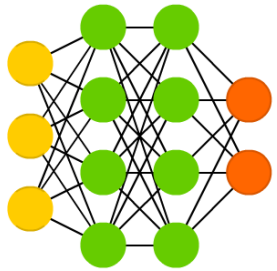
Llama 4

Gemini

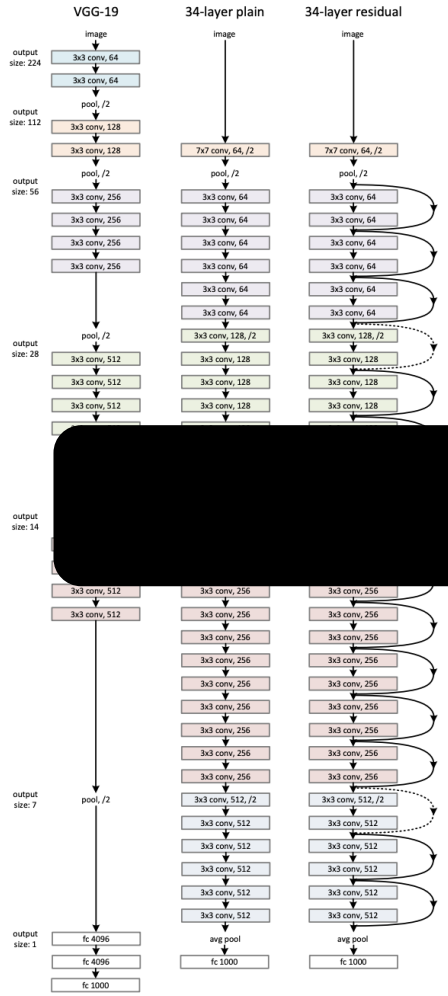


https://www.flaticon.com/free-icon/llama_1840979

It’s hard to understand how DNNs work



DNNs are still considered as **black-box**



Too complex …

https://arxiv.org/abs/1512.03385v1, https://learnopencv.com/the-future-of-image-recognition-is-here-pytorch-vision-transformer/, https://electrek.co/2019/01/04/tesla-leaks-hardware-3-self-driving-computer

Why is it important to understand DNNs?

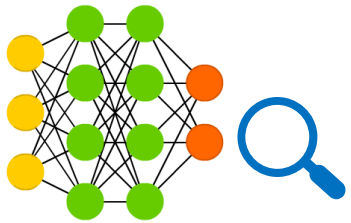
Interpretability is crucial to enable Trustworthy AI

Trust

- ✓ Users will trust a system more if they can understand it
- ✓ Important for high-stakes domains

Transparency

- ✓ Detect biases and debug issues within the model
- ✓ Editing existing models to fix issues



Interpretability

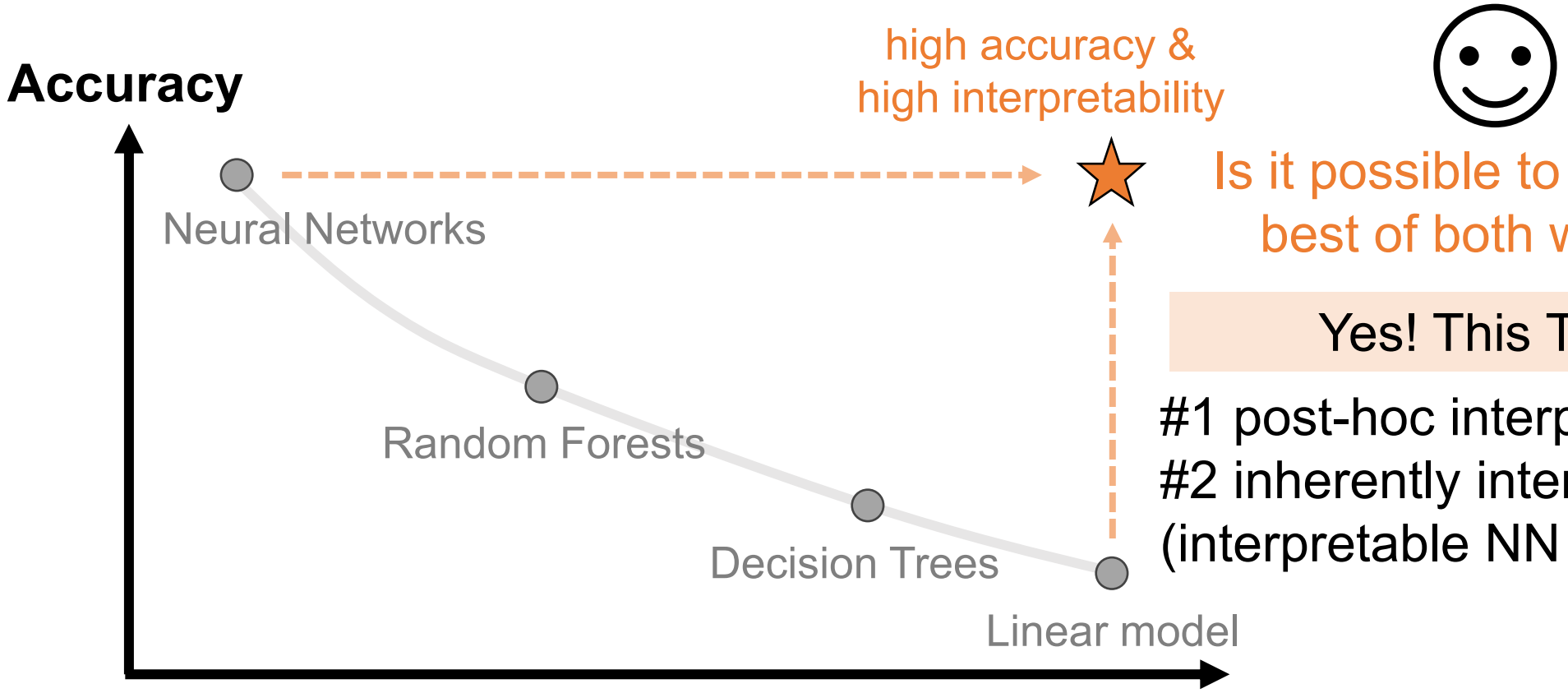
Safety

- ✓ Evaluate & monitor their capabilities
- ✓ Understand their failure modes & when we can safely deploy them

Improved models

- ✓ Gaining insight on how and what our models learn
- ✓ Helps design better models and training methods

Can we bring Interpretability into Deep Learning?



Is it possible to have the best of both worlds?

Yes! This Tutorial

#1 post-hoc interpretability tool
#2 inherently interpretable NN (interpretable NN by design)

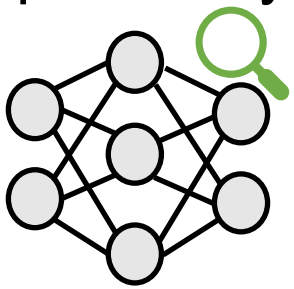
Accuracy & Interpretability trade-off exists

Goal of this Tutorial: Principled Interpretability

- Provide a unified overview of interpretability in deep learning

1

Post-hoc Interpretability tools



What knowledge has the model learned?

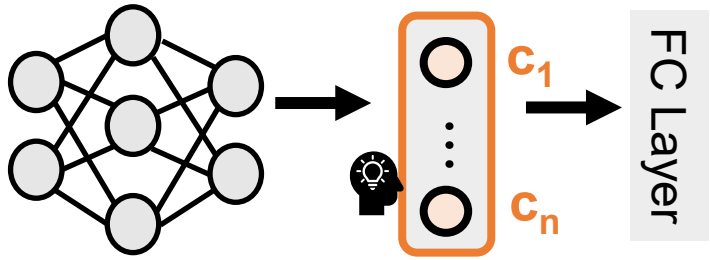
2

Rigorous Analysis to Evaluate interpretability

Which explanation is more faithful?

3

Building Inherently Interpretable DNNs

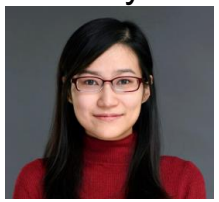


Embed human-understandable concepts c_i

Goal of this Tutorial: Principled Interpretability

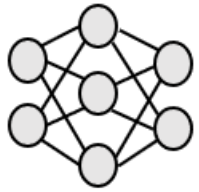
- We will focus on interpretability in deep vision models in this Tutorial, and draw connections to LLMs when applicable

Lily



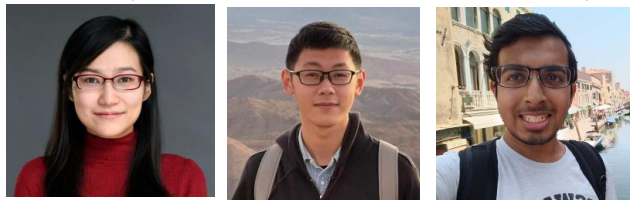
0

Fundamentals & backgrounds



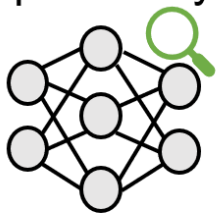
How does a deep vision model work?

Lily Ge Akshay



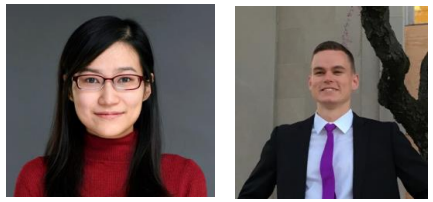
1

Post-hoc Interpretability tools



What knowledge has the model learned?

Lily Tuomas



2

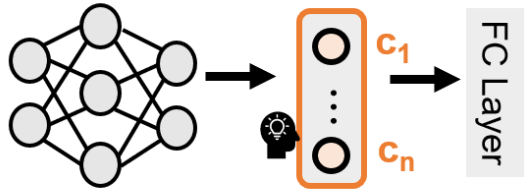
Rigorous Analysis to Evaluate interpretability

Which explanation is more faithful?

Ge Akshay

3

Building Inherently Interpretable DNNs

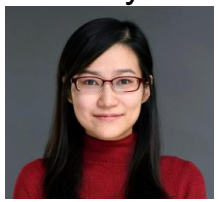


Embed human-understandable concepts c_i

Goal of this Tutorial: Principled Interpretability

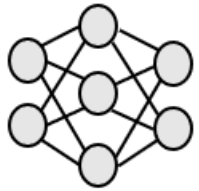
- We will focus on interpretability in deep vision models in this Tutorial, and draw connections to LLMs when applicable

Lily



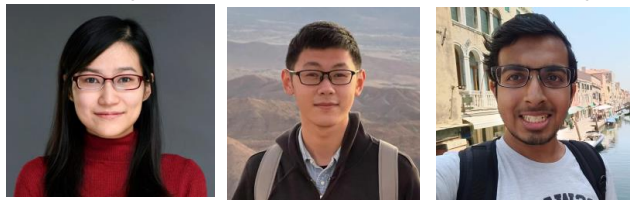
Part 1
1:00 - 1:20

Fundamentals & backgrounds



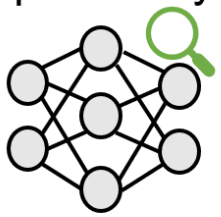
How does a deep vision model work?

Lily Ge Akshay



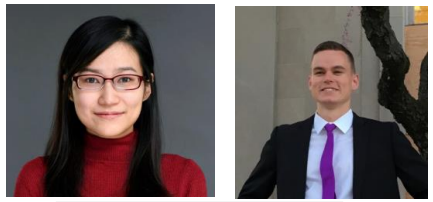
Part 2
1:20 - 2:30

Post-hoc Interpretability tools



What knowledge has the model learned?

Lily Tuomas



Part 3
2:30 - 3:00

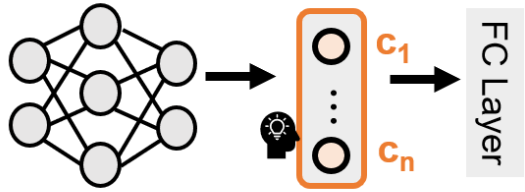
Rigorous Analysis to Evaluate interpretability

Which explanation is more faithful?

Ge Akshay

Part 4
3:15 - 3:55

Building Inherently Interpretable DNNs

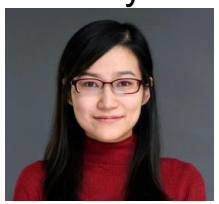


Embed human-understandable concepts c_i

Principled Interpretability Tutorial: Part 1

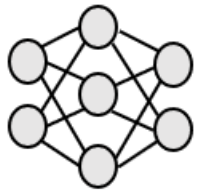
- Provide a unified overview of interpretability in deep learning

Lily



Part 1
1:00 - 1:20

Fundamentals & backgrounds



How does a deep vision model work?

Lily Ge Akshay

Part 2

Post-hoc Interpretability tools

What knowledge has the model learned?

Lily Tuomas

Part 3

Rigorous Analysis to Evaluate interpretability

Which explanation is more faithful?

Ge Akshay

Part 4

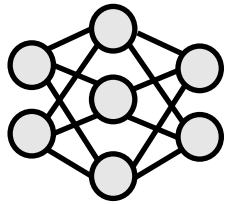
Building Inherently Interpretable DNNs

Embed human-understandable concepts c_i

Basic task: Supervised Learning

- Classification Task

Data: x



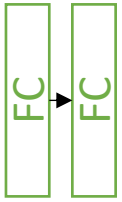
NN model
$f(x)$

Label: y

Cat

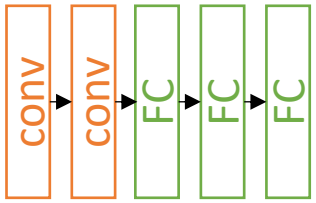
1

Multi-layer Perceptron (MLP)



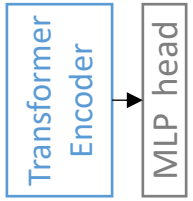
2

Convolutional Neural Network (CNN)



3

Vision Transformer (ViT)



Architecture 1: MLP

$$f(x) = W^{(2)}\sigma\left(W^{(1)}x + b^{(1)}\right) + b^{(2)}$$

1 Multi-layer Perceptron (MLP)

x

FC 256 → FC 10

e.g. 2 layer NN

$$\begin{bmatrix} -20.2 \\ 0.3 \\ \vdots \\ 90.6 \end{bmatrix}$$

Ship score
Dog score
⋮
Cat score

$f(x)$

input x

1st pixel
2nd pixel
3rd pixel
⋮
3072th pixel

weight $W^{(1)}$
bias $b^{(1)}$
activation $\sigma(\cdot)$

1st hidden layer $z^{(1)} \in \mathbb{R}^{256}$
256 neurons

1st hidden neuron
2nd hidden neuron
⋮
256th hidden neuron

weight $W^{(2)}$
bias $b^{(2)}$
activation $\sigma(\cdot)$

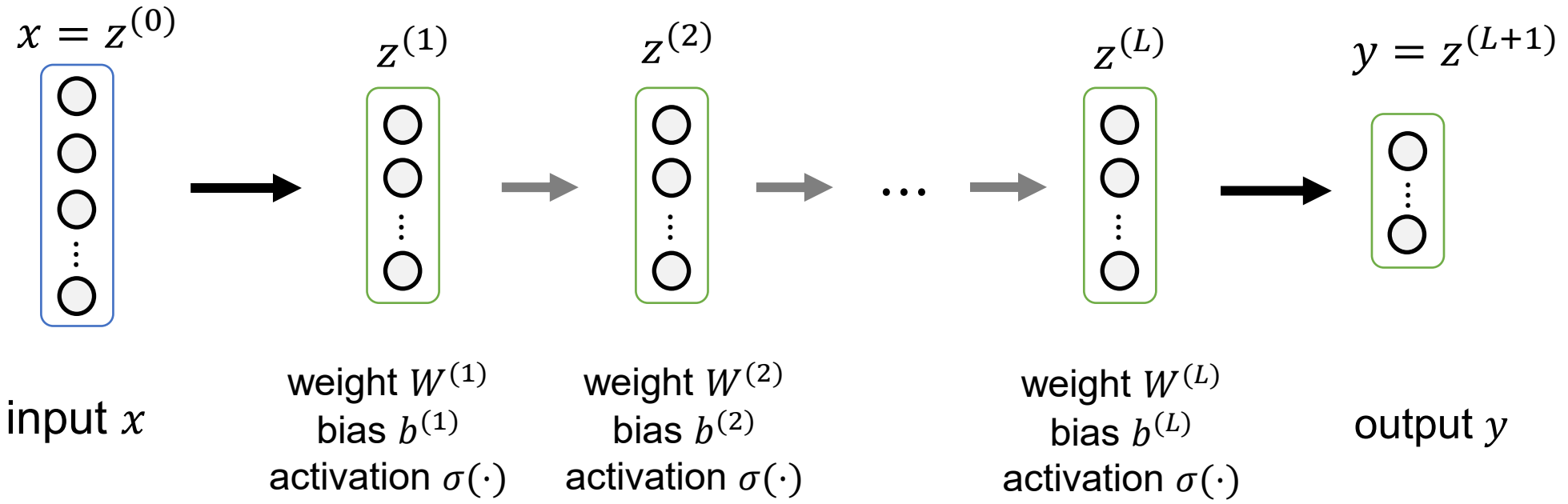
2nd layer (output) $y \in \mathbb{R}^{10}$
10 neurons (classes)

1st output neuron
⋮
10th output neuron

Note: **# of hidden neurons (256)** ≠ # of parameters (~789k) ; # of parameters is usually much larger

Architecture 1: MLP

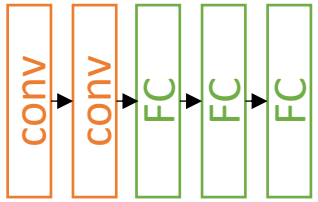
- MLP model can have many hidden layers, e.g. a (L+1)-layer MLP model
- Generally, when we say "i_{th} neuron at l_{th} layer" ⇒ hidden representation/post-activation $z_i^{(l)} \in \mathbb{R}$



Architecture 2: CNN

2 Convolutional Neural Network (CNN)

x



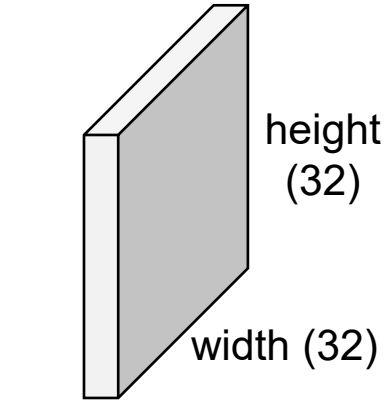
e.g. 5 layer CNN

$$\begin{bmatrix} -81 \\ -40 \\ \vdots \\ 110.8 \end{bmatrix}$$

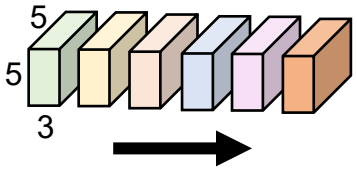
Ship score
Dog score
⋮
Cat score

$f(x)$

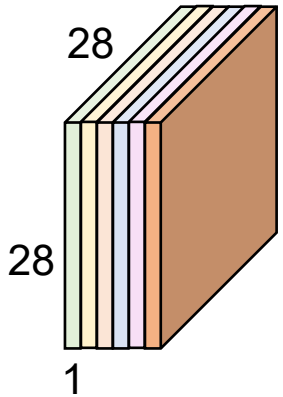
input x



1st hidden layer $z^{(1)} \in \mathbb{R}^{6\times28\times28}$
6 neurons



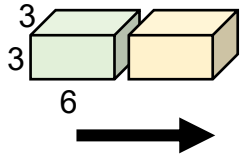
Conv filters
$W^{(1)}$, $b^{(1)}$
activation $\sigma(\cdot)$



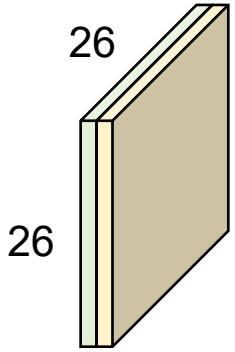
1st hidden neuron

6th hidden neuron

2nd hidden layer $z^{(2)} \in \mathbb{R}^{2\times26\times26}$
2 neurons



Conv filters
$W^{(2)}$, $b^{(2)}$
activation $\sigma(\cdot)$



1st hidden neuron

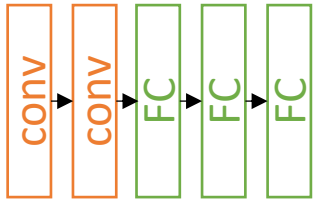
2nd hidden neuron

…

Architecture 2: CNN

2 Convolutional Neural Network (CNN)

x



e.g. 5 layer CNN

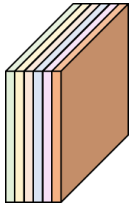
$$\begin{bmatrix} -81 \\ -40 \\ \vdots \\ 110.8 \end{bmatrix}$$

Ship score
Dog score
⋮
Cat score

$f(x)$

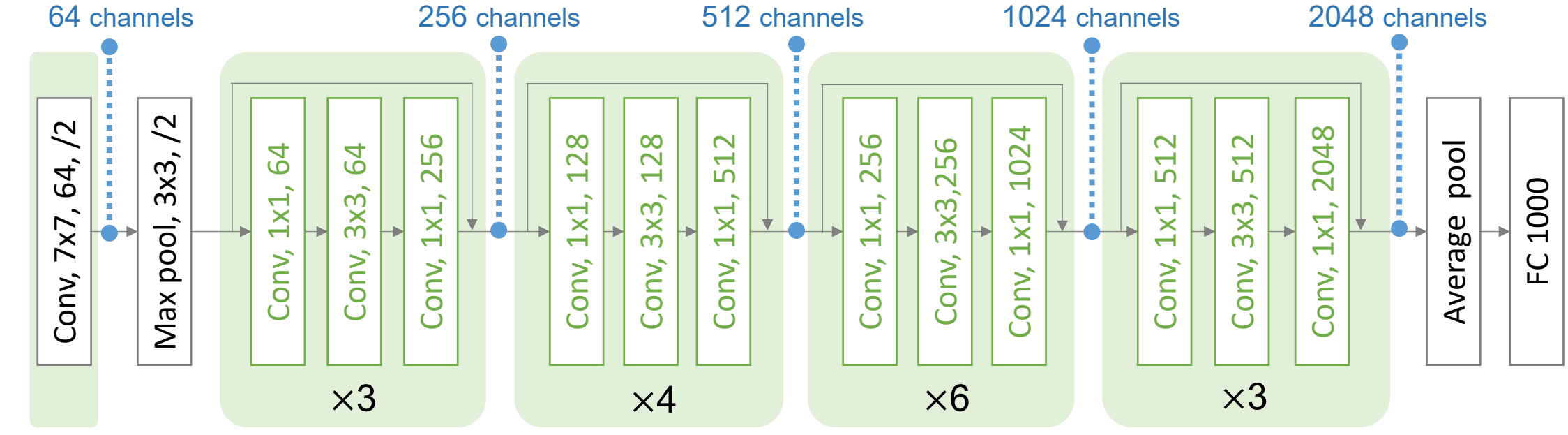
- In MLP
  - hidden layer representation at Layer l is an **1-D activation vector**: $z^{(l)} \in \mathbb{R}^{n_l}$
  - a hidden neuron is a **scalar,** i.e. has scalar activation value: $z_i^{(l)} \in \mathbb{R}$

- In CNN
  - hidden layer representation at Layer l is a **3-D activation map**: $z^{(l)} \in \mathbb{R}^{n_{l,W} \times n_{l,H} \times n_{l,D}}$
  - a hidden neuron is a **channel**, i.e. 2-D activation map: $z_i^{(l)} \in \mathbb{R}^{n_{l,W} \times n_{l,H}}$



Architecture 2: CNN

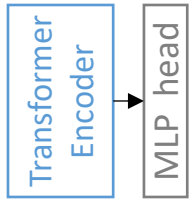
- As an example, Resnet 50 has **5 major building blocks**
- There are total 3904 hidden neurons (64+256+512+1024+2048) **in the end** of Layer 0-4
  - Note: **total # of neurons** in Layer 0 to 4 ≈ **4k** ; c.f. # of parameters ≈ 23.5 million



Architecture 3: ViT

3 Vision Transformer (ViT)

x



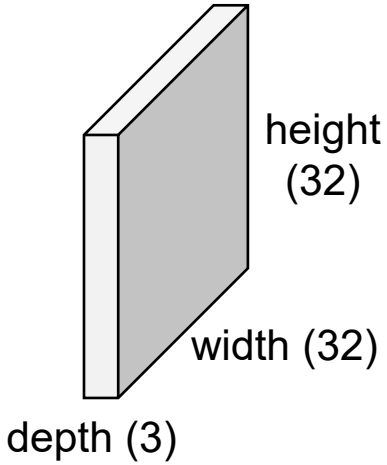
e.g. ViT-Base (ViT-B/16)

$$\begin{bmatrix} -96 \\ -30 \\ \vdots \\ 160.3 \end{bmatrix}$$

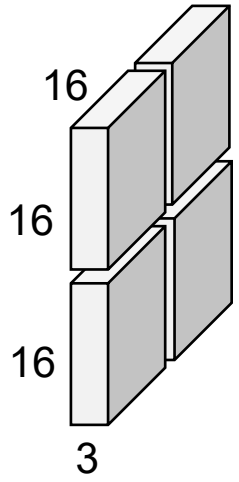
Ship score
Dog score
⋮
Cat score

$f(x)$

input x

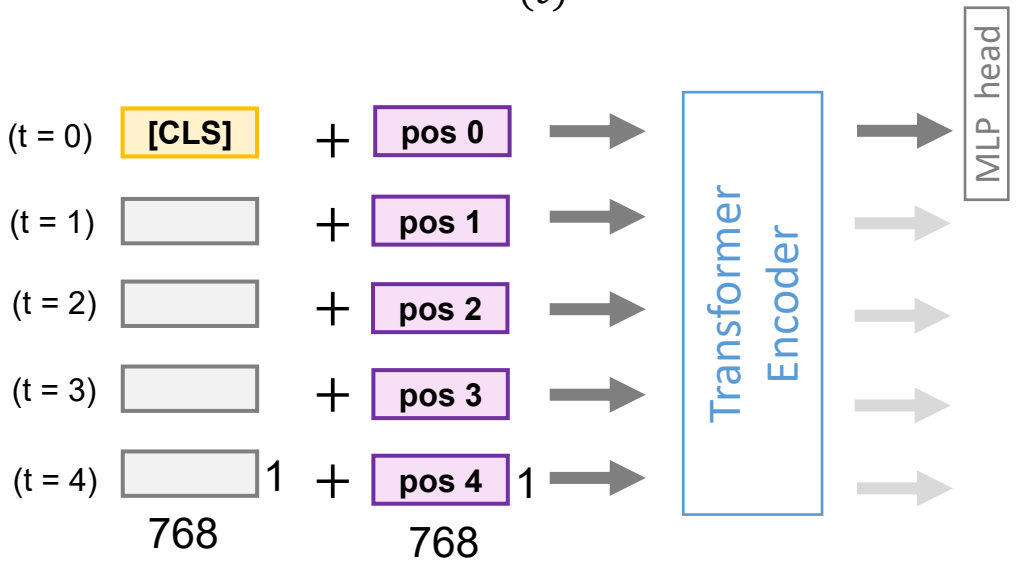


tokenize (split images into patches)



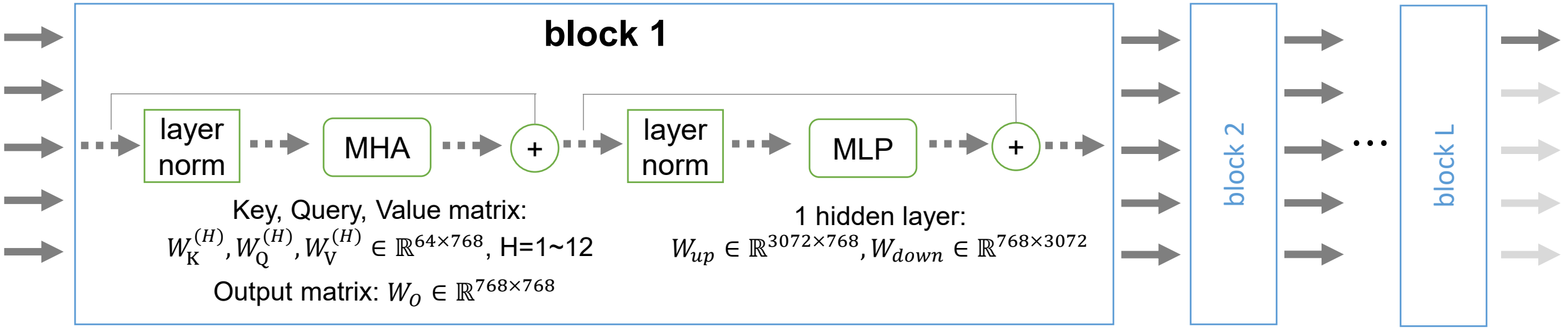
linear projection of each patch W_{emb}

input embedding vectors $z_{(t)}^{(0)} \in \mathbb{R}^{768}$



Architecture 3: ViT

- **Transformer Encoder** has L blocks, each block consists of two main module: Multi-Head Attention layer (**MHA**) and Feedforward layer (**MLP**)



Each input & output (at position t) of a block remain same dimension: $z_{(t)}^{(i-1)} \in \mathbb{R}^{768}$, $z_{(t)}^{(i)} \in \mathbb{R}^{768}$

The above diagram (dashed arrow) in block 1 only show the position-wise operation for input $z_{(t)}^{(0)}$

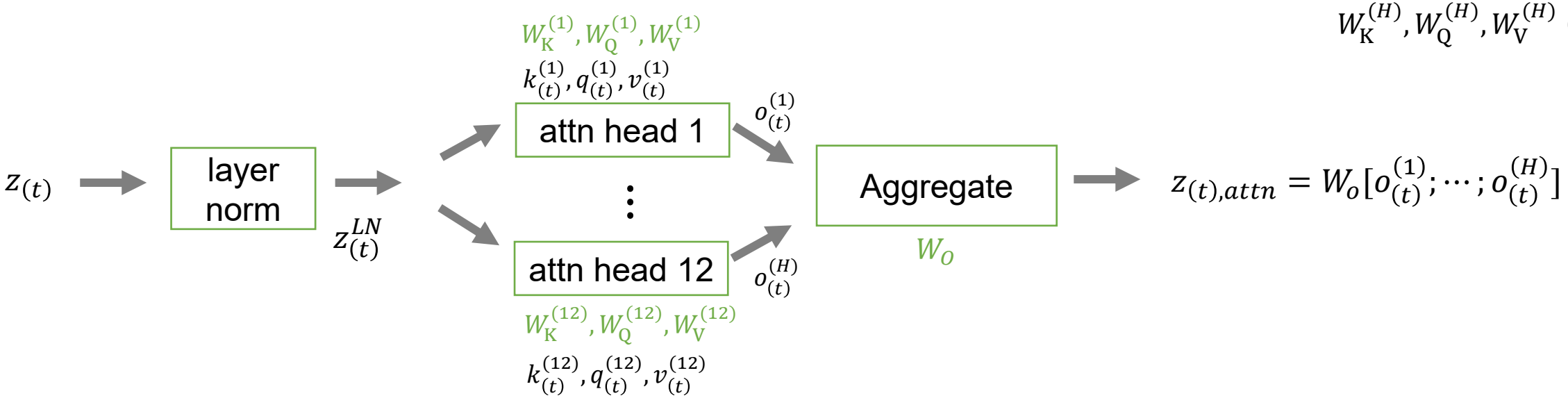
Architecture 3: ViT

- Detailed look at Multi-Head Attention layer (MHA)

Learnable parameters:

$W_O \in \mathbb{R}^{768\times768}$

$W_{\mathrm{K}}^{(H)}, W_{\mathrm{Q}}^{(H)}, W_{\mathrm{V}}^{(H)} \in \mathbb{R}^{64\times768}$



Key at Head H: $k_{(t)}^{(H)} = W_K^{(H)} z_{(t)}^{LN}$

Query at Head H: $q_{(t)}^{(H)} = W_Q^{(H)} z_{(t)}^{LN}$

Value at Head H: $v_{(t)}^{(H)} = W_V^{(H)} z_{(t)}^{LN}$

Attention: $\alpha_{t,i}^{(H)} = q_{(t)}^{(H)} \cdot k_{(i)}^{(H)}$

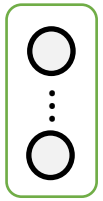
Output at head H: $o_{(t)}^{(H)} = \sum_{i=1\sim T} \alpha_{t,i}^{(H)} v_{(t)}^{(H)}$

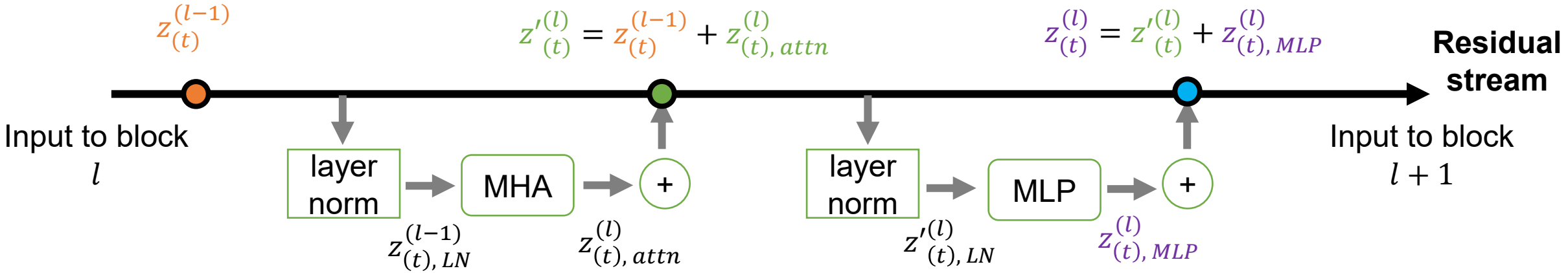
$k_{(t)}^{(H)}, q_{(t)}^{(H)}, v_{(t)}^{(H)}, o_{(t)}^{(H)} \in \mathbb{R}^{64}$ $\quad$ $z_{(t)}, z_{(t)}^{LN}, z_{(t),attn} \in \mathbb{R}^{768}$

Architecture 3: ViT

$$z_{(t)}^{(l)} = z_{(t)}^{(0)} + \sum_{i=1}^{l-1} z_{(t),\,attn}^{(i)} + \sum_{i=1}^{l-1} z_{(t),\,MLP}^{(i)}$$

- In ViT
  - hidden layer representation at Layer l for position t is an **1-D activation vector**: $z_{(t)}^{(l)} \in \mathbb{R}^{d_{model}}$, often refer to as **residual stream**
  - we also often look at a hidden neuron in the **MLP** layer, which is a **scalar,** i.e. has scalar activation value: $z_{(t),\,MLP,\,i}^{(l)} \in \mathbb{R}$



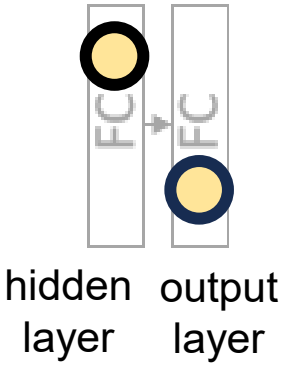


Summary

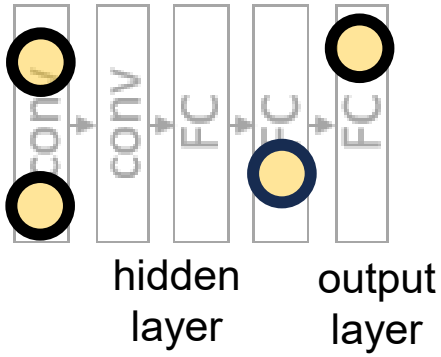
In **Part 2.1**, we will introduce methods to interpret neurons

- For an individual neuron ○, smallest unit in the network
  - It's a scalar, we look at the "activation value" (i.e. post-activation, $\sigma(\cdot)$)

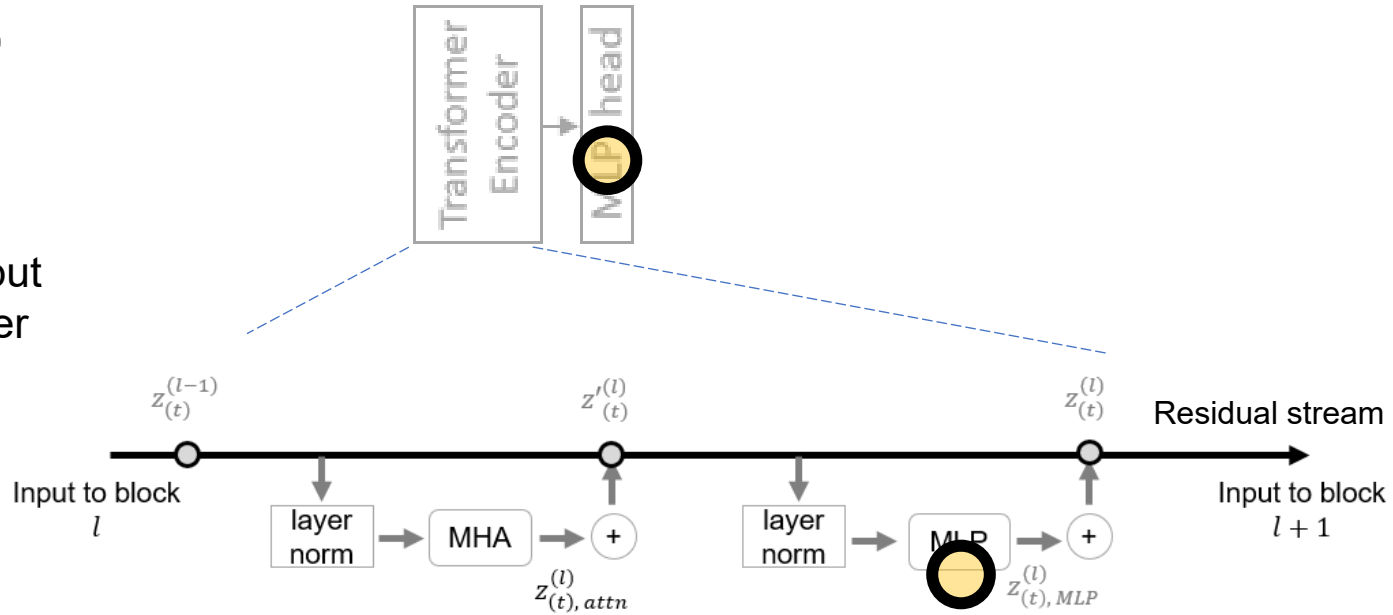
MLP



CNN



ViT

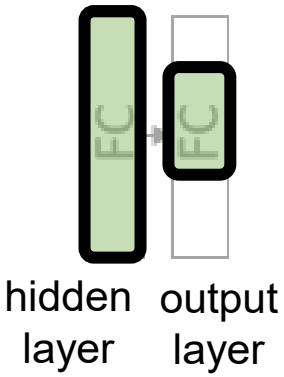


Summary

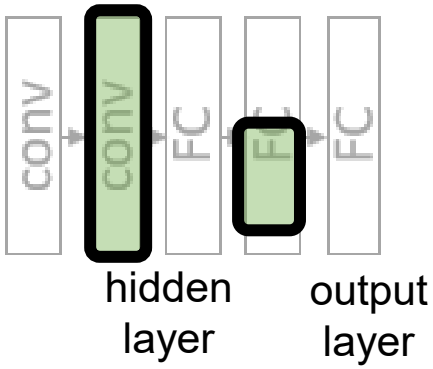
In **Part 2.2**, we will introduce methods to interpret representation vectors

- For an representation vector ▢, medium unit in the network
  - It's a vector, can be pre-activation or post-activation, $\sigma(\cdot)$

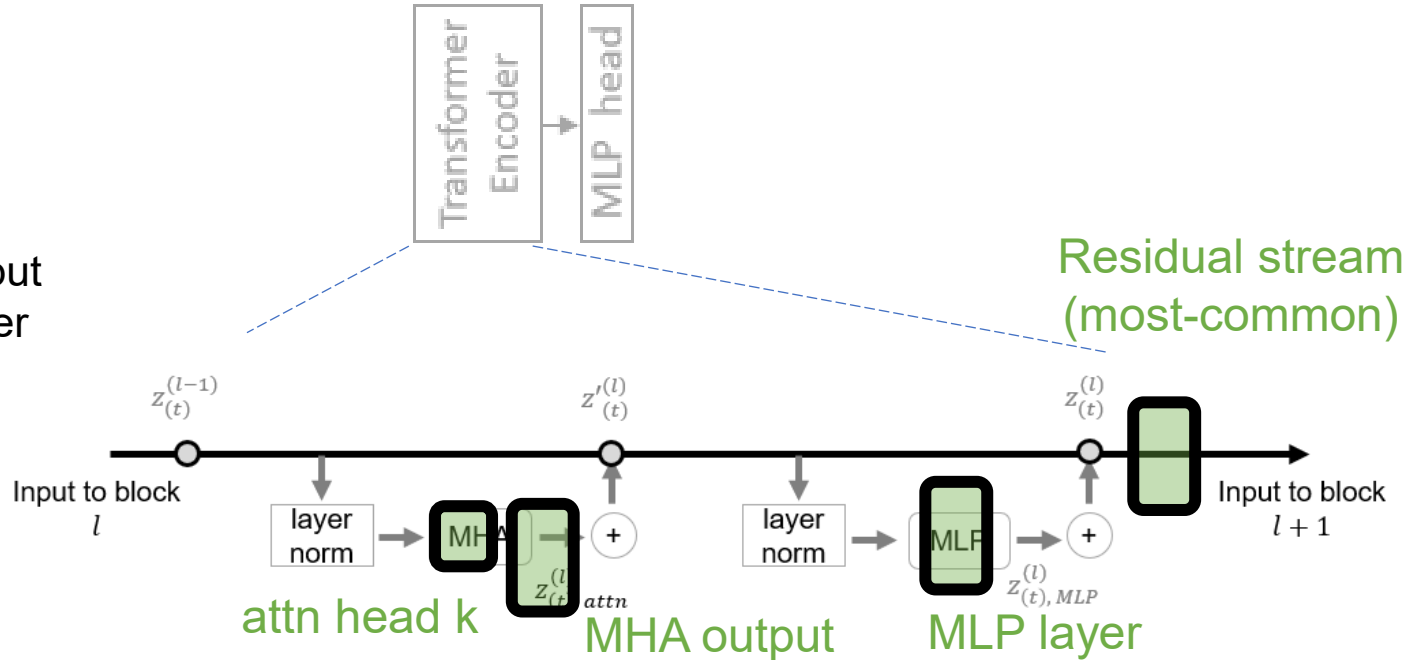
MLP



CNN



ViT



Principled Interpretability Tutorial: Part 2

https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

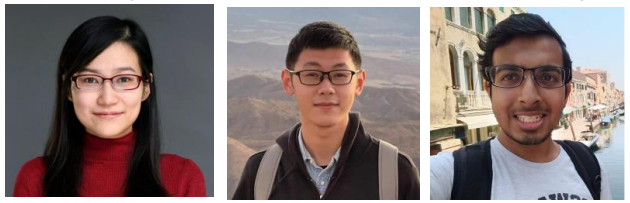
Lily

Part 1

Fundamentals & backgrounds

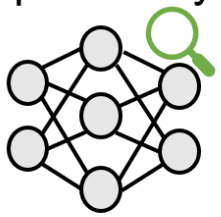
How does a deep vision model work?

Lily Ge Akshay



Part 2
1:20 - 2:30

Post-hoc Interpretability tools



What knowledge has the model learned?

Lily Tuomas

Part 3

Rigorous Analysis to Evaluate interpretability

Which explanation is more faithful?

Ge Akshay

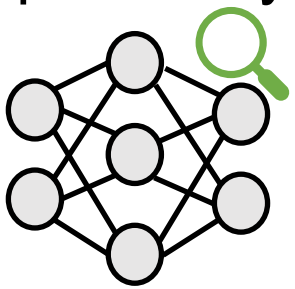
Part 4

Building Inherently Interpretable DNNs

Embed human-understandable concepts c_i

Tutorial Part 2: Overview

Post-hoc Interpretability tools



What knowledge has the model learned?

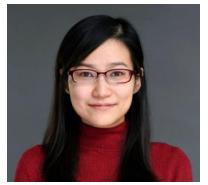
⇒ We will discuss methods from small unit to large unit

1. Understand individual neurons (**neuron-level** interpretability tools)

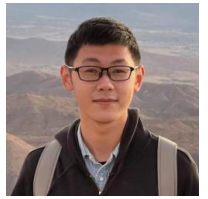
2. Understand groups of neurons / full layer (**representation-level** interpretability tools)

3. Understand multiple layers / circuits (**circuit-level** interpretability tools)

Lily



Ge

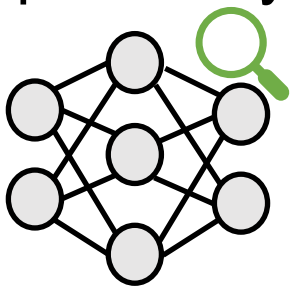


Akshay

https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

Tutorial Part 2: Overview

Post-hoc Interpretability tools

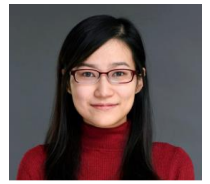


What knowledge has the model learned?

⇒ We will discuss methods from small unit to large unit

1. Understand individual neurons (**neuron-level** interpretability tools)

Lily



2. Understand groups of neurons / full layer (**representation-level** interpretability tools)

Ge

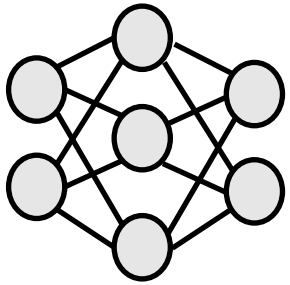
3. Understand multiple layers / circuits (**circuit-level** interpretability tools)

Akshay

https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

What is an automated interpretability tool?

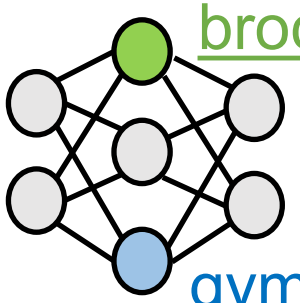
Given a DNN



Interpretability Tools

Understand model internals

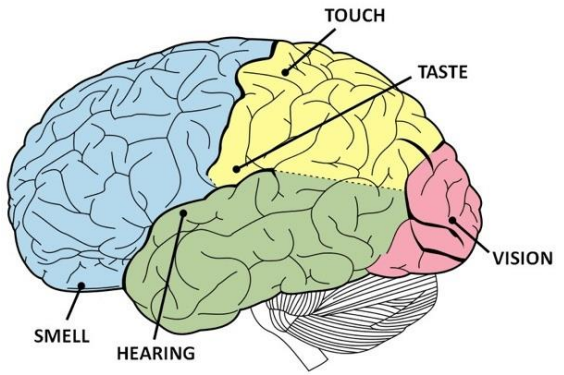
(Neuron concepts)



broccoli

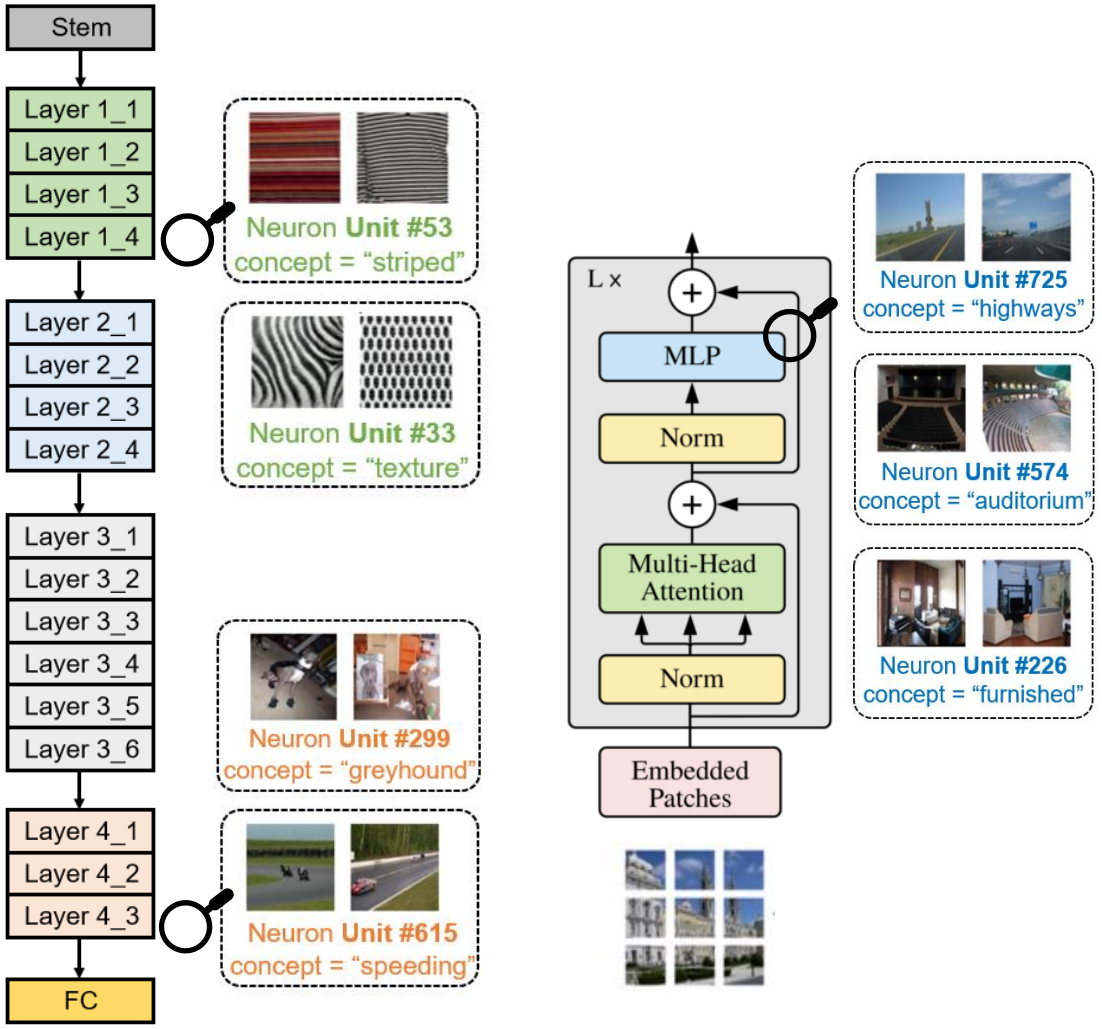
gymnastic

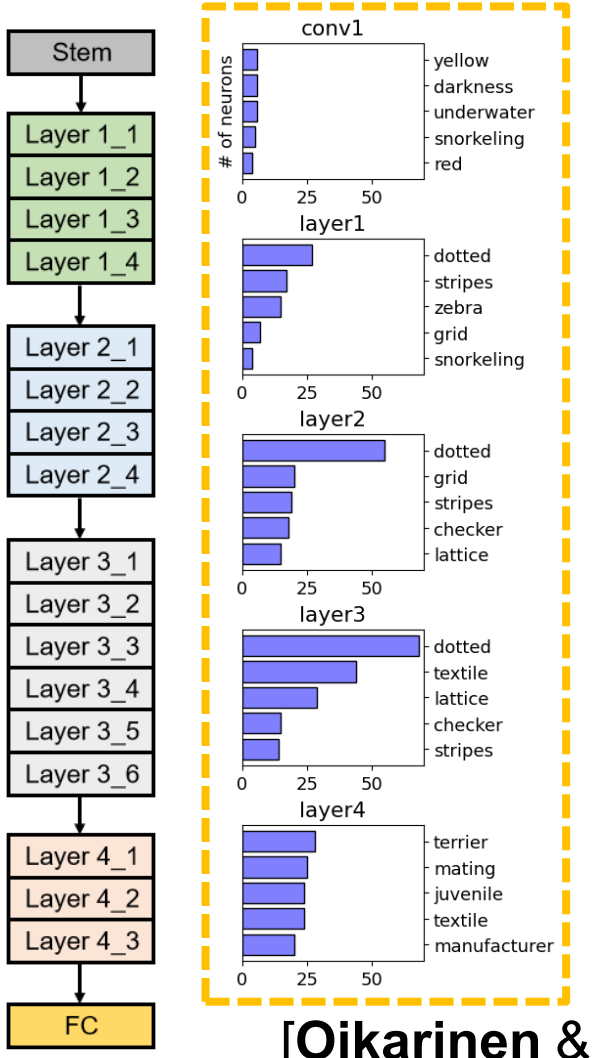
- Network Dissection (Bau & Zhou etal, CVPR'17)
- CompExp (Mu & Andreas, NeurIPS'20)
- MILAN (Hernandez etal, ICLR'22)
- CLIP-Dissect (Oikarinen & Weng, ICLR'23)
- INVERT (Bykov etal, NeurIPS'23)
- Clustered CompExp (La Rosa etal, NeurIPS'23)
- Linear Explanation (Oikarinen & Weng, ICML'24)
- Describe-n-Dissect (Bai & Iyer etal, ICML'24 MI, TMLR'25)
- MAIA (Shaham & Schwettmann etal, ICML'24)
- SAND (Srinivas etal, WACV'25)



https://buffer.com/resources/10-surprising-facts-about-how-our-brain-works/

Understand the model internals of your DNN





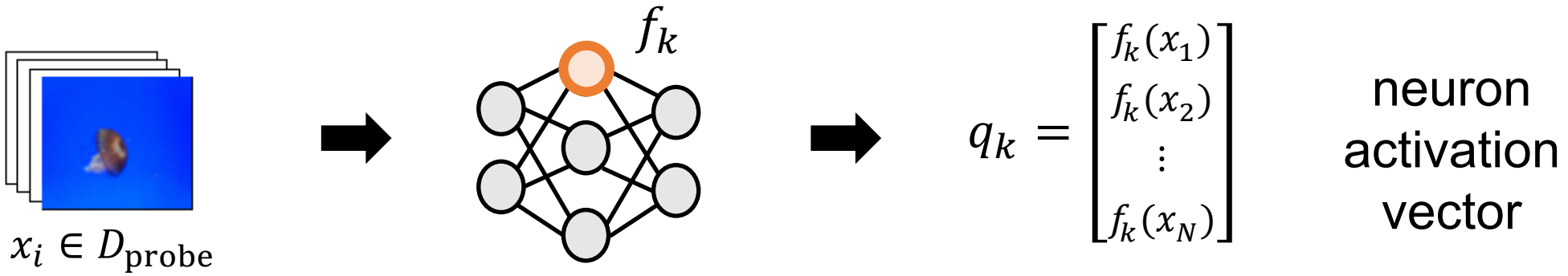
Get a Full profile of your DNN!

