

IEEE VIS 2026 · Boston

ChannelExplorer: Exploring Class Separability Through Activation Channel Visualization

Rahat Zaman · Bei Wang · Paul Rosen

SCI Institute · University of Utah

We judge deep networks by accuracy, loss, and ROC curves

Accurate, but are classes separated?

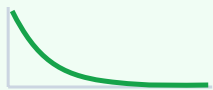
We judge deep networks by accuracy, loss, and ROC curves

OUTPUT-LEVEL METRICS

Accuracy ✓



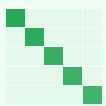
Loss ✓



ROC curve ✓



Confusion matrix ✓



Everything looks healthy

Accurate, but are classes separated?

We judge deep networks by accuracy, loss, and ROC curves

OUTPUT-LEVEL METRICS

Accuracy

Loss

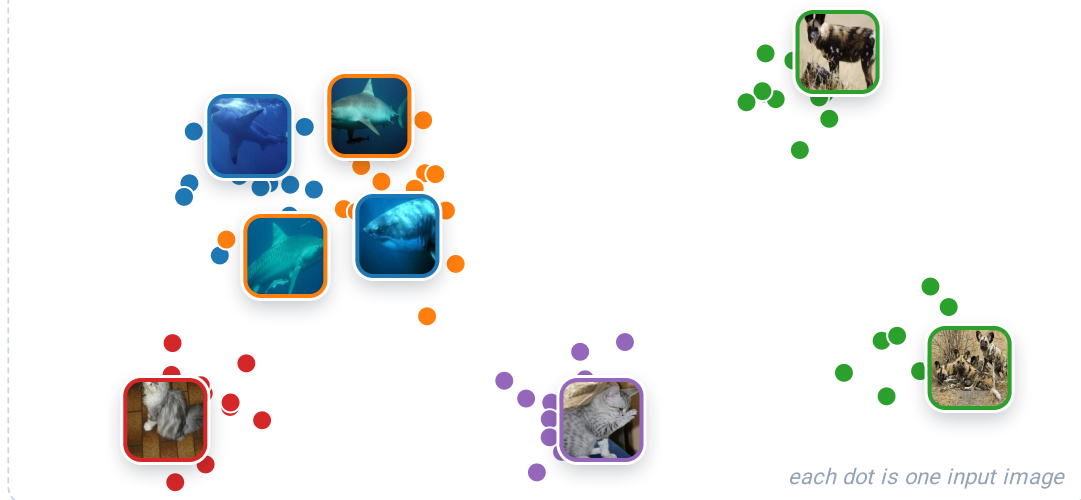
ROC curve

Confusion matrix

Everything looks healthy

INSIDE THE MODEL: ONE LAYER'S ACTIVATIONS, PROJECTED TO 2D

great white shark · tiger shark · African wild dog · Persian cat · Egyptian Mau



each dot is one input image

Accurate, but are classes separated?

We judge deep networks by accuracy, loss, and ROC curves

OUTPUT-LEVEL METRICS

Accuracy ✓

Loss ✓

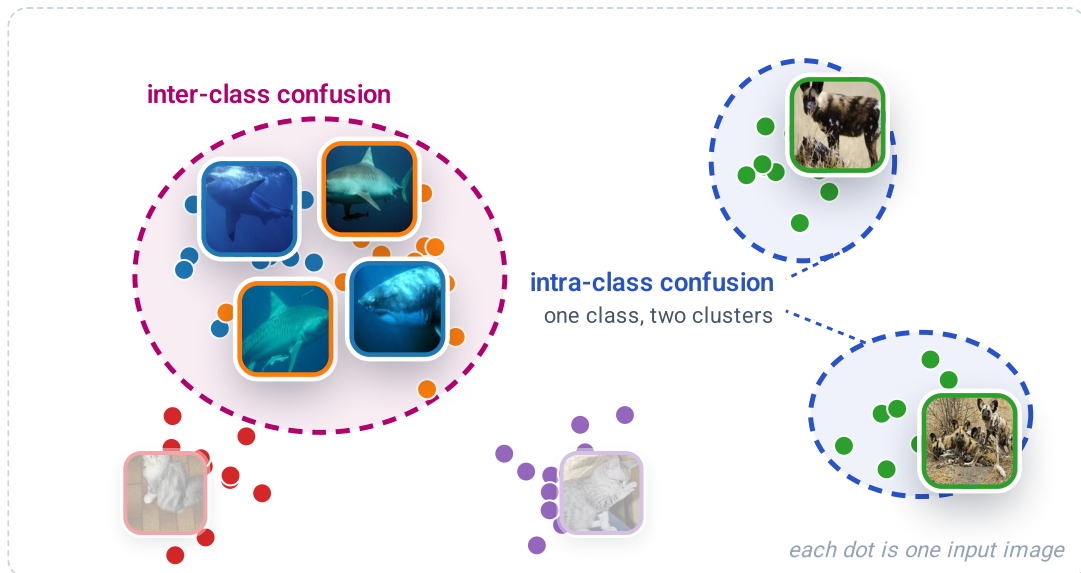
ROC curve ✓

Confusion matrix ✓

Everything looks healthy

INSIDE THE MODEL: ONE LAYER'S ACTIVATIONS, PROJECTED TO 2D

great white shark · tiger shark · African wild dog · Persian cat · Egyptian Mau



Accurate, but are classes separated?