[**Oikarinen** & **Weng**, ICLR'23 spotlight]

Key Ideas

- To identify the concept of a neuron, it generally requires two steps:

1. Record Neuron Patterns



2. Decode Neuron Patterns

find the most appropriate concept t^* that best describe the neuron activation patterns q_k

Key properties

- The key difference between different methods lies in the following choice:

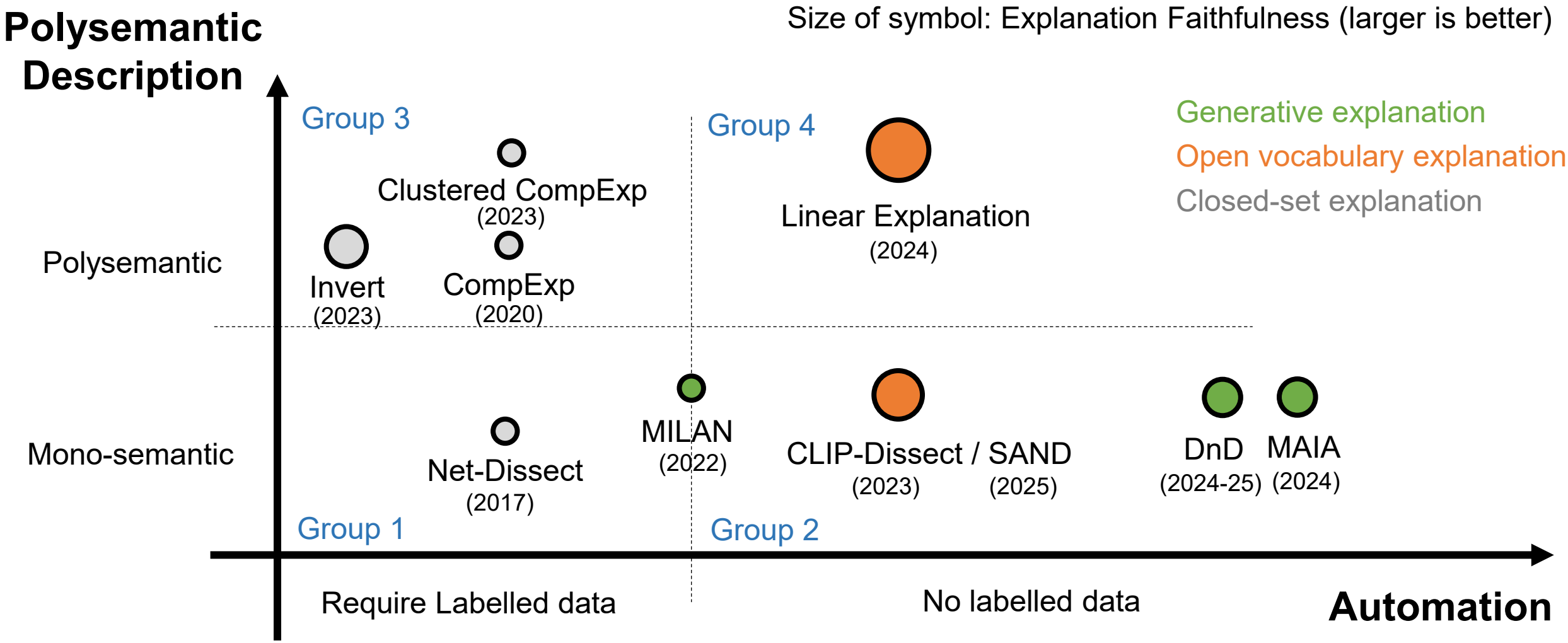
  (1) Automation:
  do we need to collect the probing dataset with concept labels?

  (2) Polysemantic Description:
  does the explanation describe all functionality of the neurons?
  does the explanation describe the neuron's full range activations?

  (3) Flexibility:
  does the explanation come from closed-set or open-set or generation?

  (4) Faithfulness:
  how well does the explanation reflect the neuron's functionalities?

Comparison


Polysemantic Description
Size of symbol: Explanation Faithfulness (larger is better)
Generative explanation
Open vocabulary explanation
Closed-set explanation
Group 3
Group 4
Clustered CompExp (2023)
Linear Explanation (2024)
Polysemantic
Invert (2023)
CompExp (2020)
Mono-semantic
Net-Dissect (2017)
MILAN (2022)
CLIP-Dissect / SAND (2023) (2025)
DnD (2024-25)
MAIA (2024)
Group 1
Group 2
Require Labelled data
No labelled data
Automation

Group 1: Mono-semantic, Required Label-Data

- Network Dissection [1]

1. Collect dataset containing a dictionary of human-labeled visual concepts

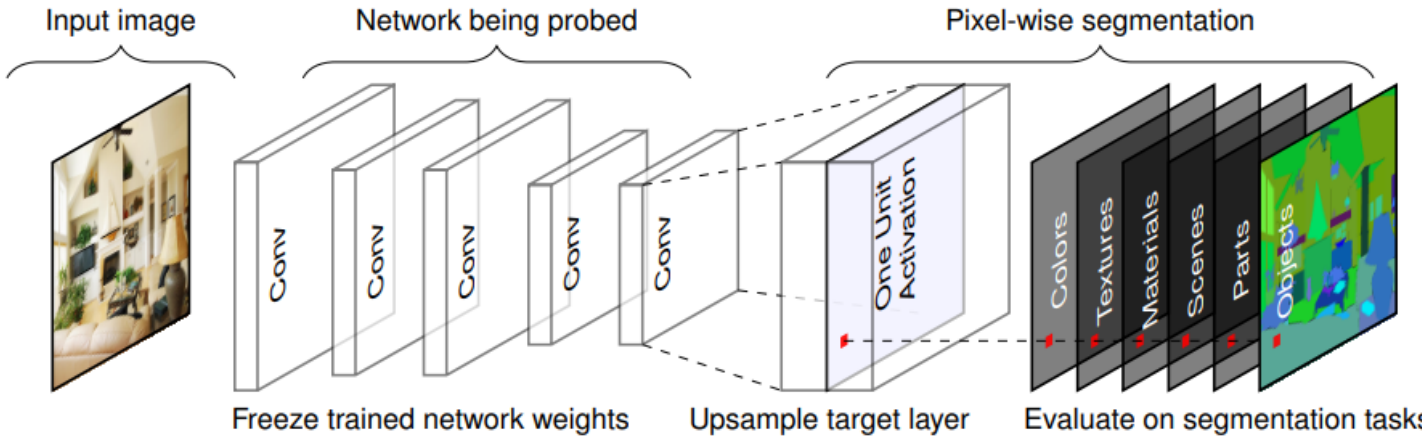
Broden Dataset

Broadly and **Den**sely annotated Dataset

ADE20K	Zhou et al, CVPR '17
Pascal Context	Mottaghi et al, CVPR '14
Pascal Part	Chen et al, CVPR '14
Open Surfaces	Bell et al, SIGGRAPH '14
Desc Textures	Cimpoi et al, CVPR '14
Colors	generated

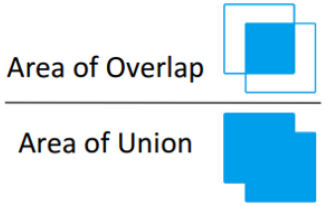
Total = 63,305 images
1,197 concepts

2. Gather response of hidden unit (conv filters) to known concepts



3. Quantify alignment of hidden unit-concept pairs (activation mask v.s. concept mask)

Intersection over Union (**IoU**) = Area of Overlap / Area of Union



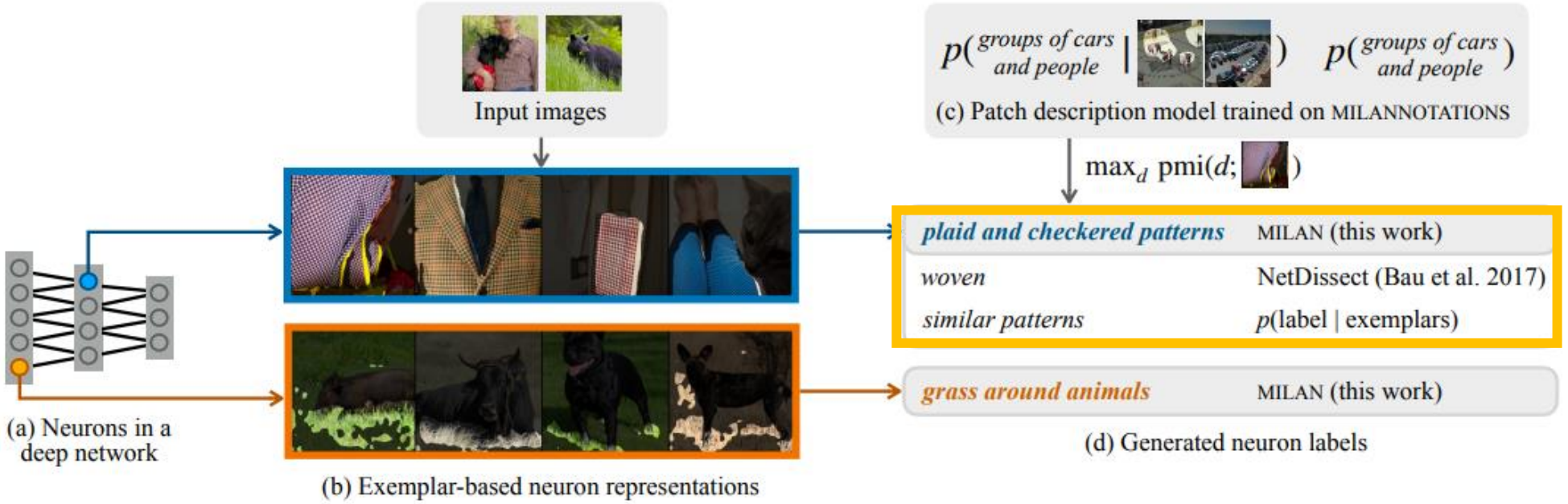
[1] Bau etal, Quantifying interpretability of Deep Visual Representations, CVPR 2017 http://netdissect.csail.mit.edu/

Group 1: Mono-semantic, Required Label-Data

- MILAN [2]

1. Collect a dataset that labels individual neurons with fine-grained natural language descriptions (MILANNONATION)

2. Use the dataset to train models that can be used to search for descriptions that best describe a new neuron through maximizing PMI

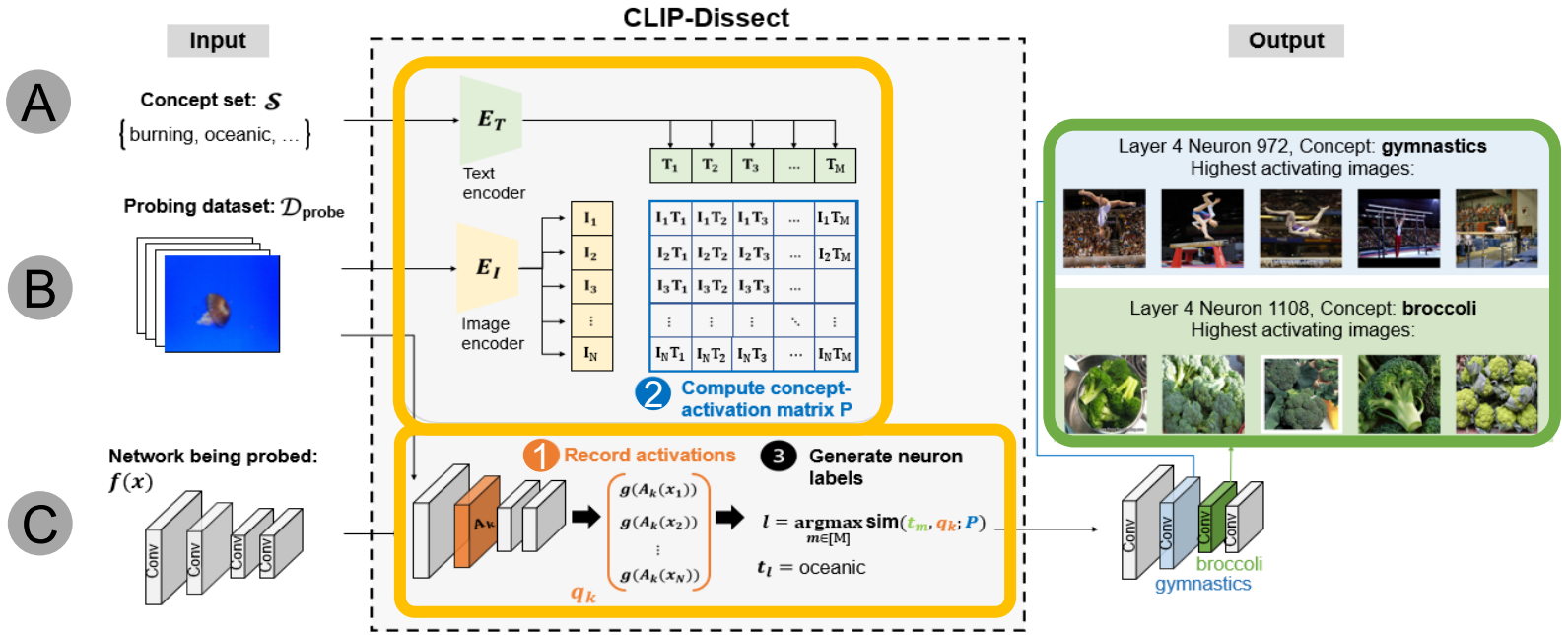


[2] Hernandez etal, Natural Language Descriptions of Deep Visual Features, ICLR 2022

Group 2: Mono-semantic, No Label-Data

- CLIP-Dissect [3]

  Use CLIP and more similarity metrics (cosine, WPMI, soft WPMI) to search for concepts that best describe neuron activities



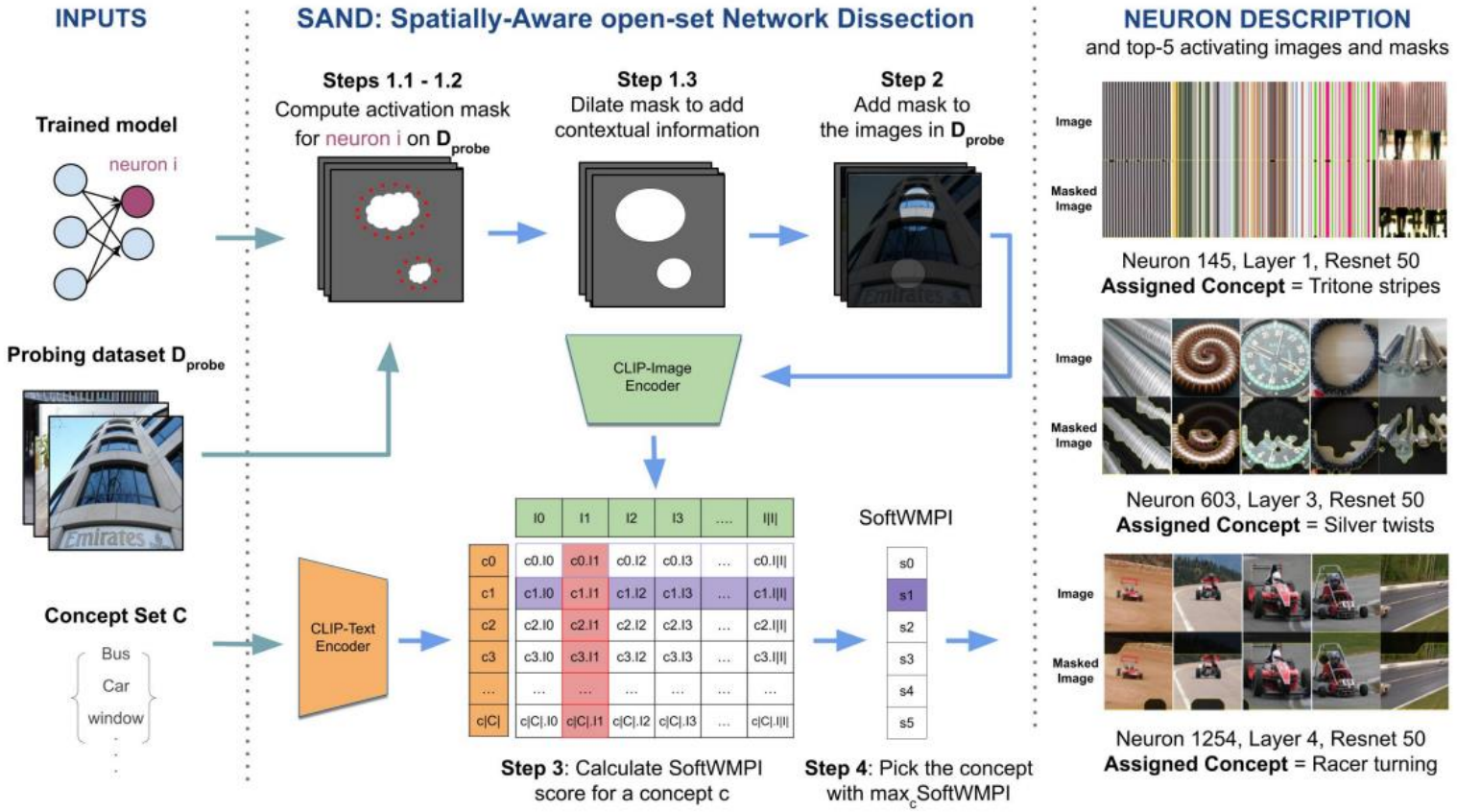
Benefits

- Concept annotation-free
- Training-Free
- Unrestricted & Open vocabulary concepts
- Efficient (10-60x faster than MILAN & Net-Dissect)

[3] Oikarinen & Weng, CLIP-Dissect: Automatic Description of Neuron Representations in Deep Vision Networks, ICLR 2023

Group 2: Mono-semantic, No Label-Data

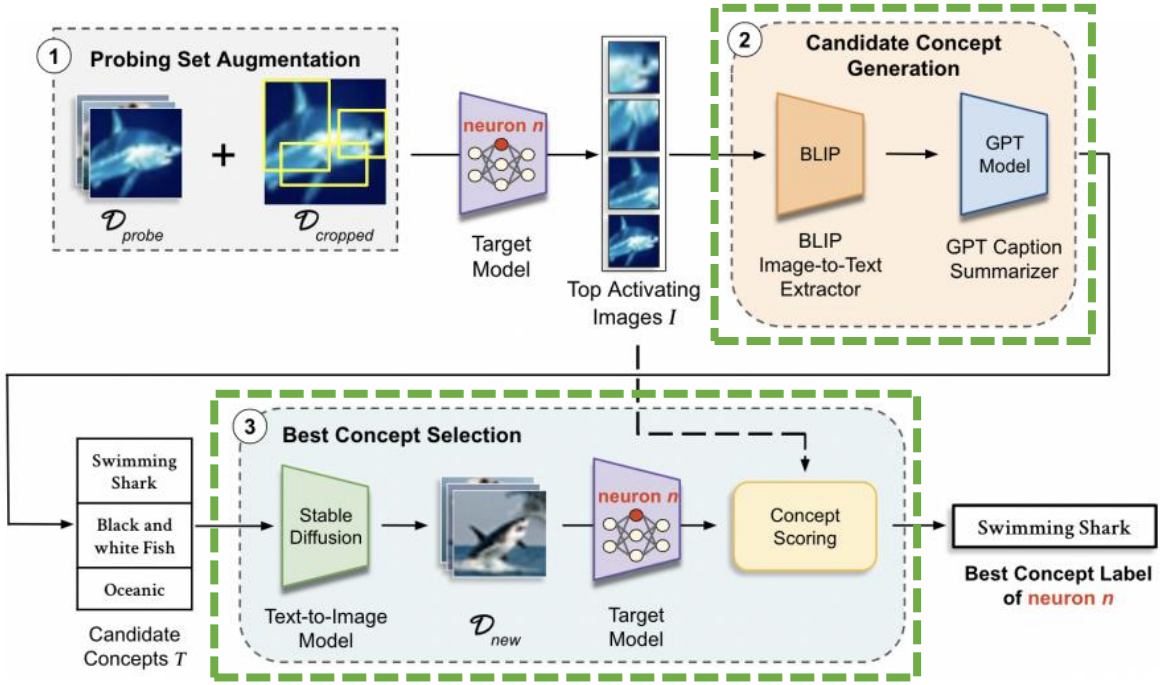
- SAND [4]
  - Build upon CLIP-Dissect pipeline
  - Consider neuron's spatial activation patterns through image masking
  - Provide higher quality of explanations on low-level concepts



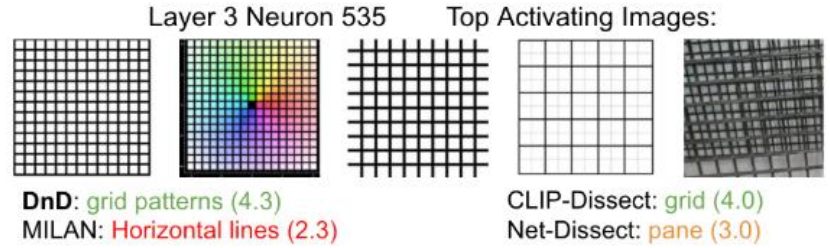
[4] Srinivas, SAND: Enhancing Open-Set Neuron Descriptions through Spatial Awareness, WACV 2025

Group 2: Mono-semantic, No Label-Data

- Describe-and-Dissect (DnD) [5]
  - Remove the need of pre-specifying concept set
  - Leverage LLMs & Image-to-text models, provide generative descriptions without collecting data



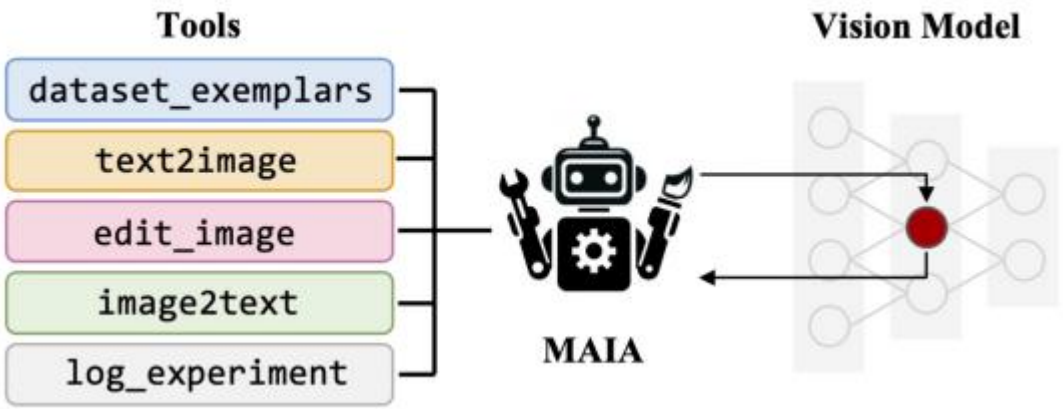
Results on ResNet-50 model trained on ImageNet

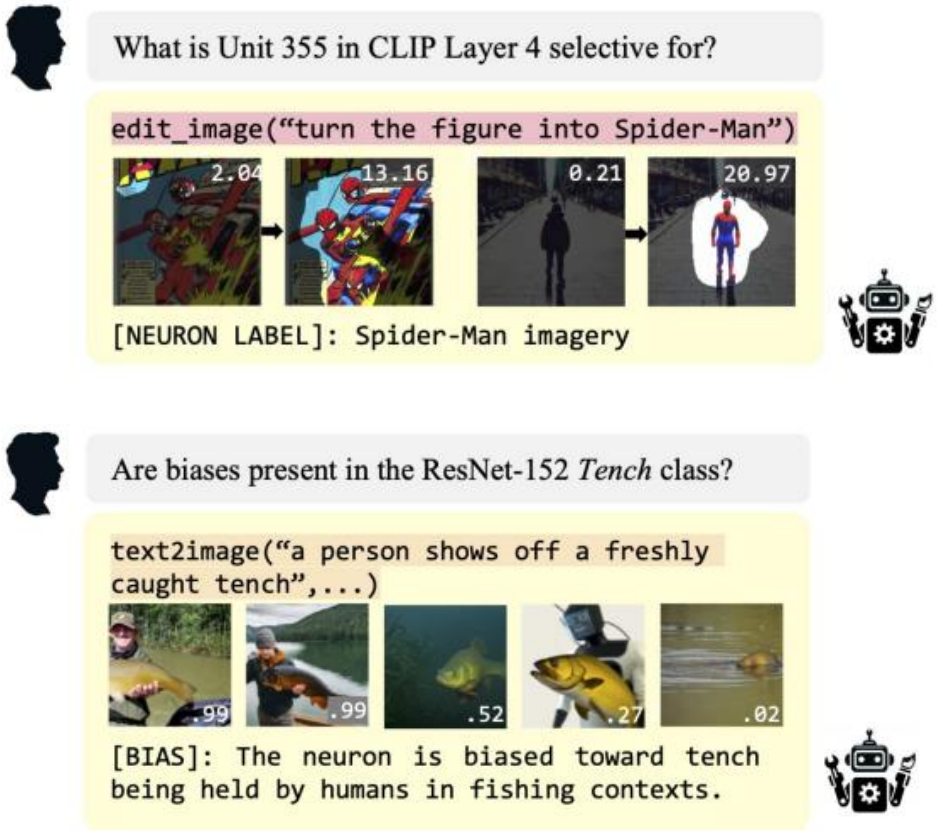


[5] Bai etal, Interpreting Neurons in Deep Vision Networks with Language Models, TMLR 2025

Group 2: Mono-semantic, No Label-Data

- MAIA [6]
  - Use LLM agent to obtain neuron explanations

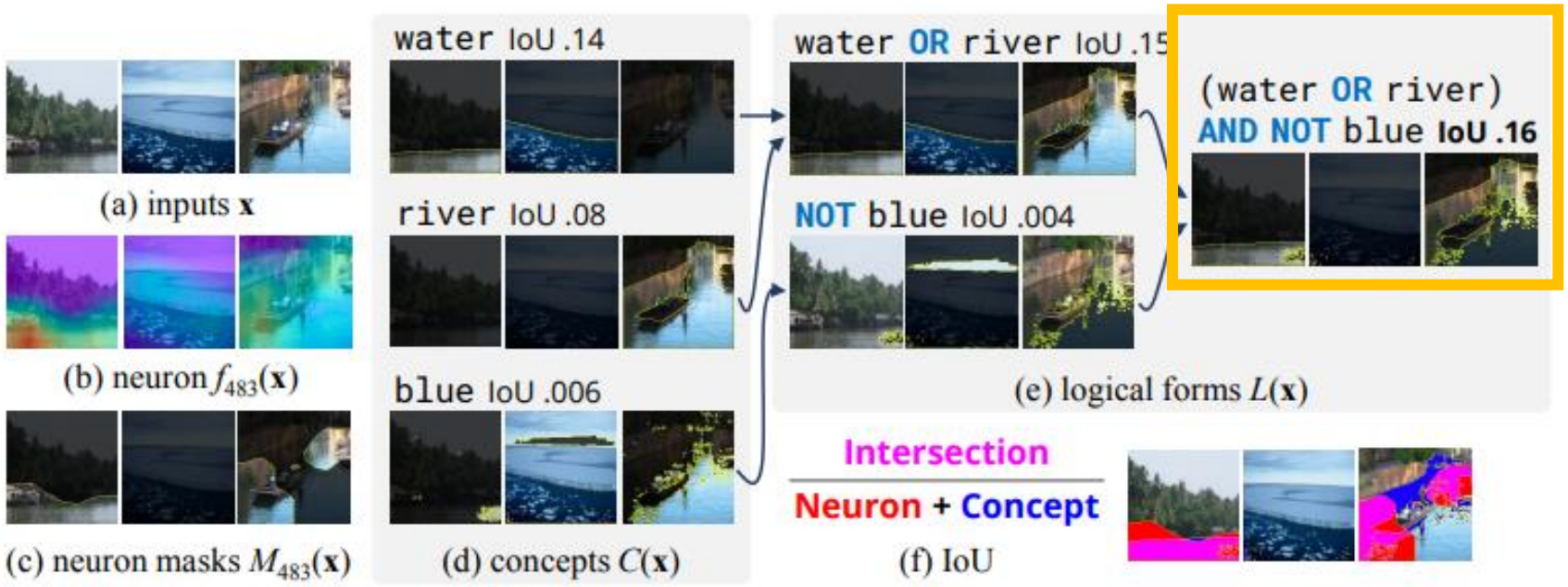




[6] Shaham etal, A Multimodal Automated Interpretability Agent, ICML 2024

Group 3: Poly-semantic, Required Label-Data

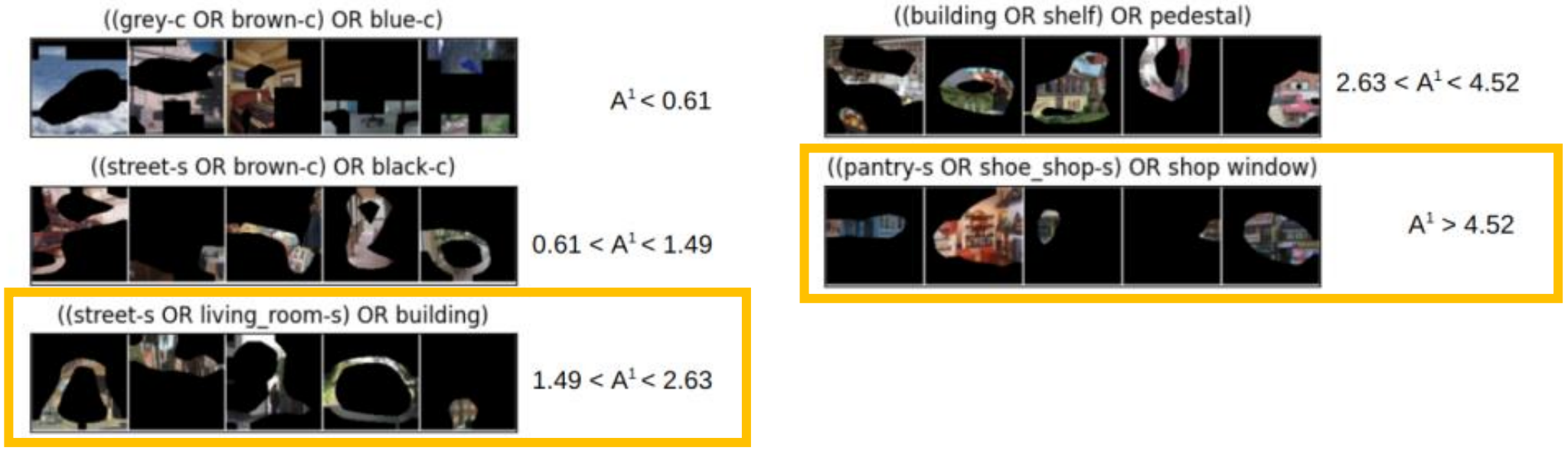
- Compositional Explanation (CompExp) [7]
  - Building upon Net-Dissect, obtain compositional explanations in logical forms “OR”, “AND”, “NOT”



[7] Mu & Andreas, Compositional Explanations of Neurons, NeurIPS 2020

Group 3: Poly-semantic, Required Label-Data

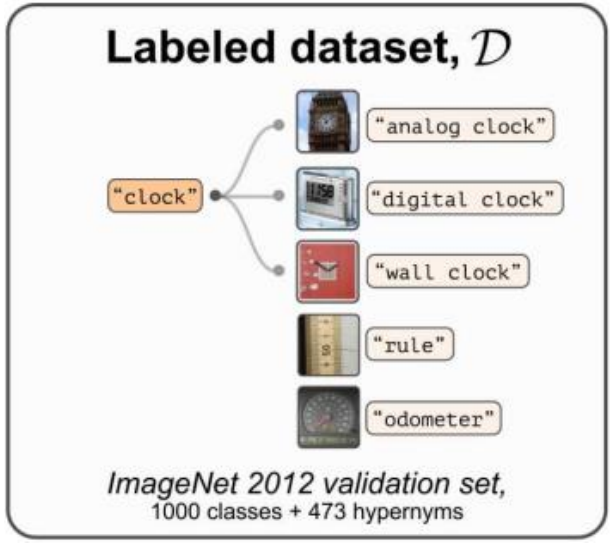
- Clustered CompExp [8]
  - Building upon CompExp, obtain a more complete compositional explanations by looking at the full range of neuron activations

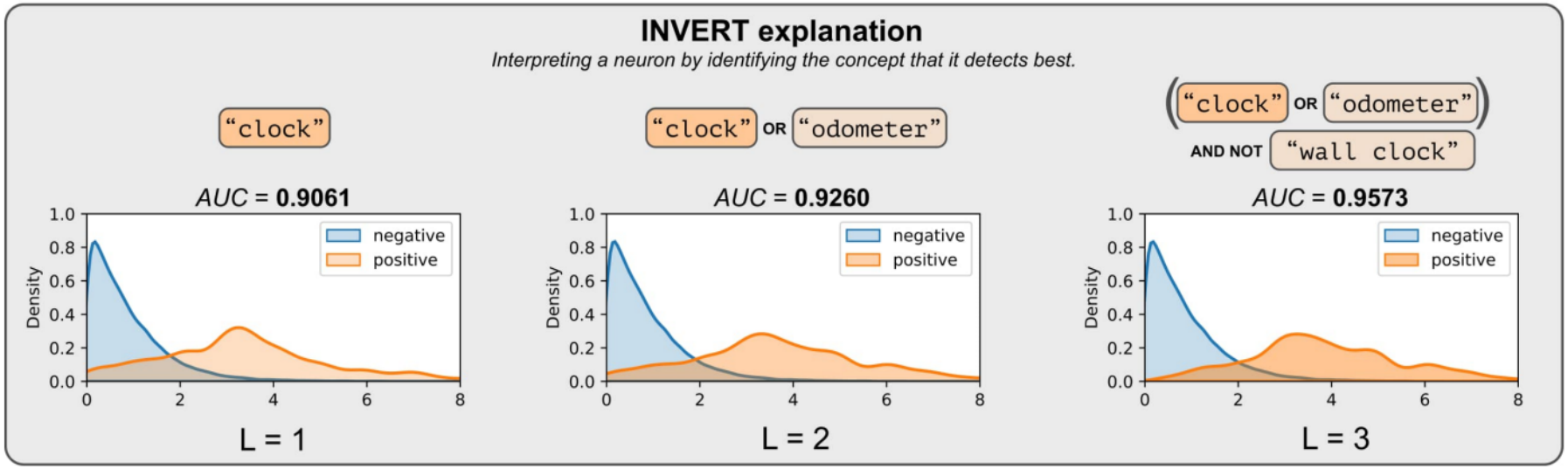


[8] La Rosa etal, Towards a fuller understanding of neurons with clustered compositional explanations, NeurIPS 2023

Group 3: Poly-semantic, Required Label-Data

- INVERT [9]
  - Use Image class-level labels (and superclass) + AUC as similarity metric to search for best descriptions for neuron activities



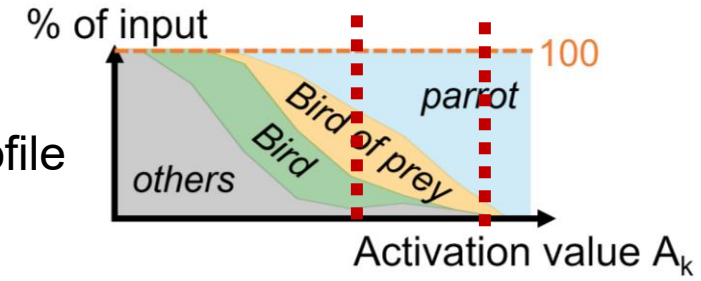


[9] Bykov etal, Labeling Neural Representations with Inverse Recognition, NeurIPS 2023

Group 4: Poly-semantic, No Label-Data

- Linear Explanation [10]
  - Explaining the "neuron activations" not just for highly activated inputs

Neuron activation profile



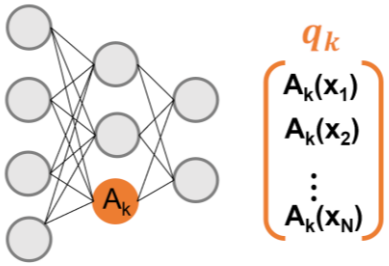
Explain neuron k with a linear combination of concepts

2.36*parrot + 2.11*(bird of prey) + 0.94*bird
[e.g. Neuron #140 in ResNet 50, layer4]

Linear Explanation for polysemantic neurons

  - can obtain linear explanation easily by solving a sparse linear regression problem

❶ **Record target neuron activations q_k**



❷ **Construct concept-activation matrix P**

Probing data

Concept			…	
Fish	1	0	…	0
Grass	1	1	…	0
Purple	0	0	…	0
⋮	⋮	⋮	⋱	⋮
Flamingo	0	0	…	1

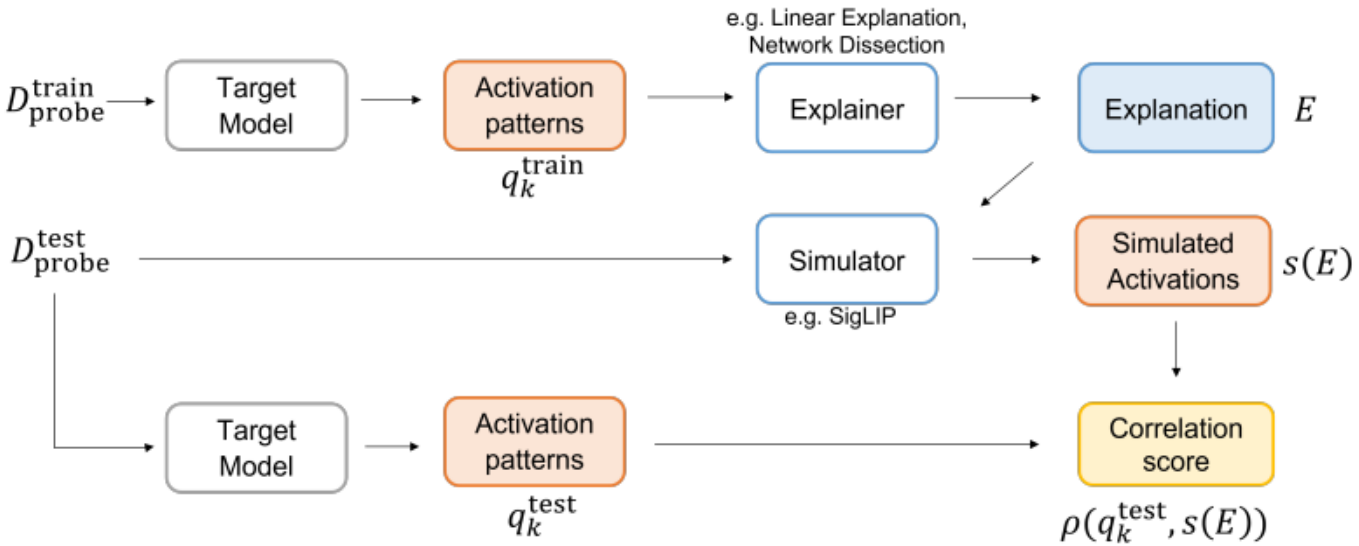
❸ **Learn sparse layer w_k to predict neuron activations**

$$\min_{w_k} \|q_k - P^\top w_k\|^2$$

[10] Oikarinen & Weng, Linear Explanations for Individual Neurons, ICML 2024

Group 4: Poly-semantic, No Label-Data

- Linear Explanation [10]
  - 1st Simulation pipeline in Vision setting to evaluate neuron description (explanation) quality
  - Importantly, explanation generation and explanation testing should use different set of probing data



Target model	Network Dissection	MILAN	CLIP-Dissect	LE (Label)	LE (SigLIP)
ResNet-50 (ImageNet)	0.1242 ± 0.002	0.0920 ± 0.002	0.1871 ± 0.002	0.2924 ± 0.002	$\mathbf{0.3772 \pm 0.002}$
ResNet-18 (Places365)	0.2038 ± 0.005	0.1557 ± 0.005	0.2208 ± 0.005	0.3388 ± 0.004	$\mathbf{0.4372 \pm 0.003}$
VGG-16 (CIFAR-100)	n/a	n/a	0.2298 ± 0.004	0.4330 ± 0.004	$\mathbf{0.4970 \pm 0.004}$
ViT-B/16 (ImageNet)	n/a	n/a	0.1722 ± 0.004	0.3243 ± 0.005	$\mathbf{0.3489 \pm 0.005}$
ViT-L/32 (ImageNet)	n/a	n/a	0.0549 ± 0.002	0.1879 ± 0.004	$\mathbf{0.2182 \pm 0.004}$

[10] Oikarinen & Weng, Linear Explanations for Individual Neurons, ICML 2024

Reference for Part 2-1

[1] Bau etal, Quantifying interpretability of Deep Visual Representations, CVPR 2017

[2] Hernandez etal, Natural Language Descriptions of Deep Visual Features, ICLR 2022

[3] Oikarinen & Weng, CLIP-Dissect: Automatic Description of Neuron Representations in Deep Vision Networks, ICLR 2023

[4] Srinivas, SAND: Enhancing Open-Set Neuron Descriptions through Spatial Awareness, WACV 2025

[5] Bai etal, Interpreting Neurons in Deep Vision Networks with Language Models, TMLR 2025

[6] Shaham etal, A Multimodal Automated Interpretability Agent, ICML 2024

[7] Mu & Andreas, Compositional Explanations of Neurons, NeurIPS 2020

[8] La Rosa, Towards a fuller understanding of neurons with clustered compositional explanations, NeurIPS 2023

[9] Bykov etal, Labeling Neural Representations with Inverse Recognition, NeurIPS 2023

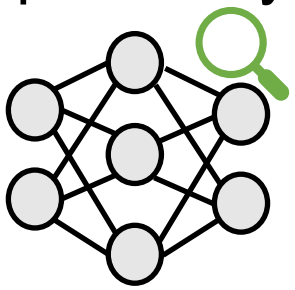
[10] Oikarinen & Weng, Linear Explanations for Individual Neurons, ICML 2024

Tutorial Part 2: DEMO TIME

Akshay

Tuomas

Post-hoc Interpretability tools



What knowledge has the model learned?

⇒ We will discuss methods from small unit to large unit

1. Understand individual neurons (**neuron-level** interpretability tools)

Lily

2. Understand groups of neurons / full layer (**representation-level** interpretability tools)

Ge

3. Understand multiple layers / circuits (**circuit-level** interpretability tools)

Akshay

https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

Unified Neuron Explanations: GitHub repo

Unified Neuron Explainer

A single framework that runs four neuron explanation methods against shared infrastructure (activation collection, CLIP features, dataset loading, CSV output).

Method	Paper	Concept source	Output
CLIP-Dissect	Oikarinen et al., ICLR 2023	Fixed text file + CLIP similarity	top-1 concept per neuron
Linear Explanations	Oikarinen & Weng, ICML 2024	Fixed text file + SigLIP/label regression	sparse linear combination per neuron
Describe-and-Dissect	Bai et al., TMLR 2025	BLIP captions + Llama 3 + SD validation	top-3 generated concepts per neuron
NetDissect	Bau et al., CVPR 2017	Broden pixel labels + IoU	top-1 Broden concept per neuron

Unified Neuron Explanations: GitHub repo

```
neuron_explainer/
├── io/
│   ├── datasets.py
│   └── output.py
├── models/
│   └── target_model.py
├── concepts/
│   ├── naming/from_file.py
│   ├── naming/generative.py
│   └── grounding/clip_grounding.py
├── activations/
│   └── collector.py
├── scoring/
│   ├── clip_match.py
│   ├── linear_sparse.py
│   └── generative_score.py
└── methods/
    ├── base.py
    ├── clip_dissect.py
    ├── linear_explanations.py
    ├── describe_and_dissect.py
    └── net_dissect.py
```

Input/Output:
Loading probing datasets, saving results

Target Model: Architecture, loading pretrained checkpoint

Concepts:
Loading concept sets,
Setting up concept supervision (e.g. ground-truth or CLIP, SigLIP)

Activations: Collecting target model activations

Scoring:
Assigning concepts to individual neurons
(e.g. comparing to CLIP, training sparse linear regressor, etc.)

Existing Methods:
Combining modular framework components to reproduce existing methods
(e.g. NetDissect, CLIP-Dissect, Linear Explanations, Describe-and-Dissect)

Demo 1: CLIP-Dissect

Demo Notebook: demo_clip_dissect.ipynb

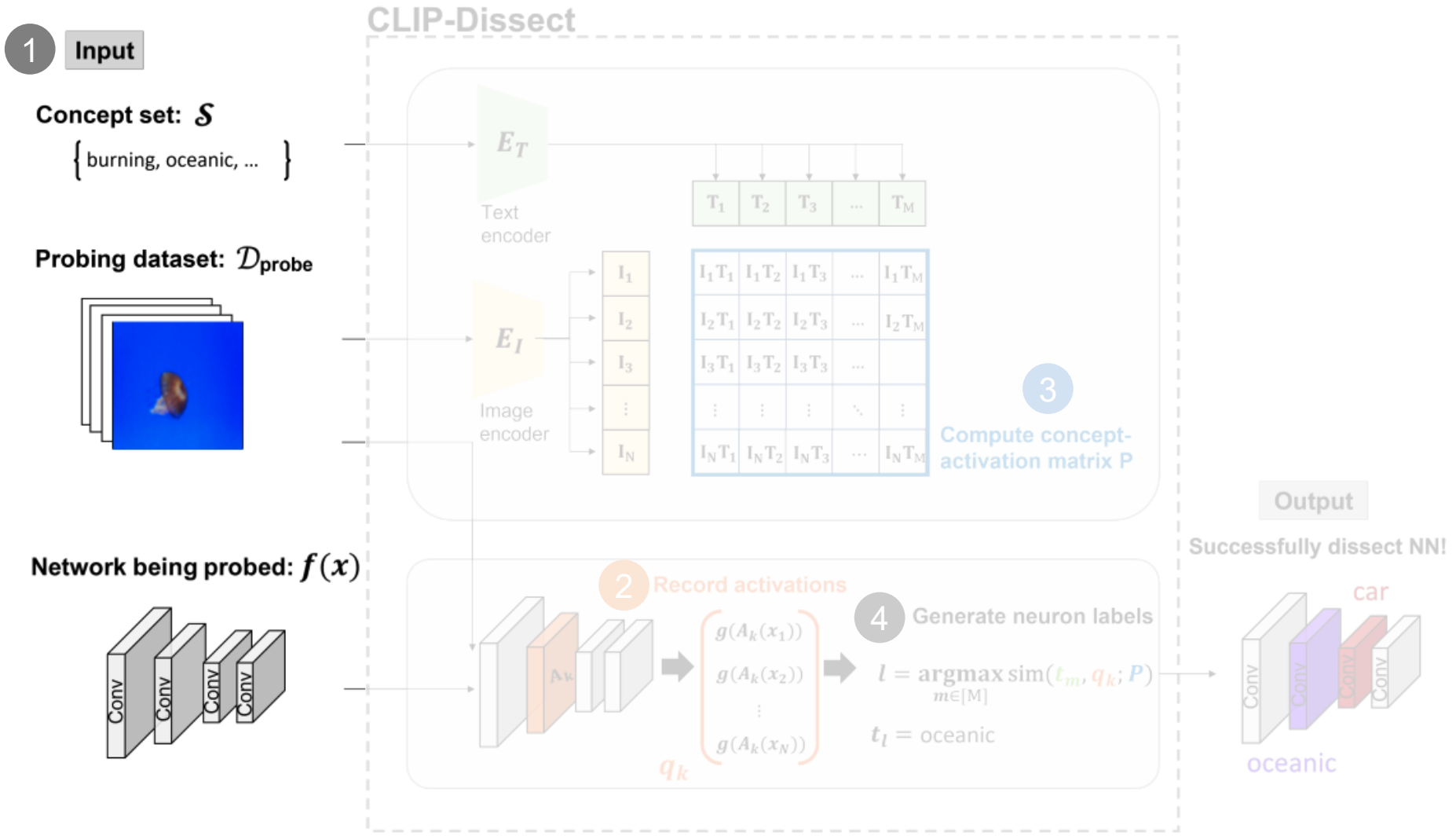
Code will be released on tutorial website

https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

Demo: CLIP-Dissect

- Method
  1. Load target model, probing dataset and concept set
  2. Collect target model's neuron activations for probing dataset
  3. CLIP grounding: collect image & text activations for probing data & concept set
  4. Score neurons against concepts

Step 1: Load model, probing data, concept set



Step 1: Load model, probing data, concept set

```python
# Load target model (auto-downloads weights the first time)
print(f"Loading target model: {MODEL_NAME} ...")
target_model, target_preprocess = load_target_model(MODEL_NAME, device)
print(f"  Parameters: {sum(p.numel() for p in target_model.parameters()):,}")
```

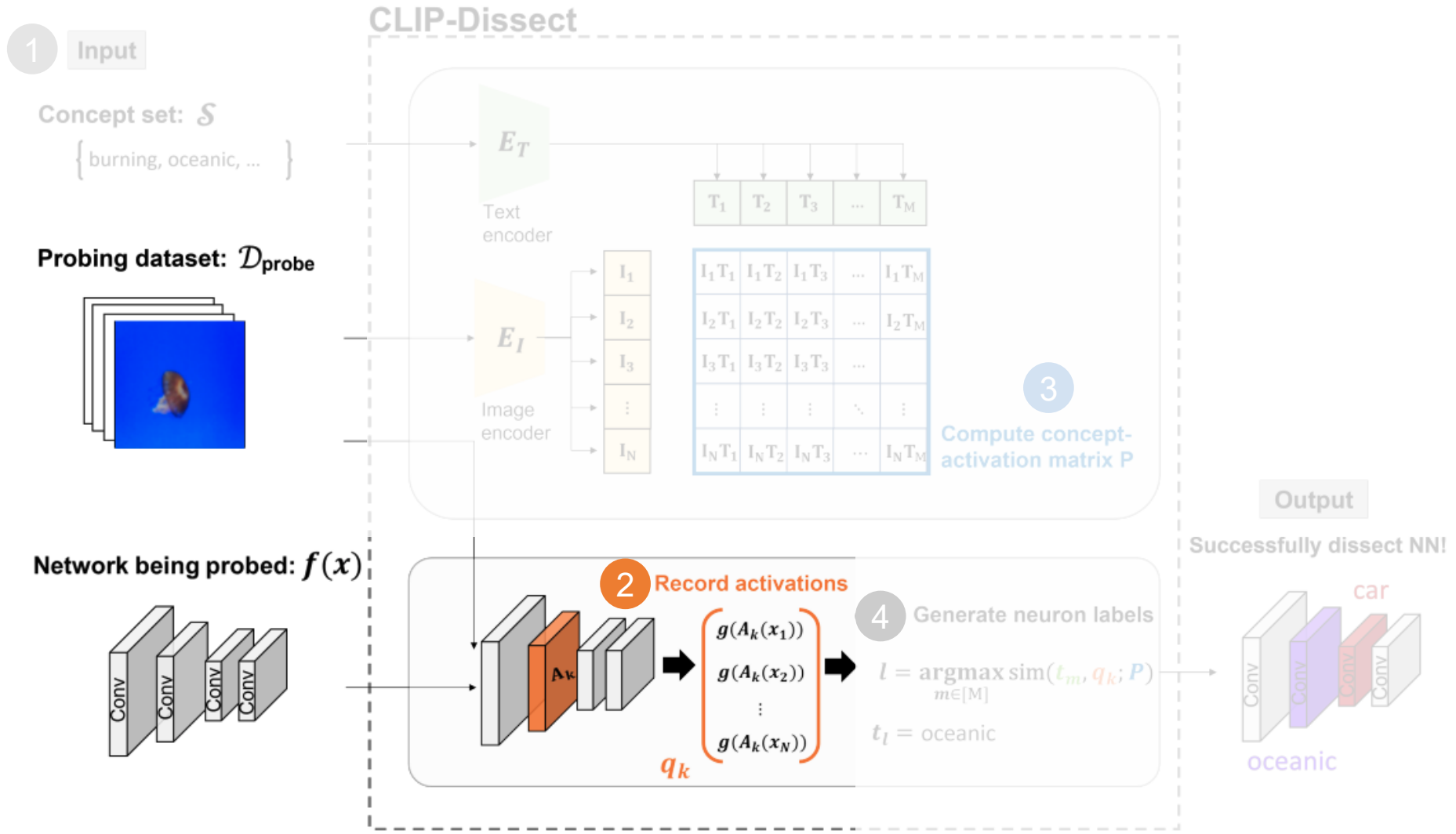
```
Loading target model: resnet56_cifar100 ...
Using cache found in C:\Users\tuoma/.cache\torch\hub\chenyaofo_pytorch-cifar-models_master
  Parameters: 861,620
```

```python
# Dataset preprocessed for the TARGET MODEL (used for activation collection)
probe_target = get_dataset(DATASET_NAME, transform=target_preprocess)

# Dataset preprocessed for CLIP (used for CLIP image features)
probe_clip = get_dataset(DATASET_NAME, transform=clip_grounder.preprocess)
```

```python
# Load concept vocabulary
concepts = load_concept_set(CONCEPT_SET_PATH)
print(f"\nConcept vocabulary: {len(concepts):,} concepts  ({CONCEPT_SET_PATH})")
print(f"  Sample: {concepts[:8]}")
```

```
Concept vocabulary: 20,000 concepts  (data/concept_sets/20k.txt)
  Sample: ['the', 'of', 'and', 'to', 'a', 'in', 'for', 'is']
```

Step 2: Collect target model neuron activations



Step 2: Collect target model neuron activations

```
act_paths = collect_activations(
    model         = target_model,
    dataset       = probe_target,
    target_layers= TARGET_LAYERS,
    batch_size    = BATCH_SIZE,
    device        = device,
    pool_mode     = POOL_MODE,
    save_dir      = SAVE_DIR,
    dataset_name = DATASET_NAME,
    model_name    = MODEL_NAME,
    num_workers   = NUM_WORKERS,
)
print("\nCached activation files:")
for layer, path in act_paths.items():
    size_mb = os.path.getsize(path) / 1e6
    print(f"  {layer:20s}  {path}  ({size_mb:.1f} MB)")
```

Specify target model (e.g. resnet56_cifar100)

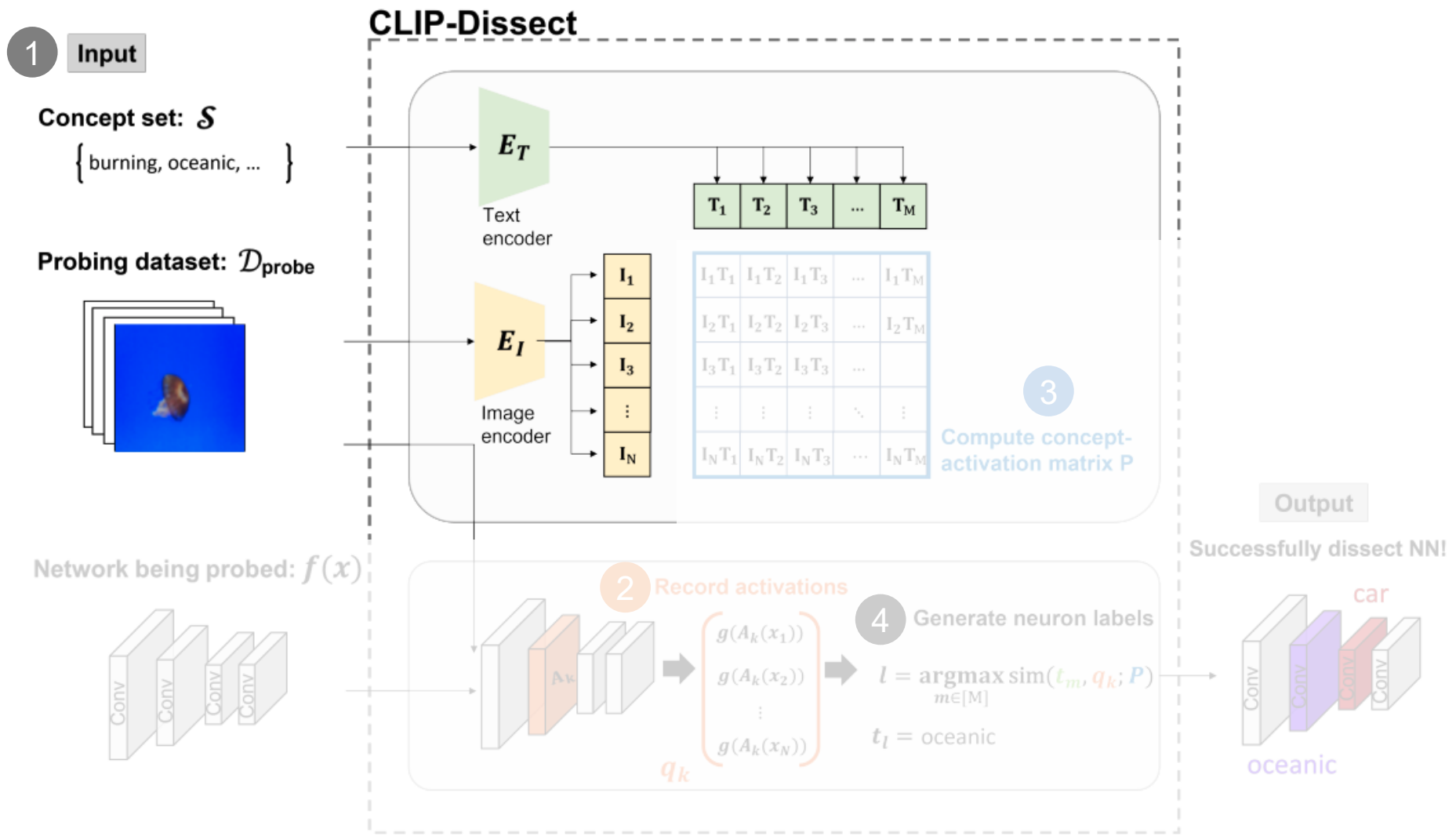
Specify probing dataset (e.g. places365_val)

Specify target model layers (e.g. layer2, layer3)

```
Collecting activations [layer2, layer3]: 100%|██████████| 286/286 [00:53<00:00,  5.36it/s]
Cached activation files:
  layer2                saved_activations\places365_val_resnet56_cifar100_layer2.pt  (4.7 MB)
  layer3                saved_activations\places365_val_resnet56_cifar100_layer3.pt  (9.3 MB)
```

Saves activations to disk (to reuse with other concept sets or other methods)

Step 3: CLIP Grounding



Step 3: CLIP Grounding

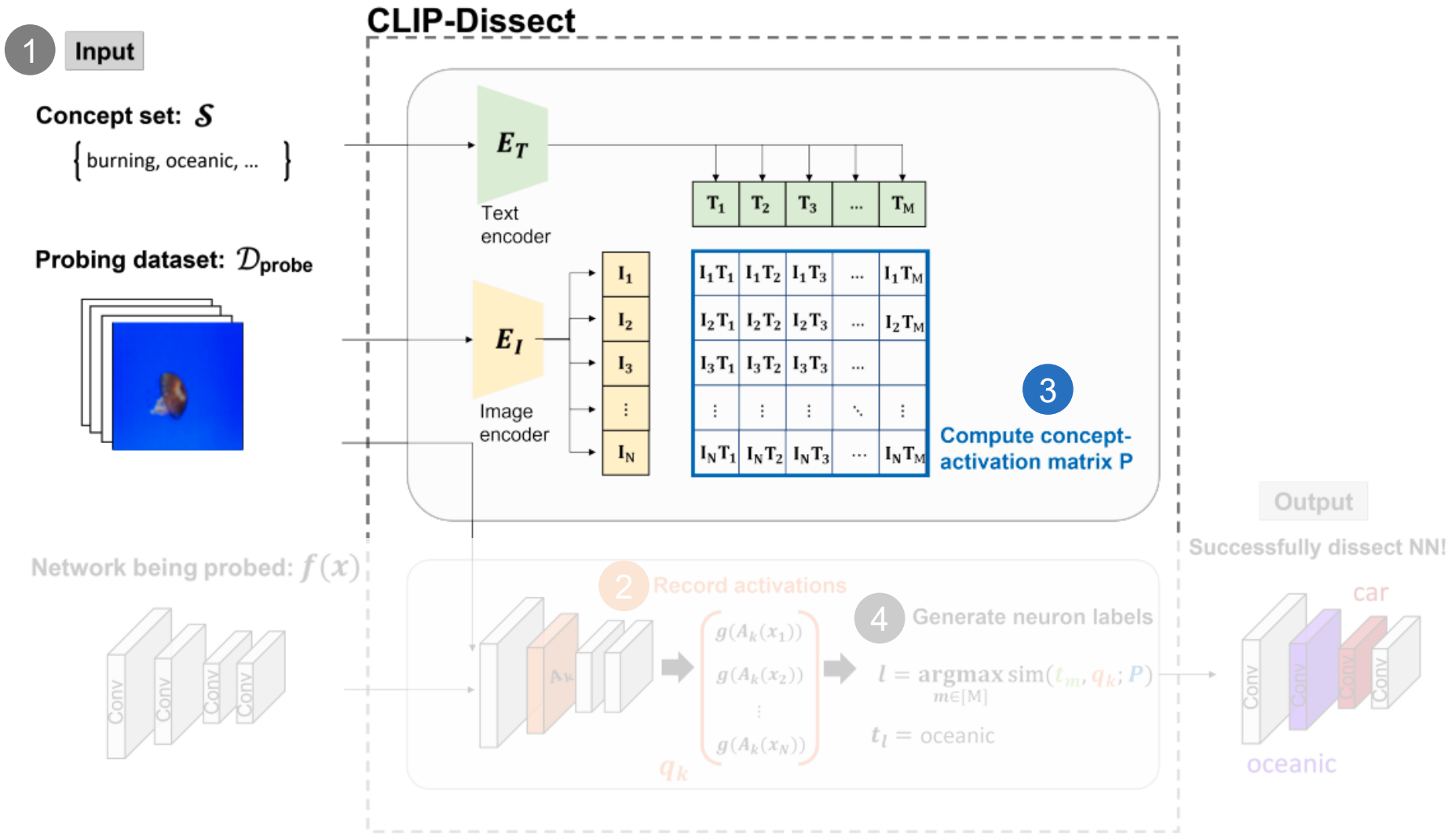
```python
# Load CLIP for grounding (downloads the first time)
print(f"\nLoading CLIP grounder: {CLIP_MODEL_NAME} ...")
clip_grounder = CLIPGrounding(CLIP_MODEL_NAME, device)
```

```
Loading CLIP grounder: ViT-B/16 ...
```

```python
print("Computing CLIP image features ...")
image_features = clip_grounder.get_image_features(
    probe_clip, batch_size=BATCH_SIZE, save_path=img_feat_path, num_workers=NUM_WORKERS
)  # [N_images, D]

print("Computing CLIP text features ...")
text_features = clip_grounder.get_text_features(
    concepts, save_path=text_feat_path
)  # [N_concepts, D]

print(f"\nImage features: {tuple(image_features.shape)}  (normalised to unit norm)")
print(f"Text features:  {tuple(text_features.shape)}")
```

```
Computing CLIP image features ...
CLIP image features: 100%|██████████| 286/286 [02:39<00:00,  1.79it/s]
Computing CLIP text features ...

Image features: (36500, 512)  (normalised to unit norm)
Text features:  (20000, 512)
```

Step 3: CLIP Grounding

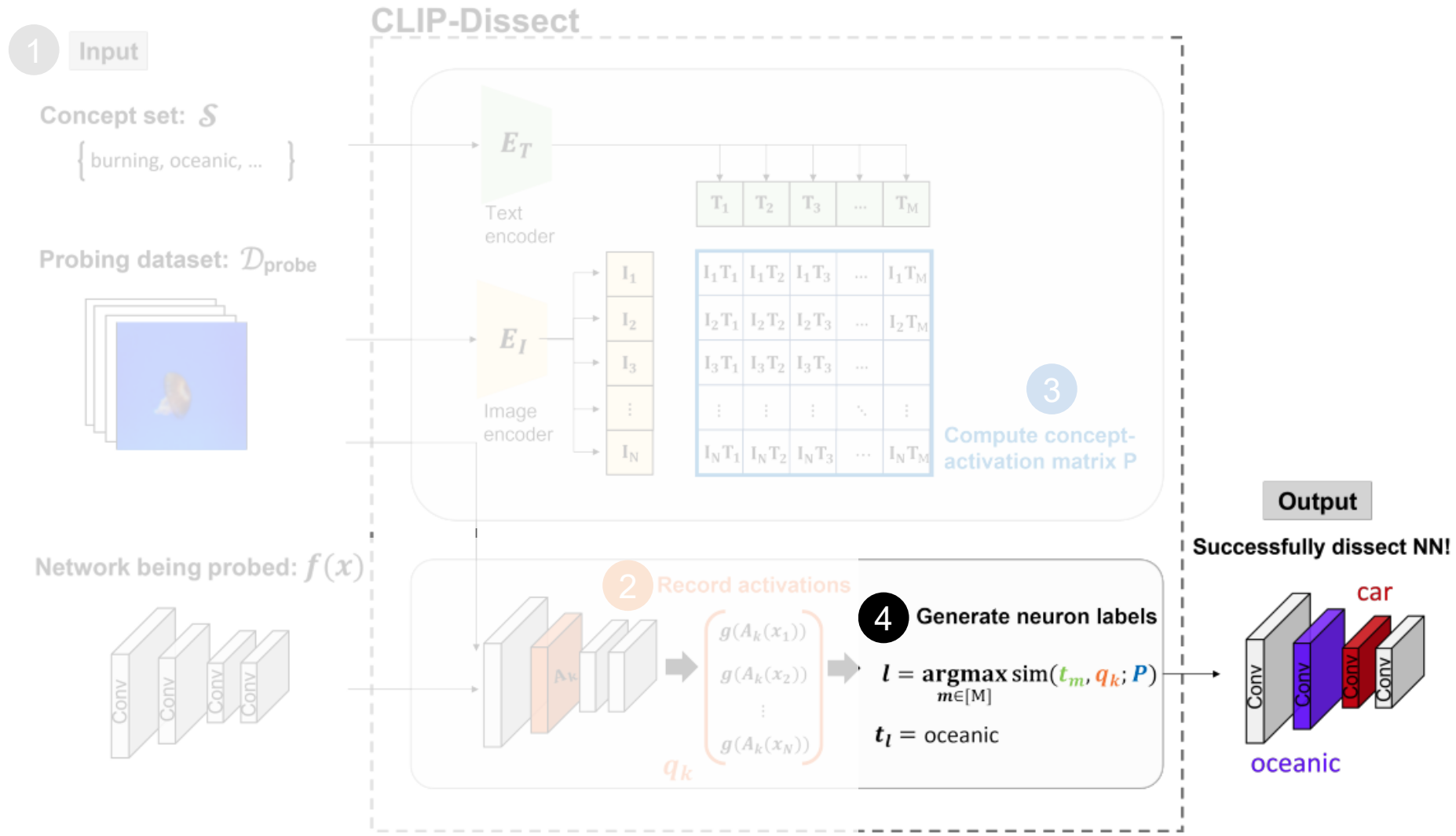


Step 3: CLIP Grounding

```python
# Build P matrix: image_features @ text_features.T  →  [N images, N concepts]
P = clip_grounder.get_concept_activation_matrix(image_features, text_features).cpu()
print(f"P matrix shape: {tuple(P.shape)}")
print(f"Value range: [{P.min():.3f}, {P.max():.3f}]  (cosine similarity)")
```

```
P matrix shape: (36500, 20000)
Value range: [-0.044, 0.365]  (cosine similarity)
```

Step 4: Score neurons against concepts



Step 4: Score neurons against concepts

```python
for layer in TARGET_LAYERS:
    target_feats = load_activations(act_paths[layer])  # [N_images, N_units]
    print(f"Scoring {target_feats.shape[1]:,} units in '{layer}' ...")

    similarities = sim_fn(P, target_feats, device=device)  # [N_units, N_concepts]

    scores, concept_ids = torch.max(similarities, dim=1)

    for unit_idx, (score, cid) in enumerate(zip(scores.cpu(), concept_ids.cpu())):
        all_rows.append({
            "layer":   layer,
            "unit":    unit_idx,
            "concept": concepts[int(cid)],
            "score":   float(score),
        })
```

```
layer,unit,concept,score
layer3,0,stripe,0.141204833984375
layer3,1,dotted,0.20458984375
layer3,2,shelving,0.195983886718750
layer3,3,wicker,0.2760009765625
layer3,4,fruits,0.15118408203125
layer3,5,stalls,0.14532470703125
layer3,6,dotted,0.114501953125
layer3,7,lattice,0.172821044921875
```

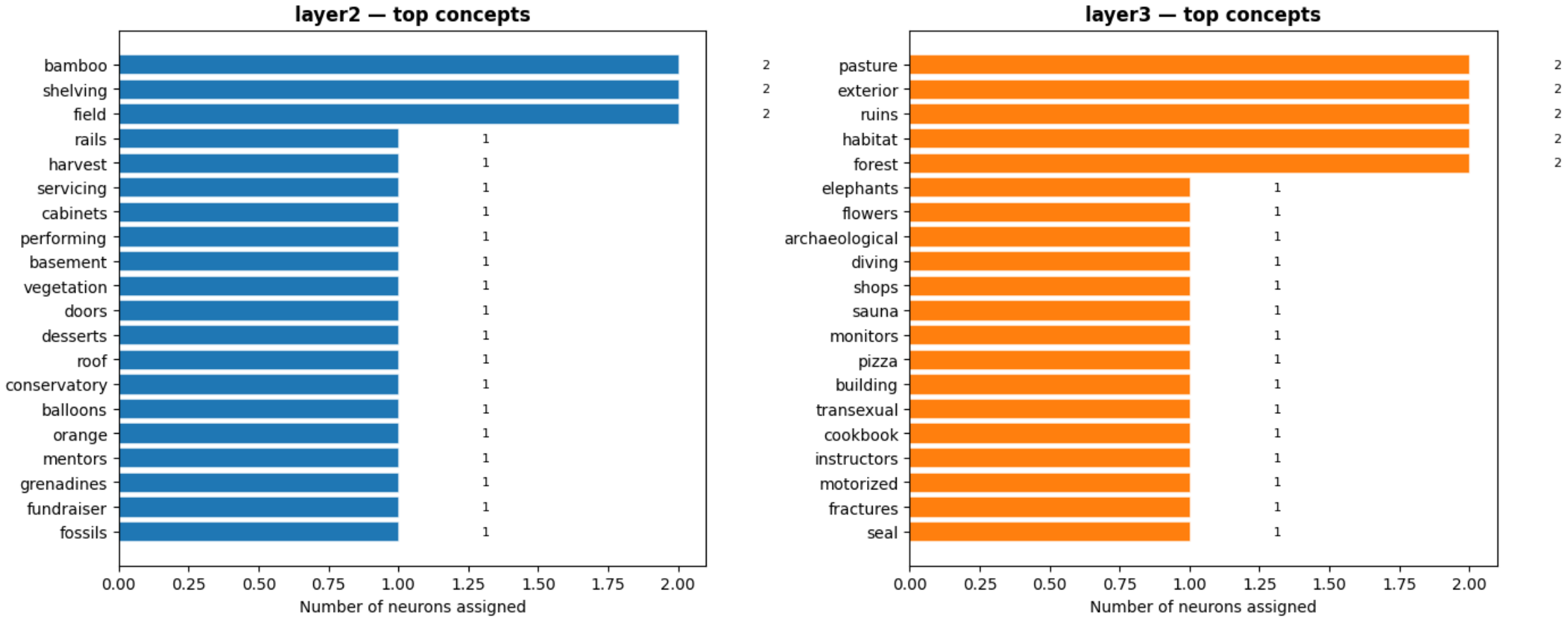
Results are saved in csv format

Qualitative Results

- e.g. Visualize top-3 neurons by CLIP-Dissect confidence

Qualitative Results

- e.g. Most common concepts per layer



Demo 2: Unified Neuron Explanations

Demo Notebook: demo_explanation.ipynb

Code will be released on tutorial website

https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

Run multiple neuron explanation methods

- With our unified repo, you can run 4 existing explanation methods:
  - CLIP-Dissect, Linear Explanations, Describe-and-Dissect, NetDissect

```
from neuron_explainer.methods.clip_dissect import CLIPDissect
from neuron_explainer.methods.describe_and_dissect import DescribeAndDissect
from neuron_explainer.methods.linear_explanations import LinearExplanations
from neuron_explainer.methods.net_dissect import NetDissect
```

```
results_by_method['linear_explanations'] = LinearExplanations().run(
    target_model_name=MODEL_NAME,
    target_layer=TARGET_LAYER,
    dataset_name=PROBE_DATASET,
    mode=METHOD_CONFIGS['linear_explanations']['mode'],
    concept_set_path=METHOD_CONFIGS['linear_explanations']['concept_set_path'],
    clip_model_name=METHOD_CONFIGS['linear_explanations']['clip_model_name'],
    batch_size=BATCH_SIZE,
    device=DEVICE,
    pool_mode=METHOD_CONFIGS['linear_explanations']['pool_mode'],
    unit_ids=sampled_units,
    max_length=METHOD_CONFIGS['linear_explanations']['max_length'],
    tolerance=METHOD_CONFIGS['linear_explanations']['tolerance'],
    glm_neuron_batch=METHOD_CONFIGS['linear_explanations']['glm_neuron_batch'],
    glm_lambda=METHOD_CONFIGS['linear_explanations']['glm_lambda'],
    save_dir=SAVE_DIR,
    result_dir=str(RESULT_DIR / 'linear_explanations'),
    result_filename='results.csv',
)
```

```
results_by_method['net_dissect'] = NetDissect().run(
    model=netdissect_model,
    layer=TARGET_LAYER,
    data_directory=BRODEN_DATA_DIRECTORY,
    categories=METHOD_CONFIGS['net_dissect']['categories'],
    batch_size=BATCH_SIZE,
    tally_batch_size=METHOD_CONFIGS['net_dissect']['tally_batch_size'],
    tally_ahead=METHOD_CONFIGS['net_dissect']['tally_ahead'],
    quantile=METHOD_CONFIGS['net_dissect']['quantile'],
    output_dir=str(RESULT_DIR / 'net_dissect'),
    result_filename='results.csv',
)
```

Compare multiple neuron explanation methods

- Qualitative examples (ImageNet-pretrained ResNet50)

unit	CLIP-Dissect	Linear Explanations	Describe-and-Dissect
153	control tower, indoor (score=0.620)	3.74*fixture + 3.07*tandem + 1.43*infrastructure + 0.94*food + 0.07 (corr=0.446)	elevated viewpoints | elevated views of spaces | industrial or corporate architecture
197	minibike (score=0.782)	2.96*poisoning + 2.63*wheeled vehicle + 2.30*tam-o'-shanter + 1.66*bike + 1.54*apse + 0.12 (corr=0.50	vehicles with riders | vehicles in various states | various vehicle compositions
237	ram (adult male sheep) (score=0.457)	6.33*forager + 5.14*garb + 4.59*labor + 3.98*grandparent + 1.83*chef + 0.06 (corr=0.580)	rural themed pastoral scenes | formal procession scenes | agricultural and ceremonial processions
296	jacuzzi, indoor (score=1.356)	3.40*bath + 2.92*work surface + 1.46*cooking + 0.06 (corr=0.688)	people in bathtubs | bathing and relaxation scenes | people in bathing situations
385	swimming pool, indoor (score=0.386)	3.52*whirlpool + 2.01*equipment + 1.87*food + 1.75*recreation + 1.34*interior + 0.03 (corr=0.598)	spring and coil designs background | spring and coil structures | blue plastic springs
617	leaf (score=1.013)	10.99*savory + 9.73*macrofauna + 3.83*veined + 0.15 (corr=0.495)	leaf insect habitats | insect on a leaf | nature with leaves and insects
879	green color (score=0.564)	8.89*snorkel + 4.19*emerald + 2.36*recreation + 2.06*fiberglass + 0.99*bath + 0.14 (corr=0.452)	various shades of green | various green items | variations of the color green
1326	carousel (score=0.758)	3.76*swing + 3.69*carousel + 2.02*veranda + 1.67*four-poster bed + 0.76*dining + 0.07 (corr=0.488)	carousel scenes with animals | carousels and animals | carousel scenes
1497	utility room (score=1.379)	5.83*bath + 3.52*interior + 3.11*kitchen + 0.14 (corr=0.710)	bathroom fixtures and decor | bathroom settings
1617	pool table (score=1.299)	4.42*bongo + 2.53*furniture + 1.78*poolroom, home + 1.14*cook + 0.12 (corr=0.528)	billiard game environments | billiard related settings | billiard and game spaces

CLIP-Dissect: single concept per neuron

Linear Explanations: Linear combination of concepts per neuron

Describe-and-Dissect: Concepts generated by image captioning model + LLM

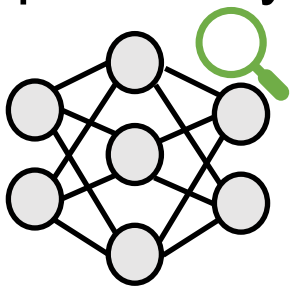
Compare multiple neuron explanation methods

- Qualitative examples (ImageNet-pretrained ResNet50)

Neuron ID	Concept	Top-5 activating images
1326	**CLIP-Dissect**: “carousel” **Linear Expl**: “3.76 * swing + 3.69 * carousel + …” **Describe-n-Dissect**: “carousels and animals”	
1617	**CLIP-Dissect**: “pool table” **Linear Expl**: “4.42 * bongo + 2.53 * furniture + …” **Describe-n-Dissect**: “billiard game environments”	

Tutorial Part 2: Overview

Post-hoc Interpretability tools



What knowledge has the model learned?

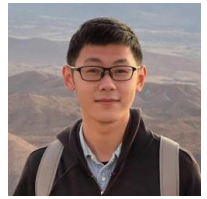
⇒ We will discuss methods from small unit to large unit

1. Understand individual neurons (**neuron-level** interpretability tools)

Lily

2. Understand groups of neurons / full layer (**representation-level** interpretability tools)

Ge



3. Understand multiple layers / circuits (**circuit-level** interpretability tools)

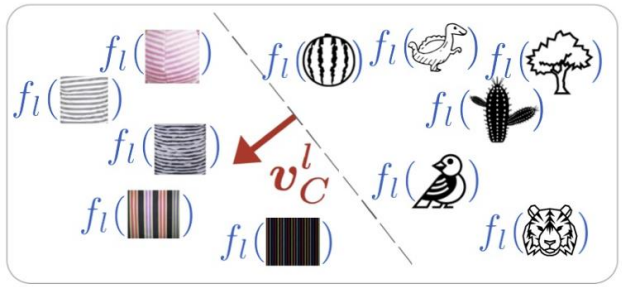
Akshay

https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

Medium units

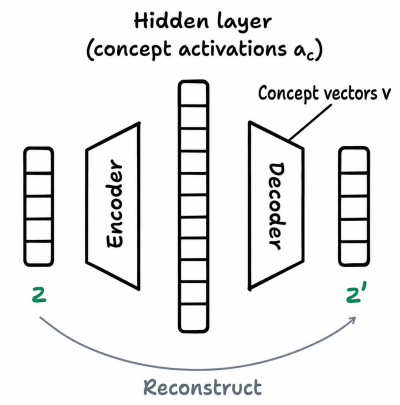
- We focus on a larger unit than single neuron: representation vector z
  - Specifically, we will look at model internal representation vector $z \in \mathbb{R}^d$, including residual stream, attention head output, and full MLP layer.
  - There are 3 main methods to understand a representation vectors:

1



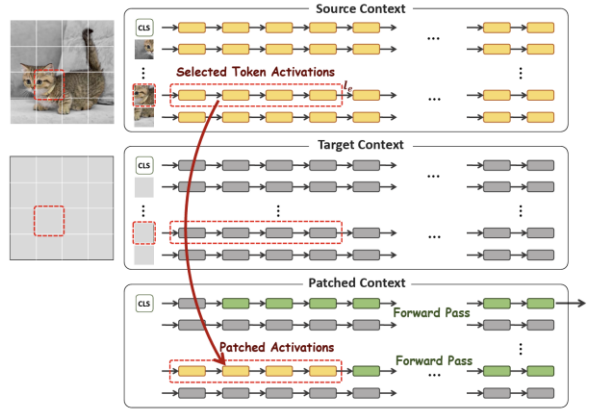
TCAV

2



Sparse Autoencoder

3

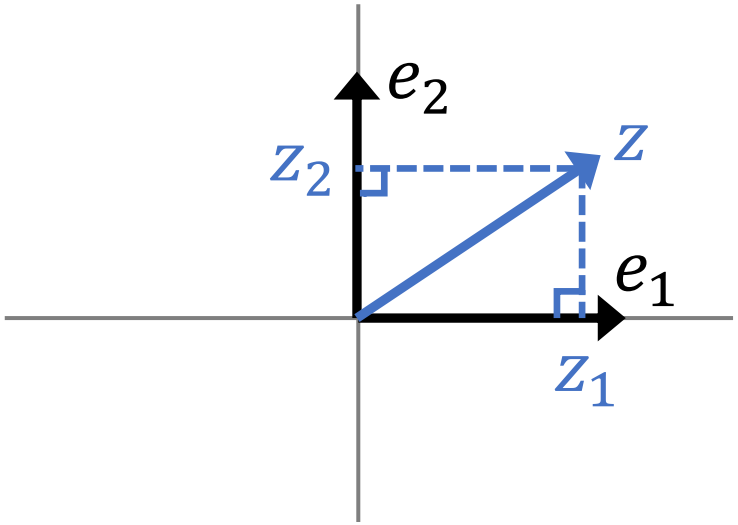


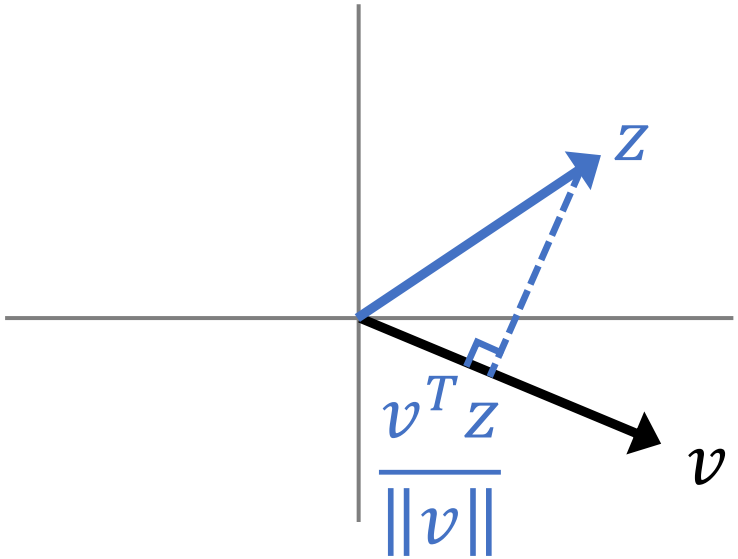
Activation patching

Linear direction

- **Hypothesis:** Concepts are represented by linear directions v in representation space (Z)
- Neurons can be regarded as **special directions** in representation space, with elementary vectors ($v = e_k$):

$z_k = e_k^\top z$, $e_k = (0, \ldots, 1, 0, \ldots, 0)$ is the k-th standard basis vector.





Linear direction

Can we extend interpretability study to more general direction v in the representation space?

Two questions:

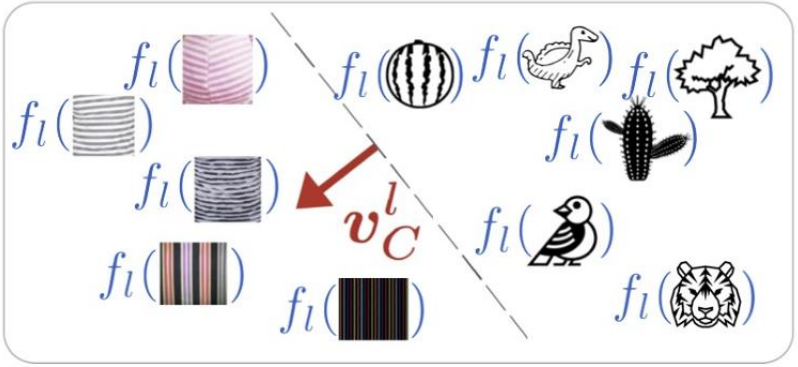
1. *How do we find direction of interest, from infinity number of directions in the representation space?*
   - **Supervised:** TCAV, find the direction corresponding to predefined concept.
   - **Unsupervised:** SAE, automatically discover important directions.
2. *How can we apply these directions?*
   - **Model steering (intervention):** Steer model output by injecting the direction.

Beyond single direction: Activation patching.

Method 1: TCAV

TCAV [1] learns Concept Activation Vector (CAV) as *linear direction* in representation space.

- Collect positive/negative samples for target concept.
- Train a binary linear classifier, v_C^l as a linear CAV for concept *C* at layer *l.*
  - The learned weight of linear classifier is v_C^l
- Limitation: Require concept labels



Concept vector for “stripe”

[1] Kim, et al. "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)." *ICML*, 2018.

Method 2: Sparse Autoencoder

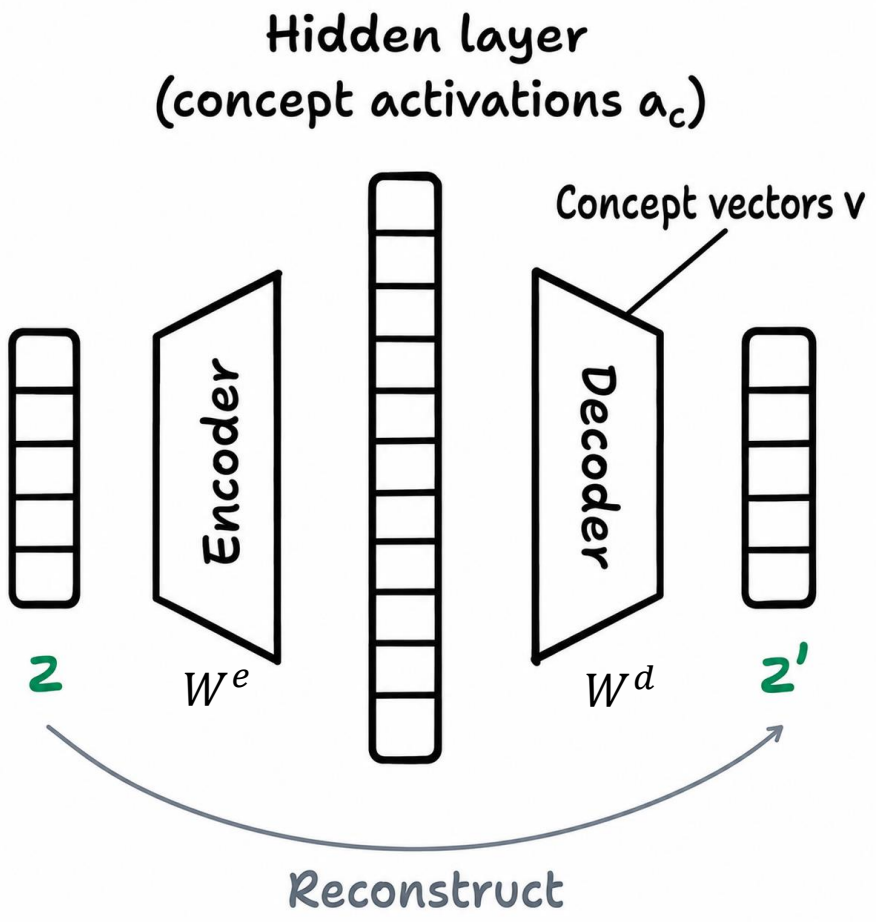
Sparse Autoencoder discovers concepts automatically from representation.

- **Encoder:** Map representation to a sparse latent a_c (concept)

$$a_i = \sigma(W^e_{\{i,:\}} z)$$

- **Decoder:** Reconstruct representation from latent (concept)

$$z' = \sum_i a_i W^d_{\{i,:\}}$$



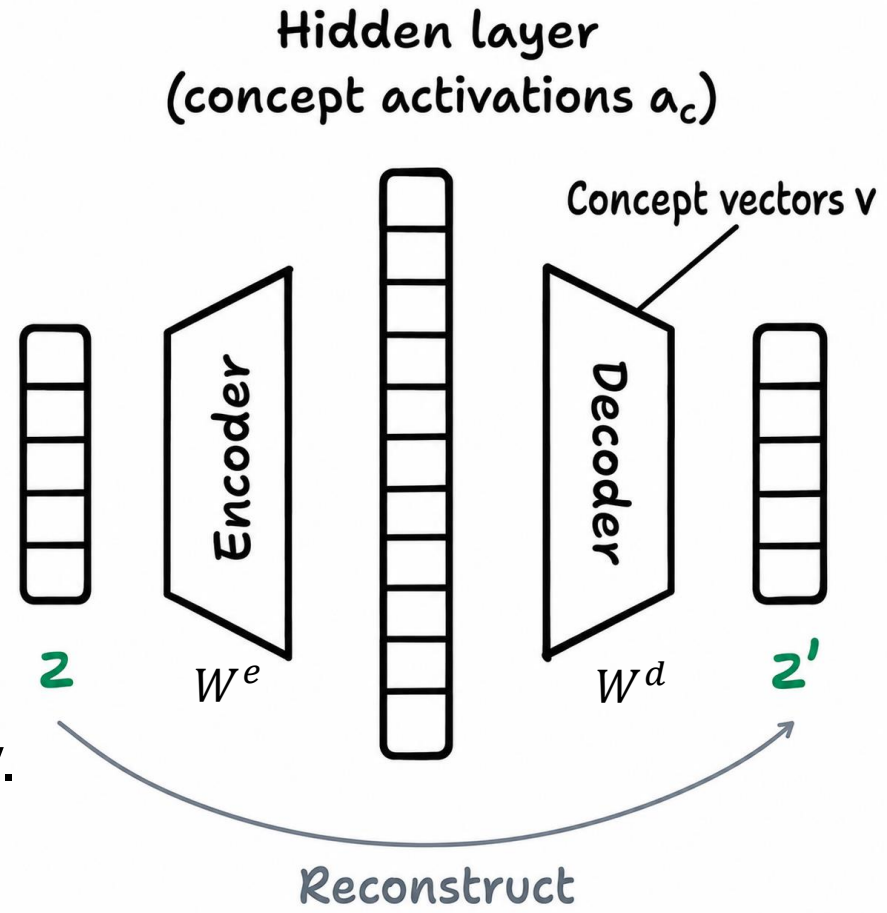
[2] Pach, et al. "Sparse Autoencoders Learn Monosemantic Features in Vision-Language Models" NeurIPS 2025

Method 2: Sparse Autoencoder

SAE is trained to recover the original representation from a sparse latent.

$$\min \|z' - z\|_2^2 + \lambda \sum_i |a_i|$$

Reconstruction loss L_1 loss for sparsity



- Original SAE uses ReLU activation to create sparsity.
- Modern SAE introduces more activations: JumpReLU, TopK, BatchTopK…

Method 2: Sparse Autoencoder

Example SAE neurons [2] from CLIP ViT-L/14

[2] Pach, et al. "Sparse Autoencoders Learn Monosemantic Features in Vision-Language Models" NeurIPS 2025

Method 2: Interpreting SAE neurons

SAE does not tell us the concept correspond to each latent neuron.

- **Manual inspection:** Check top-activate images.
- **DNCBM [3]:** Use CLIP model and compare decoder weight v_c with concept's CLIP text embedding.

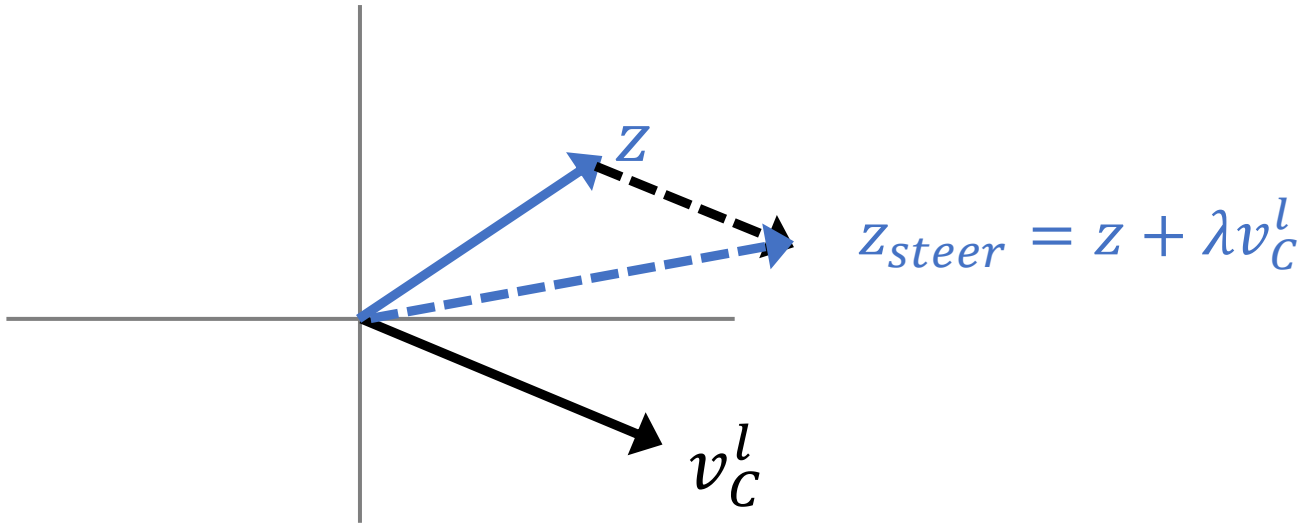
More generally, we can treat SAE latent as individual neurons, and use all methods we introduce in small-unit part. (e.g. CLIP-Dissect [4]).

[3] Rao, et al. "Discover-then-name: Task-agnostic concept bottlenecks via automated concept discovery." *ECCV 2024*
[4] Oikarinen and Weng, "CLIP-Dissect: Automatic Description of Neuron Representations in Deep Vision Networks," ICLR 2023

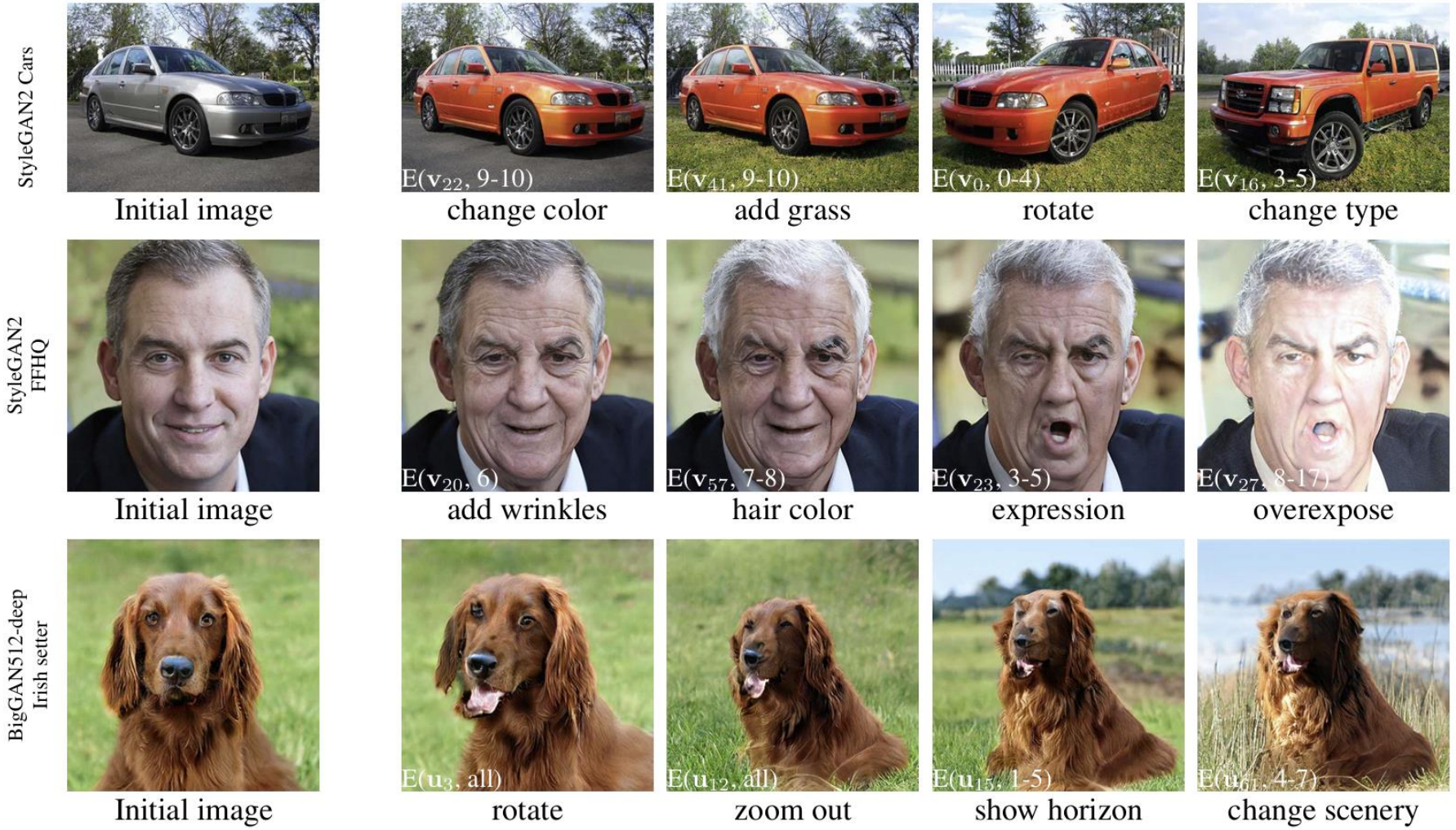
Application: Model Steering

- After extracting concept vectors v_c from SAE/TCAV/… , it can be applied to steer the output of the model.
- By injecting the vector v_C^l into specific layers, generated image can be steered.

$$z_{steer} = z + \lambda v_C^l$$



Application: Model Steering



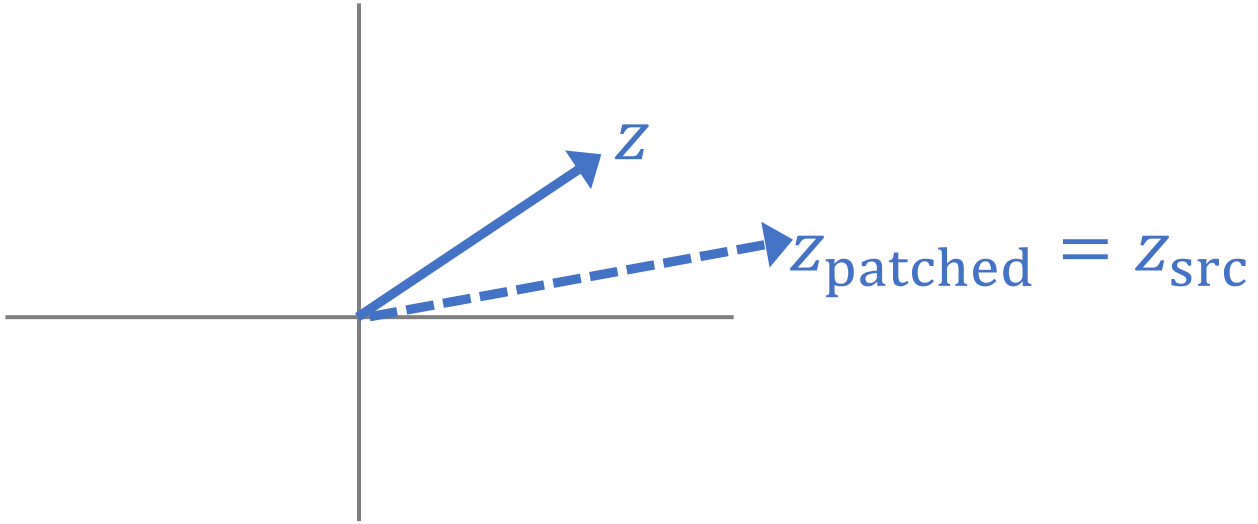
[5] Härkönen, et al. "Ganspace: Discovering interpretable gan controls" NeurIPS 2020

Method 3: Activation patching

- Activation patching replace representation in target input with that from source input.

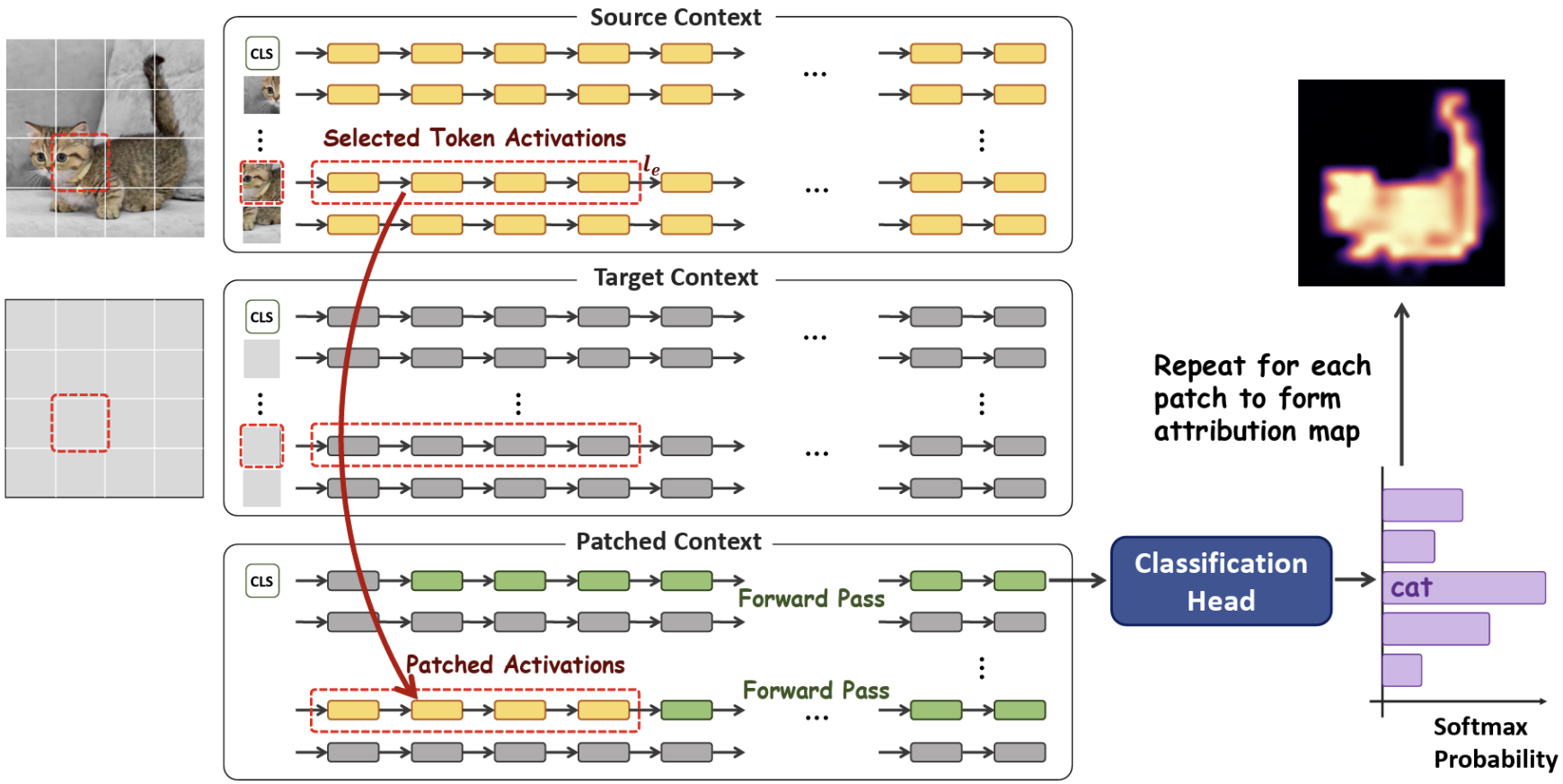
$$z_{\text{patched}} = z_{\text{src}}$$

- Allow evaluating the impact of individual token (image patch) to final prediction.



[6] Izadi, et al. "Causal Attribution via Activation Patching." *arXiv* 2026.

Method 3: Activation patching



Attributing contribution of individual token (image patch) via activation patching.

[6] Izadi, et al. "Causal Attribution via Activation Patching." *arXiv* 2026.

Connections to Language models

These introduced methods are also widely applied in language models:

- SAE [7] and activation patching (ROME [8]) are also applied to analyze LLMs.
- TCAV's idea is also applied for LLM, more known as linear probing [9]

[7] Bricken et al. "Towards Monosemanticity: Decomposing Language Models With Dictionary Learning," Transformer Circuits / Anthropic, 2023.
[8] Meng et al. "Locating and Editing Factual Associations in GPT," NeurIPS, 2022.
[9] Conneau et al. "What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties," ACL, 2018.

Special representation space

Besides general methods, there are some special representation spaces allowing specific methods on them:

1. **CLIP image embedding space:** This space is semantically aligned with CLIP text-embeddings, allowing interpretation with text.
2. **Attention head outputs:** Besides output representation z, attention head also provide attention map, attributes contribution to each token.