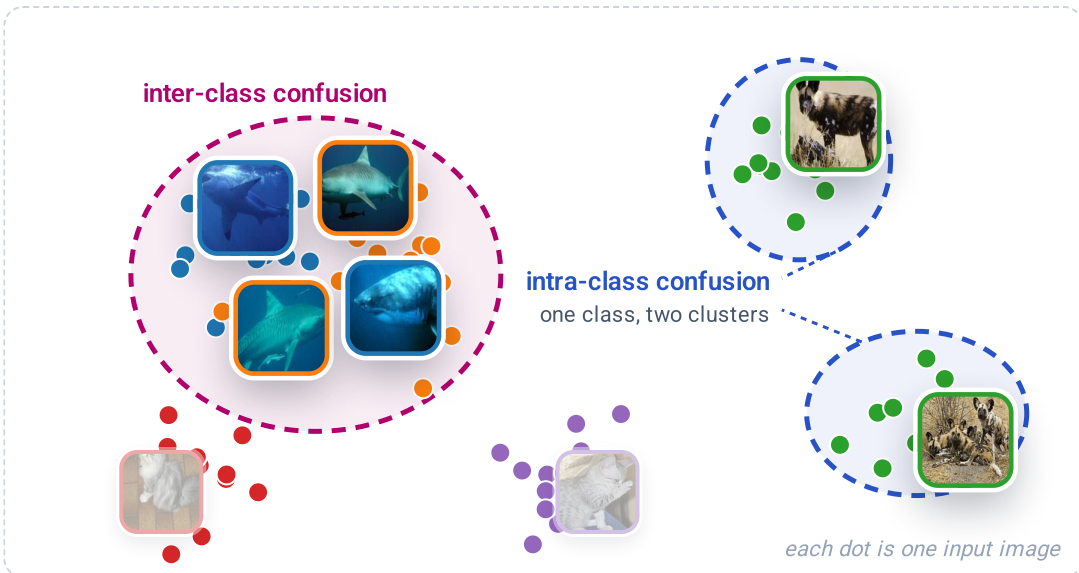
We judge deep networks by accuracy, loss, and ROC curves

Class separability

How **distinct** the internal representations of different classes are.

INSIDE THE MODEL: ONE LAYER'S ACTIVATIONS, PROJECTED TO 2D

great white shark · tiger shark · African wild dog · Persian cat · Egyptian Mau



Class Separability and why it matters

IDENTIFYING SEPARABILITY PROBLEMS

Class Separability and why it matters

IDENTIFYING SEPARABILITY PROBLEMS

1. makes a model more **interpretable**

Class Separability and why it matters

IDENTIFYING SEPARABILITY PROBLEMS

1. makes a model more **interpretable**
2. gives clearer **decision boundaries**

Class Separability and why it matters

IDENTIFYING SEPARABILITY PROBLEMS

1. makes a model more **interpretable**
2. gives clearer **decision boundaries**
3. exposes **mislabeled or imbalanced** data

Class Separability and why it matters

IDENTIFYING SEPARABILITY PROBLEMS

1. makes a model more **interpretable**
2. gives clearer **decision boundaries**
3. exposes **mislabeled or imbalanced** data
4. tells us **where to retrain or regularize**, instead of retraining the whole model

But finding that layer or channel is hard

Class Separability and why it matters

IDENTIFYING SEPARABILITY PROBLEMS

1. makes a model more **interpretable**
2. gives clearer **decision boundaries**
3. exposes **mislabeled or imbalanced** data
4. tells us **where to retrain or regularize**, instead of retraining the whole model

But finding that layer or channel is hard



→ **InceptionV3** · layer `mixed10` · output 8 × 8 × 2,048

Class Separability and why it matters

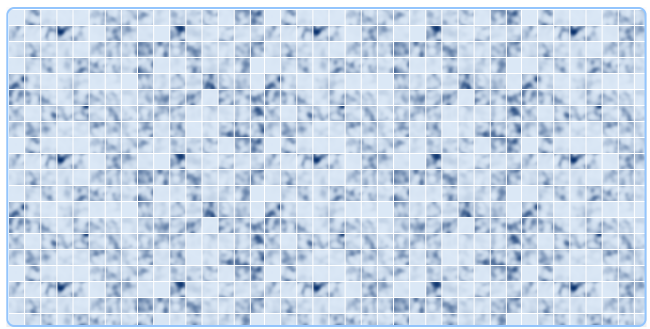
IDENTIFYING SEPARABILITY PROBLEMS

1. makes a model more **interpretable**
2. gives clearer **decision boundaries**
3. exposes **mislabeled or imbalanced** data
4. tells us **where to retrain or regularize**, instead of retraining the whole model