Special representation space #1: SpLiCE

- SpLiCE studies the image embedding of CLIP model. This is a special representation space aligned with CLIP text embedding.

- Utilizing alignment with text embedding, SpLiCE decomposes z into linear combination of text embeddings.

$$z = \sum_i c_i v_i$$

CLIP text embedding of i-th concept

CLIP image embedding

i-th concept activation, sparse

[10] Bhalla, et al. "Interpreting clip with sparse linear concept embeddings (splice)." *NeurIPS 2024*

Special representation space #1: SpLiCE

wedding sign	0.06
cow	0.04
shop logo	0.04
sweets	0.04
sweetened	0.03
swing	0.02

"A statue of a cow sitting under a sign on grass."

powder	0.08
snowboard	0.07
skiing	0.07
slopes	0.06
winter sports	0.04
ski resort	0.03

"A field that has some snow and a man skiing in it"

cat	0.05
shower	0.05
tabby cat	0.03
scratching	0.03
shampoo	0.03
sink	0.02

"A cat is looking at a bathtub in a bathroom with a toilet."

vitamin c	0.10
oranges	0.08
fresh orange	0.06
depression	0.04
unhappy	0.04
food background	0.03
macro shot	0.03

"A knife stabbing an orange with a sad face."

red umbrella	0.07
umbrella	0.06
woman walks	0.03
businessman	0.03
autumn winter	0.01
orange background	0.01
paved	0.01

"Adult with orange umbrella walks on bricked open area."

wild horses	0.10
blackandwhite	0.08
bw	
mare	0.07
beach sand	0.03
horse	0.02
resting	0.01
kissing	0.01

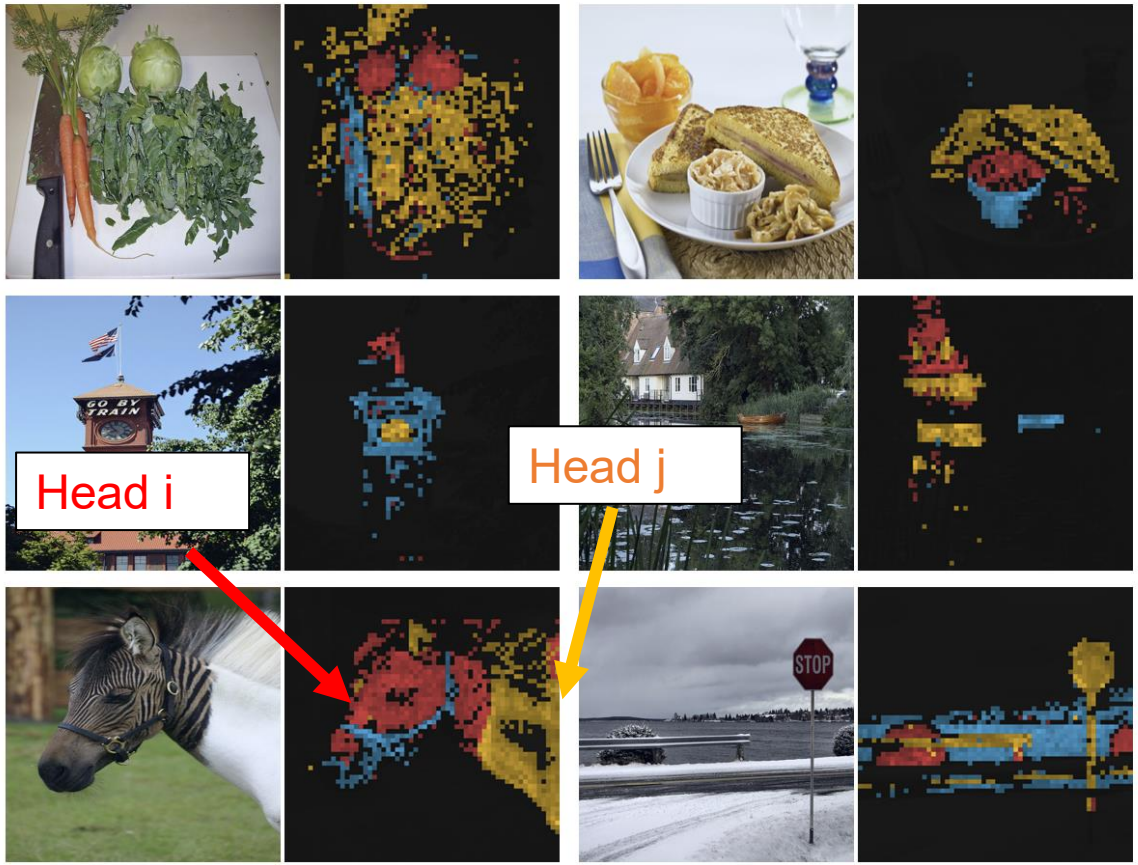
"A foal on the beach laying next to a horse."

[10] Bhalla, et al. "Interpreting clip with sparse linear concept embeddings (splice)." *NeurIPS 2024*

Special representation space #2: Attention heads

- Different attention heads attend to different region of the image.
- Attention maps quantifies attention to each patch and visualize it.

$$A = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

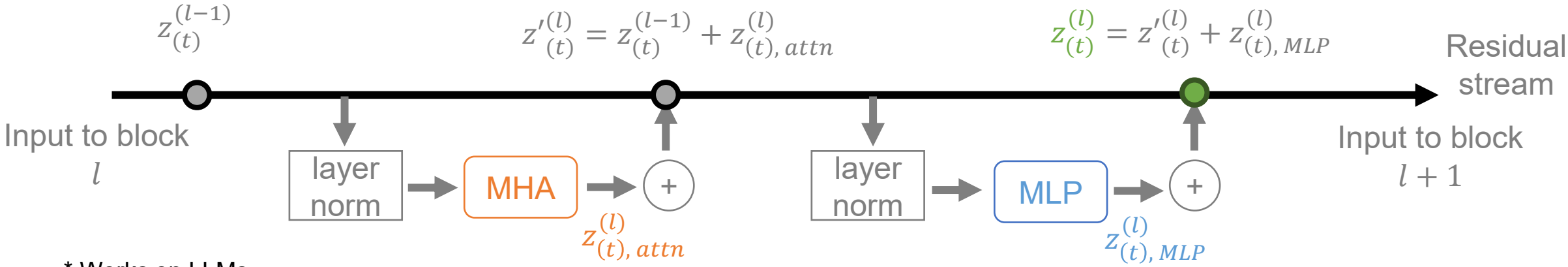


Different colors correspond to different attention heads at the last layer self-attention on CLS token.

[11] Caron, et al. "Emerging properties in self-supervised vision transformers." *ICCV* 2021.

Summary

Component	Method 1: Linear Probe / TCAV	Method 2: SAE	Method 3: Activation Patching
Transformer, Residual Stream	[22] Zhang et al.*	[13] Kravets et al.; [14] Zhang	[12] Żukowska et al.
Transformer, Attention Output	[16] Basile et al.	[21] Lieberum et al. *	[15] Chereddy; [12] Żukowska et al.
Transformer, MLP / FFN Output	[17] Chen et al.	[21] Lieberum et al. *	[15] Chereddy
CNN, Conv Block Output	[1] Kim et al.; [20] De Santis et al.	[19] Gorton	[18] Kowalska & Kwaśnicka



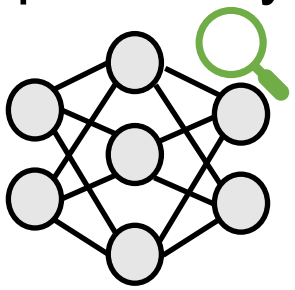
* Works on LLMs

References

[1] Kim et al. “Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV),” ICML, 2018.
[2] Pach et al. “Sparse Autoencoders Learn Monosemantic Features in Vision-Language Models,” NeurIPS, 2025.
[3] Rao et al. “Discover-then-Name: Task-Agnostic Concept Bottlenecks via Automated Concept Discovery,” ECCV, 2024.
[4] Oikarinen and Weng, “CLIP-Dissect: Automatic Description of Neuron Representations in Deep Vision Networks,” ICLR 2023
[5] Härkönen et al. “GANSpace: Discovering Interpretable GAN Controls,” NeurIPS, 2020.
[6] Izadi et al. “Causal Attribution via Activation Patching,” arXiv, 2026.
[7] Bricken et al. “Towards Monosemanticity: Decomposing Language Models with Dictionary Learning,” Transformer Circuits / Anthropic, 2023.
[8] Meng et al. “Locating and Editing Factual Associations in GPT,” NeurIPS, 2022.
[9] Conneau et al. “What You Can Cram into a Single $&!#* Vector: Probing Sentence Embeddings for Linguistic Properties,” ACL, 2018.
[10] Bhalla et al. “Interpreting CLIP with Sparse Linear Concept Embeddings (SpLiCE),” NeurIPS, 2024.
[11] Caron et al. “Emerging Properties in Self-Supervised Vision Transformers,” ICCV, 2021.
[12] Żukowska et al. “Seeing Through Circuits: Faithful Mechanistic Interpretability for Vision Transformers,” arXiv, 2026.
[13] Kravets et al. “Interpretability Transfer from Language to Vision via Sparse Autoencoders,” ICML, 2026.
[14] Zhang. “Sparse but Not Simpler: A Multi-Level Interpretability Analysis of Vision Transformers,” arXiv, 2026.
[15] Chereddy. “Attention Gathers, MLPs Compose: A Causal Analysis of an Action-Outcome Circuit in VideoViT,” AAAI Workshop, 2026.
[16] Basile et al. “Head Pursuit: Probing Attention Specialization in Multimodal Transformers,” NeurIPS, 2025.
[17] Chen et al. “Probing the Mid-Level Vision Capabilities of Self-Supervised Learning,” CVPR, 2025.
[18] Kowalska and Kwaśnicka. “From Networks to Circuits: A Structured Approach to AI Interpretability,” arXiv, 2025.
[19] Gorton. “The Missing Curve Detectors of InceptionV1: Applying Sparse Autoencoders to InceptionV1 Early Vision,” arXiv, 2024.
[20] De Santis et al. “Visual-TCAV: Concept-Based Attribution and Saliency Maps for Post-hoc Explainability in Image Classification,” arXiv, 2024.
[21] Lieberum et al. “Gemma Scope: Open Sparse Autoencoders Everywhere All at Once on Gemma 2,” arXiv, 2024.
[22] Zhang et al. “Controlling Large Language Models Through Concept Activation Vectors”, AAAI 2025

Tutorial Part 2: Overview

Post-hoc Interpretability tools



What knowledge has the model learned?

⇒ We will discuss methods from small unit to large unit

1. Understand individual neurons (**neuron-level** interpretability tools)

2. Understand groups of neurons / full layer (**representation-level** interpretability tools)

3. Understand multiple layers / circuits (**circuit-level** interpretability tools)

Lily

Ge

Akshay

https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

Introduction: Larger Unit

- Why larger units?
  - Neuron explanations or layer representations may be highly localized
  - Beyond single neuron/layer, can we understand a DNN across layers?
  - Discover circuit of neurons across layers in a DNN

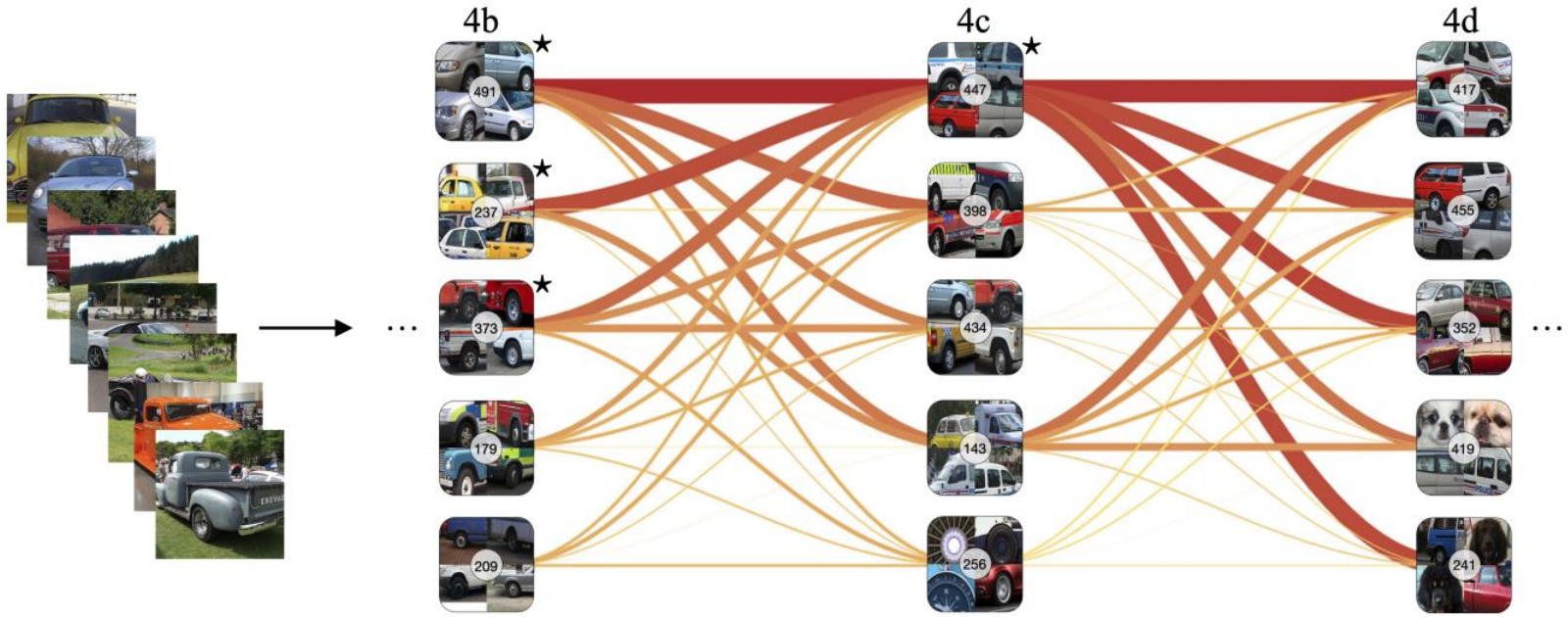
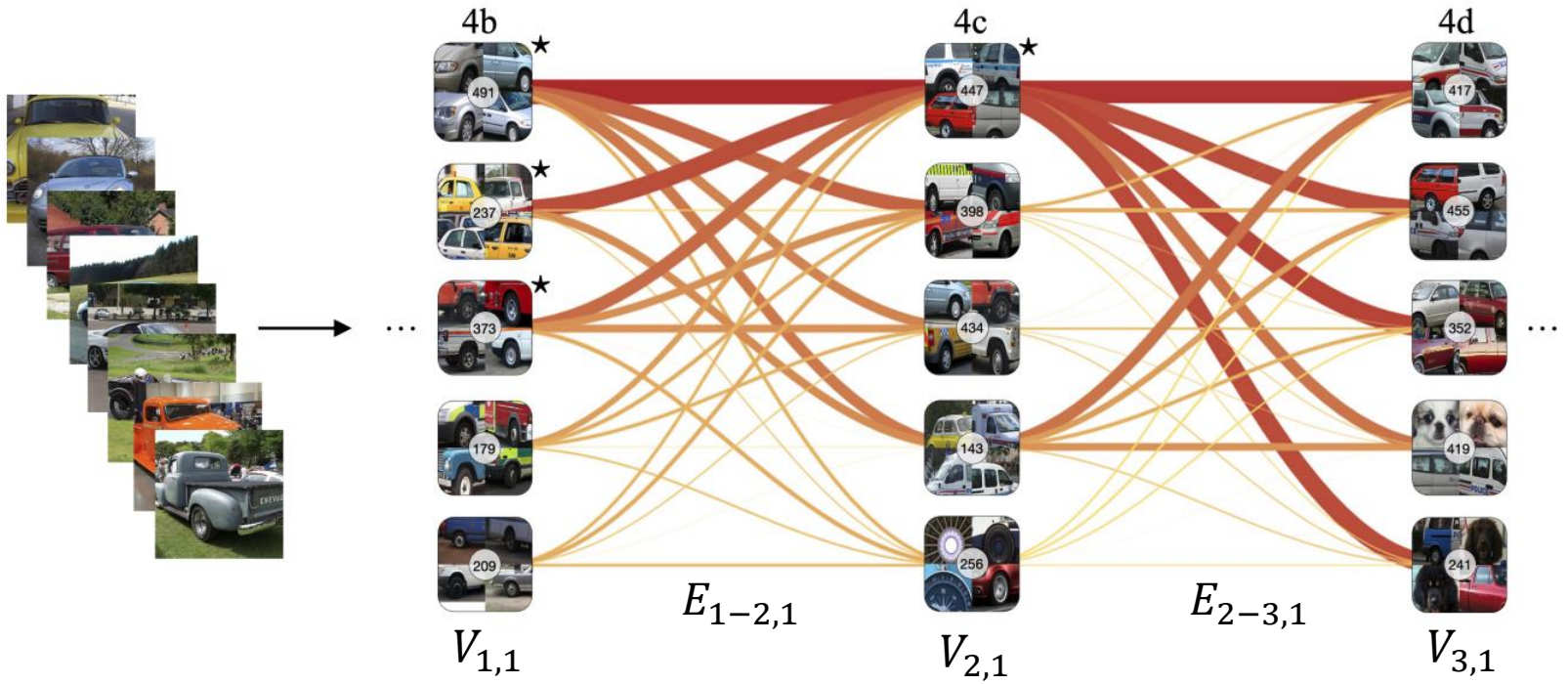


Image source: [1] Rajaram et al., “Automatic Discovery of Visual Circuits”, NeurIPS23W

Key Ideas

- A circuit can be parameterized as a directed acyclic graph $G = (V, E)$
  - Vertices V → neurons or tokens or other smaller units in the DNN
  - Edges E → weights connecting the vertices in the graph



Key Ideas

- A circuit can be parameterized as a directed acyclic graph $G = (V, E)$
  - Vertices V → neurons or tokens or other smaller units in the DNN
  - Edges E → weights connecting the vertices in the graph

- Desirable properties for discovered circuits
  - Let $G_f = (V_f, E_f)$ be the full computational graph of the DNN → $G \subseteq G_f$
  - Let $\mathcal{M}$ be a metric of interest
    - e.g. class accuracy, to discover a circuit for classifying a particular class
  - Circuit faithfulness: circuit should behave similar to the full DNN w.r.t. $\mathcal{M}$

$$|\mathcal{M}(G_f) - \mathcal{M}(G)| \leq \epsilon \text{ where } |V| \ll |V_f|$$

[2] Żukowska et al., “Seeing Through Circuits: Faithful Mechanistic Interpretability for Vision Transformers”, arXiv 2604.14477

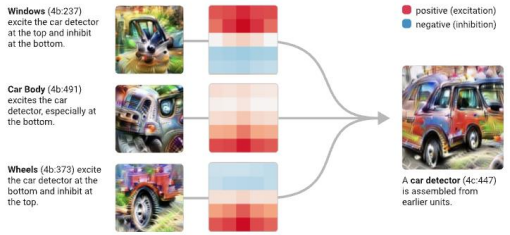
Characteristics of Circuit Discovery Methods

- The key differences between methods lie in the following choices:
  - Q1. Automation
    - Is the circuit constructed automatically or by manual inspection by human researchers?
    - For Q2 and Q3: are they completely automated or involve manual inspection?
  - Q2. Method for identifying vertices V (smaller units)
    - E.g. using neuron interpretability (CLIP-Dissect[12]) or sparse autoencoders[13], etc.
  - Q3. Method for identifying edges E
    - E.g. using model weights or activation correlation or inter-neuron sensitivity, etc.
  - Identifying V and E (Q2, Q3) can be simultaneous or sequential (in any order)

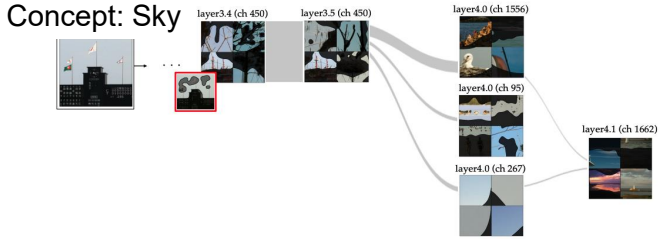
[12] Oikarinen and Weng, “CLIP-Dissect”, ICLR23
[13] Joseph et al., “Prisma”, CVPR25W

Overview of Circuit Discovery Methods

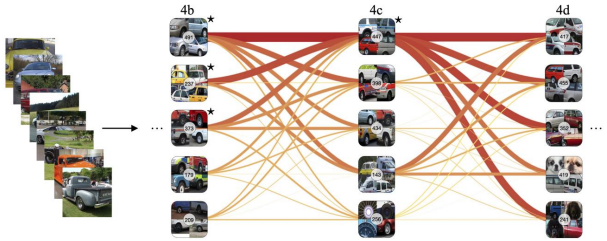
Type 1: Manual Inspection
Q1: Both vertices & edges manually
Q2: Vertices? → Feature visualization
Q3: Edges? → Visualizing weights



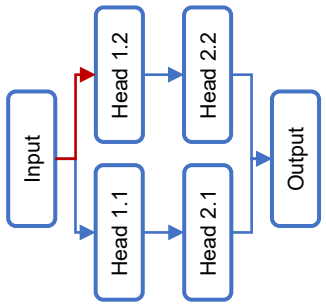
Type 2: Concept Graph
Q1: Both automatically
Q2: Neurons as concepts
Q3: Sensitivity & similarity b/w vertices



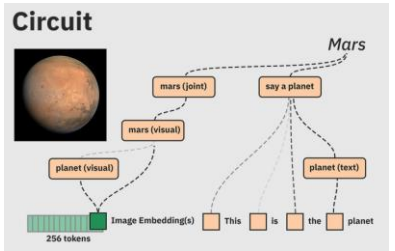
Type 3: Activation Correlation
Q1: Both automatically
Q2: Vertices that maximize attribution
Q3: Attribution score b/w vertices



Type 4: Activation Patching
Q1: Both automatically
Q2: Vertices whose edges are present in circuit
Q3: Edges which causally affect output

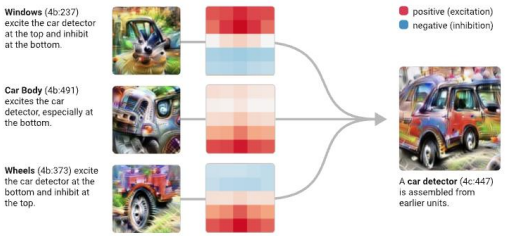


Type 5: SAE/Transcoder based
Q1: Vertices automatically, edges manually
Q2: Vertices: SAE/transcoder decomposed tokens
Q3: Manual inspection of attribution graph & features



Overview of Circuit Discovery Methods

Type 1: Manual Inspection
Q1: Both vertices & edges manually
Q2: Vertices? → Feature visualization
Q3: Edges? → Visualizing weights



Type 2: Concept Graph
Q1: Both automatically
Q2: Neurons as concepts
Q3: Sensitivity & similarity b/w vertices

Type 3: Activation Correlation
Q1: Both automatically
Q2: Vertices that maximize attribution
Q3: Attribution score b/w vertices

Type 4: Activation Patching
Q1: Both automatically
Q2: Vertices whose edges are present in circuit
Q3: Edges which causally affect output

Type 5: SAE/Transcoder based
Q1: Vertices automatically, edges manually
Q2: Vertices: SAE/transcoder decomposed tokens
Q3: Manual inspection of attribution graph & features

Type 1: Manual Inspection

- How to identify vertices V? Considering CNN layer channels as neurons
  - Feature visualization[3]:
    - Adversarially optimize an image to highly activate a particular neuron

$$\mathbf{x}^* = \arg \max_{\mathbf{x}\ s.t.\ ||\mathbf{x}||=\rho} h_{ij}(\theta, \mathbf{x})$$

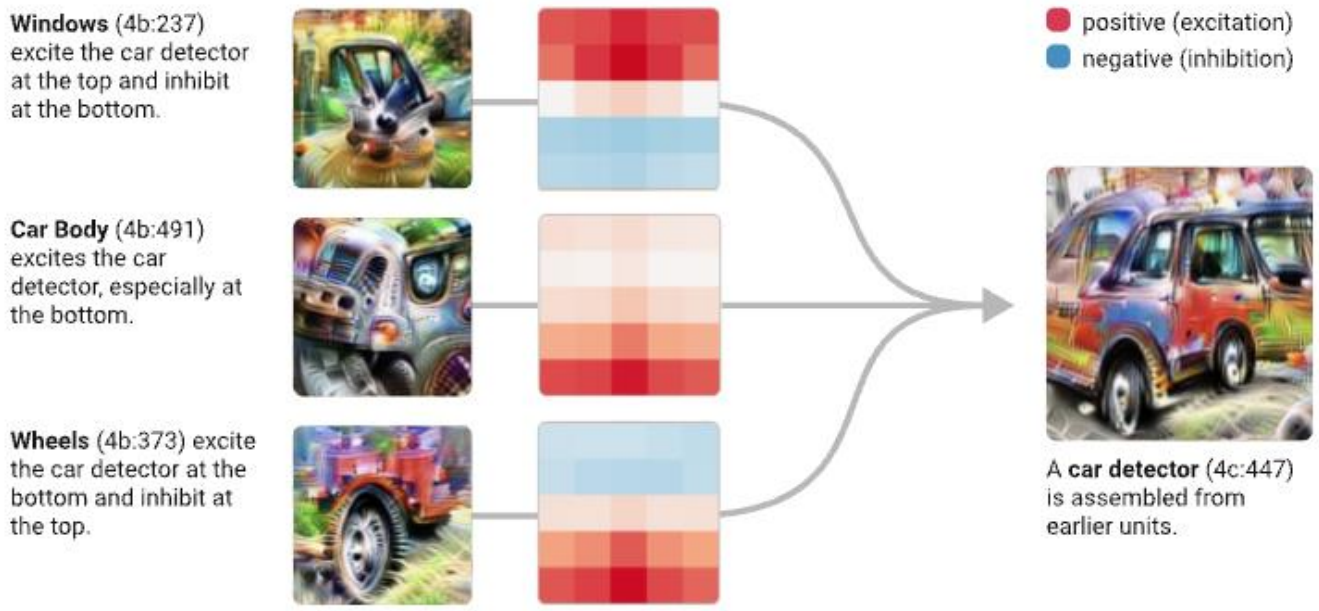
  - Manually inspect optimized images and find relevant neurons [4, 5]


A **car detector** (4c:447)

[3] Olah et al., “Feature Visualization”, Distill 2017
[4] Olah et al., “Zoom In: An Introduction to Circuits”, Distill 2020
[5] Cammarata et al., “Curve Circuits”, Distill 2021

Type 1: Manual Inspection

- How to identify edges E? Considering CNN layer channels as neurons (vertices)
  - Consider conv. weights with highest L2-norm between selected vertices
  - Manual inspection to find relevant edges



[4] Olah et al., “Zoom In: An Introduction to Circuits”, Distill 2020
[5] Cammarata et al., “Curve Circuits”, Distill 2021

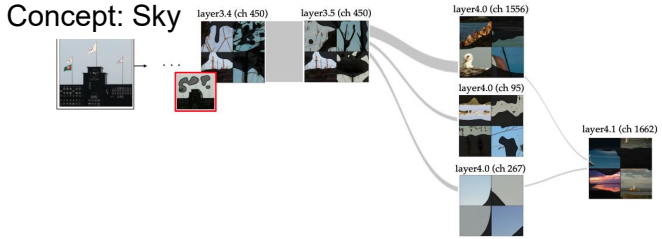
Overview of Circuit Discovery Methods

Type 1: Manual Inspection

Q1: Both vertices & edges manually
Q2: Vertices? → Feature visualization
Q3: Edges? → Visualizing weights

Type 2: Concept Graph

Q1: Both automatically
Q2: Neurons as concepts
Q3: Sensitivity & similarity b/w vertices



Type 3: Activation Correlation

Q1: Both automatically
Q2: Vertices that maximize attribution
Q3: Attribution score b/w vertices

Type 4: Activation Patching

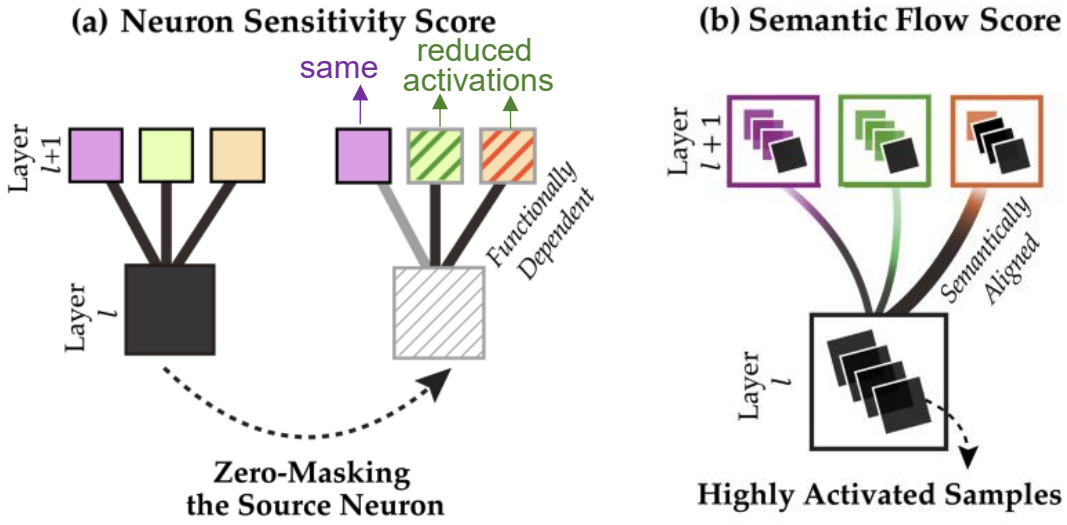
Q1: Both automatically
Q2: Vertices whose edges are present in circuit
Q3: Edges which causally affect output

Type 5: SAE/Transcoder based

Q1: Vertices automatically, edges manually
Q2: Vertices: SAE/transcoder decomposed tokens
Q3: Manual inspection of attribution graph & features

Type 2: Concept Graph Construction [6-8]

- How to identify vertices V and edges E?[8]
  - Root nodes (starting nodes)
    - For a given query image, pick nodes/neurons with > top-1% activations
  - Other nodes in the circuit
    - For each root node in layer l, check all neurons in the next layer $l+1$

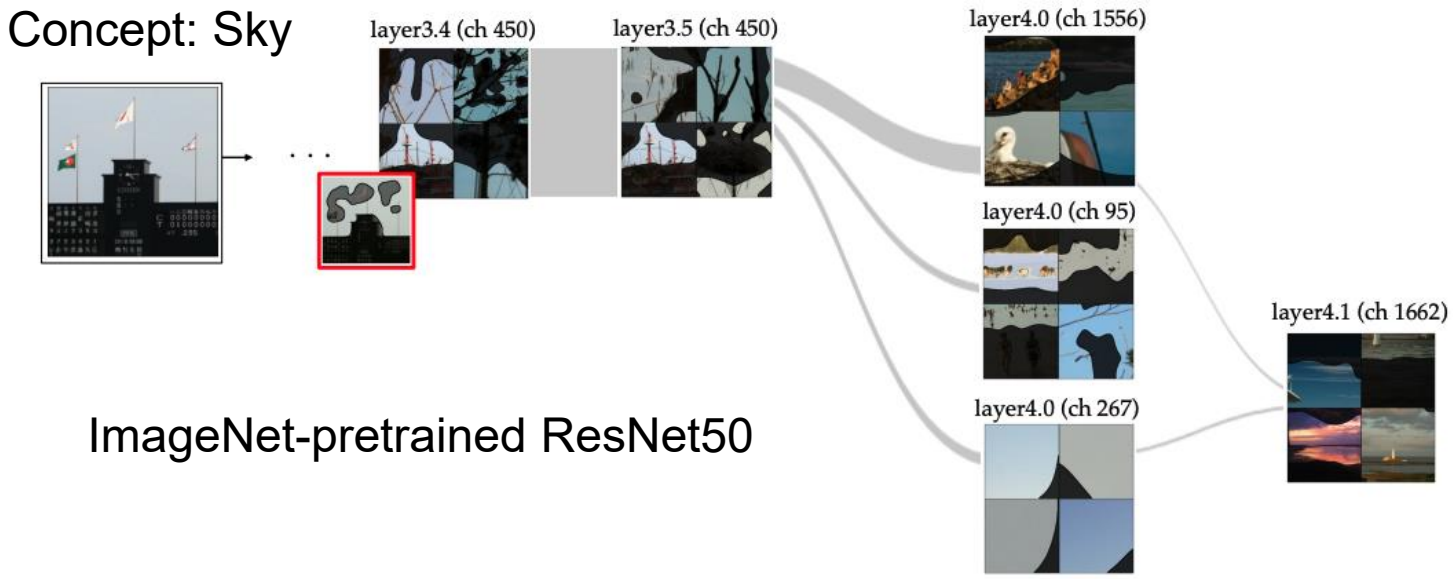


Edge weight = $(S_{NS} + S_{SF})$
given $S_{NS} > \tau_{NS}$ and $S_{SF} > \tau_{SF}$

[6] Visual Concept Connectome, CVPR24; [7] NeurFlow, ICLR25, [8] Granular Concept Circuits, ICCV25

Type 2: Concept Graph Construction [6-8]

- How to identify vertices V and edges E?[8]
  - Root nodes (starting nodes)
    - For a given query image, pick nodes/neurons with > top-1% activations
  - Other nodes in the circuit
    - For each root node in layer l, check all neurons in the next layer $l+1$



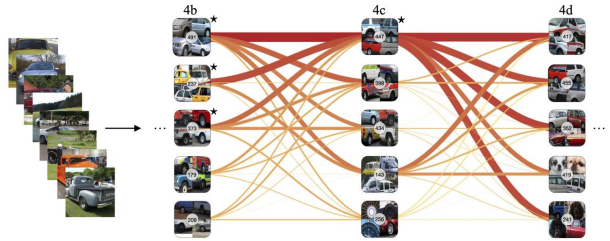
[8] Granular Concept Circuits, ICCV25

Overview of Circuit Discovery Methods

Type 1: Manual Inspection
Q1: Both vertices & edges manually
Q2: Vertices? → Feature visualization
Q3: Edges? → Visualizing weights

Type 2: Concept Graph
Q1: Both automatically
Q2: Neurons as concepts
Q3: Sensitivity & similarity b/w vertices

Type 3: Activation Correlation
Q1: Both automatically
Q2: Vertices that maximize attribution
Q3: Attribution score b/w vertices



Type 4: Activation Patching
Q1: Both automatically
Q2: Vertices whose edges are present in circuit
Q3: Edges which causally affect output

Type 5: SAE/Transcoder based
Q1: Vertices automatically, edges manually
Q2: Vertices: SAE/transcoder decomposed tokens
Q3: Manual inspection of attribution graph & features

Type 3: Activation Correlation based

- How to identify vertices V and edges E?[1, 9]
  - Quantify activation correlation across layers with an attribution matrix

$$a_i = f(x)[l_i] \rightarrow \text{layer } l_i \text{ activations}$$

$$A[x, l_i, m, n] = \frac{\partial \|l_{i+1,n}(a_i)\|_2}{\partial a_{i,m}} \rightarrow \text{attribute } l_{i+1} \text{ neurons to } l_i \text{ neurons}$$

$$A = \sum_x A[x] \rightarrow \text{aggregate over dataset}$$

  - Iteratively build and refine circuit by maximizing attribution at each layer

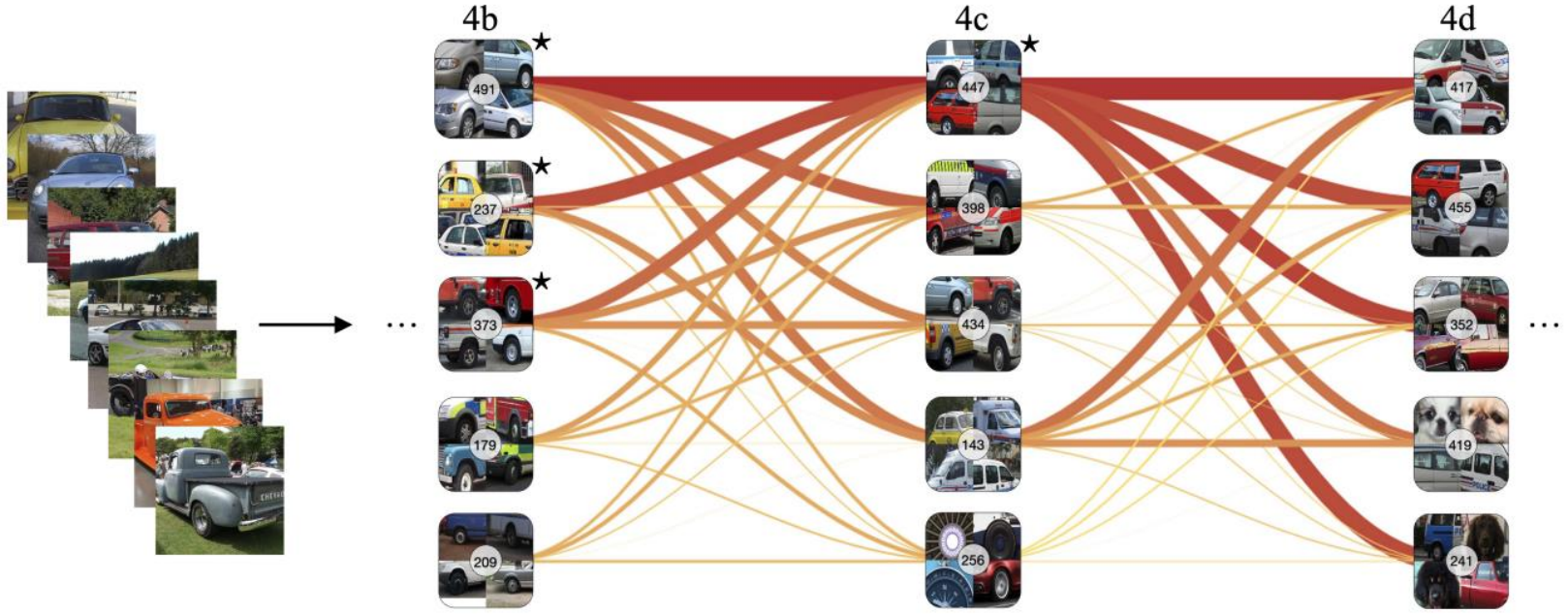
for $i = L$ to 1: top-k $\arg\max_m \sum(A[l_i, m, n])$ → select layer l_i neurons that maximize attribution with l_{i+1}

for $i = 1$ to L: top-k $\arg\max_n \sum(A[l_{i-1}, m, n])$ → select layer l_i neurons that maximize attribution with l_{i-1}

[9] Wang et al., “Discovering Influential Neuron Path in Vision Transformers”, ICLR25
[1] Rajaram et al., “Automatic Discovery of Visual Circuits”, NeurIPS23W

Type 3: Activation Correlation based

- How to identify vertices V and edges E?[1, 9]



Vertices selected iteratively by maximizing the attribution score
Edges shown are proportional to the attribution score

[1] Rajaram et al., “Automatic Discovery of Visual Circuits”, NeurIPS23W

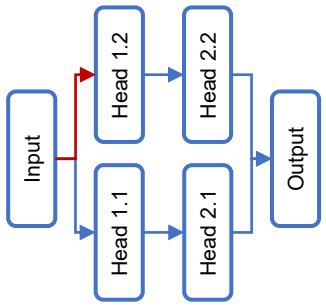
Overview of Circuit Discovery Methods

Type 1: Manual Inspection
Q1: Both vertices & edges manually
Q2: Vertices? → Feature visualization
Q3: Edges? → Visualizing weights

Type 2: Concept Graph
Q1: Both automatically
Q2: Neurons as concepts
Q3: Sensitivity & similarity b/w vertices

Type 3: Activation Correlation
Q1: Both automatically
Q2: Vertices that maximize attribution
Q3: Attribution score b/w vertices

Type 4: Activation Patching
Q1: Both automatically
Q2: Vertices whose edges are present in circuit
Q3: Edges which causally affect output

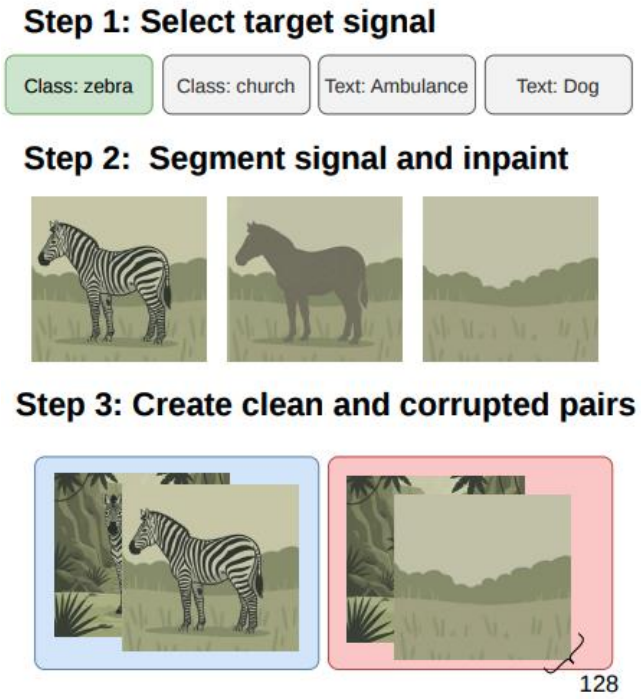


Type 5: SAE/Transcoder based
Q1: Vertices automatically, edges manually
Q2: Vertices: SAE/transcoder decomposed tokens
Q3: Manual inspection of attribution graph & features

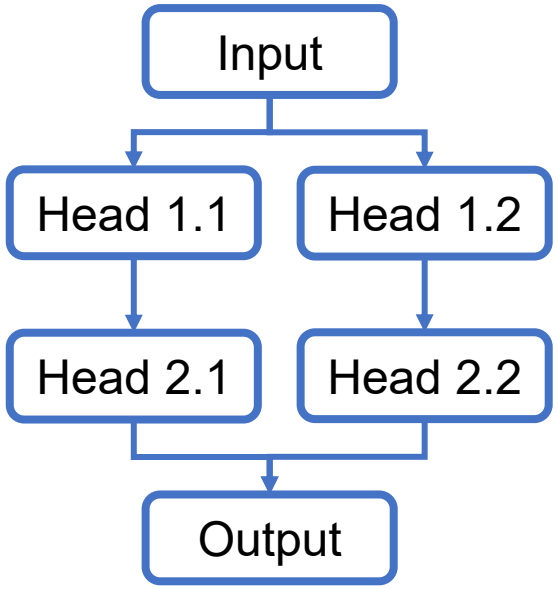
Type 4: Activation Patching based

- Key idea: Identify edges E that are important

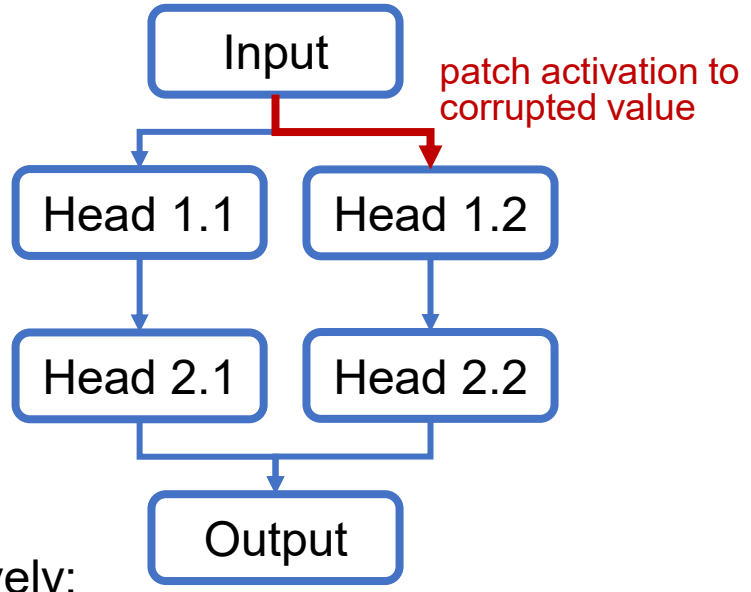
A. Clean-Corrupted Data Creation



B. Save Clean/Corrupted Activations



C. Build Circuit



Iteratively:
Exclude (nodes and) **edge** from circuit if
$\|\mathcal{M}(o_{clean}) - \mathcal{M}(o_{corrupt})\|_2 < \tau$

[2] Żukowska et al., "Seeing Through Circuits", arXiv 2604.14477

Overview of Circuit Discovery Methods

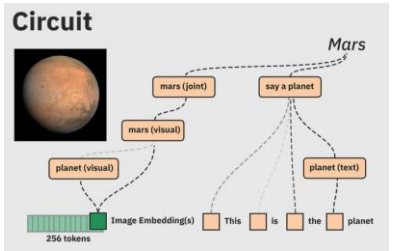
Type 1: Manual Inspection
Q1: Both vertices & edges manually
Q2: Vertices? → Feature visualization
Q3: Edges? → Visualizing weights

Type 2: Concept Graph
Q1: Both automatically
Q2: Neurons as concepts
Q3: Sensitivity & similarity b/w vertices

Type 3: Activation Correlation
Q1: Both automatically
Q2: Vertices that maximize attribution
Q3: Attribution score b/w vertices

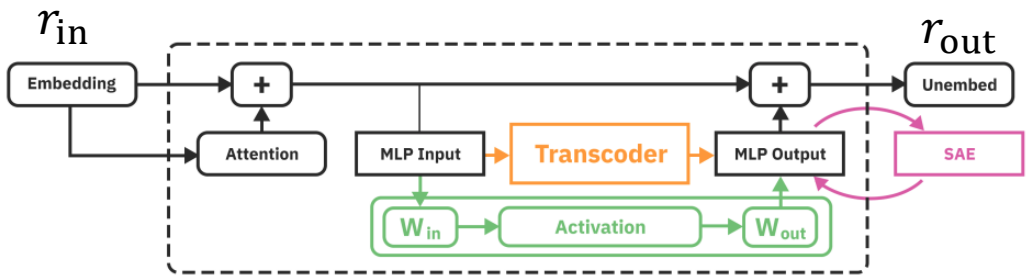
Type 4: Activation Patching
Q1: Both automatically
Q2: Vertices whose edges are present in circuit
Q3: Edges which causally affect output

Type 5: SAE/Transcoder based
Q1: Vertices automatically, edges manually
Q2: Vertices: SAE/transcoder decomposed tokens
Q3: Manual inspection of attribution graph & features



Type 5: SAE/Transcoder based

- How to identify vertices V and edges E?[10]
  - Use intermediate layer tokens (& input tokens & output logits) as vertices
  - Use transcoder to decompose tokens into interpretable components
  - For a given input, create an attribution graph with edges $E = A_{s \to t}$



$$A_{s \to t} = a_s w_{s \to t} \in \mathbb{R}$$

Attribution of vertex s to vertex t

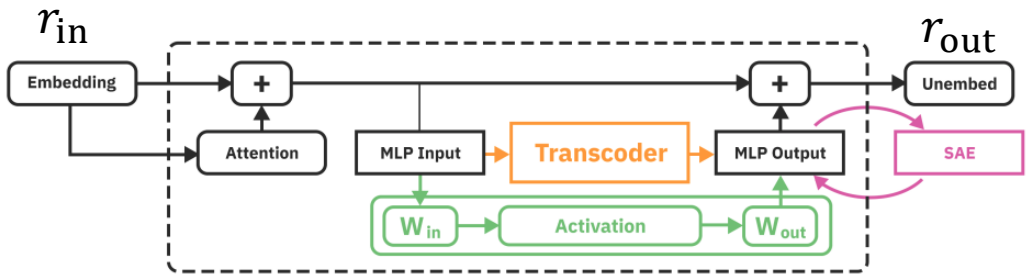
Derivative of pre-activation of t w.r.t. activation of s

Activation magnitude of node s

[10] Yang et al., “Circuit Tracing in Vision–Language Models”, CVPR26F

Type 5: SAE/Transcoder based

- How to identify vertices V and edges E?[10]
  - Use intermediate layer tokens (& input tokens & output logits) as vertices
  - Use transcoder to decompose tokens into interpretable components
  - For a given input, create an attribution graph with edges $E = A_{s \to t}$



$$A_{s \to t} = a_s w_{s \to t} \in \mathbb{R}$$

Attribution of vertex s to vertex t

Jacobian matrix: How much does vertex s affect vertex t (computed via autograd)

Apply chain rule for the transcoder: $w_{s \to t} = f_{\text{dec}}^{(s)\top} J_{s \to t} f_{\text{enc}}^{(t)}$

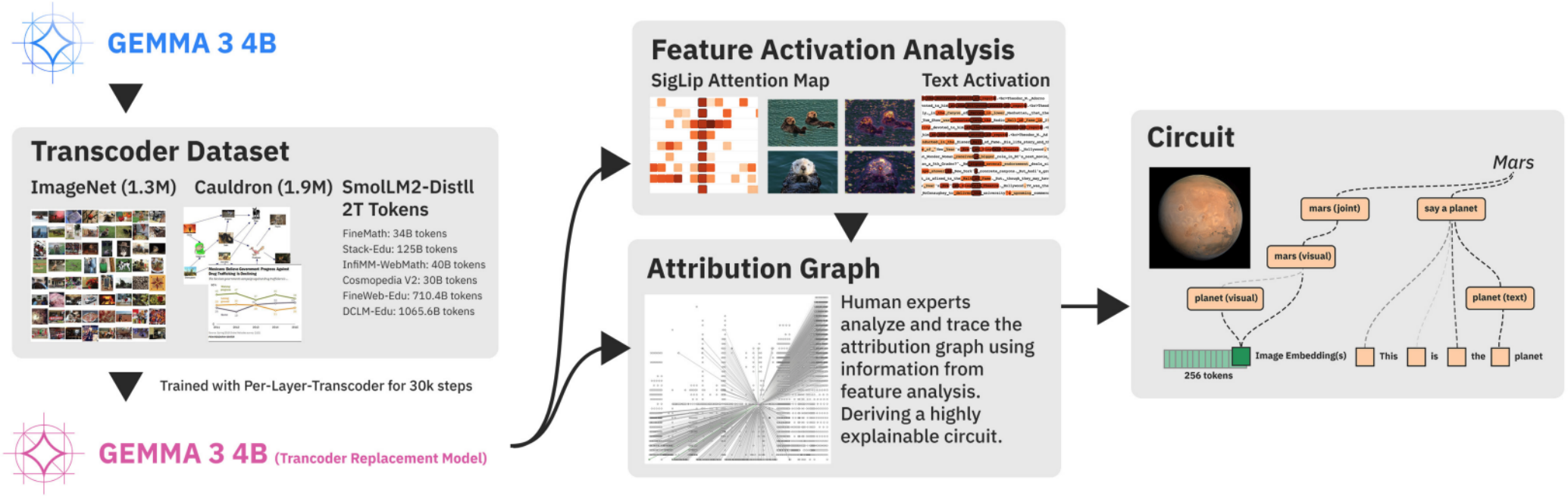
Transcoder decoder vector for vertex s

Transcoder encoder vector for vertex t

[10] Yang et al., “Circuit Tracing in Vision–Language Models”, CVPR26F

Type 5: SAE/Transcoder based

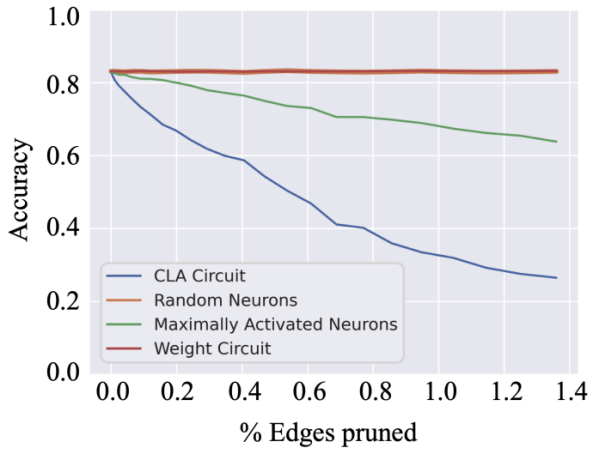
- Final circuit identification[10]
  - Feature activation visualizations:
    - Top-k activating examples & vision encoder attention maps for each vertex (token)
  - Manual inspection of attribution graph and feature activation visualizations



[10] Yang et al., “Circuit Tracing in Vision–Language Models”, CVPR26F

Evaluation

- Most vision circuit discovery works use their own evaluation methods
  - There is scope to create a more unified evaluation benchmark
  - E.g. evaluation benchmarks in LLM circuit discovery[11]
- Evaluation methods
  - Interventions with circuits should causally affect model outputs
    - Pruning the circuit (i.e. setting circuit node activations to zero)
    - Activating the circuit (i.e. setting all other nodes outside the circuit to zero)

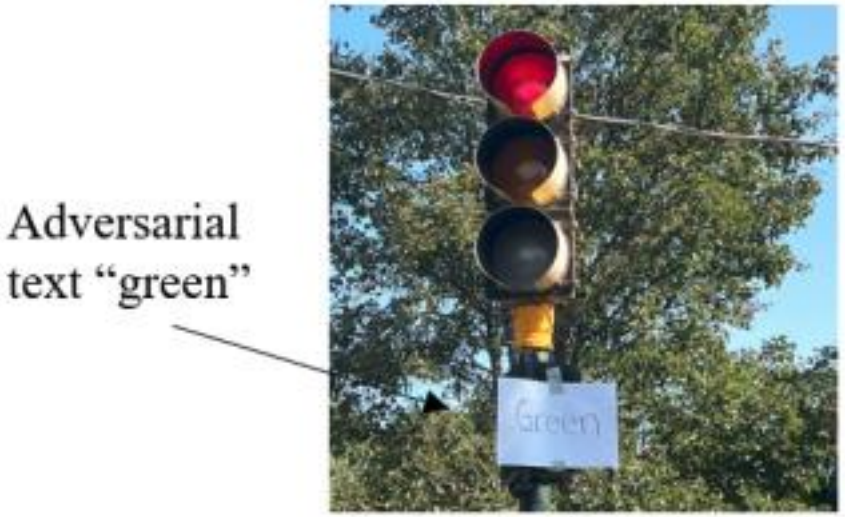


E.g. from [1], pruning particular class' circuit neurons reduces class accuracy

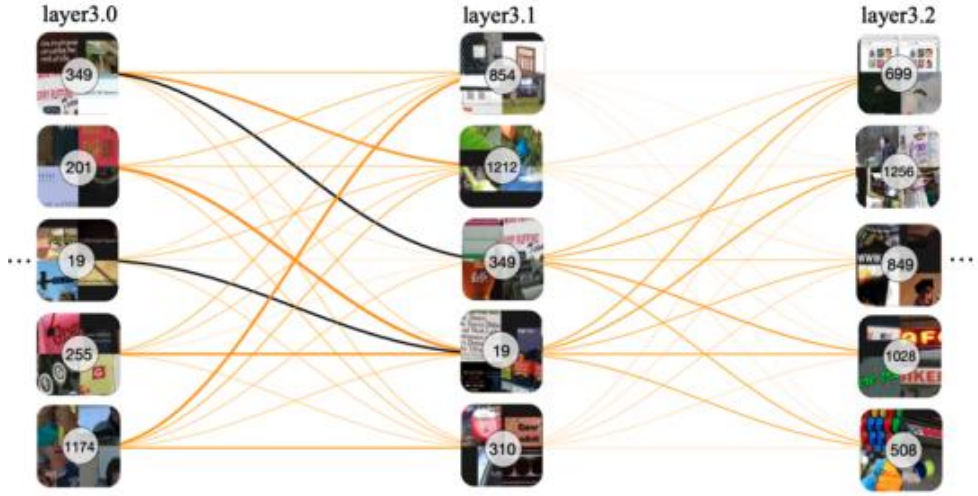
[1] Rajaram et al., "Automatic Discovery of Visual Circuits", NeurIPS23W
[11] Mueller et al., "MIB: A Mechanistic Interpretability Benchmark", ICML25

Usecases

- Interventions to prevent text-based adversarial attacks



CLIP: circuit for images containing text



CLIP zero-shot probabilities

red traffic light	0.036
green traffic light	**0.964**

Probabilities after pruning circuit

red traffic light	**0.874**
green traffic light	0.126

[1] Rajaram et al., “Automatic Discovery of Visual Circuits”, NeurIPS23W

References

1. Rajaram et al., “Automatic Discovery of Visual Circuits”, NeurIPS 2023 Workshop
2. Żukowska et al., “Seeing Through Circuits: Faithful Mechanistic Interpretability for Vision Transformers”, arXiv 2604.14477
3. Olah et al., “Feature Visualization”, Distill 2017
4. Olah et al., “Zoom In: An Introduction to Circuits”, Distill 2020
5. Cammarata et al., “Curve Circuits”, Distill 2021
6. Kowal et al., “Visual Concept Connectome: Open World Concept Discovery and their Interlayer Connections in Deep Models”, CVPR 2024
7. Cao et al., “NeurFlow: Interpreting Neural Networks through Neuron Groups and Functional Interactions”, ICLR 2025
8. Kwon et al., “Granular Concept Circuits: Toward a Fine-Grained Circuit Discovery for Concept Representations”, ICCV 2025
9. Wang et al., “Discovering Influential Neuron Path in Vision Transformers”, ICLR 2025
10. Yang et al., “Circuit Tracing in Vision–Language Models: Understanding the Internal Mechanisms of Multimodal Thinking”, CVPR 2026 Findings
11. Mueller et al., “MIB: A Mechanistic Interpretability Benchmark”, ICML 2025
12. Oikarinen and Weng, “CLIP-Dissect: Automatic Description of Neuron Representations in Deep Vision Networks”, ICLR 2023
13. Joseph et al., “Prisma: An Open Source Toolkit for Mechanistic Interpretability in Vision and Video”, CVPR 2025 Workshop

Principled Interpretability Tutorial: Part 3

https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

Lily

Part 1

Fundamentals & backgrounds

How does a deep vision model work?

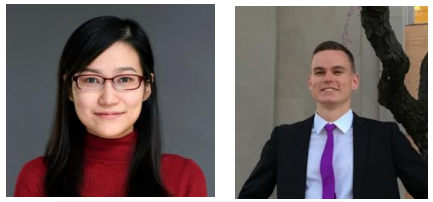
Lily Ge Akshay

Part 2

Post-hoc Interpretability tools

What knowledge has the model learned?

Lily Tuomas



Part 3
2:30 - 3:00

Rigorous Analysis to Evaluate interpretability

Which explanation is more faithful?

Ge Akshay

Part 4

Building Inherently Interpretable DNNs

Embed human-understandable concepts c_i

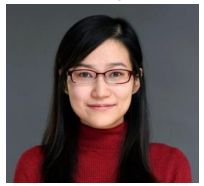
Tutorial Part 3: Overview

Rigorous Analysis to Evaluate interpretability

Which explanation is more faithful?

1. Methods to evaluate explanations

Lily



2. DEMO
(neuron-eval & sanity tests for reliable metric)

Tuomas

https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

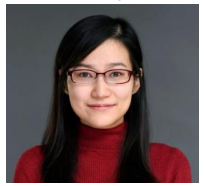
Tutorial Part 3: Overview

Rigorous Analysis to Evaluate interpretability

Which explanation is more faithful?

1. Methods to evaluate explanations

Lily



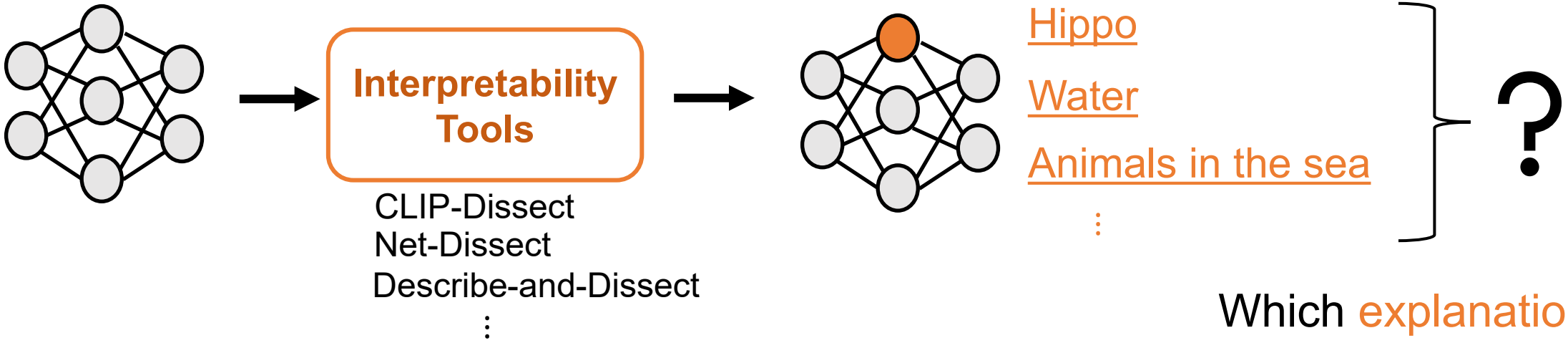
2. DEMO
(neuron-eval & sanity tests for reliable metric)

Tuomas

https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

Challenge

- Many methods have been proposed to explain neurons:



Which explanation should we trust?

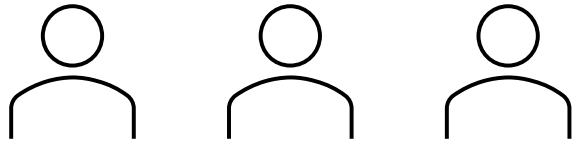
- However, it is not clear:
  - *Which methods produce best explanations*?
  - *How should we evaluate how accurate / faithful an explanation is?*

Existing Evaluation – 1. human study

- Currently, one of the most common methods to evaluate neuron explanation is through crowdsourced/human study

Show human a set of highly activated images for the neuron of interest and a neuron explanation (e.g. “Red”) from a method, then ask human to rate it

Q: Does the explanation match the images?



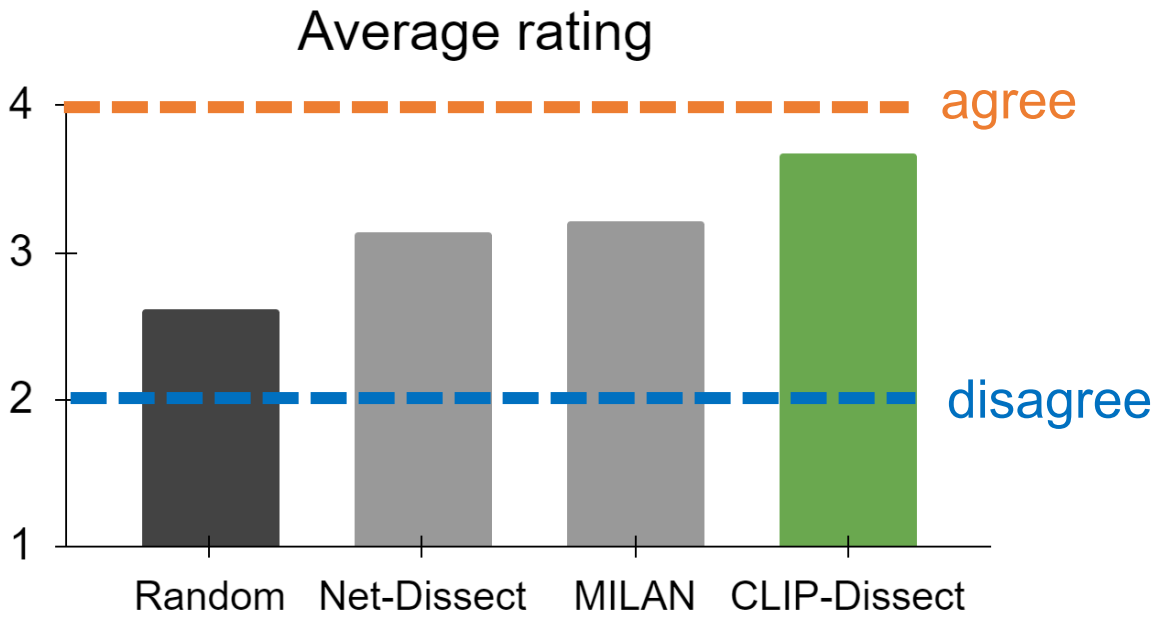
Choice: No, Maybe, Yes

Choice: Strongly disagree, disagree, not agree nor disagree, agree, strongly agree

Existing Evaluation – 1. human study

- Large scale crowdsourced eval:

Total 1520 neurons across 2 models & 10 layers



As we will discuss shortly, it turns out that this is actually not a good way for evaluation

[**Oikarinen** & **Weng**, ICLR'23 spotlight]

Existing Evaluation – 2. labelled dataset

- Use curated labelled dataset (e.g. Broden, Imagenet val)

  Calculate how much the explanation (through concept masks in the labelled dataset) overlap with (binarized) neuron activations using IOU score

Given neuron of interest: unit 2

activated images

Given Explanation: Lamp

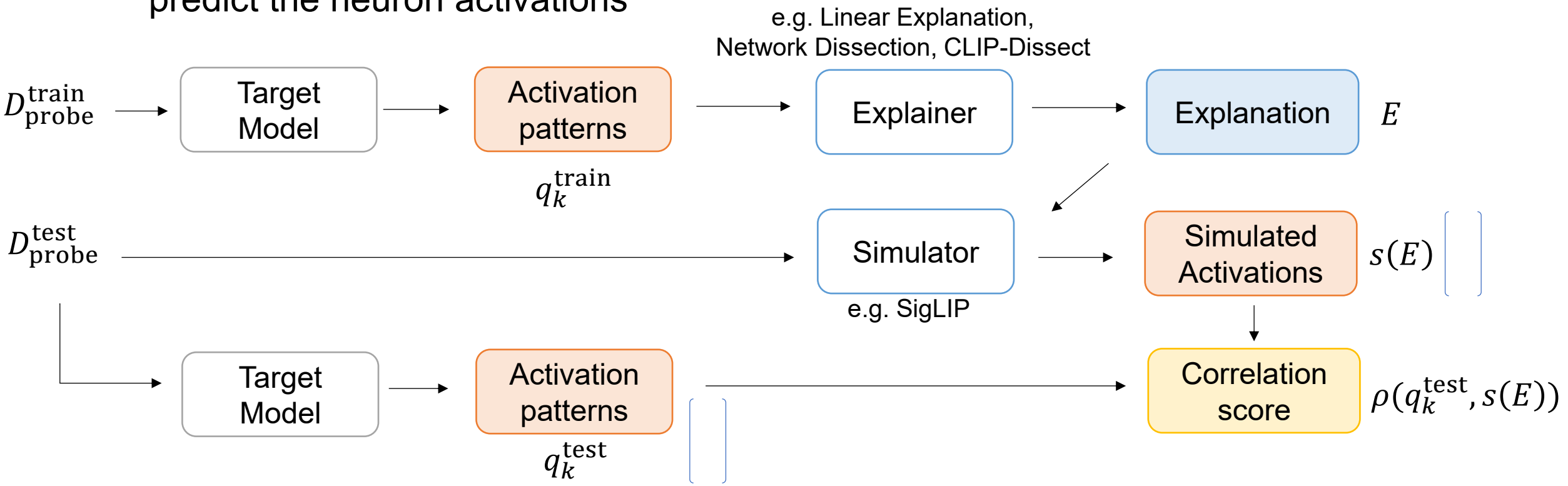
concept mask of lamp

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

[**Bau etal**, CVPR'17; **La Rosa etal**, NeurIPS'23]

Existing Evaluation – 3. Simulation

- Use a simulator to simulate *how much a neuron will activate on the given input*
  - **Key idea:** a good explanation should allow the simulator model to accurately predict the neuron activations



[**Oikarinen** & **Weng**, ICML'24]

Existing Evaluation – 3. Simulation

- Simulation results:

Target model	Network Dissection	MILAN	CLIP-Dissect	LE (Label)	LE (SigLIP)
ResNet-50 (ImageNet)	0.1242 ± 0.002	0.0920 ± 0.002	0.1871 ± 0.002	0.2924 ± 0.002	**0.3772 ± 0.002**
ResNet-18 (Places365)	0.2038 ± 0.005	0.1557 ± 0.005	0.2208 ± 0.005	0.3388 ± 0.004	**0.4372 ± 0.003**
VGG-16 (CIFAR-100)	n/a	n/a	0.2298 ± 0.004	0.4330 ± 0.004	**0.4970 ± 0.004**
ViT-B/16 (ImageNet)	n/a	n/a	0.1722 ± 0.004	0.3243 ± 0.005	**0.3489 ± 0.005**
ViT-L/32 (ImageNet)	n/a	n/a	0.0549 ± 0.002	0.1879 ± 0.004	**0.2182 ± 0.004**

Table 2. Average correlation scores between simulated and actual neuron activations, across all neurons in the second to last layer of the respective models. For ViT models we report the MLP neurons in the last transformer block. We do not include Network Dissection and MILAN results for the last two models, as those methods are designed for 2d activations, while the final layers of these models have effectively scalar activations.

[**Oikarinen** & **Weng**, ICML'24]

Challenge of Existing Evaluation approaches

- We have discussed 3 main existing evaluation methods:
  - *How are they connected to each other?*
  - *Which one should we use?*
  - *Is there a systematic way to perform evaluation?*

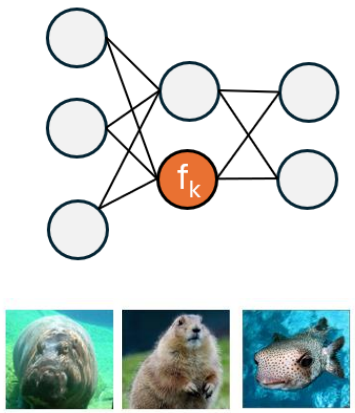
⇨ Motivation of **Neuron-Eval**:
A unified framework to evaluate and compare neuron explanation quantitatively

[**Oikarinen etal**, ICML'25]

Neuron Eval

How faithful is this explanation?

- A unified framework to *evaluate* and *compare* neuron explanations quantitatively



Prior work as a special case #1

Crowdsourced Evaluation on Highly Activating Input (Zhou etal'15, Bau etal'17, Oikarinen etal' 23)

$$s := \mathbb{P}(\text{concept}|\text{neuron is active})$$

$$= \frac{B(a_k) \cdot B(c_t)}{||B(a_k)||_1}$$

$$= \frac{TP}{TP + FN}$$

$$\text{TP} = B(a_k) \cdot B(c_t),$$

$$\text{FN} = B(a_k) \cdot \overline{B(c_t)},$$

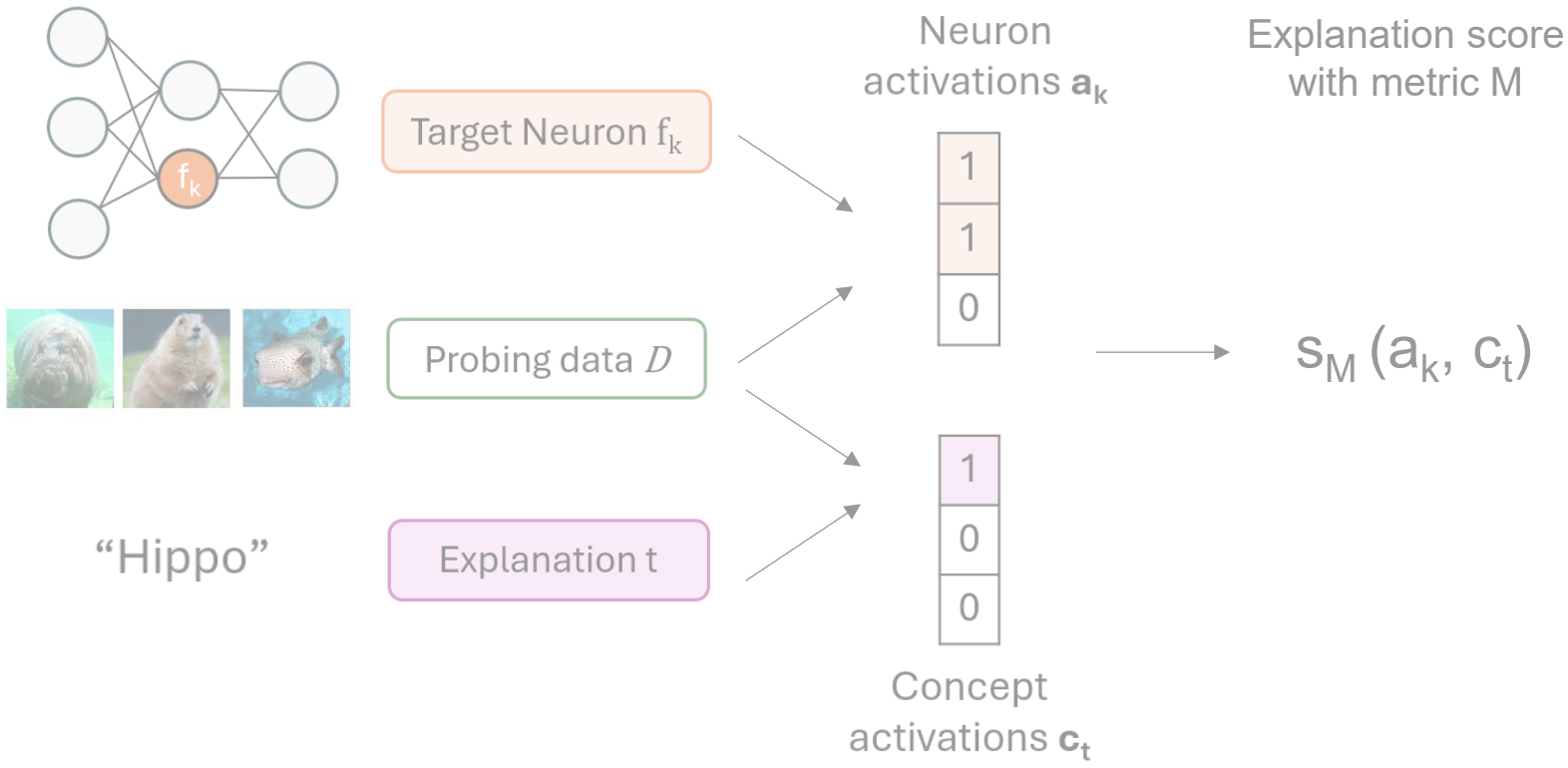
Can be viewed as binary classification: predict neuron activations

- M = Recall
- c_t from Amazon Turk raters

Oikarinen etal, *Evaluating Neuron Explanations: A Unified Framework with Sanity Tests*, ICML 25

Neuron Eval

- A unified framework to *evaluate* and *compare* neuron explanations quantitatively



Prior work as a special case #2

IoU with Labeled Data (Bau etal'17)

- M = IoU
- c_t from curated labeled dataset (Broden)

Prior work as a special case #3

Automated Simulation Evaluation with Language Models & VLM (Bills etal'23, Oikarinen etal'24)

- M = Correlation coefficient
- c_t from model predicted neuron activations (pseudo labels)

Oikarinen etal, *Evaluating Neuron Explanations: A Unified Framework with Sanity Tests*, ICML 25

Neuron Eval

We can represent diverse evaluations from **19** previous works and **18** different metrics under our framework!

- 13 existing + 5 new from this work
- can handle individual neurons, TCAV, linear probing, SAE features etc

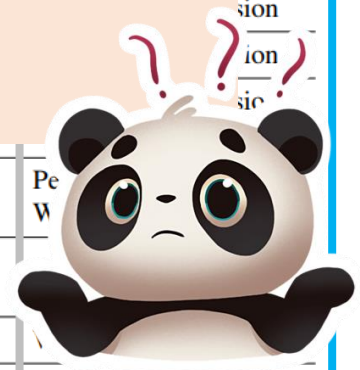
Potential Issues

- Different papers use different evaluation metrics without much justification
- No clear idea what metrics are good or reliable

NeuronEval: A General Framework to Evaluate Neuron Explanations

Metric M	**Study**	**Concept Source** c_t	**Granularity**	**Domain**
Recall	(Zhou et al., 2015)	Crowdsourced	Whole Input	Vision
∼Recall	(Bau et al., 2017; Oikarinen & Weng, 2023) (Oikarinen et al., 2023; Bai et al., 2025)	Crowdsourced	Whole Input	Vision
Precision	(Srinivas et al., 2025)	Generative	Whole Input	Vision
F1-score	(Huang et al., 2023) (Gurnee et al., 2023)	Generative + Model Labeled data	Whole Input Per-token	Language Language
IoU				sion
Accu				sion
∼AU				ion
Inver				sio
Correlation(T&R)	(Bills et al., 2023)	Model	Pe	
Correlation	(Oikarinen & Weng, 2024)	Model	W	
Spearman Correlation(T&R)	(Bricken et al., 2023; Templeton et al., 2024)	Model		
∼WPMI	(Oikarinen & Weng, 2023)	Model		
MAD	(Kopf et al., 2024)	Generative	Whole Input	Vision
∼MAD	(Shaham et al., 2024)	Generative	Whole Input	Vision
	(Singh et al., 2023)	Generative	Whole Input	Language

Which score should we use and trust?



Oikarinen etal, *Evaluating Neuron Explanations: A Unified Framework with Sanity Tests*, ICML 25

Neuron Eval: Sanity Tests

- To address this, a **Sanity Test** is introduced to *evaluate* the **metric M**
  - These are the *necessary* tests for a good evaluation metric M
  - Motivated by the failure of M = Recall and Precision (shown in Toy example in the next slide):
    - Recall favors the *generic* concepts:
      - e.g. “pet” neuron, but “animal” can have same perfect score
    - Precision favors the *specific* concepts:
      - e.g. “pet” neuron, but “dog”, “cat” can have same perfect score

Oikarinen etal, *Evaluating Neuron Explanations: A Unified Framework with Sanity Tests*, ICML 25

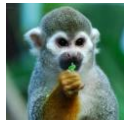
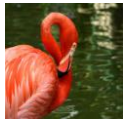
Neuron Eval: Sanity Tests

- Failure example of using Recall/Precision as evaluation metric

Probing data D_{test} | Neuron: pet $B(a_k)$ | c_{dog} | c_{cat} | c_{pet} | c_{animal}





$B(a_k)$	c_{dog}	c_{cat}	c_{pet}	c_{animal}
1	1	0	1	1
1	0	1	1	1
1	1	0	1	1
0	0	0	0	1
0	0	0	0	1
0	0	0	0	1

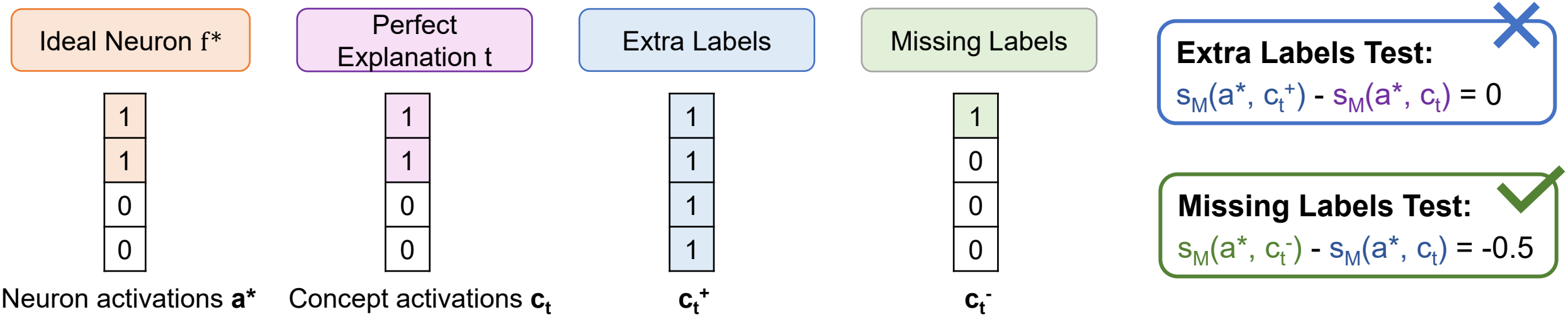
Results of different evaluation metrics:

Concept t	s_{Recall}	$s_{\text{Precision}}$	s_{IoU}
Dog	2/3 = 0.67	2/2 = 1	2/3 = 0.67
Cat	1/3 = 0.33	1/1 = 1	1/3 = 0.33
Pet	**3/3 = 1**	**3/3 = 1**	**3/3 = 1**
Animal	3/3 = 1	3/6 = 0.5	3/6 = 0.5

- *Correct concept* **Pet** gets perfect score with all metrics
- Recall incorrectly gives perfect score to superclass *Animal*
- Precision incorrectly gives perfect score to subclasses *Dog* and *Cat*

Meta-Evaluation: Sanity Tests

- For a reliable metric, it should be able to differentiate between
  - correct concept v.s. too specific concept → missing labels test $[c_t^-]_i = \begin{cases} [c_t]_i & \text{with probability } 0.5 \\ 0 & \text{with probability } 0.5 \end{cases}$
  - correct concept v.s. too general concept → extra labels test $[c_t^+]_i = \begin{cases} 1 & \text{if } [c_t]_i = 1, \text{ else with probability } \frac{||c_t||_1}{n - ||c_t||_1} \\ 0 & \text{otherwise} \end{cases}$
  - i.e. we should observe changes in $\Delta s(k) = \mathbb{E}_{c_{t_k}^{\pm}} [s_{\mathbf{M}}(a_k, c_{t_k}^{\pm}) - s_{\mathbf{M}}(a_k, c_{t_k})]$



Oikarinen etal, *Evaluating Neuron Explanations: A Unified Framework with Sanity Tests*, ICML 25

Sanity test for reliable evaluation metric

- Surprisingly, most existing Evaluation Metrics do not pass these tests ☹
  - Among **18** metrics M (13 existing + 5 new), only **5** (3 existing + 2 new) passed the tests:

Meta-Evaluation #1	(I) Missing Labels Test		(II) Extra Labels Test		Pass
	Experimental	Theoretical	Experimental	Theoretical	
Recall	98.66%	100.00%	0.00%	0.00%	×
Precision	45.73%	0.00%	99.81%	100.00%	×
F1-score	93.68%	100.00%	99.82%	100.00%	✓
IoU	93.62%	100.00%	99.81%	100.00%	✓
Accuracy	23.79%	60.00%	70.37%	69.68%	×
Balanced Accuracy	98.65%	100.00%	53.67%	60.00%	×
Inverse Balanced Accuracy	64.18%	60.00%	99.50%	100.00%	×
AUC	94.96%	100.00%	59.18%	60.00%	×
Inverse AUC	52.81%	60.00%	99.99%	100.00%	×
Correlation	99.41%	100.00%	99.92%	100.00%	✓
Correlation (T&R)	87.83%	100.00%	60.26%	43.64%	×
Spearman Correlation	64.05%	67.20%	49.21%	44.08%	×
Spearman Correlation (T&R)	80.04%	100.00%	59.81%	19.68%	×
Cosine	99.45%	100.00%	99.26%	100.00%	✓
WPMI	95.89%	100.00%	58.84%	100.00%	×
MAD	59.81%	60.00%	99.34%	100.00%	×
AUPRC	95.61%	100.00%	99.46%	100.00%	✓
Inverse AUPRC	99.15 %	100.00%	95.58%	89.54%	×

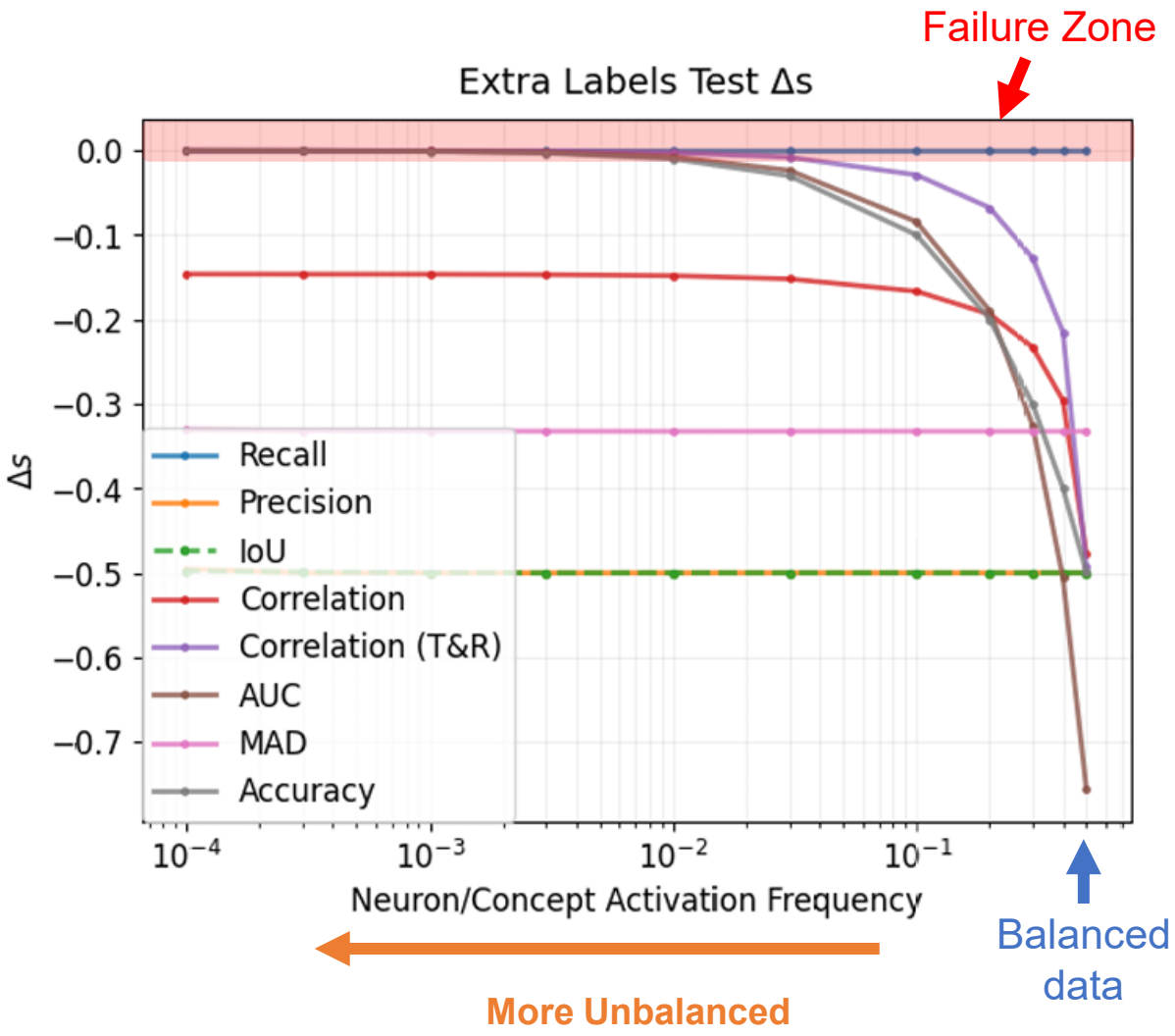
Reliable evaluation metric

F1-score, IoU, Correlation, Cosine (new), AUPRC (new)

☺

Oikarinen etal, *Evaluating Neuron Explanations: A Unified Framework with Sanity Tests*, ICML 25

Insights from the Sanity test results

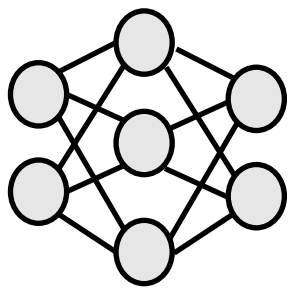


Only Recall fails the Extra label test on balanced data – concept is present 50% of the time

- As data becomes more unbalanced, metrics like AUC, Accuracy, Corr (T&R) starts to fail

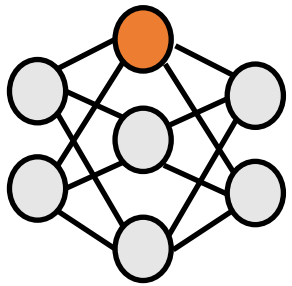
Remaining Challenges

- Human evaluation is still the gold standard to evaluate explanations



Interpretability Tools

CLIP-Dissect
Net-Dissect
Describe-and-Dissect
⋮



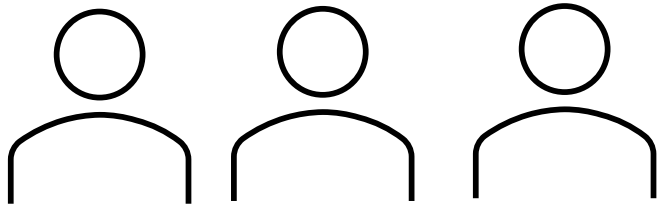
Hippo
Water
Animals in the sea
⋮

$s_M(a_k, c_t)$

Which explanation is better?

1 Should use the reliable metrics M from Neuron-Eval

2



Challenges

- Almost all published evaluations used metrics that failed the sanity checks

NeuronEval: A General Framework to Evaluate Neuron Explanations

Metric M	Study	Concept Source c_t	Granularity	Domain
Recall	(Zhou et al., 2015)	Crowdsourced	Whole Input	Vision
∼Recall	(Bau et al., 2017; Oikarinen & Weng, 2023) (Oikarinen et al., 2023; Bai et al., 2025)	Crowdsourced	Whole Input	Vision
Precision	(Srinivas et al., 2025)	Generative	Whole Input	Vision
F1-score	(Huang et al., 2023)	Generative + Model	Whole Input	Language
	(Gurnee et al., 2023)	Labeled data	Per-token	Language
IoU	(Bau et al., 2017; Mu & Andreas, 2020) (La Rosa et al., 2024)	Labeled data	Per-pixel	Vision
Accuracy	(Koh et al., 2020)	Labeled data	Whole Input	Vision
∼AUC	(Zimmermann et al., 2023)	Crowdsourced	Whole Input	Vision
Inverse AUC	(Bykov et al., 2023)	Labeled data	Whole Input	Vision
	(Kopf et al., 2024)	Generative	Whole Input	Vision
Correlation(T&R)	(Bills et al., 2023)	Model	Per-token	Language
Correlation	(Oikarinen & Weng, 2024)	Model	Whole Input	Vision
Spearman Correlation(T&R)	(Bricken et al., 2023; Templeton et al., 2024)	Model	Per-token	Language
∼WPMI	(Oikarinen & Weng, 2023)	Model	Whole Input	Vision
MAD	(Kopf et al., 2024)	Generative	Whole Input	Vision
∼MAD	(Shaham et al., 2024)	Generative	Whole Input	Vision
	(Singh et al., 2023)	Generative	Whole Input	Language

Existing human studies typically only measure recall, which is not reliable

No existing crowd-sourced study uses passing reliable metrics yet

Challenges

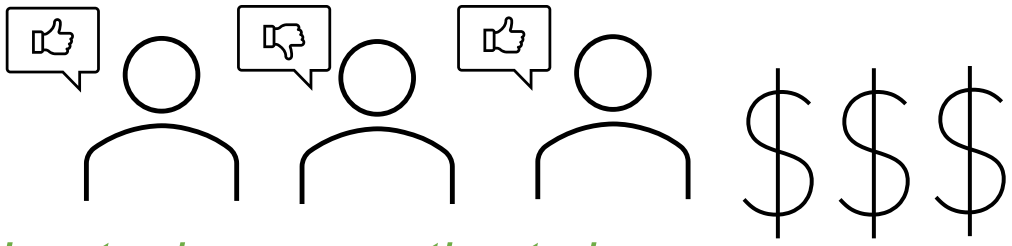
- To use reliable metric (e.g. correlation) to conduct human study, this introduces new challenges

$$s_\rho(a_k, c_t) = \mathrm{E}_{x_i \sim P} \frac{\left([a_k]_i - \mu(a_k)\right) \cdot \left([c_t]_i - \mu(c_t)\right)}{\sigma(a_k)\sigma(c_t)}$$

Challenge 1: Labeling Cost
Labeling all inputs is very expensive.

Can we get accurate estimates with a subset of labels?

Challenge 2: Rating noise
Human raters are often noisy.



How to do aggregation to improve both reliability and cost?

Oikarinen etal, *Beyond Top Activations: Efficient and Reliable Crowdsourced Evaluation of Automated Interpretability. CVPR 2026*

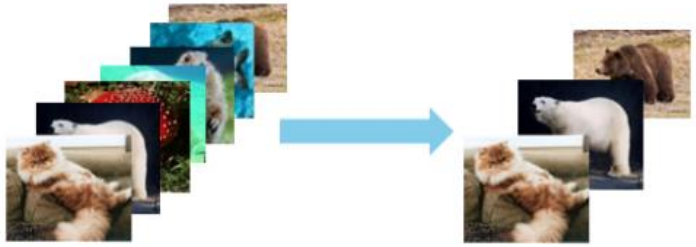
Challenges

Challenge 1: Labeling Cost
Labeling all inputs is very expensive.

- To collect labels for a concept i.e. **ground beetle**, we use this interface and ask user to select all inputs with **ground beetle**
- Issue: To measure correlation, we need labels for every input in the dataset
  - Even if most of them don't have the concept.
- Can we do better?

Solution #1: Model-Guided Importance Sampling

Challenge 1: Labeling Cost
Labeling all inputs is very expensive.



Instead of annotating concepts on the entire dataset, estimate s_ρ from a subset using importance sampling $\mathbb{E}_{x\sim\mathcal{P}}[h(x)] = \mathbb{E}_{x\sim\mathcal{Q}}[\frac{p(x)}{q(x)}h(x)]$

Step 1

Optimal sampling distn Q* that minimize the variance of estimate is

$$q^*(x) \propto |h(x)|p(x)$$
$$\implies q^*(x_i) \propto |\bar{a}_{ki} \cdot \bar{c}_{ti}|$$

Step 2

As label c_{ti} is unknown beforehand, use SigLIP to estimate it (model guidance)

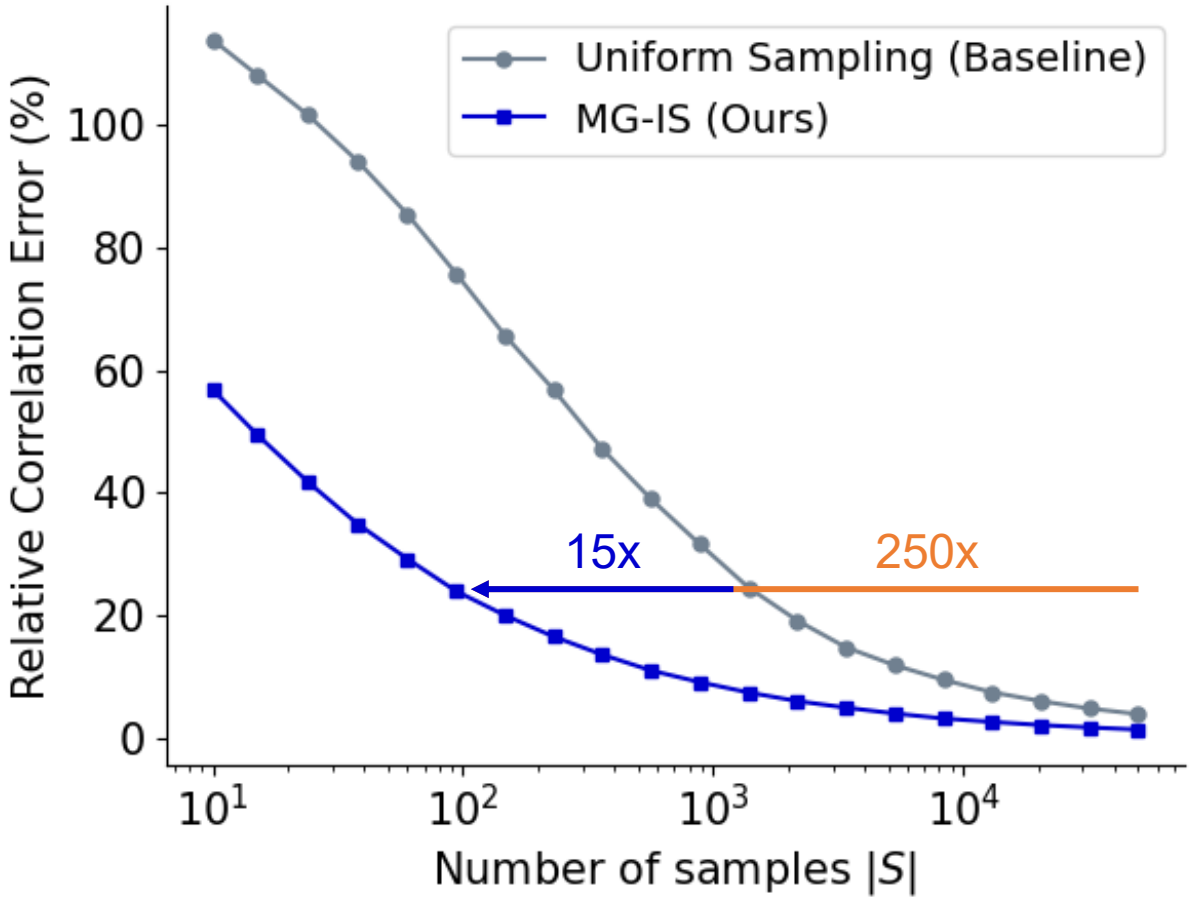
$$q^{siglip}(x_i) = \frac{|[\bar{a}_k]_i \cdot [\bar{c}_t^{siglip}]_i|}{\sum_{x_i\in\mathcal{D}} |[\bar{a}_k]_i \cdot [\bar{c}_t^{siglip}]_i|}$$

Step 3

Finally, mix q^{siglip} with original distribution p, set γ=0.2

$$q^{\text{MG-IS}}(x) = (1-\gamma)q^{siglip}(x) + \gamma p(x)$$
$$p(x) = \frac{1}{|\mathcal{D}|} \; \forall \; x \in \mathcal{D}$$

Solution #1: Model-Guided Importance Sampling



MG-IS reduce # of samples needed to annotate compared to uniform sampling!

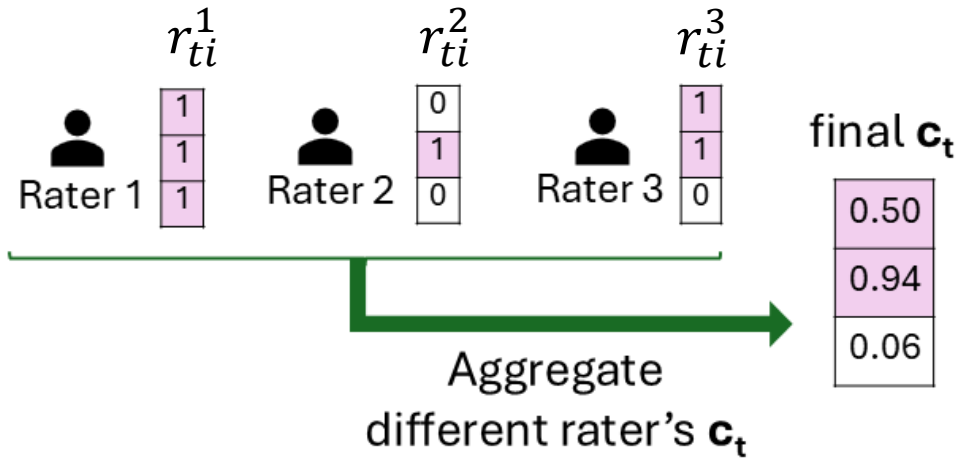
- Leads to 15x reduction in human study cost compare with uniform sampling
- Instead of labelling the full set of 50,000 examples, we can get accurate estimates with subset of ~200 examples

Solution #2: Bayes Rating Aggregation

Challenge 2: Rating noise
Human ratings are often noisy.

Need to collect multiple annotations, what's the best way to combine ratings?

Use Bayesian method for rating aggregation, instead of average rating or majority vote



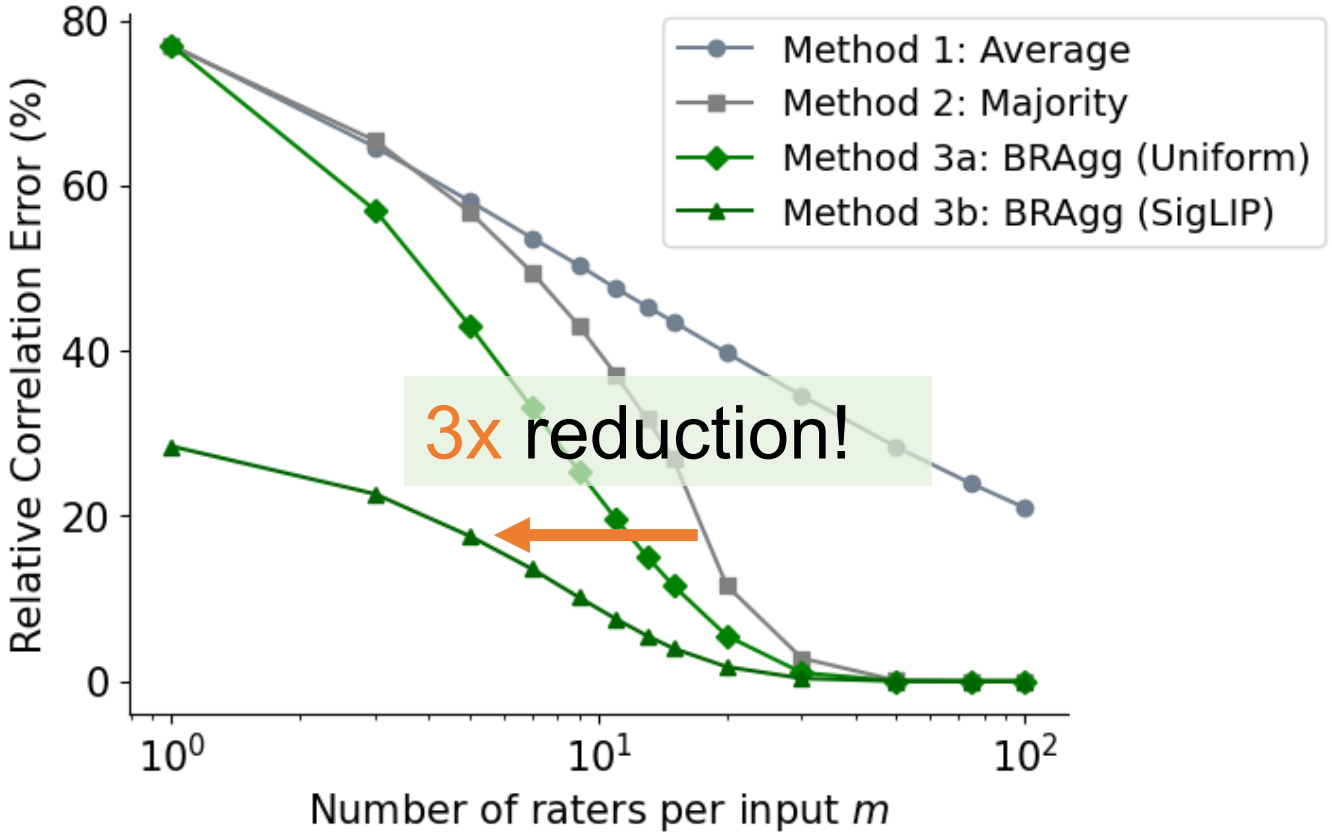
$[c_t]_i = P([c_t^*]_i = 1|R_{ti})$, posterior probability

$$[c_t]_i = \frac{\mathbb{P}(R_{ti}|C_{ti}) \cdot \mathbb{P}(C_{ti})}{\mathbb{P}(R_{ti}|C_{ti}) \cdot \mathbb{P}(C_{ti}) + \mathbb{P}(R_{ti}|\neg C_{ti})\mathbb{P}(\neg C_{ti})}$$

Term 1 : a function of annotation error rate η

Term 2 : confidence of concept exists in input x_i

Solution #2: Bayes Rating Aggregation

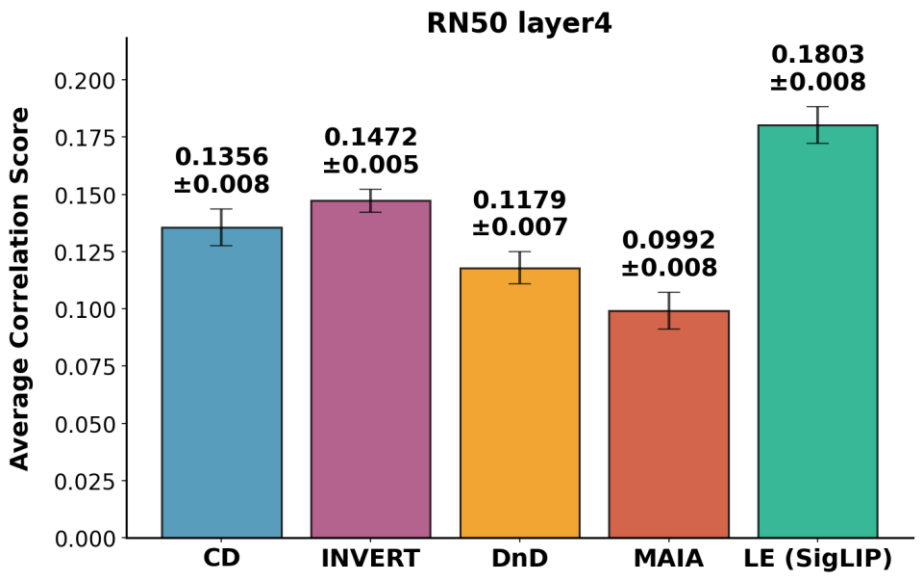


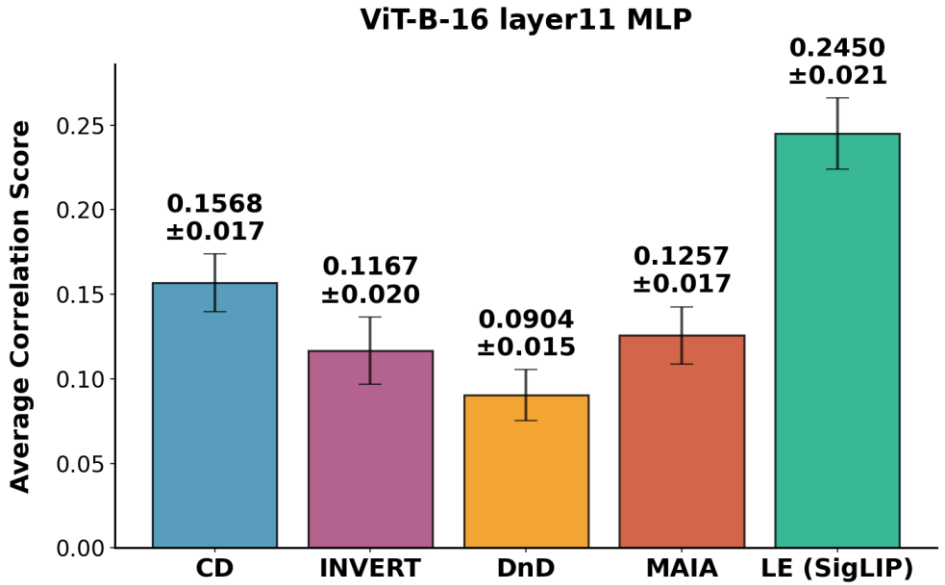
Test 2 version of BRAgg with different methods to calculate prior probability

- BRAgg (SigLIP) reduces number of raters needed by ~3x over standard methods like Majority vote
- Combining BRAgg and MG-IS, it reduce final study cost from **~$90,000 to ~$2160**!

Results: Comparing Neuron Explanation Methods

- Compared existing automated interpretability methods: CLIP-Dissect (CD), INVERT, Describe-and-Dissect (DnD), MAIA, and Linear Explanation (LE)
- Across two different models, we found that Linear Explanations consistently produced the best explanations





Tutorial Part 3: Overview

Rigorous Analysis to Evaluate interpretability

Which explanation is more faithful?

1. Methods to evaluate explanations

Lily

2. DEMO
(neuron-eval & sanity tests for reliable metric)

Tuomas

https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

Demo: Neuron Evaluation

Demo Notebook: demo_evaluation_metrics.ipynb

Code will be released on tutorial website

https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

Coffee Break

(15 mins)

https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

Principled Interpretability Tutorial: Part 4

https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

Lily

Part 1

Fundamentals & backgrounds

How does a deep vision model work?

Lily Ge Akshay

Part 2

Post-hoc Interpretability tools

What knowledge has the model learned?

Lily Tuomas

Part 3

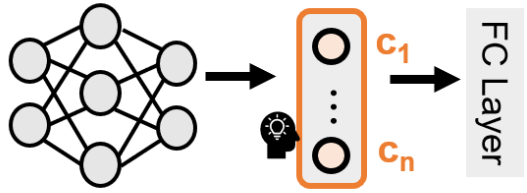
Rigorous Analysis to Evaluate interpretability

Which explanation is more faithful?

Ge Akshay

Part 4
3:15 - 3:55

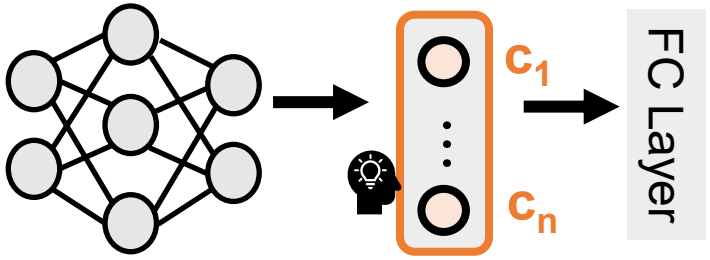
Building Inherently Interpretable DNNs



Embed human-understandable concepts c_i

Tutorial Part 4: Overview

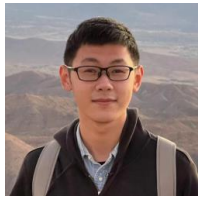
Building Inherently Interpretable DNNs



Embed human-understandable concepts c_i

1. Learning Interpretable DNN for **classification** and **regression**

Ge



2. Learning Interpretable DNN for **generative models**

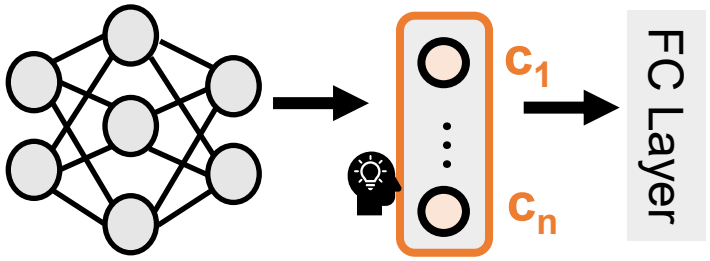
Akshay

https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

Ge

Tutorial Part 4: Overview

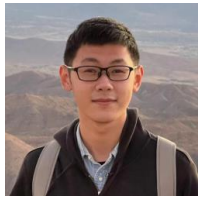
Building Inherently Interpretable DNNs



Embed human-understandable concepts c_i

1. Learning Interpretable DNN for **classification** and **regression**

Ge



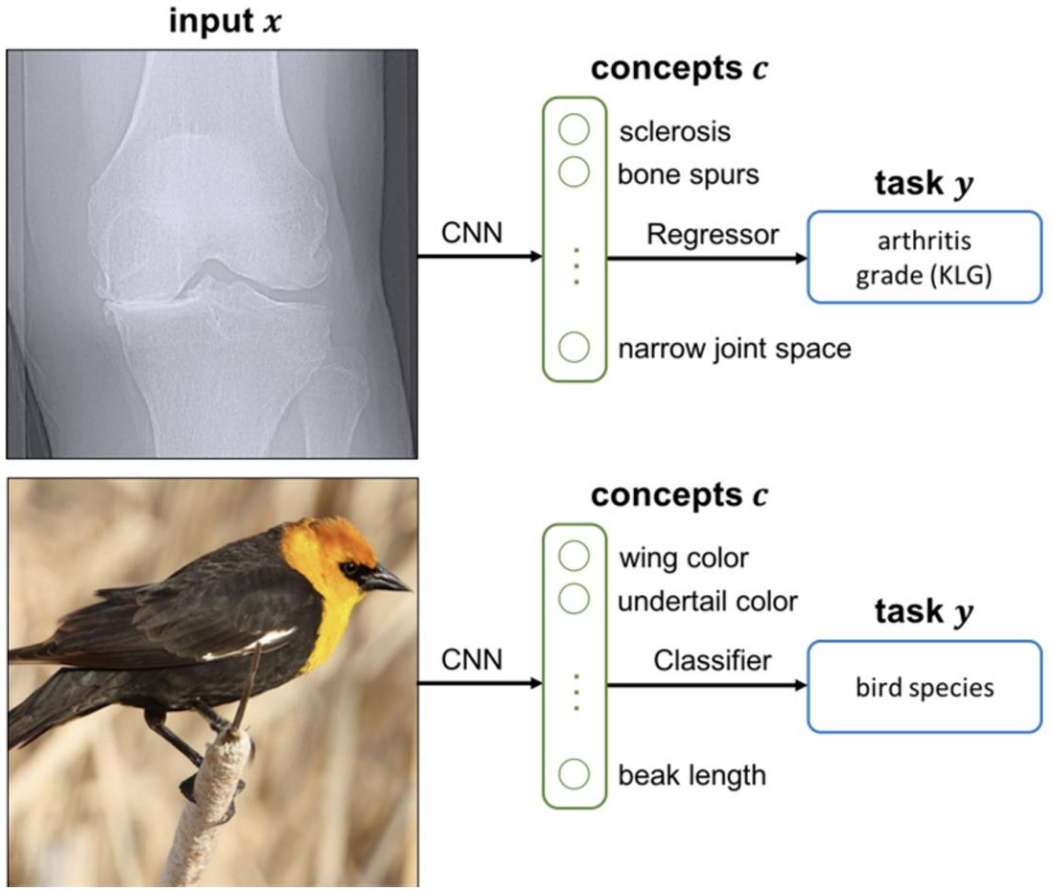
2. Learning Interpretable DNN for **generative models**

Akshay

https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

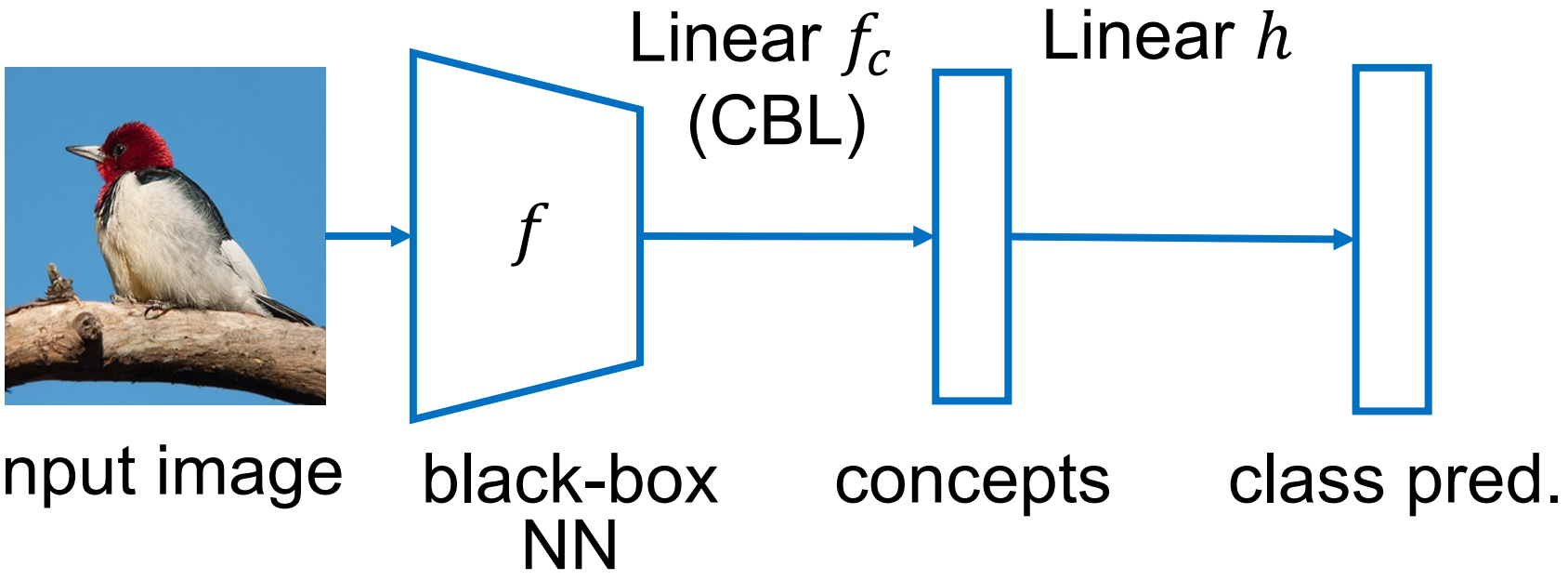
Concept bottleneck for classification

- Concept Bottleneck [1] is a layer where each neuron encodes an interpretable concept.
- Model first predicts the concepts, then predict final label based on the concept.



[1] Koh, et al. "Concept bottleneck models." *ICML 2020*

CBM architecture



① How to decide the candidate concept set?

② How to learn target concepts?

What do we care about CBM? 1. Performance

- Common classification metric, accuracy, does not take interpretability into account.

- Besides normal accuracy, another metric **ANEC** [2] is proposed to measure **model performance under specific sparsity level.**

$$NEC(W_F) = \frac{1}{C}\sum_{i=1}^{C}\sum_{j=1}^{k}\mathbf{1}\{(W_F)_{ij} \neq 0\}$$

# of classes

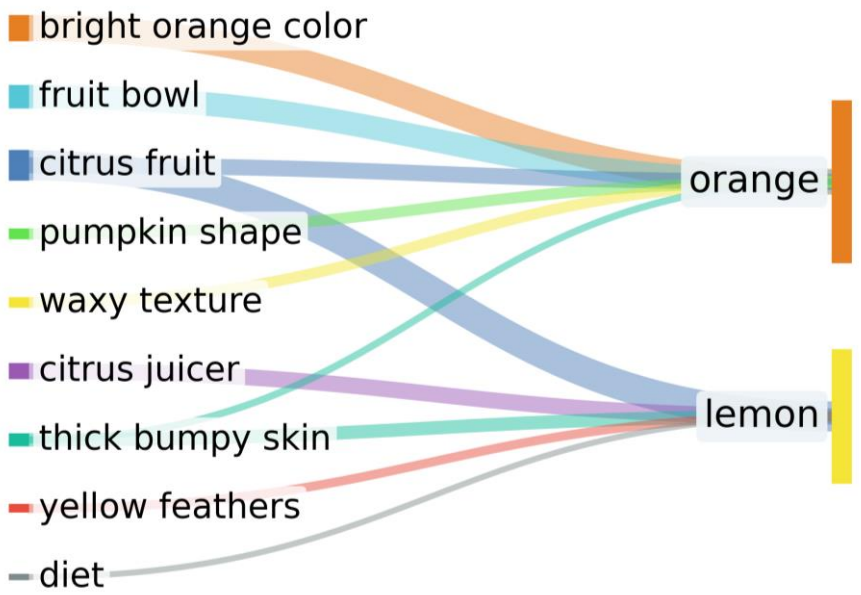
# of concepts

Final classifier weight

[2] Srivastava, et. al. "Vlg-cbm: Training concept bottleneck models with vision-language guidance." *NeurIPS 2024*

What do we care about CBM? 1. Performance

ANEC [2] measures accuracy with **limited "number of effective concepts" (NEC)** used for predicting each class, ensuring the interpretability of final model.



[2] Srivastava, et. al. "Vlg-cbm: Training concept bottleneck models with vision-language guidance." *NeurIPS 2024*

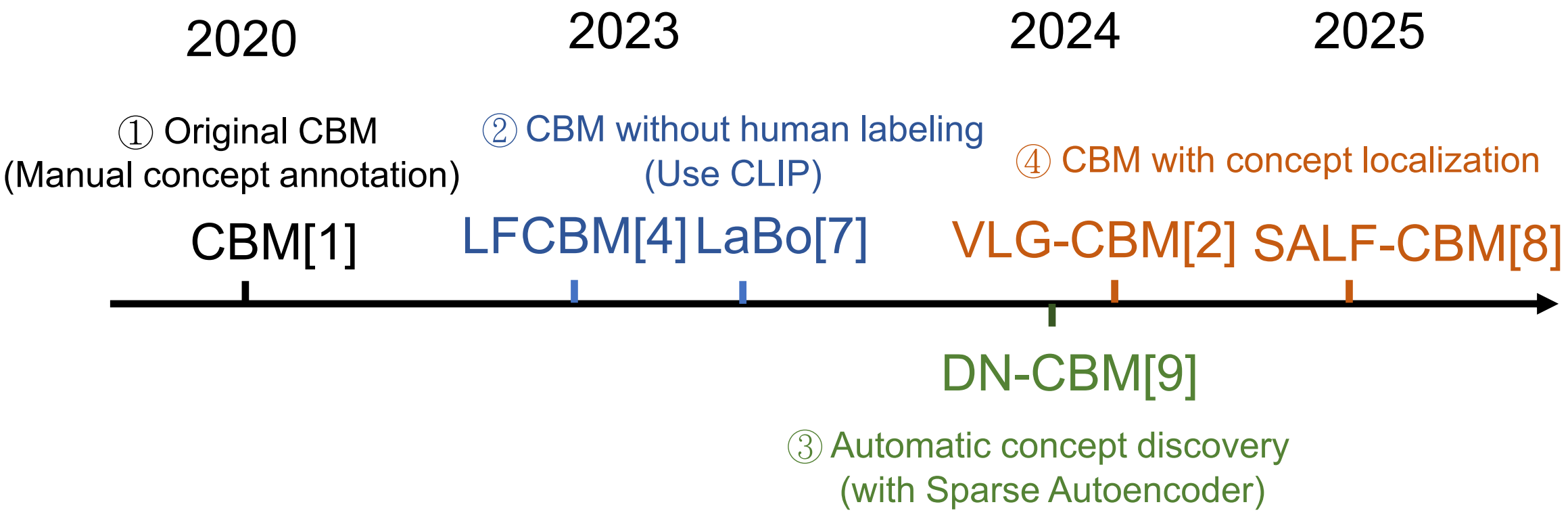
What do we care about CBM? 2. Interpretability

To evaluate interpretability, two key aspects:

1. **Does the CBL neuron faithfully match corresponding concept?**
   - Common methods include manual inspection of top-activate images.
   - More generally, can apply Neuron Evaluation method in part 3, e.g. Neuron Eval [3].
2. **Can we use CBL neuron to intervene model prediction?**
   - Intervene model output by correcting CBL prediction.

[3] Oikarinen, et. al. “Evaluating Neuron Explanations: A Unified Framework with Sanity Checks" *ICML 2025*

CBM timeline



Related works to CBM

- **SCoRe** [Morgado & Vasconcelos, CVPR 2017]
  - Similar idea to CBM that predicts attributes first, then use attributes to do downstream prediction.
  - Mainly focus on zero-shot recognition
- **SENN** [Alvarez-Melis & Jaakkola, NeurIPS 2018]
  - Learn concepts for downstream classification without supervision
  - Use input-dependent weight instead of fixed weights in CBM
- **CEM** [Espinosa Zarlenga et al., NeurIPS 2022]
  - Replace scalar concept bottleneck to concept embedding vector.
  - The goal is to increase the capacity of CBM models, but under NEC evaluation, capacity are similar to CBM.
- **Interpretable-by-design models** [Chattopadhyay et al., TPAMI 2022]
  - Based on information pursuit and their variants to select queries that are sufficient for classifications [Chattopadhyay etal, ICLR 2024; Kolek etal, ICCV 2025; Ge etal, MICCAI 2025]

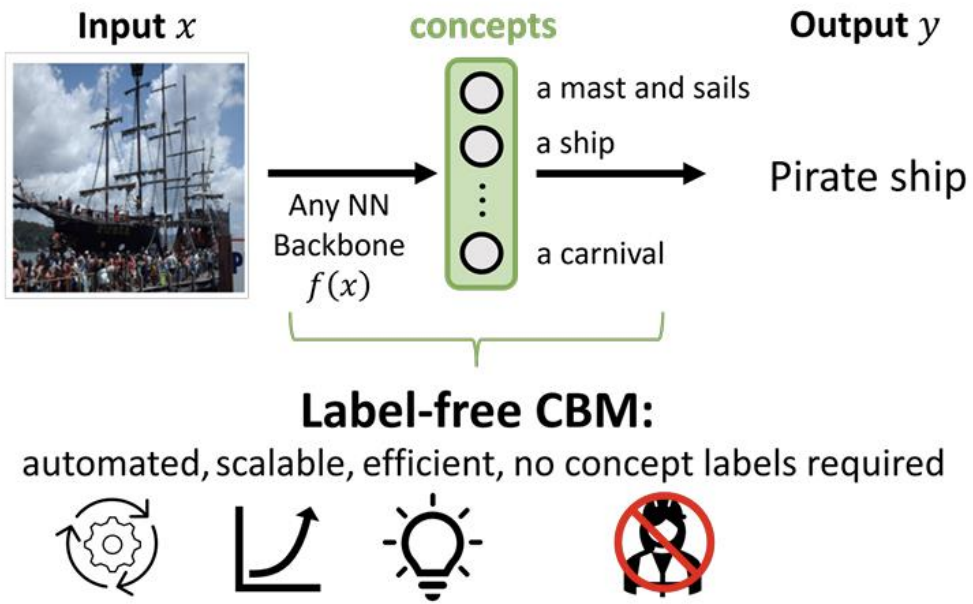
Extension of concept bottleneck model

- Can we train the concept bottleneck without human labeling?
  - LF-CBM (Oikarinen et. al. 2023), LaBo (Yang, et al. 2023)
- Can we ground the concept in the image?
  - VLG-CBM (Srivastava, et. al. 2024), SALF-CBM (Benou, et. al. 2025)
- Can we automatically discover concepts from model?
  - DN-CBM (Rao, et al. 2024)

LFCBM

Automate CBM construction pipeline.

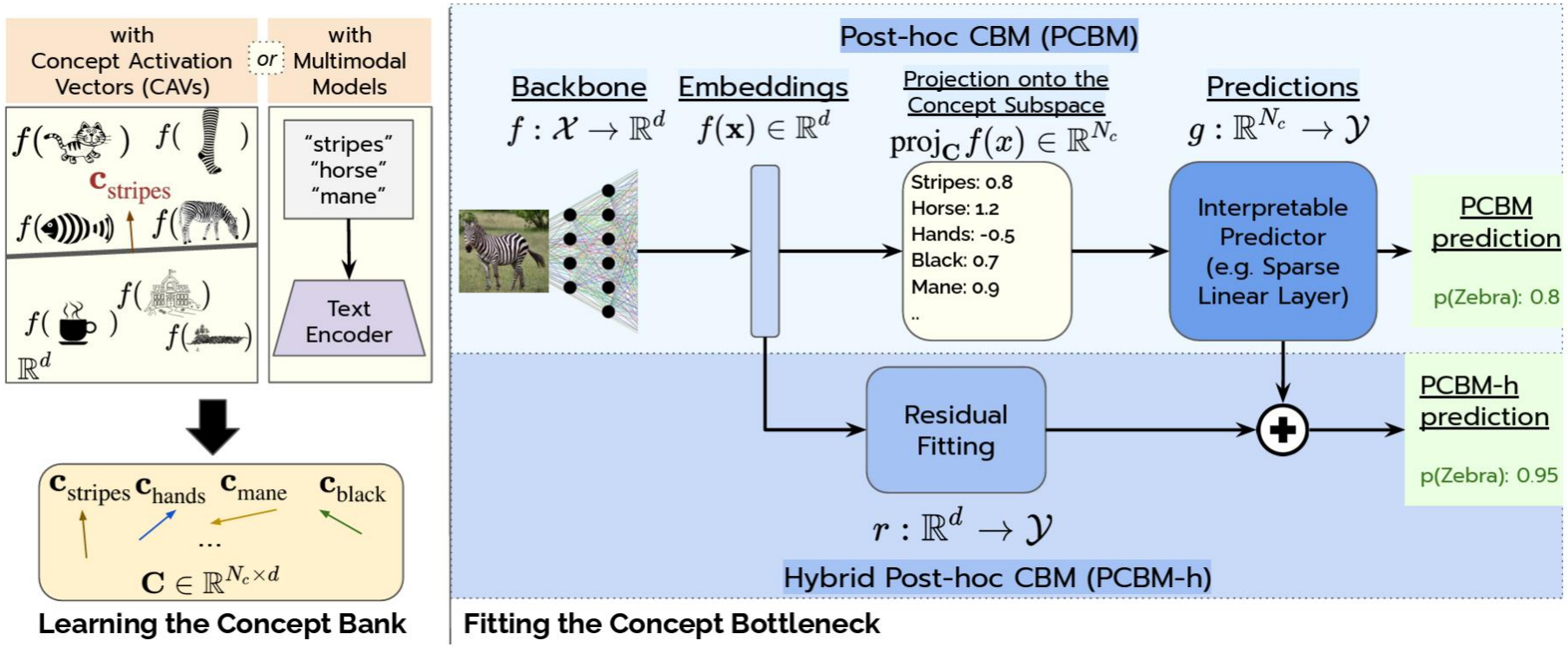
- Construct concept set using LLM.
- Train concept bottleneck from CLIP pseudo-labels (CLIP-Dissect [5]), hence allowing flexible backbone.
- Sparse linear classifier to predict downstream classification label.



[4] Oikarinen et al. "Label-free Concept Bottleneck Models." *ICLR 2023*

[5] Oikarinen et al. “CLIP-Dissect: Automatic Description of Neuron Representations in Deep Vision Networks." *ICLR 2023*

Post-hoc CBM

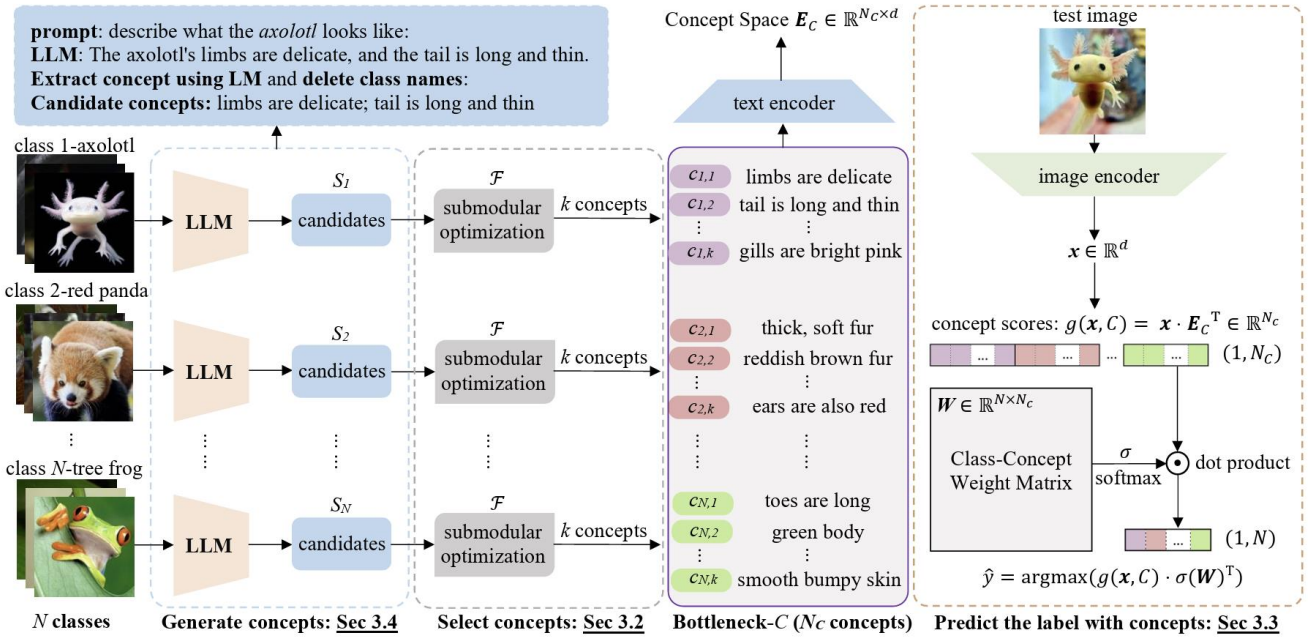


- P-CBM trains concept bottleneck from concept bank (require concept annotation) or multimodal models (CLIP) with backbone as clip encoder
- Introduce hybrid (black-box) residual fitting to improve model performance

[6] Yuksekgonul et al. Post-hoc Concept Bottleneck Models, ICLR 2023

LaBo

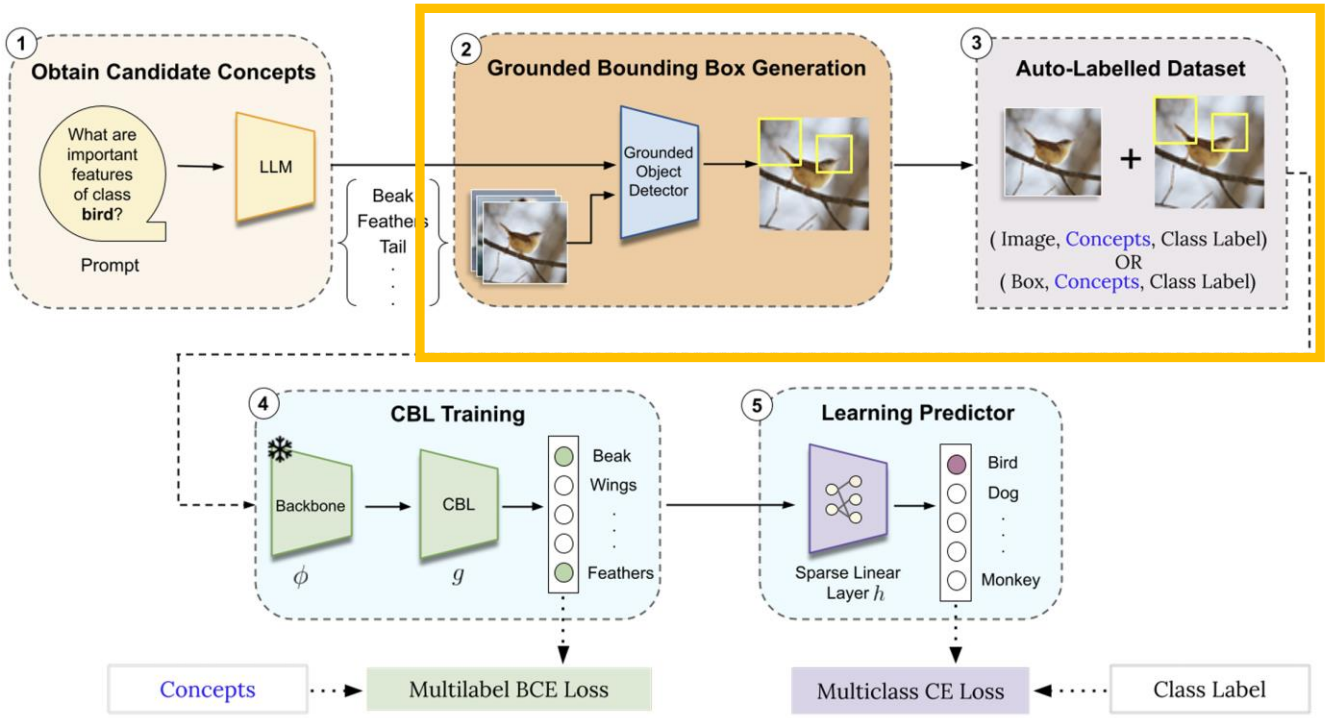
- Generate candidates using LLM, and select concepts with submodular optimization.
- Use CLIP model to get concepts scores for input image.
- Limitation: backbone architecture needs to use CLIP image encoder.



[7] Yang, et al. "Language in a bottle: Language model guided concept bottlenecks for interpretable image classification." *CVPR 2023*

VLG-CBM

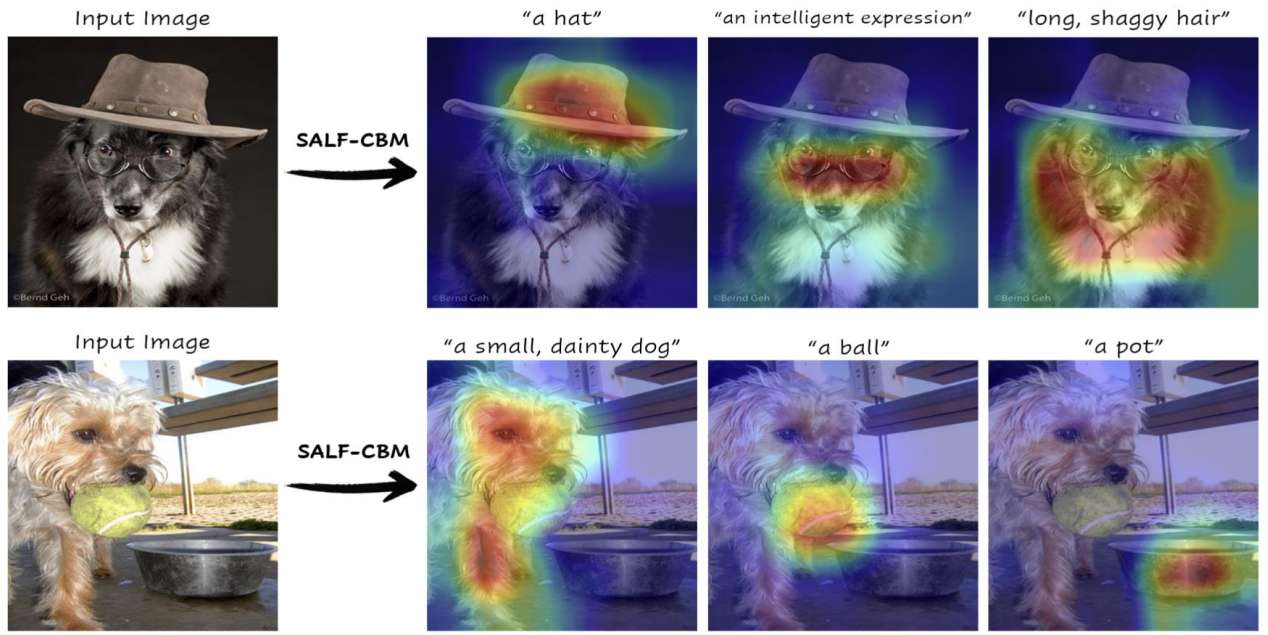
- Use Grounding-Dino to generate bounding box for each concept.
- With bounding box, filter non-visual concepts and augment the dataset.



[2] Srivastava, et. al. "Vlg-cbm: Training concept bottleneck models with vision-language guidance." *NeurIPS 2024*

SALF-CBM

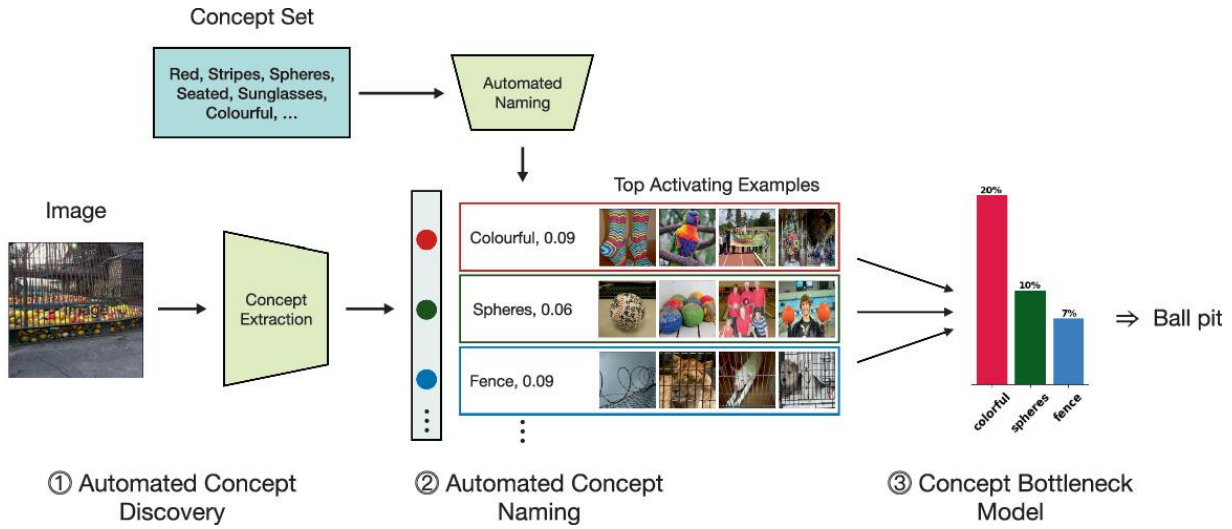
- Extend scalar concept prediction to special concept map prediction from LF-CBM.
- Extract local concept similarity with CLIP, using a red circle to highlight target region.



[8] Benou, et. al. "Show and tell: Visually explainable deep neural nets via spatially-aware concept bottleneck models.“ CVPR 2025.

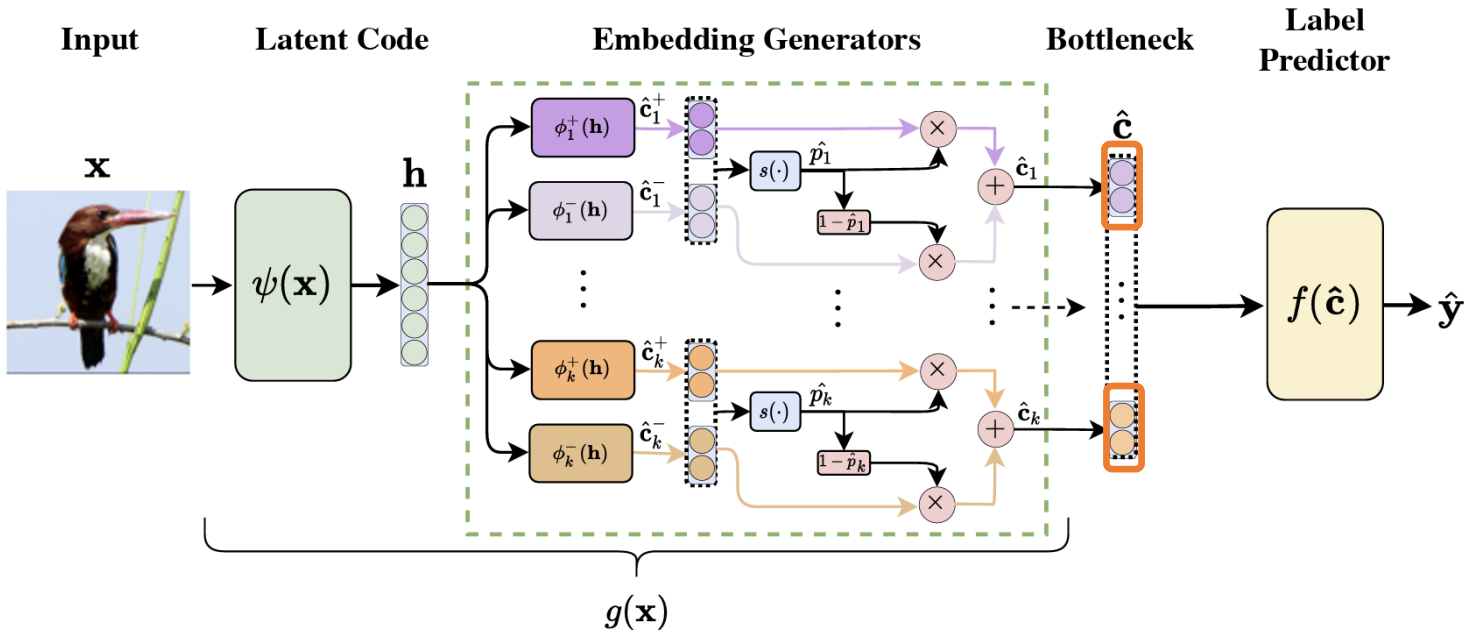
DN-CBM

- Use a SAE to automatically extract concepts.
- Name SAE neurons by comparing the cosine similarity between decoder weight and text.
- Limitation: Require backbone to be CLIP encoder.



[9] Rao, et al. "Discover-then-name: Task-agnostic concept bottlenecks via automated concept discovery." *ECCV* 2024.

CEM



- Replace scalar bottleneck to vector embedding.
- For each concept, two embeddings are learned for active and inactive. A weighted mixture is used for downstream classification, based on predicted activation probability.

[10] Espinosa Zarlenga, et al. "Concept embedding models: Beyond the accuracy-explainability trade-off." *NeurIPS 2022*

More CBMs

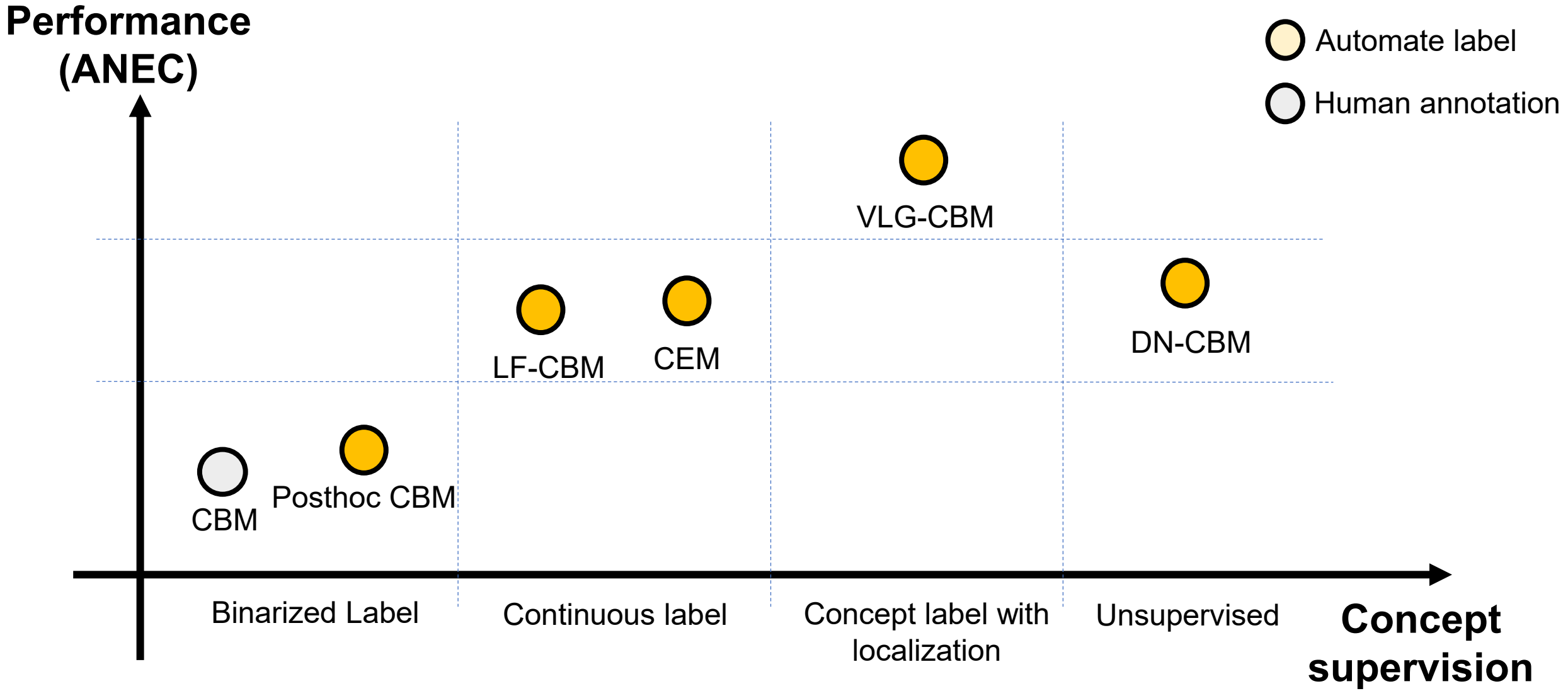
- V2C-CBM (He et al. AAAI 2025)
- DCBM (Prasse et al. ICML 2025)
- PS-CBM (Zhao et al. AAAI 2026)
- M-CBM (De Santis et al. ICLR 2026)
- …

[9] He et al. “V2C-CBM: Building Concept Bottlenecks with Vision-to-Concept Tokenizer,” AAAI, 2025.
[10] Prasse et al. “DCBM: Data-Efficient Visual Concept Bottleneck Models,” ICML, 2025.
[11] Zhao et al. “Partially Shared Concept Bottleneck Models,” AAAI, 2026.
[12] De Santis et al. “Learning Concept Bottleneck Models from Mechanistic Explanations,” ICLR, 2026.

Comparison between CBMs (Classification)

Method	Concept set generation	Concept supervision form	Model architecture
CBM [1]	Manual collected	Binary labels (human-annotated attributes)	CBL at any layer
LF-CBM [4]	LLM-generated	Continuous CLIP similarity	Linear head after backbone
P-CBM [6]	LLM-generated	Binarized CLIP similarity	Linear head after backbone
LaBo [7]	LLM-generated	Continuous CLIP similarity	Linear head after backbone
VLG-CBM [2]	LLM-generated	Binary labels from vision-language grounding	Linear/MLP head after backbone
DN-CBM [9]	Auto discovery by SAE	N/A	SAE on backbone representation

Comparison between CBMs (Classification)



References

[1] Koh et al. "Concept bottleneck models." *ICML 2020*

[2] Srivastava et. al. "Vlg-cbm: Training concept bottleneck models with vision-language guidance." *NeurIPS 2024*

[3] Oikarinen et. al. “Evaluating Neuron Explanations: A Unified Framework with Sanity Checks" *ICML 2025*

[4] Oikarinen et al. "Label-free Concept Bottleneck Models." *ICLR 2023*

[5] Oikarinen et. al. “CLIP-Dissect: Automatic Description of Neuron Representations in Deep Vision Networks." *ICLR 2023*

[6] Yuksekgonul et al. Post-hoc Concept Bottleneck Models, *ICLR 2023*

[7] Yang et al. "Language in a bottle: Language model guided concept bottlenecks for interpretable image classification." *CVPR 2023*

[8] Benou et. al. "Show and tell: Visually explainable deep neural nets via spatially-aware concept bottleneck models.“ *CVPR 2025*

[9] Rao et al. "Discover-then-name: Task-agnostic concept bottlenecks via automated concept discovery." *ECCV* 2024

[10] Espinosa Zarlenga et al. "Concept embedding models: Beyond the accuracy-explainability trade-off." *NeurIPS 2022*

[11] He et al. “V2C-CBM: Building Concept Bottlenecks with Vision-to-Concept Tokenizer,” *AAAI 2025*

[12] Prasse et al. “DCBM: Data-Efficient Visual Concept Bottleneck Models,” *ICML 2025*

[13] Zhao et al. “Partially Shared Concept Bottleneck Models,” *AAAI 2026*

[14] De Santis et al. “Learning Concept Bottleneck Models from Mechanistic Explanations,” *ICLR 2026*

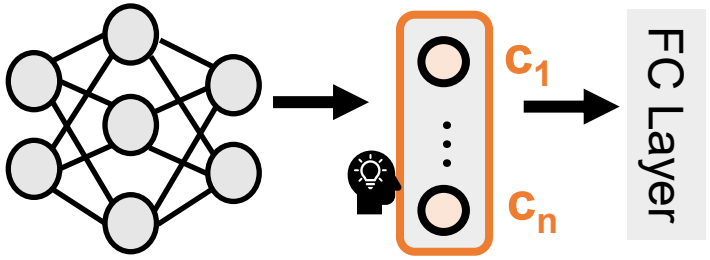
[15] Chattopadhyay etal, “Bootstrapping Variational Information Pursuit with Large Language and Vision Models for Interpretable Image Classification”, *ICLR 2024*

[16] Kolek etal, “Learning Interpretable Queries for Explainable Image Classification with Information Pursuit”, *ICCV 2025*

[17] Ge etal, “IP-CRR: Information Pursuit for Interpretable Classification of Chest Radiology Reports”, *MICCAI 2025*

Tutorial Part 4: DEMO TIME

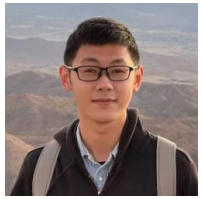
Building Inherently Interpretable DNNs



Embed human-understandable concepts c_i

1. Learning Interpretable DNN for **classification** and **regression**

Ge



2. Learning Interpretable DNN for **generative models**

Akshay

https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

Concept explanation examples.

Concept explanations

GT: 001.Black_footed_Albatross

LF-CBM
→ 001.Black_footed_Alb ✓

1. three toes on each foo (+4.924)
2. a small, green body (+1.806)
3. a purple-red head and (+1.698)
4. a bright orange breast (-1.667)
5. large feet (-1.047)

VLG-CBM
→ 001.Black_footed_Alb ✓

1. dark wingtips (+4.465)
2. black feet (+3.585)
3. a yellow beak (+3.120)
4. yellow bill (+1.493)
5. white body (+1.014)

CEM
→ 001.Black_footed_Alb ✓

1. a long, pointed wings (+0.034)
2. a pink bill (+0.030)
3. a white wingbar (+0.023)
4. dark wingtips (+0.023)
5. white underwings (+0.022)

Original CBM
→ 001.Black_footed_Alb ✓

1. has shape swallow-like (+0.189)
2. has crown color grey (+0.173)
3. has nape color brown (+0.169)
4. has forehead color buf (+0.144)
5. has wing pattern solid (+0.140)

LaBo
→ 003.Sooty_Albatross ✗

1. a raucous voice (+2.218)
2. a large blue-grey bird (+2.144)
3. a loud cawing sound (+1.907)
4. a black crest on the h (+1.139)
5. a red crest on the hea (+1.101)

GT: 020.Yellow_breasted_Chat

LF-CBM
→ 020.Yellow_breasted_ ✓

1. a dark blue-black uppe (+2.920)
2. large feet (-1.937)
3. iridescent blue-green (+1.720)
4. Glossy black wings (+1.716)
5. a red head (+1.308)

VLG-CBM
→ 020.Yellow_breasted_ ✓

1. a black patch on the b (+3.995)
2. a black tail (+2.516)
3. a black throat (+2.296)
4. yellowish legs (+0.461)
5. greenish-yellow back a (+0.440)

CEM
→ 020.Yellow_breasted_ ✓

1. a small, dark body (+0.036)
2. a black cap and back (+0.035)
3. a black head and upper (+0.030)
4. a seabird (+0.030)
5. three toes on each foo (+0.024)

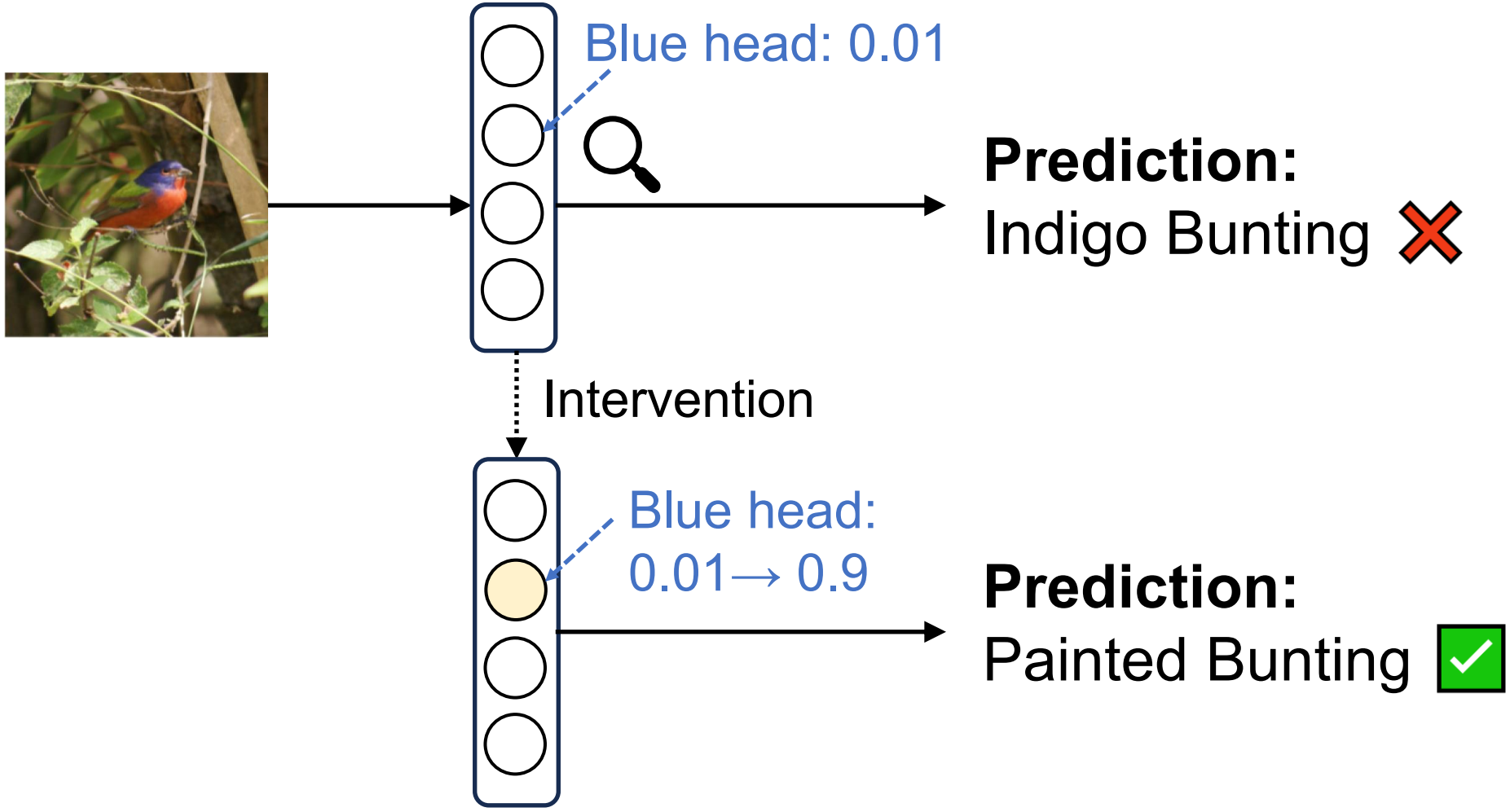
Original CBM
→ 161.Blue_winged_Warb ✗

1. has upper tail color g (+0.129)
2. has shape perching-lik (+0.122)
3. has shape upright-perc (+0.110)
4. has forehead color gre (+0.107)
5. has bill color grey (+0.102)

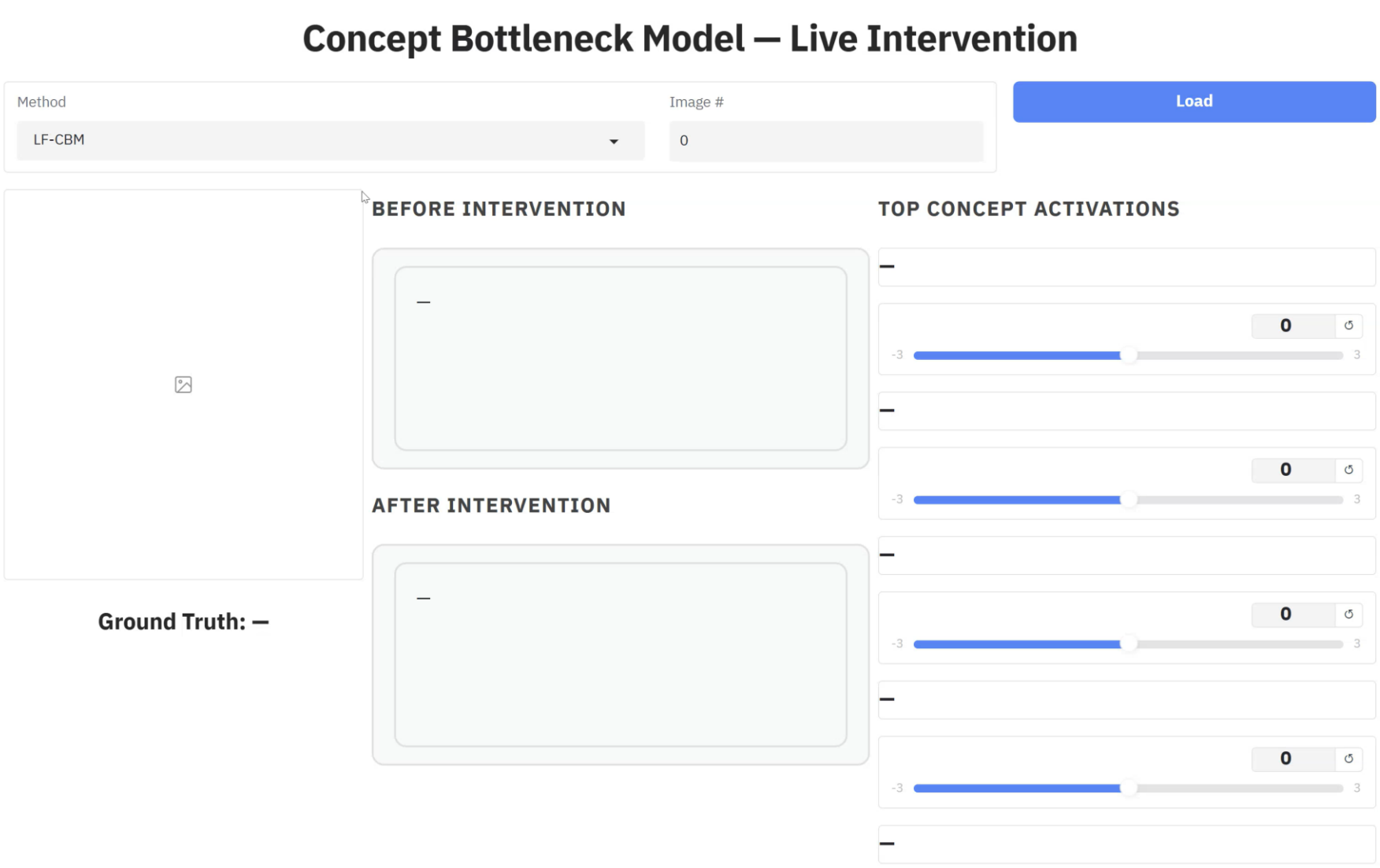
LaBo
→ 020.Yellow_breasted_ ✓

1. a buffy breast (+3.364)
2. a brown and white colo (+2.857)
3. a black eye-stripe (+1.930)
4. a black bill with a ye (+1.919)
5. a red head (+1.495)

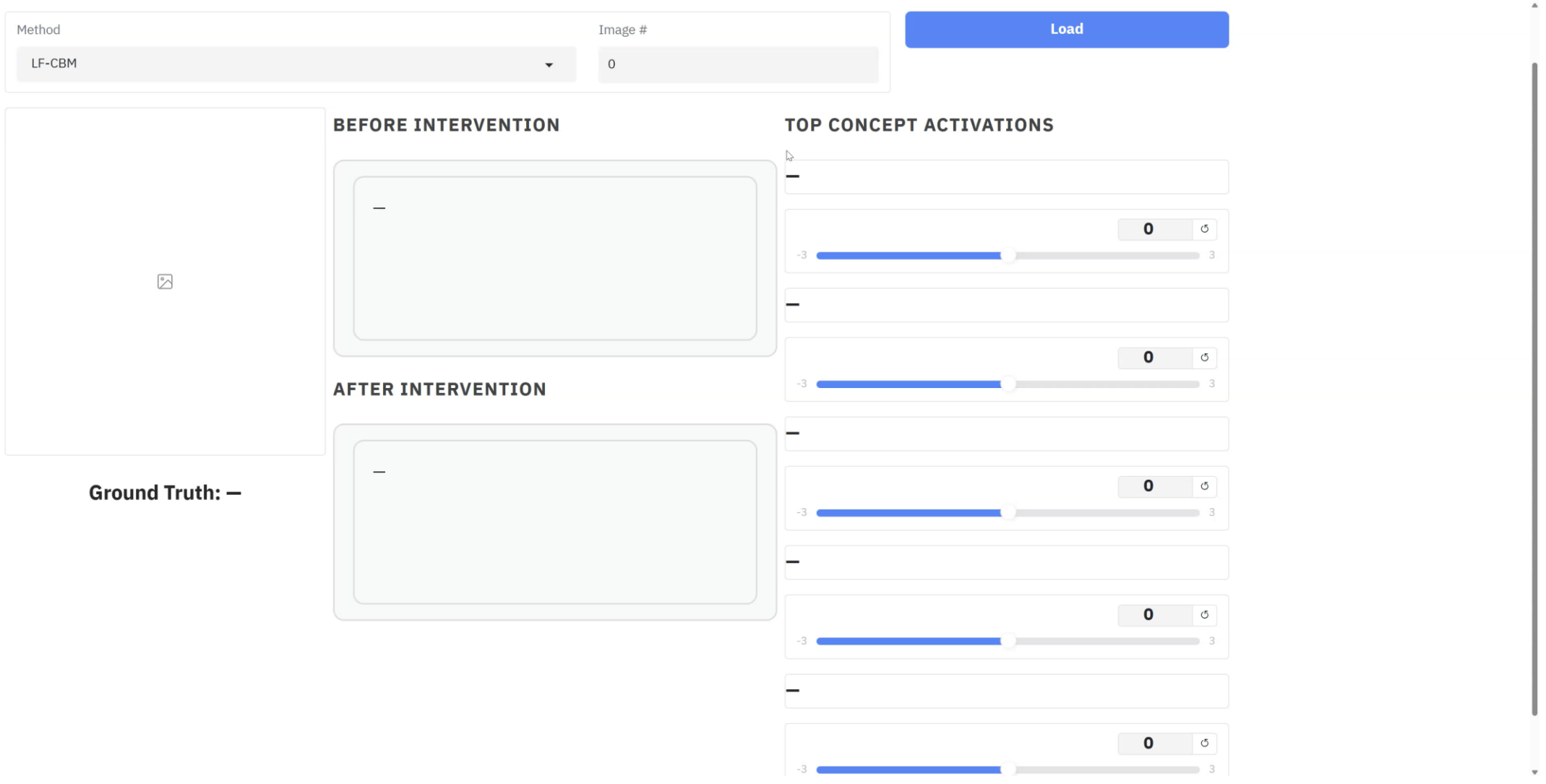
CBM intervention



Demo – CBM Intervention #1

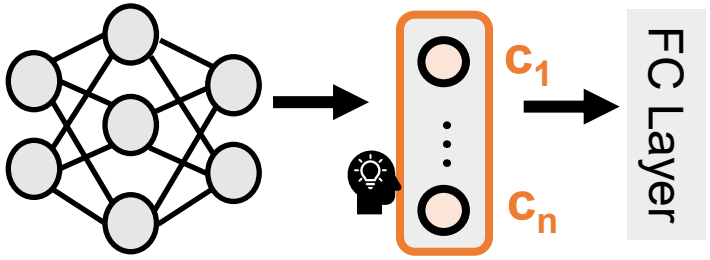


Demo – CBM Intervention #2



Tutorial Part 4: Overview

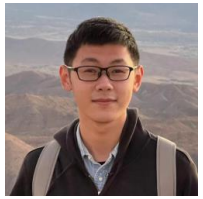
Building Inherently Interpretable DNNs



Embed human-understandable concepts c_i

1. Learning Interpretable DNN for **classification** and **regression**

Ge



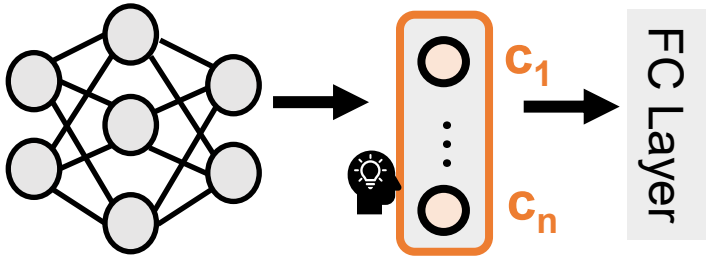
2. Learning Interpretable DNN for **generative models**

Akshay

https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

Tutorial Part 4: Overview

Building Inherently Interpretable DNNs



Embed human-understandable concepts c_i

1. Learning Interpretable DNN for **classification** and **regression**

Ge

2. Learning Interpretable DNN for **generative models**

Akshay

https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

Timeline: Interpretable Generative Models

2016-20

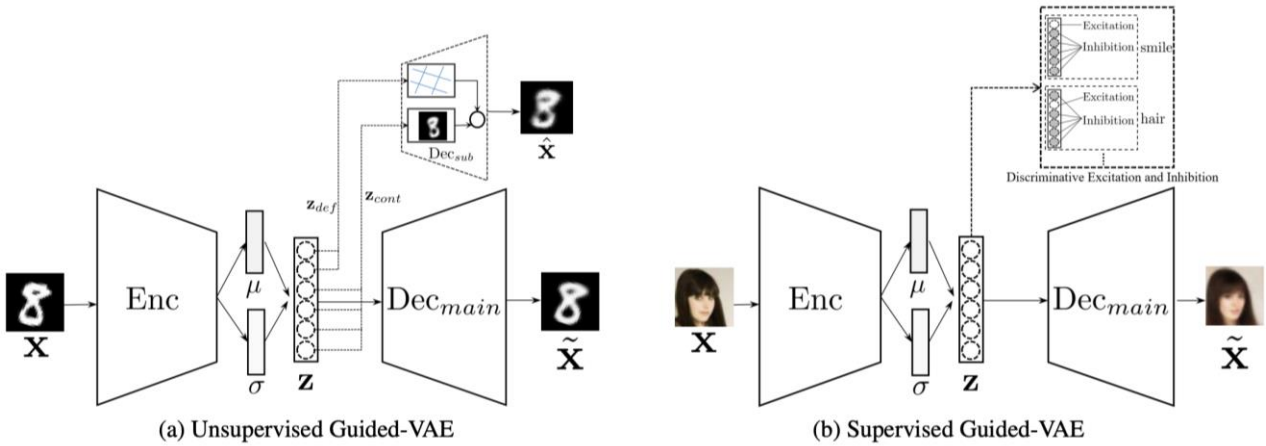
① Disentangled Representation Learning[1-4]

β-VAE, InfoGAN

[1] Higgins et al., "β-VAE", ICLR17; [2] Chen et al. "Isolating Sources of Disentanglement in VAEs", NeurIPS18
[3] Chen et al., "InfoGAN", NeurIPS16; [4] Ding et al., "Guided Variational Autoencoder for Disentanglement Learning", CVPR20

Prior work on Interpretable Generative Models

- Disentangled representation learning with GANs/VAEs
  - β-VAE and follow-up works [1, 2], InfoGAN[3] → unsupervised discovery
  - Guided-VAE[4] → follow-up to β-VAE including supervised concept learning
  - Limitations:
    - Restricted to single family of generative models (either VAE or GAN)
    - Need to train the generative model from scratch (i.e. expensive for larger models)



(a) Unsupervised Guided-VAE (b) Supervised Guided-VAE

[1] Higgins et al., "β-VAE", ICLR17; [2] Chen et al. "Isolating Sources of Disentanglement in VAEs", NeurIPS18
[3] Chen et al., "InfoGAN", NeurIPS16; [4] Ding et al., "Guided Variational Autoencoder for Disentanglement Learning", CVPR20

Timeline: Interpretable Generative Models

2016-20

① Disentangled Representation Learning[1-4]

β-VAE, InfoGAN

2019-20

② Post-hoc Interpretability[5, 6]

GAN-Dissect, -Rewriting

Limitations:

Unsupervised Discovery,
Specific to GAN/VAE family

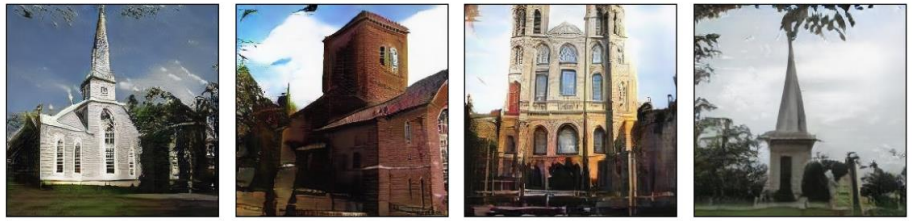
[1] Higgins et al., "β-VAE", ICLR17; [2] Chen et al. "Isolating Sources of Disentanglement in VAEs", NeurIPS18
[3] Chen et al., "InfoGAN", NeurIPS16; [4] Ding et al., "Guided Variational Autoencoder for Disentanglement Learning", CVPR20
[5] Bau et al., "GAN Dissection", ICLR19; [6] Bau et al., "Rewriting a Deep Generative Model", ECCV20

Prior work on Interpretable Generative Models

- Post-hoc interpretability in GANs
  - GAN-Dissect [5] uses neuron-interpretability on GANs
    - By intervening on neurons related to desired concepts, generated images can be edited
  - [6] performs rank-1 edits on GAN weights to learn/unlearn concepts
  - Limitations: post-hoc methods and model itself is not inherently interpretable

(a) Generate images of churches

(b) Identify GAN units that match trees

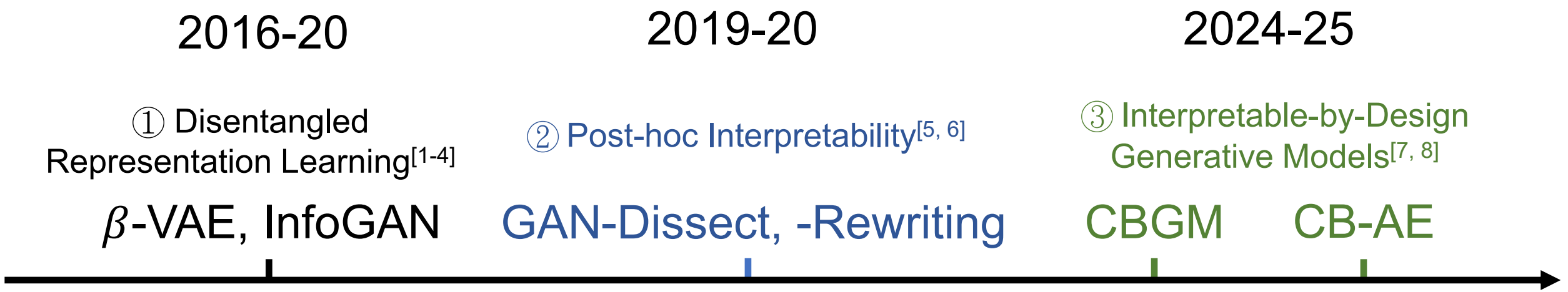


(c) Ablating units removes trees

(d) Activating units adds trees

[5] Bau et al., “GAN Dissection: Visualizing and understanding generative adversarial networks”, ICLR19
[6] Bau et al., “Rewriting a Deep Generative Model”, ECCV20

Timeline: Interpretable Generative Models



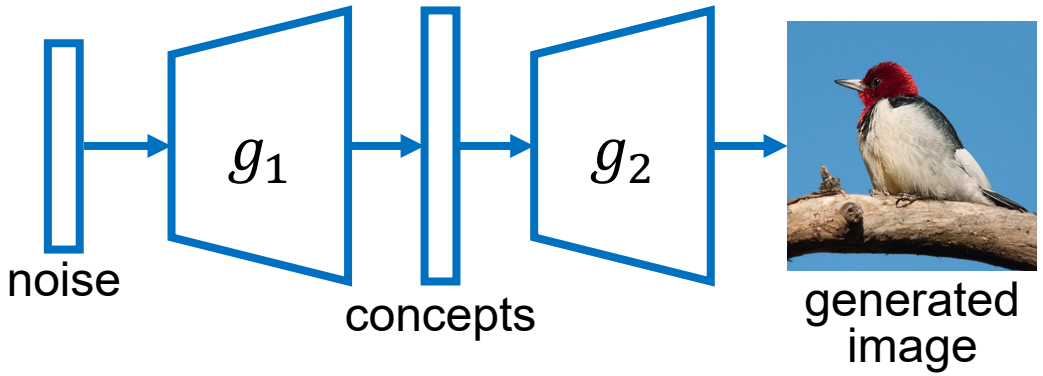
Limitations:
- Unsupervised Discovery,
- Specific to GAN/VAE family

Limitations:
- Only post-hoc editing,
- Not interpretable-by-design

[1] Higgins et al., "β-VAE", ICLR17; [2] Chen et al. "Isolating Sources of Disentanglement in VAEs", NeurIPS18
[3] Chen et al., "InfoGAN", NeurIPS16; [4] Ding et al., "Guided Variational Autoencoder for Disentanglement Learning", CVPR20
[5] Bau et al., "GAN Dissection", ICLR19; [6] Bau et al., "Rewriting a Deep Generative Model", ECCV20
[7] Ismail et al., "Concept Bottleneck Generative Models", ICLR24; [8] Kulkarni et al., "Interpretable Generative Models through Post-Hoc CBs", CVPR25

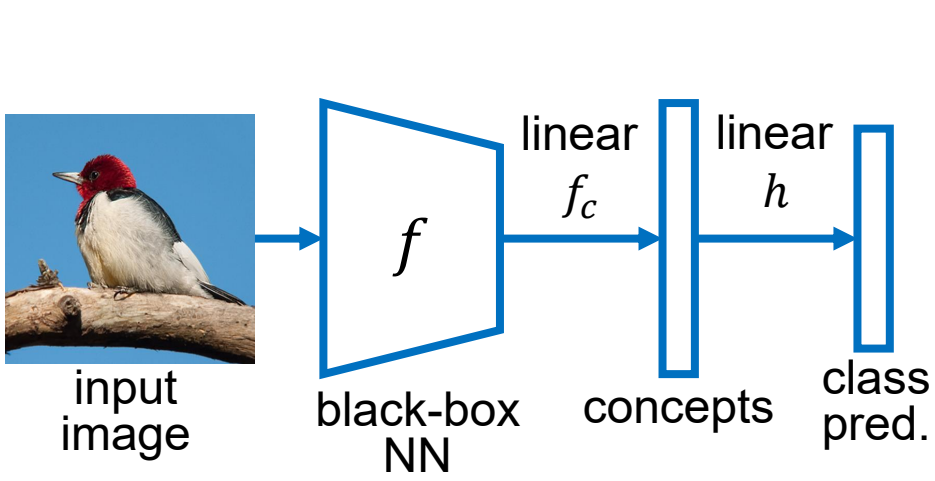
Why Generative CBMs?

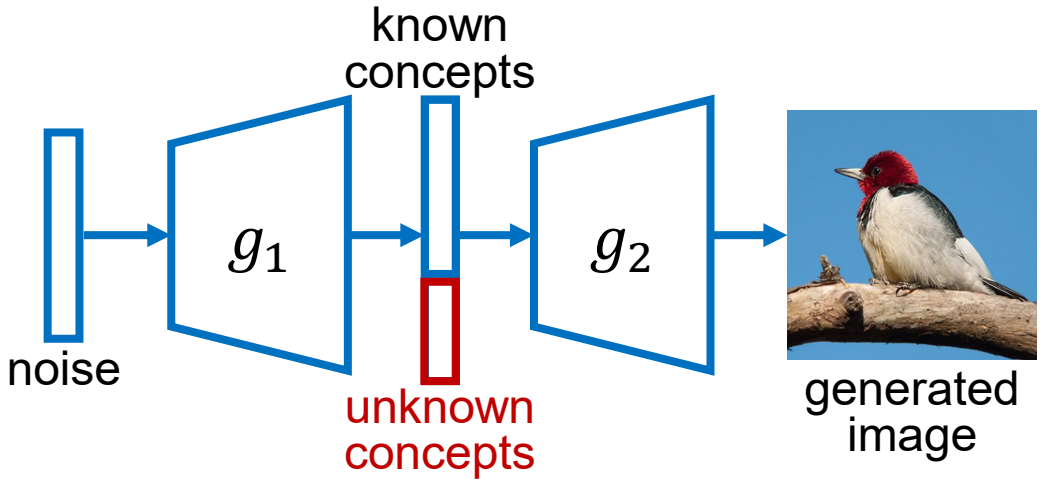
- Interpretability
  - Find which concepts are used during generation
  - Usecase: Monitor harmful or biased concepts during generation
- Steerability
  - Control generation w.r.t. human-interpretable concepts
  - Usecase: Fine-grained control on generated images



Challenges in Generative CBMs

- Differences compared to classification CBMs
  - Single linear layer based on concepts is usually insufficient for generation
    - Solution: Concept bottleneck between two deep models
  - Too few concepts may cause mode collapse (low diversity) in generation
    - Solution: Encode unknown concepts as additional information
  - Steering is more difficult with non-linear mapping of concepts → image
    - Solution: Explicitly include training objectives for steerable generation

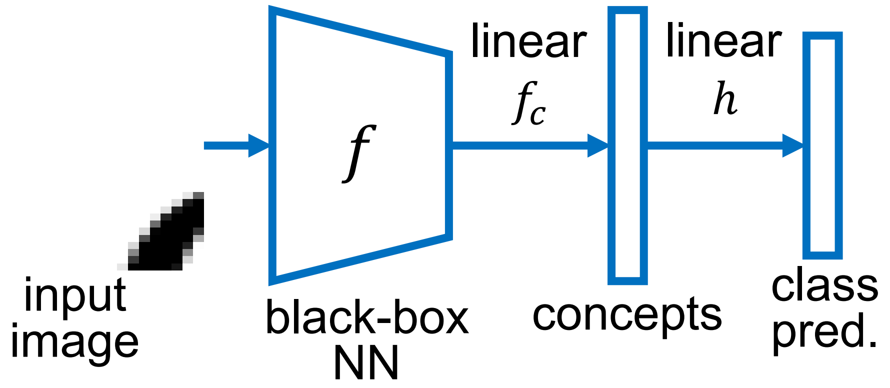


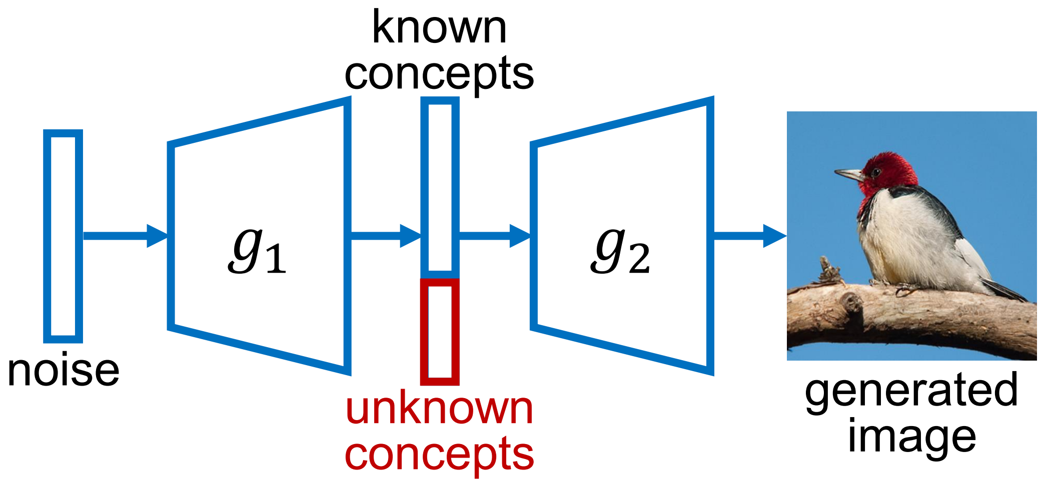


Challenges in Generative CBMs

- Differences compared to classification CBMs
  - Single linear layer based on concepts is usually insufficient for generation
    - Solution: Concept bottleneck between two deep models
  - Steering is more difficult with non-linear mapping of concepts → image
    - Solution: Explicitly include training objectives for steerable generation

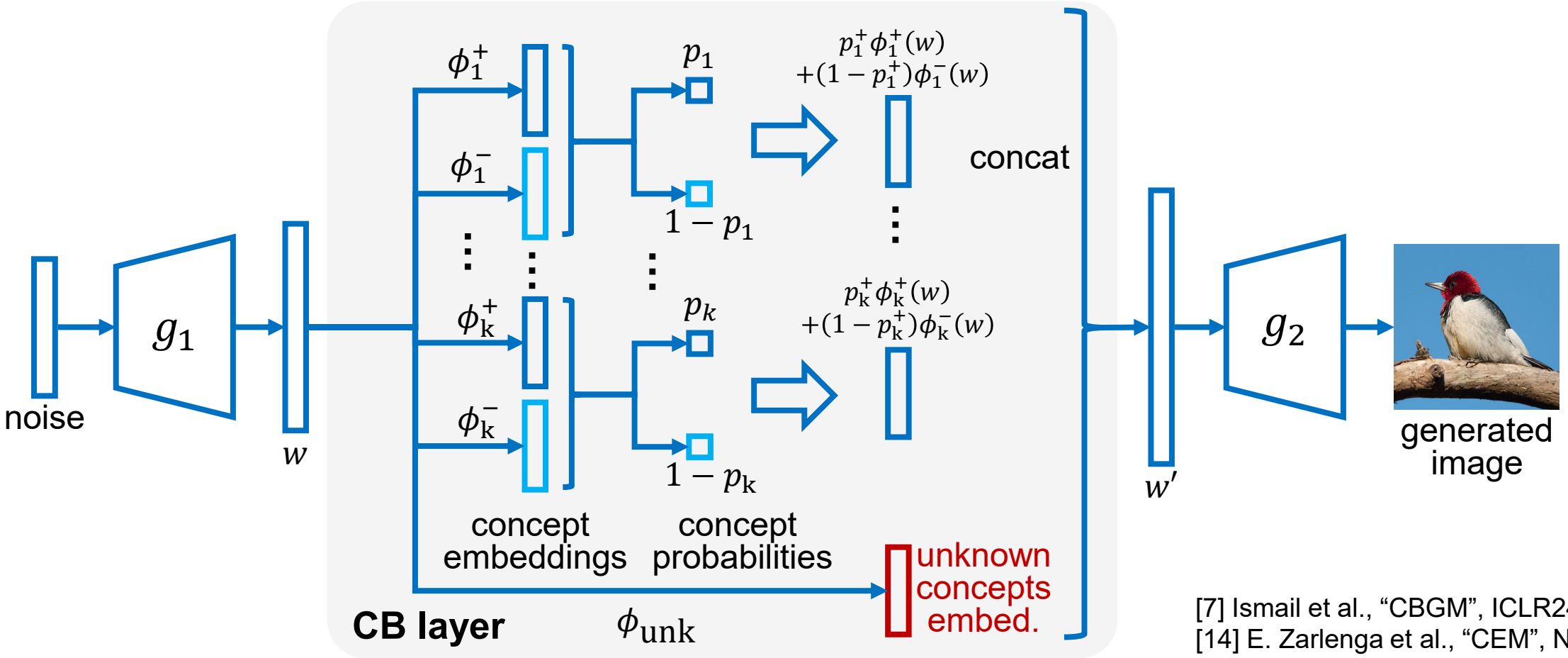
First two works to address this problem:
CBGM (ICLR 2024) and CB-AE (CVPR 2025)





Concept Bottleneck Generative Models (ICLR24)

- Extends Concept Embedding Models[14] to image generation



[7] Ismail et al., “CBGM”, ICLR24
[14] E. Zarlenga et al., “CEM”, NeurIPS22

Concept Bottleneck Generative Models (ICLR24)

- CBGM training
  - Train generative model + CBL from scratch
  - Training objectives:
    - Task loss → generative model objectives (e.g. GAN or VAE or diffusion model losses)
    - Concept loss → BCE loss b/w concept probabilities and ground-truth concepts
    - Orthogonality loss → minimize similarity b/w known and unknown concept embeddings

$$\min_{g_1, g_2, CBL} \left[\mathcal{L}_{task} + \mathcal{L}_{con} + \mathcal{L}_{orth}\right]$$

$$\mathcal{L}_{con,i} = y_i \log p_i + (1 - y_i) \log(1 - p_i)$$

$$\mathcal{L}_{orth} = \sum_{i=1}^{k} \langle w'_i, w'_{unk} \rangle$$

[7] Ismail et al., “Concept Bottleneck Generative Models”, ICLR24

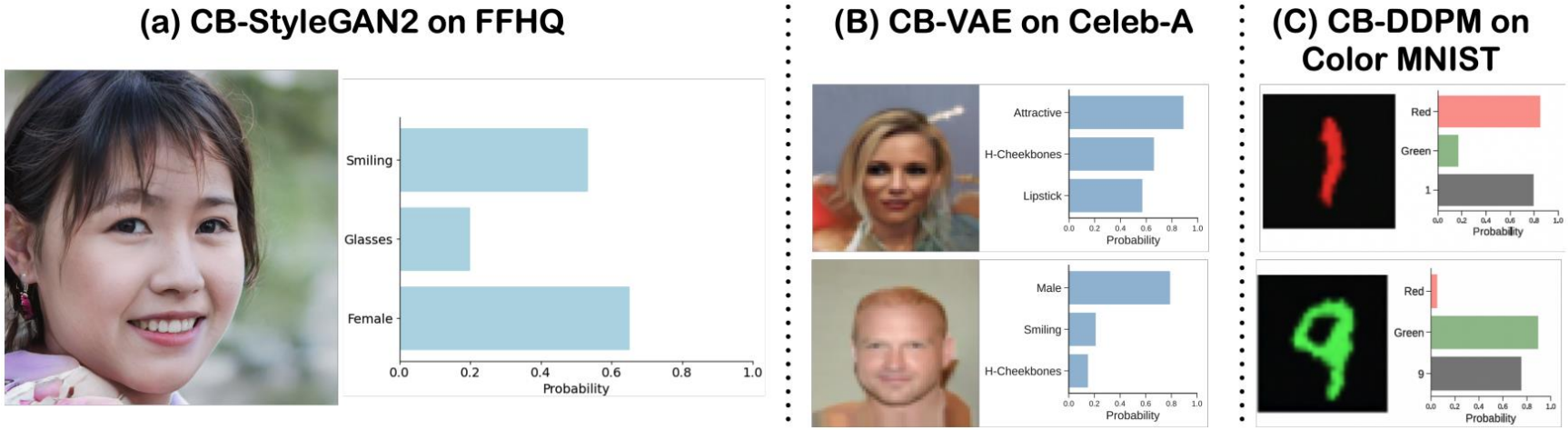
Concept Bottleneck Generative Models (ICLR24)

- Results

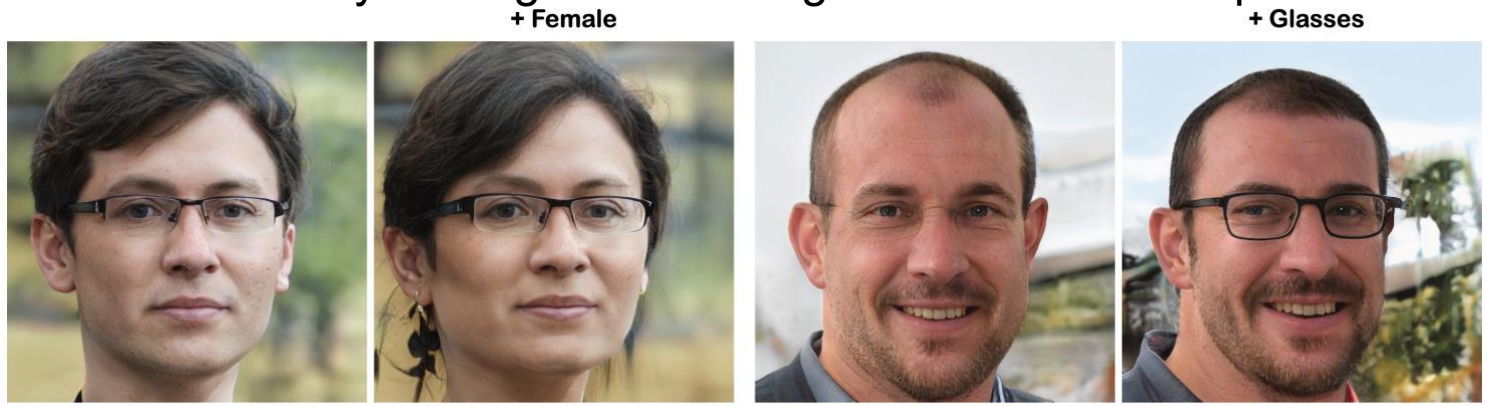
Gen Model	FID
StyleGAN2	9.0
CB-StyleGAN2	9.1
Diffusion	9.1
CB-Diffusion	9.3
VAE	8.4
CB-VAE	8.2

Similar generative performance as without CBL

Interpretability: Get concept predictions for each generated image

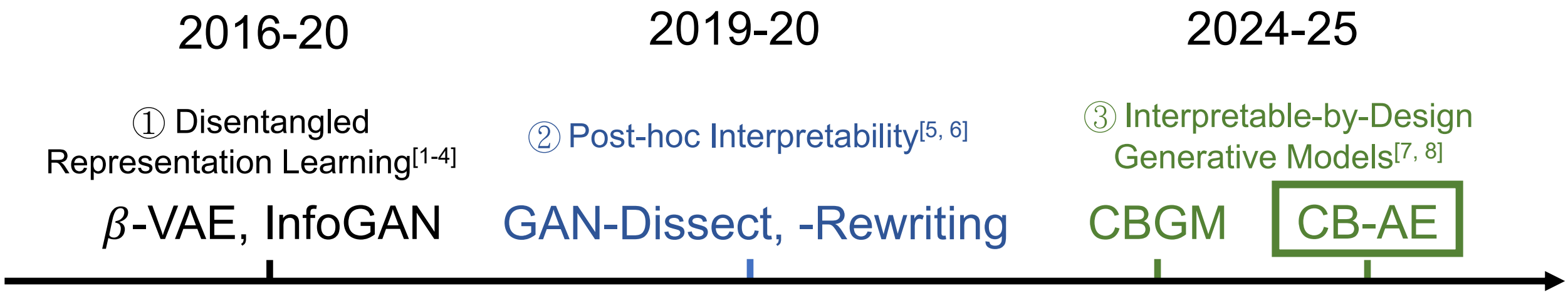


Steerability: Fine-grained editing w.r.t. known concepts



[7] Ismail et al., “Concept Bottleneck Generative Models”, ICLR24

Timeline: Interpretable Generative Models



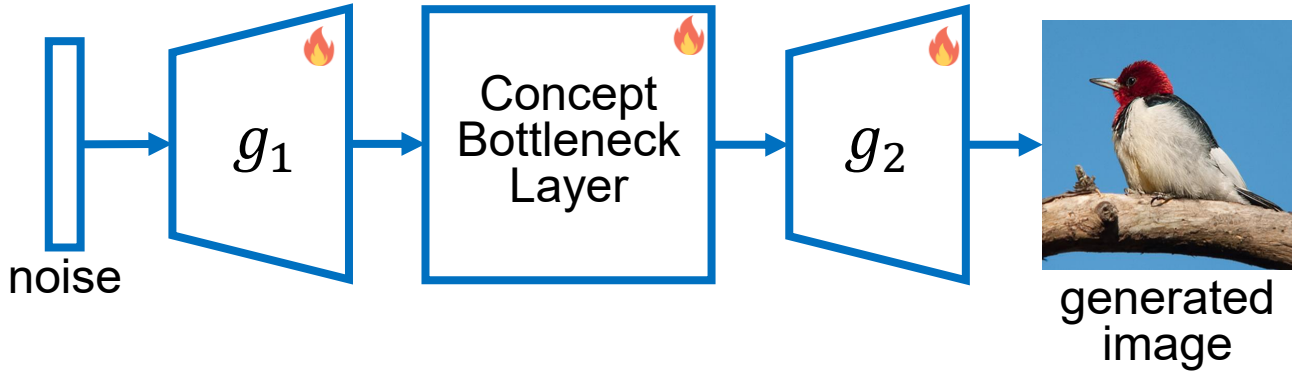
Limitations:
Unsupervised Discovery,
Specific to GAN/VAE family

Limitations:
Only post-hoc editing,
Not interpretable-by-design

[1] Higgins et al., "β-VAE", ICLR17; [2] Chen et al. "Isolating Sources of Disentanglement in VAEs", NeurIPS18
[3] Chen et al., "InfoGAN", NeurIPS16; [4] Ding et al., "Guided Variational Autoencoder for Disentanglement Learning", CVPR20
[5] Bau et al., "GAN Dissection", ICLR19; [6] Bau et al., "Rewriting a Deep Generative Model", ECCV20
[7] Ismail et al., "Concept Bottleneck Generative Models", ICLR24; [8] Kulkarni et al., "Interpretable Generative Models through Post-Hoc CBs", CVPR25

Concept Bottleneck AutoEncoder (CVPR25)

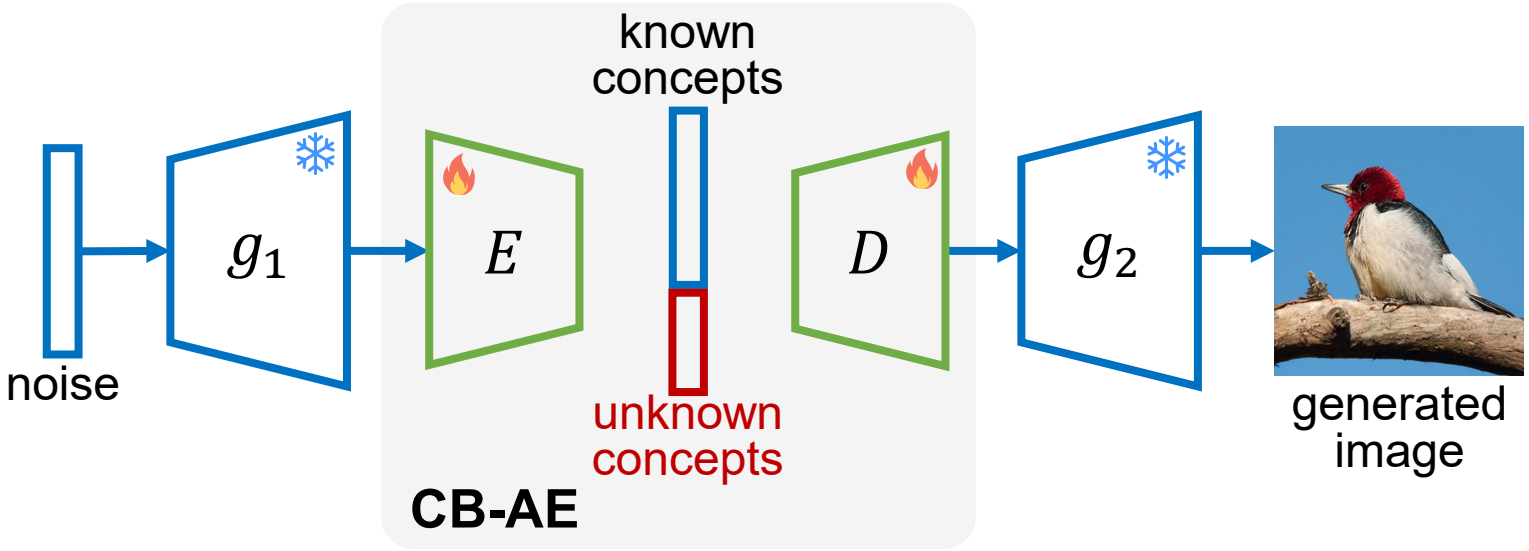
- Limitations of CBGM (ICLR24):
  - Expensive retraining of the entire generative model
    - E.g. 10 V100-days for DDPM (diffusion model) training
  - Concept annotations for real images
    - Expensive to obtain annotations
    - Generative models like GANs cannot always accurately invert real images to latents
      - Additional training parameters and objective to invert real images to GAN latents



[8] Kulkarni et al., “Interpretable Generative Models through Post-Hoc Concept Bottlenecks”, CVPR25

Concept Bottleneck AutoEncoder (CVPR25)

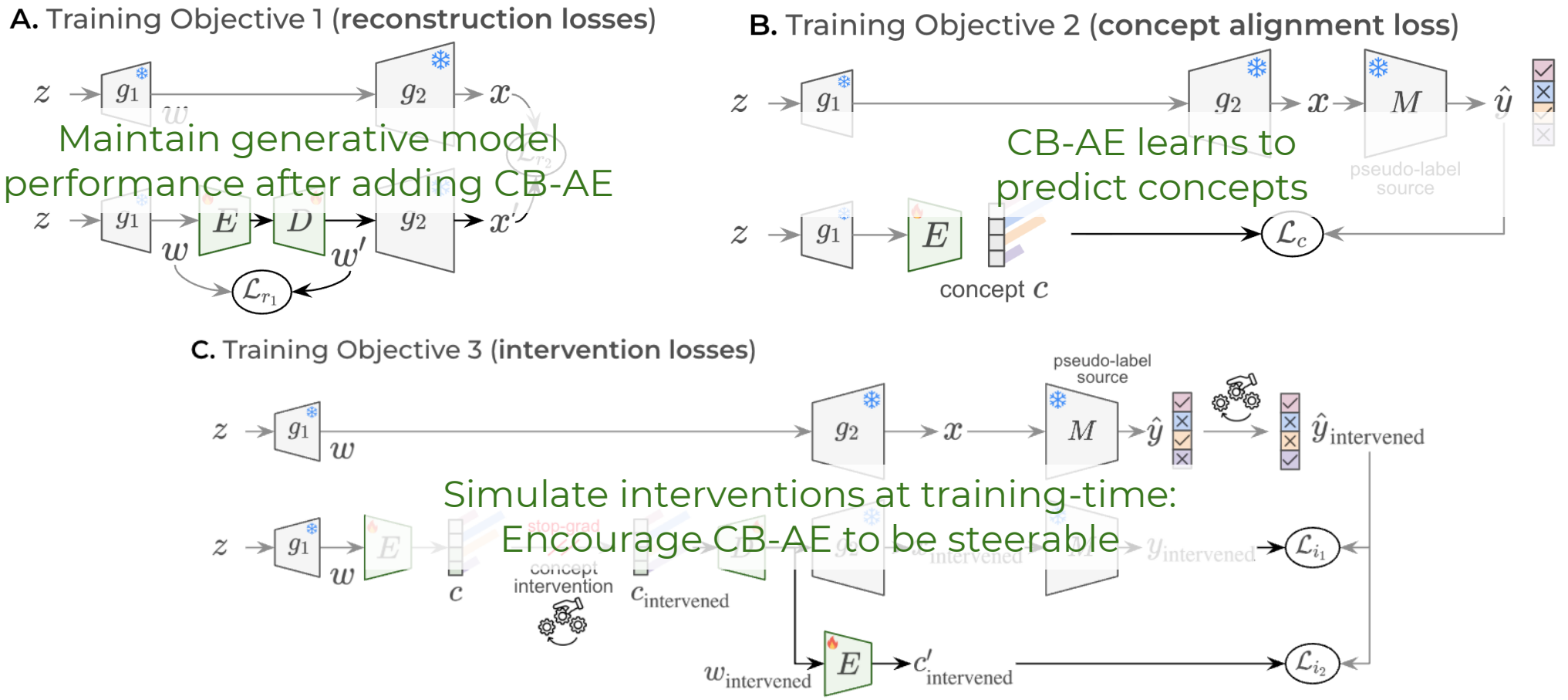
- CB-AE addresses these limitations:
  - Efficiently train only CB-AE with frozen, pretrained generator
  - Train without real images or concept annotations



[8] Kulkarni et al., “Interpretable Generative Models through Post-Hoc Concept Bottlenecks”, CVPR25

Concept Bottleneck AutoEncoder (CVPR25)

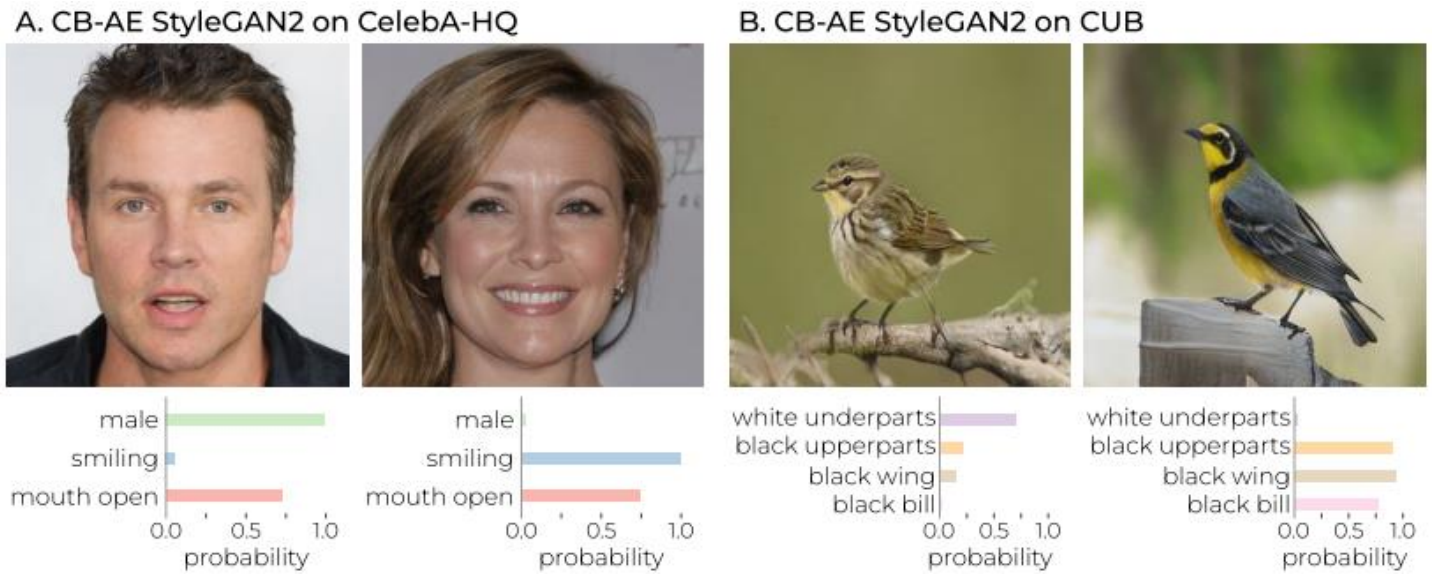
- CB-AE training



[8] Kulkarni et al., "Interpretable Generative Models through Post-Hoc Concept Bottlenecks", CVPR25

Concept Bottleneck AutoEncoder (CVPR25)

- Results: Interpretability



Concept Accuracy

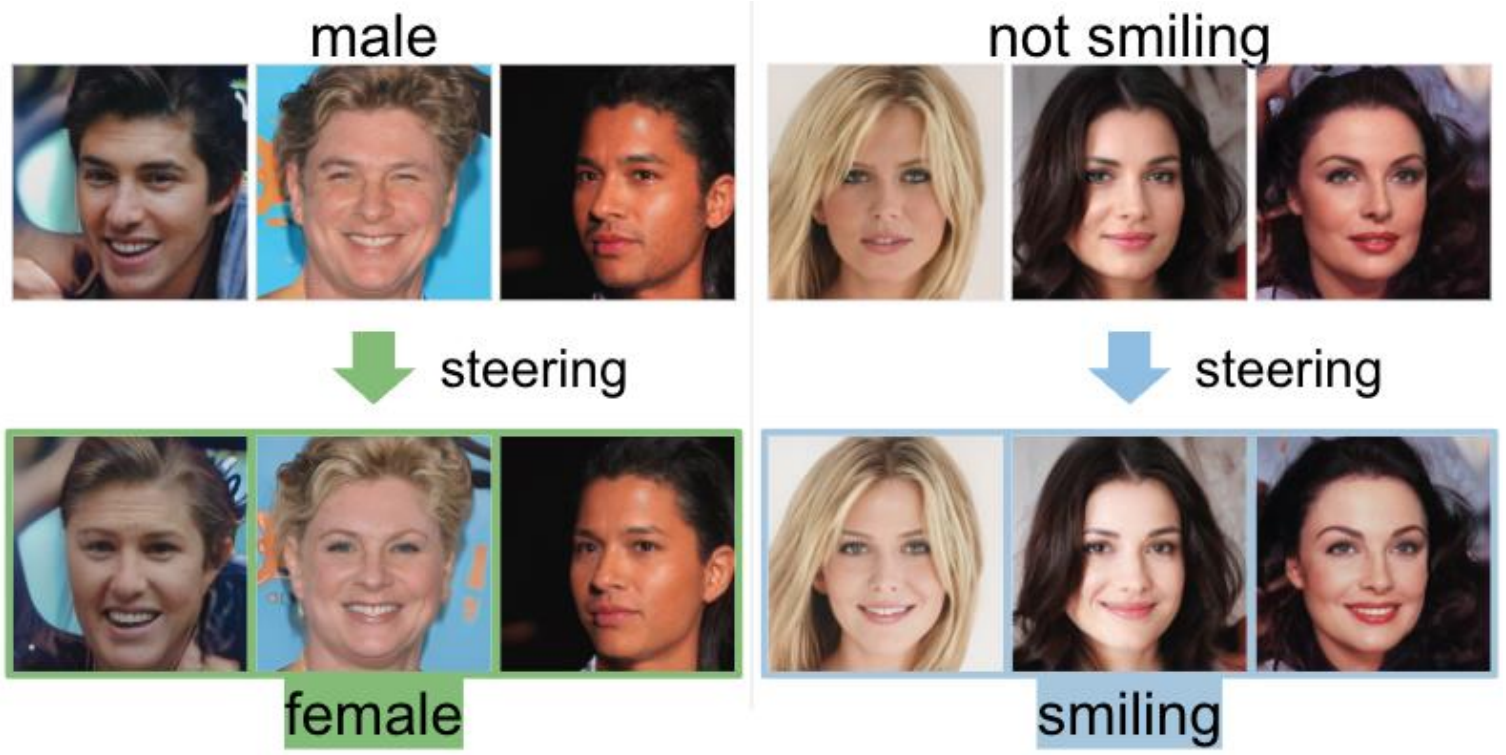
Conc. Acc. (%)	CelebA-HQ		CUB
	DDPM	StyGAN2	StyGAN2
CB-AE	89.79	86.04	81.33

Good concept accuracy across datasets and models

[8] Kulkarni et al., “Interpretable Generative Models through Post-Hoc Concept Bottlenecks”, CVPR25

Concept Bottleneck AutoEncoder (CVPR25)

- Results: Steerability



Intervention Success / Steerability

Steerability (%)	CelebA (64×64)	
	GAN	DDPM
CBGM	25.60	13.80
CB-AE	**47.34**	**23.72**

CB-AE improves steerability by ~15%

[8] Kulkarni et al., “Interpretable Generative Models through Post-Hoc Concept Bottlenecks”, CVPR25

Timeline: Interpretable Generative Models

2016-20

① Disentangled Representation Learning[1-4]

β-VAE, InfoGAN

Limitations:
Unsupervised Discovery,
Specific to GAN/VAE family

2019-20

② Post-hoc Interpretability[5, 6]

GAN-Dissect, -Rewriting

Limitations:
Only post-hoc editing,
Not interpretable-by-design

2024-25

③ Interpretable-by-Design Generative Models[7, 8]

CBGM CB-AE

Limitations:
Concept leakage,
Imperfect steering

2026
Variational Hard CB[9]
Concept Leakage Prevention,
More Comprehensive Evaluation

2025
Energy-based CB[10]
Improving Interpretability

2026
CB-Diffusion[11]
Improving Steerability

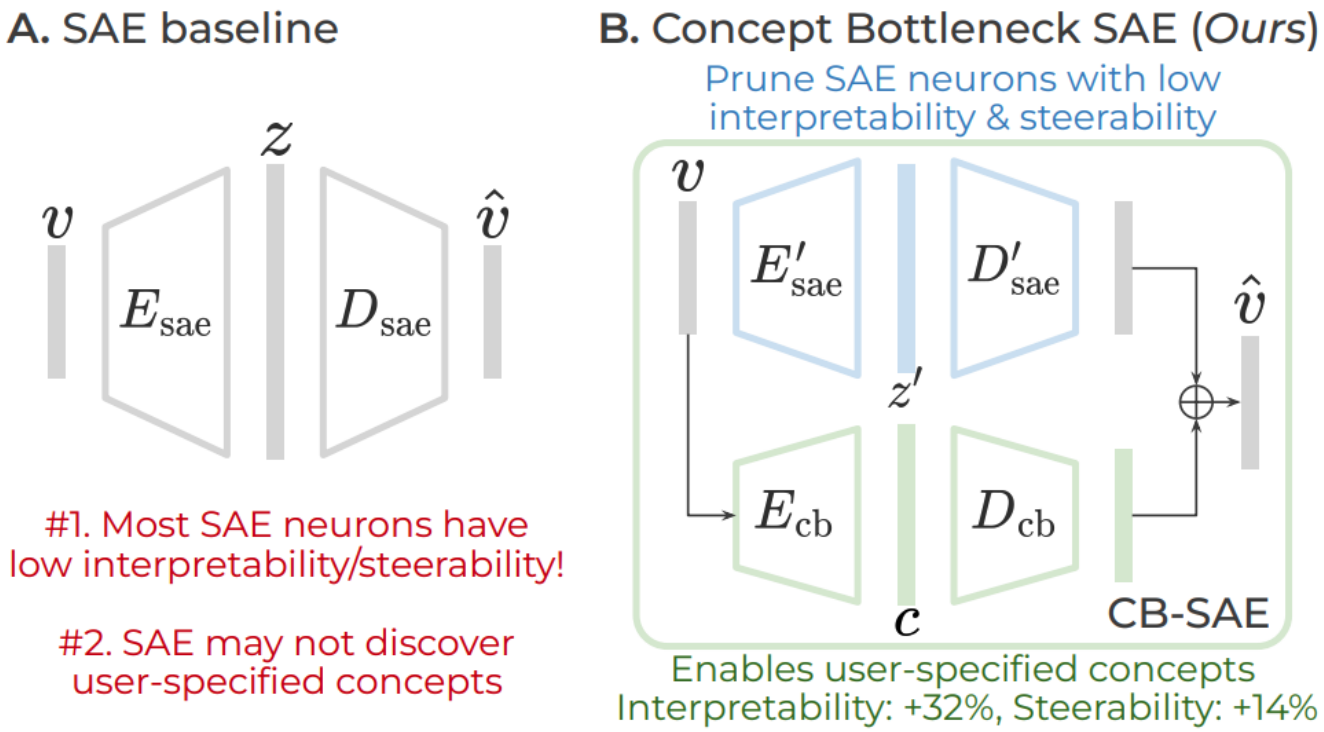
2025-26
CB-LLM[12], CB-SAE[13]
Applicability to Text-based
& Multimodal Generative LLMs

Concept Bottleneck SAEs (CVPR 2026)

- If interested, welcome to drop by our poster on Friday morning

Poster Session 1: June 5th, 10:45 am to 12:45 pm

ExHall A-F, Poster #269

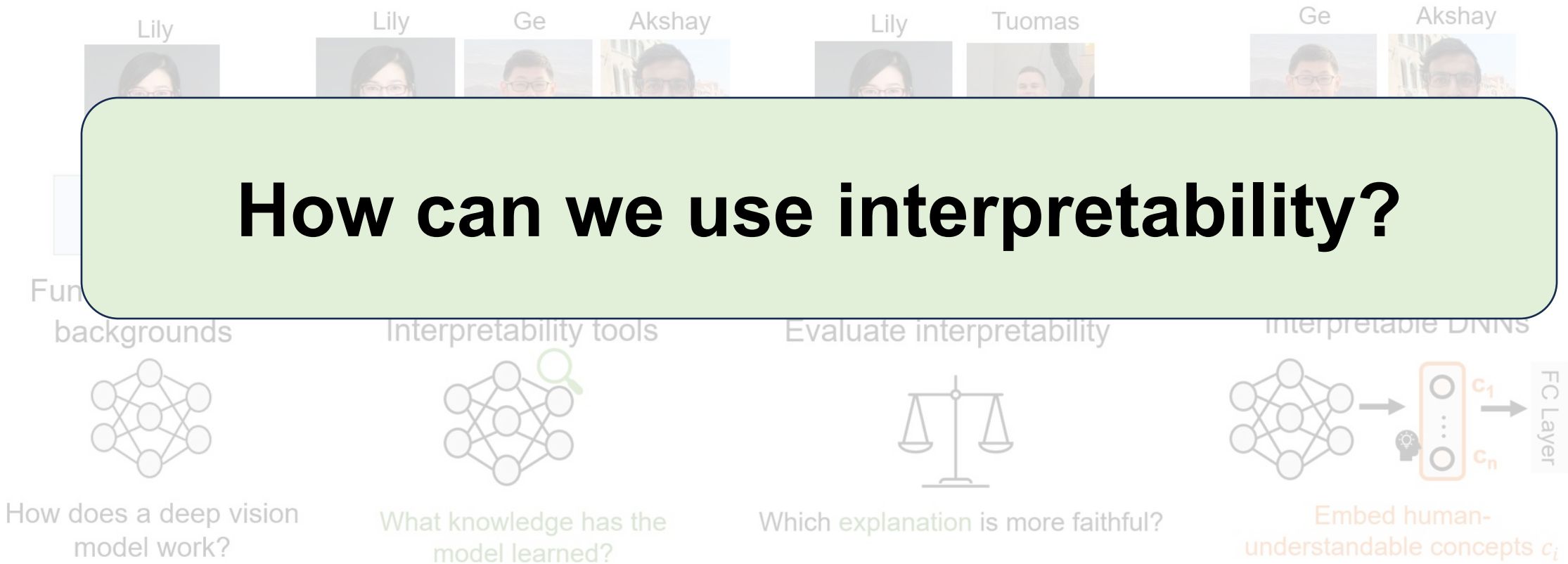


References

1. Higgins et al., “β−VAE: Learning Basic Visual Concepts with a Constrained Variational Framework”, ICLR 2017
2. Chen et al. “Isolating Sources of Disentanglement in VAEs”, NeurIPS 2018
3. Chen et al., “InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets”, NeurIPS 2016
4. Ding et al., “Guided Variational Autoencoder for Disentanglement Learning”, CVPR 2020
5. Bau et al., “GAN-Dissection: Visualizing and Understanding Generative Adversarial Networks”, ICLR 2019
6. Bau et al., “Rewriting a Deep Generative Model”, ECCV 2020
7. Ismail et al., “Concept Bottleneck Generative Models”, ICLR 2024
8. Kulkarni et al., “Interpretable Generative Models through Post-hoc Concept Bottlenecks”, CVPR 2025
9. Martínez-García et al., “A Probabilistic Hard Concept Bottleneck for Steerable Generative Models”, ICLR 2026
10. Kim et al., “CoBELa: Steering Transparent Generation via Concept Bottlenecks on Energy Landscapes”, ICCV 2025 Workshop
11. Enouen and Galhotra, “Concept Bottleneck Diffusion for Steerable Generation”, ICLR 2026 Workshop
12. Sun et al., “Concept Bottleneck Large Language Models”, ICLR 2025
13. Kulkarni et al., “Interpretable and Steerable Concept Bottleneck Sparse Autoencoders”, CVPR 2026
14. Espinosa Zarlenga et al., “Concept Embedding Models: Beyond the Accuracy-Explainability Trade-Off”, NeurIPS 2022

Principled Interpretability Tutorial: Part 5

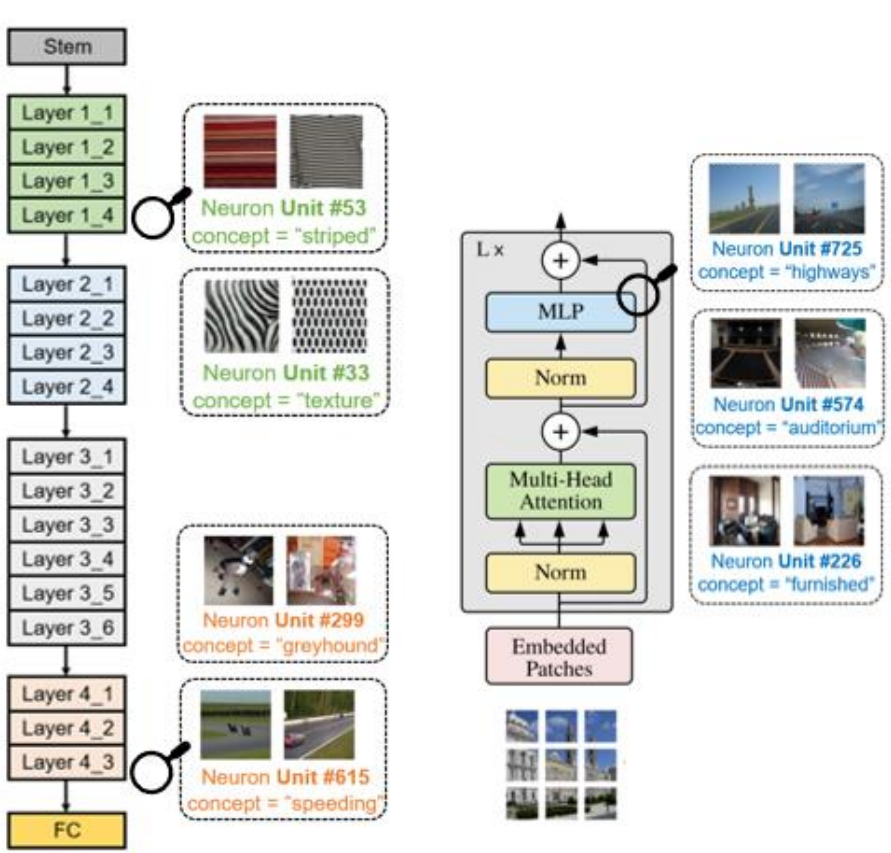
https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/

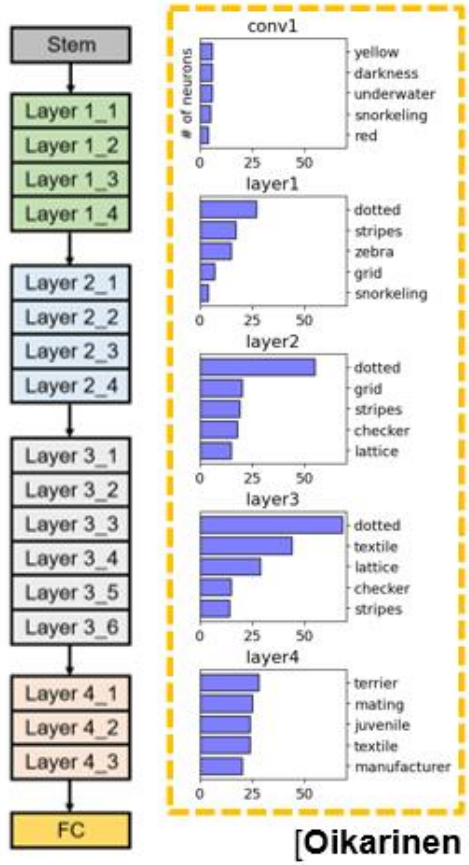


How can we use interpretability?

Recall in the PART 2 & demo

- Post-hoc interpretability tools helps us understand better the models





Get a Full profile of your DNN!

[**Oikarinen** & **Weng**, CLIP-Dissect, ICLR'23]

Recall in the PART 4 & demo

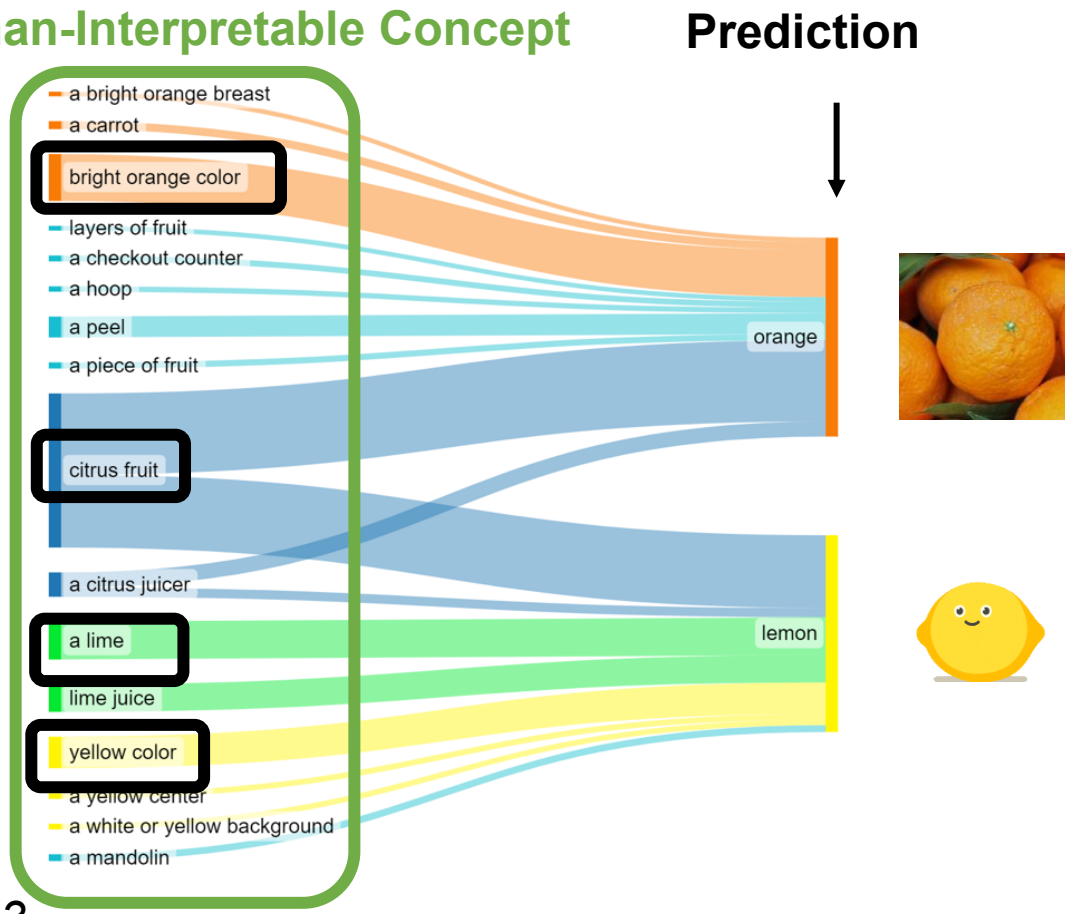
- Inherently interpretable DNN helps us understand better the models

We can easily visualize final layer weights, and see whether they make sense!

For example, we can see that:

orange = “citrus fruit” + “bright orange color”
(prediction) (concepts)

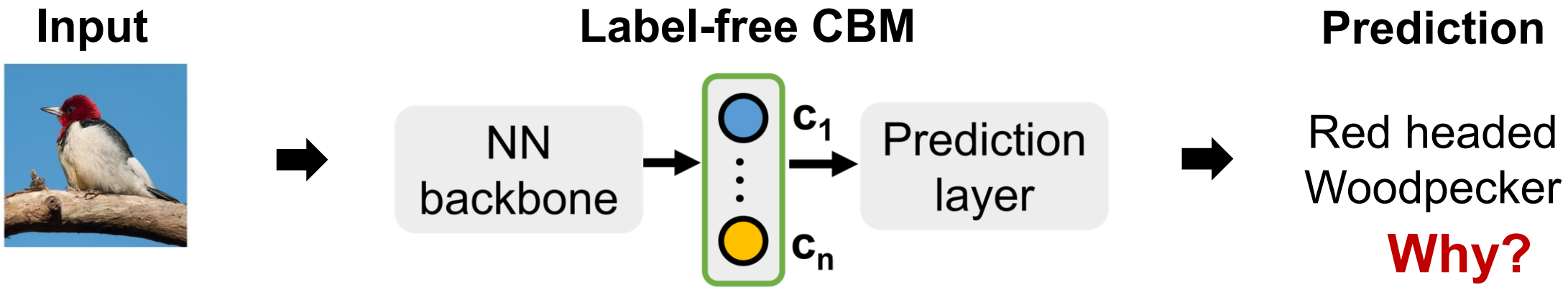
lemon = “citrus fruit” + “lime” + “yellow color”
(prediction) (concepts)



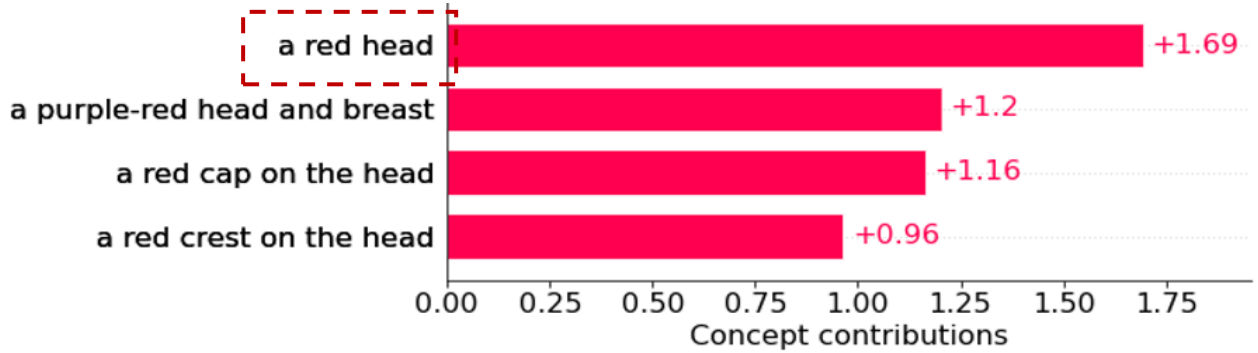
Oikarinen et al. "Label-free Concept Bottleneck Models." *ICLR 2023*

Recall in the PART 4 & demo

- Inherently interpretable DNN helps us understand better the models



We can explain individual decision by plotting concept contributions!



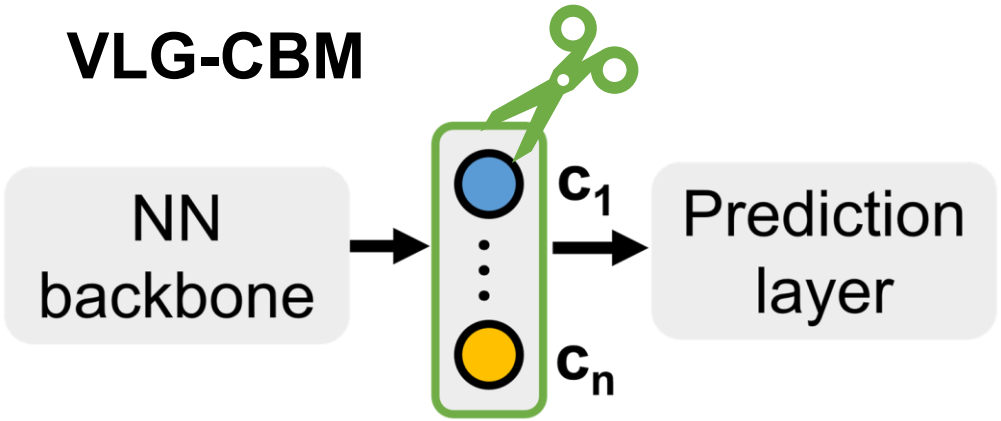
For example, “a red head” makes this image a *Red-headed Woodpecker*

Oikarinen et al. "Label-free Concept Bottleneck Models." *ICLR 2023*

Recall in the PART 4 & demo

Input

VLG-CBM



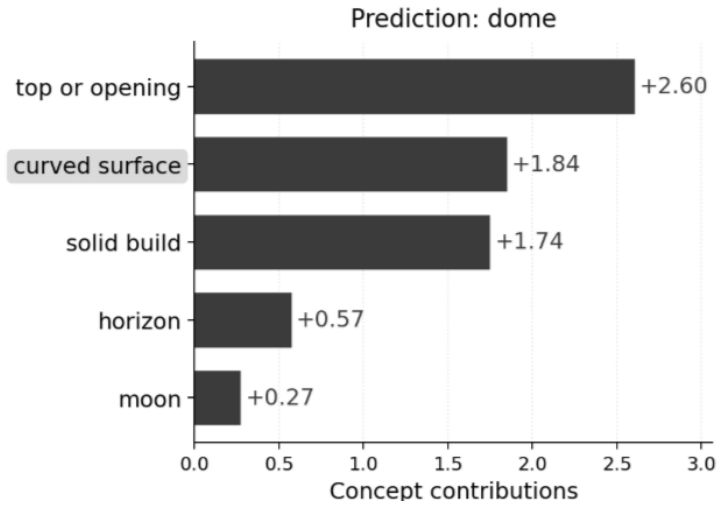
Prediction

dome

Incorrect!

Intervene the wrong concept

(deactivate “curved surface”)



tile roof

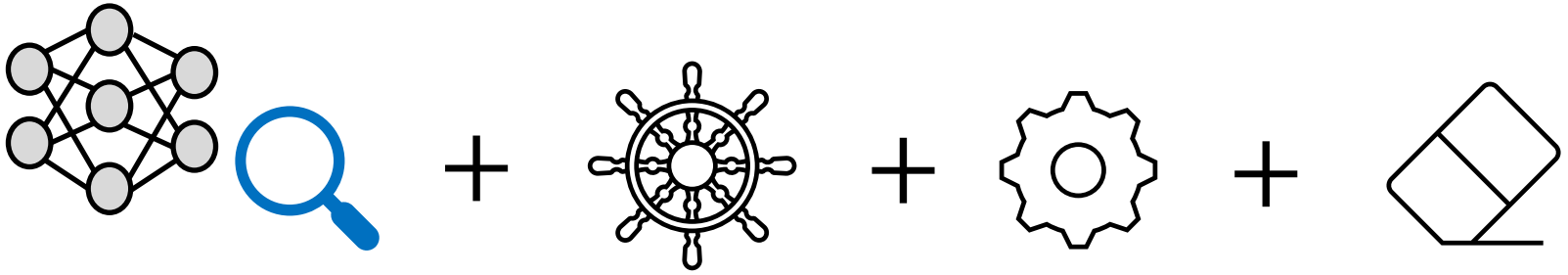
Correct!

Srivastava et. al. "Vlg-cbm: Training concept bottleneck models with vision-language guidance." *NeurIPS 2024*

Interpretability as a Control Interface

- **Actionable interpretability:** turn explanations into **control** variables

Control AI model behaviors easily

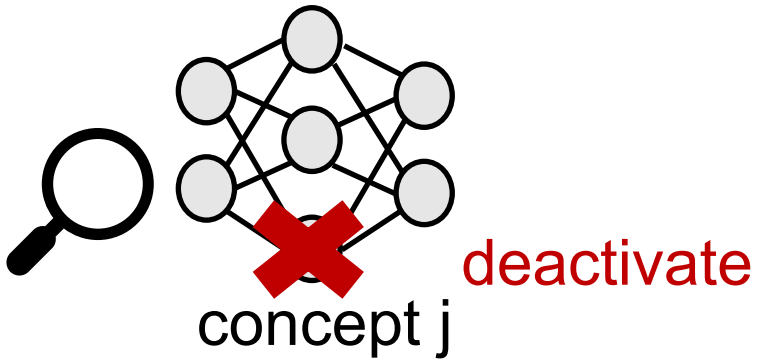


Understand Steer Modify Forget/ Unlearn

Use case #1

- Unlearning with Post-hoc neuron interpretability tool (e.g. CLIP-Dissect)

Unlearn concepts by simply deactivating the related neurons



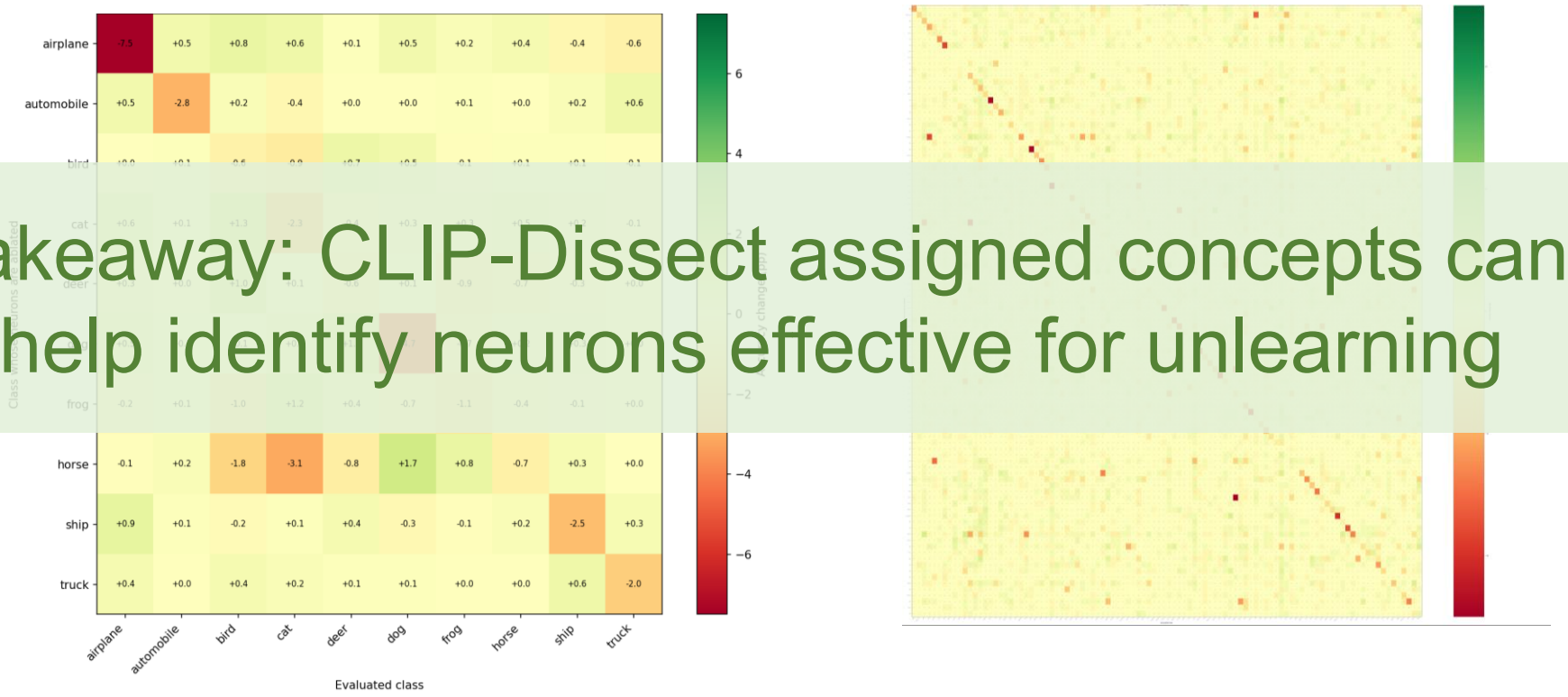
Disable class-related concepts and observe how the related class & unrelated class's accuracy change

Model's related class accuracy drop (diagonal terms) while other unrelated class ~not affected

Oikarinen et. al. "CLIP-Dissect: Automatic Description of Neuron Representations in Deep Vision Networks." *ICLR 2023*

Use case #1: Unlearning with CLIP-Dissect

- Unlearning evaluation (CIFAR10/100, ResNet18, layer 4)



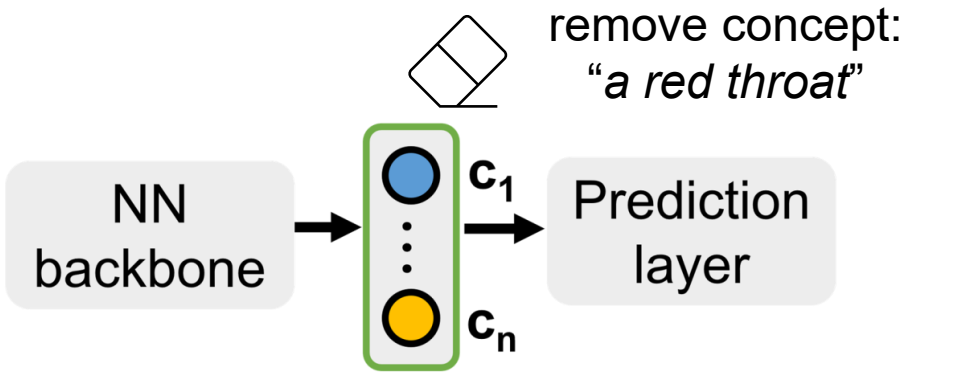
Takeaway: CLIP-Dissect assigned concepts can help identify neurons effective for unlearning

green: ↑ acc.
yellow: ~ acc.
red: ↓ acc.

Oikarinen et. al. “CLIP-Dissect: Automatic Description of Neuron Representations in Deep Vision Networks." *ICLR 2023*

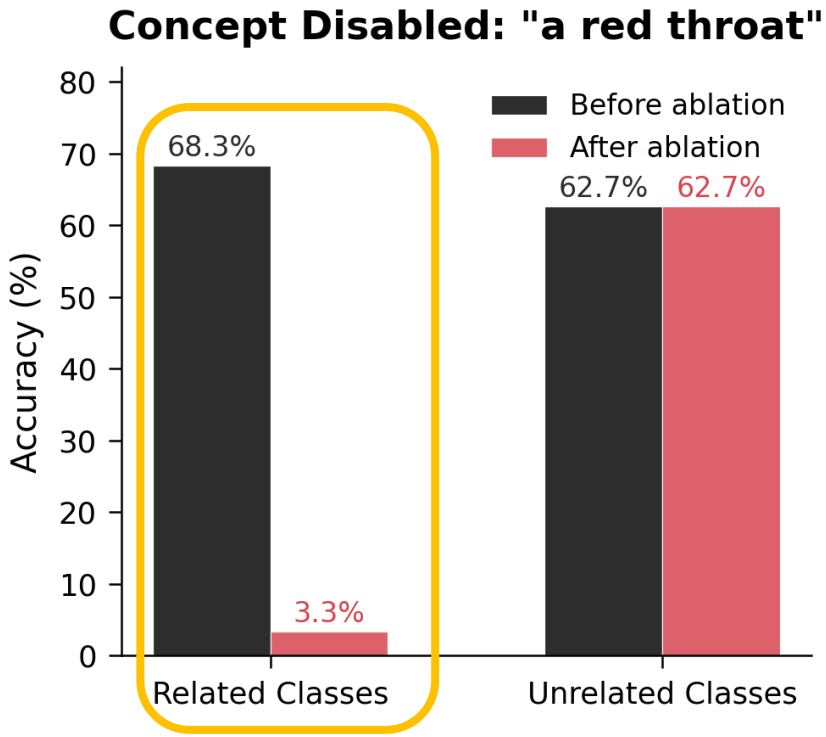
Use case #2

- Unlearning with inherently interpretable DNN (e.g. VLG-CBM)



Anna hummingbird

Ruby throated hummingbird

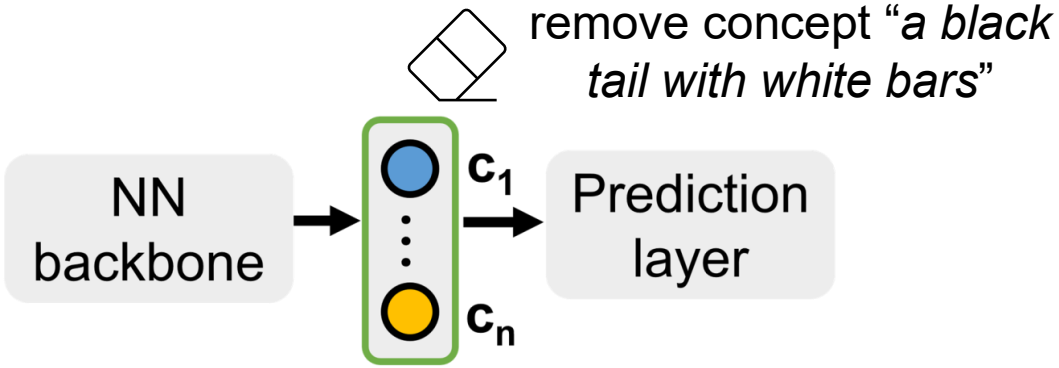


Related: Anna Hummingbird, Ruby throated Hummingbird

Srivastava et. al. "Vlg-cbm: Training concept bottleneck models with vision-language guidance." *NeurIPS 2024*

Use case #2

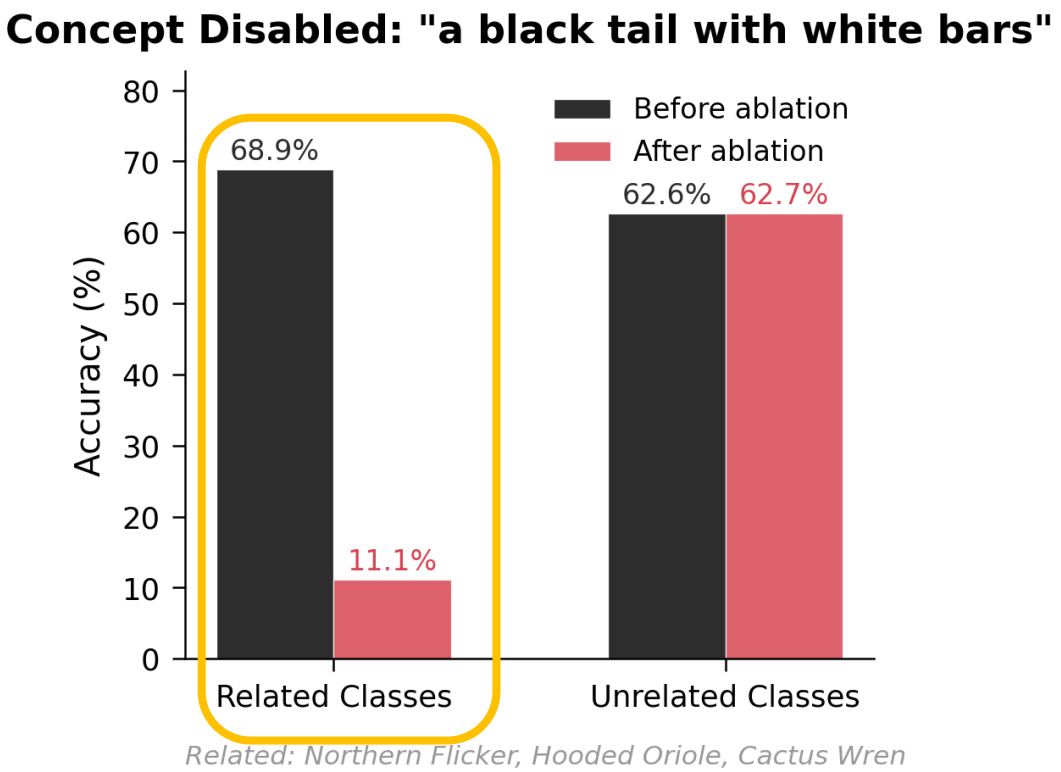
- Unlearning with inherently interpretable DNN (e.g. VLG-CBM)



Northern Flicker

Hooded Oriole

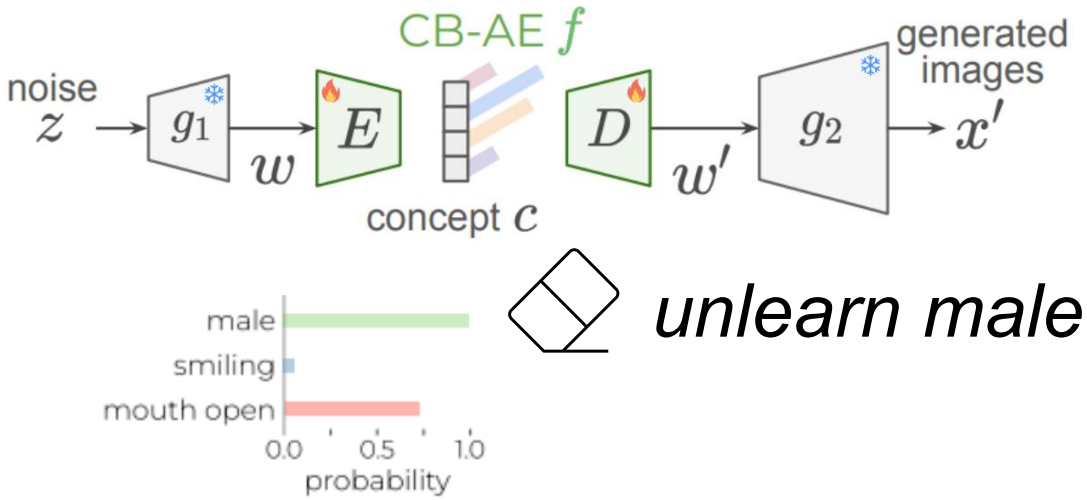
Cactus Wren



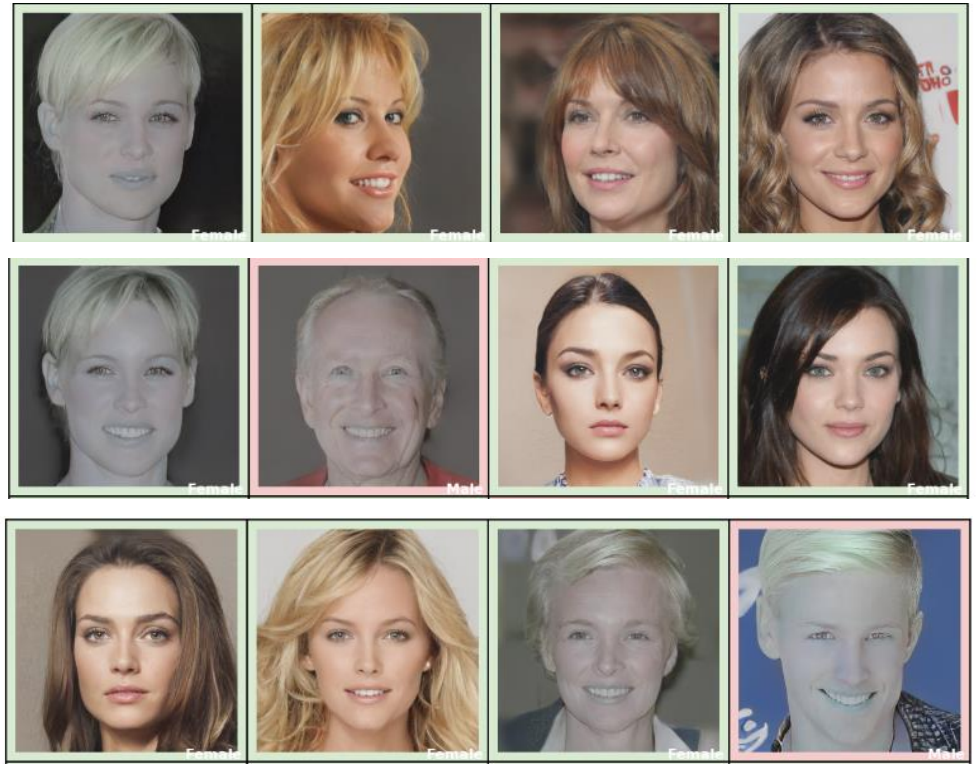
Srivastava et. al. "Vlg-cbm: Training concept bottleneck models with vision-language guidance." *NeurIPS 2024*

Use case #3

- Steering & Unlearning with inherently interpretable DNN (e.g. CB-AE)



unlearn male



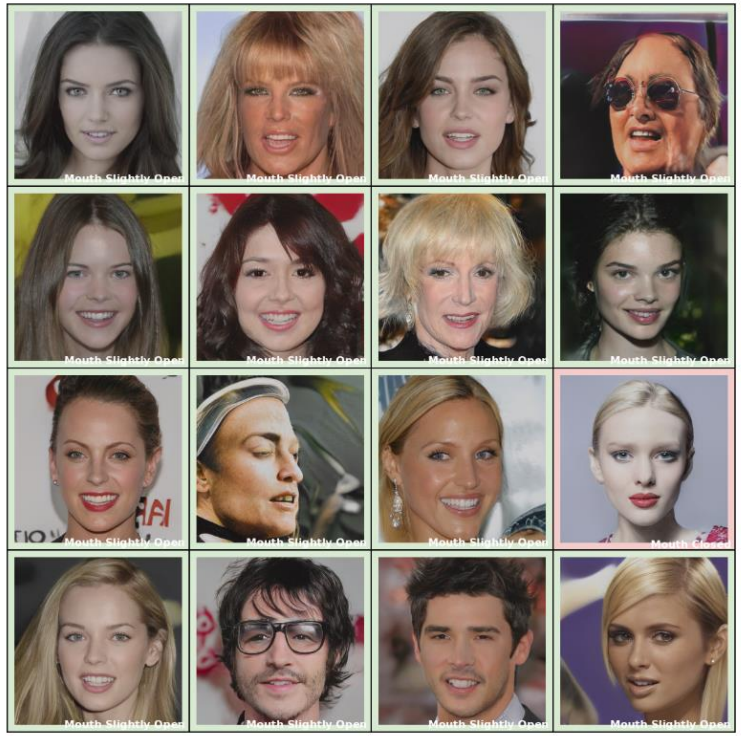
Now the controlled model output is mostly female!

Kulkarni et. al. “Interpretable Generative Models through Post-hoc concept bottlenecks" *CVPR 2025*

Use case #3

- Steering & Unlearning with inherently interpretable DNN (e.g. CB-AE)

unlearn “mouth closed”



unlearn “not smiling”

Kulkarni et. al. “Interpretable Generative Models through Post-hoc concept bottlenecks" *CVPR 2025*

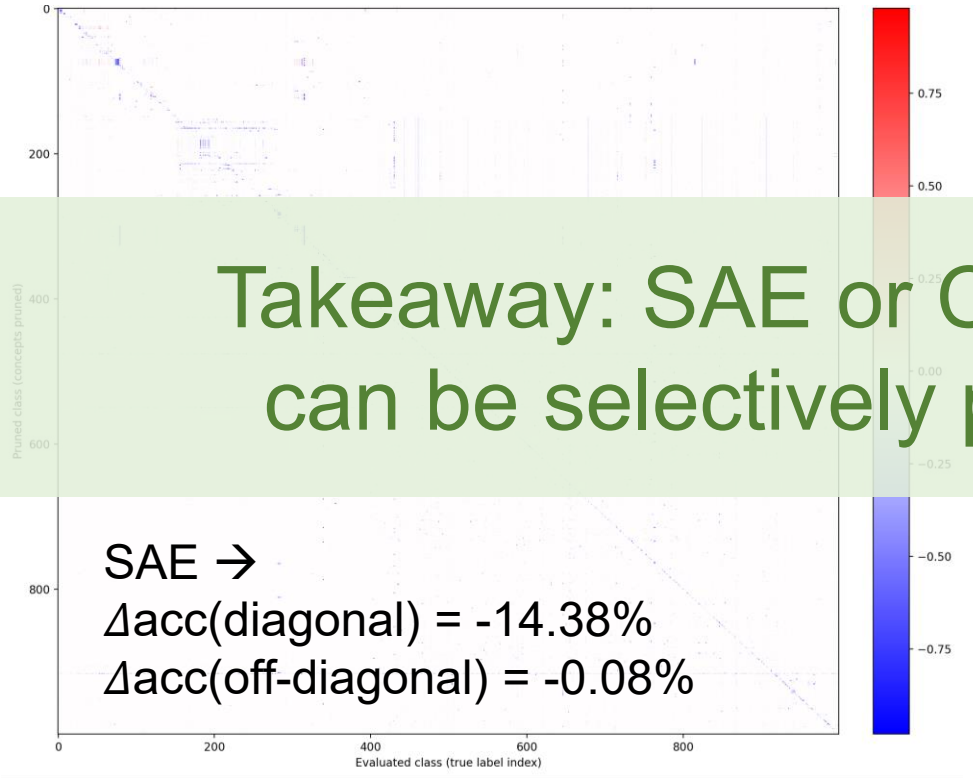
Use case #4: Unlearning with SAE/CB-SAE

- Setup
  - SAE/CB-SAE trained on ImageNet-1k dataset for CLIP image encoder
  - Apply CLIP-Dissect to SAE/CB-SAE neurons to find class-related neurons
  - Unlearning evaluation with zero-shot ImageNet classification
    - Ablate SAE/CB-SAE neurons related to a particular class during test accuracy evaluation

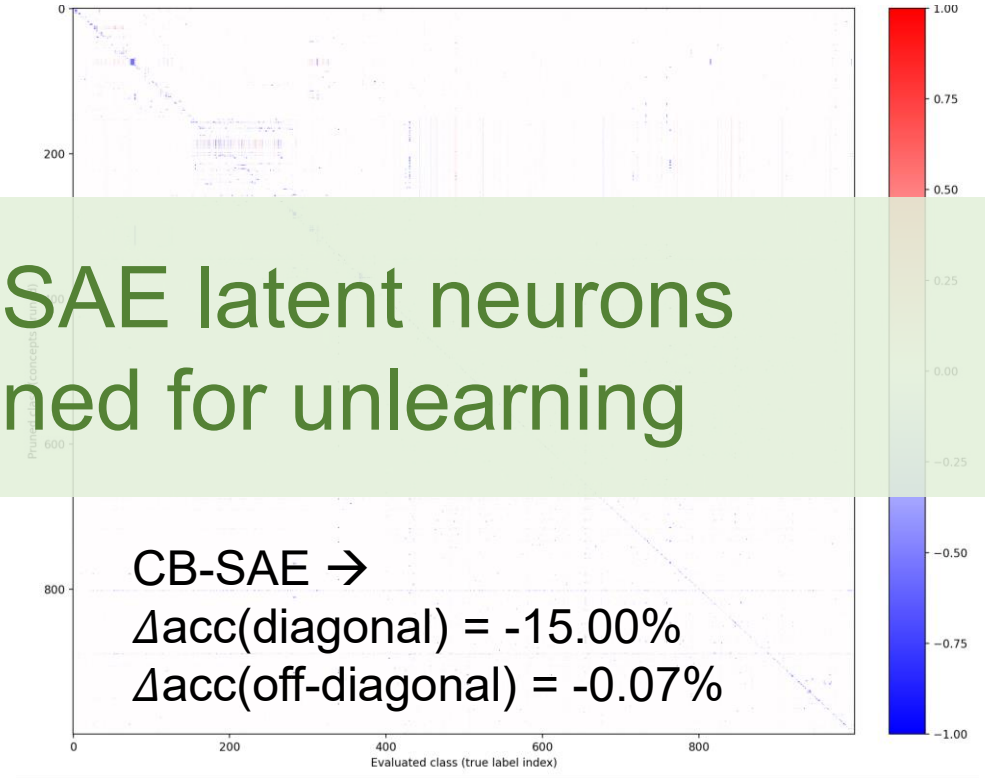
Kulkarni et al., “Interpretable and Steerable Concept Bottleneck Sparse Autoencoders”, CVPR 2026

Use case #4: Unlearning with SAE/CB-SAE

- Unlearning evaluation (SAE/CB-SAE with CLIP zero-shot classifier)



SAE →
Δacc(diagonal) = -14.38%
Δacc(off-diagonal) = -0.08%



CB-SAE →
Δacc(diagonal) = -15.00%
Δacc(off-diagonal) = -0.07%

Takeaway: SAE or CB-SAE latent neurons can be selectively pruned for unlearning

Kulkarni et al., “Interpretable and Steerable Concept Bottleneck Sparse Autoencoders”, CVPR 2026

Interpretability for scientific discovery

GOODFIRE

Research

Using Interpretability to Identify a Novel Class of Alzheimer's Biomarkers

An AI model was trained to detect Alzheimer's from blood samples. We opened it up to understand how—and found that DNA fragment length patterns dominate its decision-making. We distilled this insight into a human-interpretable classifier that generalizes better than the biomarker classes previously reported in the literature when tested on an independent cohort.

Want et al. Using Interpretability to Identify a Novel Class of Biomarkers for Alzheimer's Detection, Goodfire, 2026

Interpretability for unlearning unsafe contents

- Real-world usecase: Unlearning unsafe image generation
  - Problem: Harmful prompts in SD 1.4/2.1 lead to inappropriate generated images
  - Solution: Unlearning (suppressing) harmful concepts via SAE

Table 2. **Nudity unlearning evaluation on the I2P benchmark.** The best result for each metric is highlighted in bold.

Method	Armpits	Belly	Buttocks	Feet	Breasts (F)	Genitalia (F)	Breasts (M)	Genitalia (M)	Total	CLIPScore (↑)	FID (↓)
FMN (Zhang et al., 2024a)	43	117	12	59	155	17	19	2	424	30.39	13.52
CA (Kumari et al., 2023)	153	180	45	66	298	22	67	7	838	**31.37**	16.25
AdvUn (Zhang et al., 2024b)	8	**0**	**0**	13	**1**	1	**0**	**0**	28	28.14	17.18
Receler (Huang et al., 2024)	48	32	3	35	20	**0**	17	5	160	30.49	15.32
MACE (Lu et al., 2024)	17	19	2	39	16	**0**	9	7	111	29.41	**13.42**
CPE (Lee et al., 2025)	10	8	2	8	6	1	3	2	40	31.19	13.89
UCE (Gandikota et al., 2024)	29	62	7	29	35	5	11	4	182	30.85	14.07
SLD-M (Schramowski et al., 2023)	47	72	3	21	39	1	26	3	212	30.90	16.34
ESD-x (Gandikota et al., 2023)	59	73	12	39	100	6	18	8	315	30.69	14.41
ESD-u (Gandikota et al., 2023)	32	30	2	19	27	3	8	2	123	30.21	15.10
SAeUron	**7**	1	3	**2**	4	**0**	**0**	1	**18**	30.89	14.37
SD v1.4	148	170	29	63	266	18	42	7	743	31.34	14.04
SD v2.1	105	159	17	60	177	9	57	2	586	31.53	14.87

Metric: Number of inappropriate images successfully generated (lower is better)

Cywinski & Deja, “SAeUron: Interpretable Concept Unlearning in Diffusion Models with Sparse Autoencoders”, ICML25

Open Challenges

- Faithful explanations & Reliable evaluation
- Human-grounded validation
- Mechanistic understanding at scale
- Actionable intervention and control
- Interpretable-by-design foundation models

Reliable Evaluation of Interpretability

- **Challenge:** We still lack widely accepted methods for determining whether an explanation is trustworthy
- **Questions:**
  - How do we measure faithfulness?
    - Neuron-eval [1] discussed in PART 3 is just a first step; recent work [2] on developing theory for trustworthy interpretation
  - How do we quantify uncertainty in explanations?
    - moreover, explanations can be compromised with random noises [3] or adversary [3, 4]
  - How do we evaluate explanations across models, datasets, and modalities? [5]

[1] Oikarinen etal, Evaluating Neuron Explanations: A Unified Framework with Sanity Checks, ICML 2025

[2] Yan etal, Faithful and Stable Neuron Explanations for Trustworthy Mechanistic Interpretability, arxiv 2025

[3] Srivastava etal, Corrupting Neuron Explanations of Deep Visual Features, ICCV 2023

[4] Sinha etal, Understanding and Enhancing Robustness of Concept-based Models, AAAI 2023

[5] Lee etal, Disentangled Concepts Speak Louder Than Words: Explainable Video Action Recognition, NeurIPS 2025

Human-Grounded Interpretability

- **Challenge:** What kind of explanations are useful for human understanding

- **Questions:**
  - What makes an explanation understandable
  - How do different users require different explanations
  - How do we efficiently and reliably evaluate human interpretability
    - Methods discussed in PART 3 is an initial step, more efforts needed

Actionable Interpretability

- **Challenge:** Explanations should enable intervention, not just observation

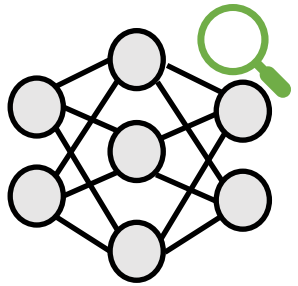
- **Questions:**
  - How to effectively steer model?
    - Covered in Part 4, active area of research
  - How to effectively perform editing and unlearning?
    - Covered in Part 5
  - How to effectively detect failures before deployment?
  - How to improve model design?
    - Covered in Part 4, active area of research

The Future of Interpretability

Trustworthy Interpretability

Ensure the faithfulness, reliability, and actionability

Post-hoc interpretation

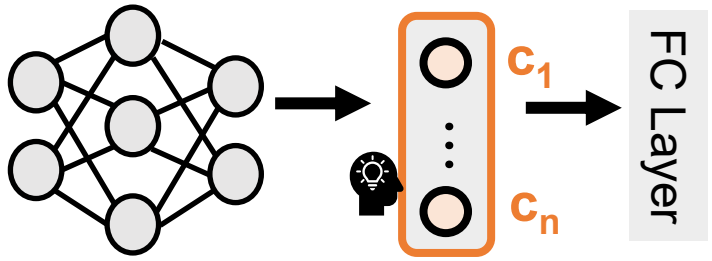


+

Evaluation metrics

+

Inherently Interpretable DNNs



Summary & Resource

All materials will be available at our tutorial website:

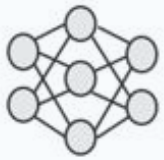
https://lilywenglab.github.io/cvpr2026-principled-interpretability-tutorial/ lweng@ucsd.edu

Principled Interpretability in Vision Models, CVPR 2026 Tutorial

Lily

Part 1

Fundamentals & backgrounds



How does a deep vision model work?

Lily Ge Akshay

Part 2

Post-hoc Interpretability tools

What knowledge has the model learned?

Lily Tuomas

Part 3

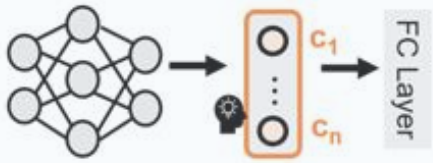
Rigorous Analysis to Evaluate interpretability

Which explanation is more faithful?

Ge Akshay

Part 4

Building Inherently Interpretable DNNs



Embed human-understandable concepts c_i

Lily

Part 5

Application & Future of Interpretability

Actionable and Trustworthy Interpretability

CVPR Workshop Tomorrow (Thursday)!

Trustworthy, Robust, Uncertainty-Aware, and Explainable Visual Intelligence and Beyond (TRUE-V)

June 4th (Thursday), 8 AM, Location: Room 705/707

Workshop Organizers:

Lily Weng
UC San Diego

Nghia Hoang
Washington State U

Tammy Riklin Raviv
Ben Gurion U

Giuseppe Raffa
Intel Labs

Arno Blaas
Apple

Eunji Kim
Amazon

Bhavya Kailkhura
LLNL

Kowshik Thopalli
LLNL