But finding that layer or channel is hard



→ **InceptionV3** · layer `mixed10` · output 8 × 8 × 2,048



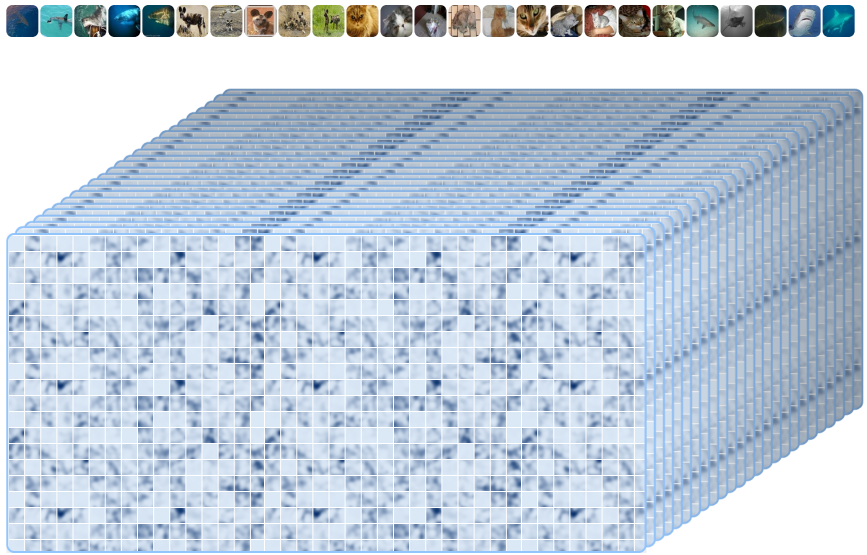
2,048 activation maps for **one** image

Class Separability and why it matters

IDENTIFYING SEPARABILITY PROBLEMS

1. makes a model more **interpretable**
2. gives clearer **decision boundaries**
3. exposes **mislabeled or imbalanced** data
4. tells us **where to retrain or regularize**, instead of retraining the whole model

But finding that layer or channel is hard



51,200 maps for **25** images, in **one** layer

Class Separability and why it matters

IDENTIFYING SEPA

1. makes a r
2. gives clea
3. exposes r
4. tells us wh retraining

But finding th

Key Visualization Challenges

C1 **Abundance of activation channels**

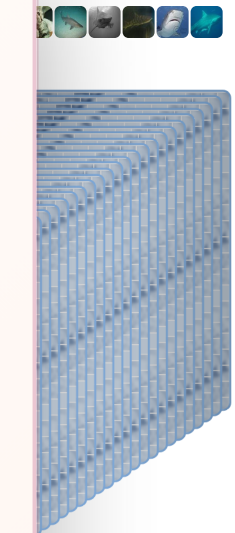
analyzing more than a few images, across thousands of channels, becomes tedious

C2 **Activation similarity is hard to perceive**

images side by side are hard to compare, and so are magnitudes across images

C3 **Layer and channel contributions are hard to identify**

few mechanisms quantify which layers or channels help the task



Trace class separability from the dataset,
to each layer, down to individual activation channels,
by comparing summaries instead of raw activations.

Trace class separability from the dataset, to each layer, down to individual activation channels, by comparing summaries instead of raw activations.

ChannelExplorer coordinated views, linked selections

Scatterplot View
inter- and intra-class confusion in each layer

Jaccard Similarity View
exact overlap of strongly activated channels

Heatmap View
channel patterns that identify classes, or not

any model with image-like layers — CNN · ResNet · GAN · U-Net · Stable Diffusion

TensorFlow PyTorch

Model overview InceptionV3 · last Inception block

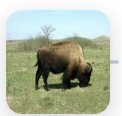
- **Layers** as nodes, **weights** as edges



Model overview

InceptionV3 · last Inception block

- **Layers** as nodes, **weights** as edges
- Inputs **flow** through every layer



input

mixed9

Details
Activations
Scatterplot View
Jaccard Similarity View
Confusion Hierarchy
Heatmap View

conv2d_85

Details
Activations
Scatterplot View
Jaccard Similarity View
Confusion Hierarchy
Heatmap View

conv2d_87

Details
Activations
Scatterplot View
Jaccard Similarity View
Confusion Hierarchy
Heatmap View

conv2d_88

Details
Activations
Scatterplot View
Jaccard Similarity View
Confusion Hierarchy
Heatmap View

mixed10

Details
Activations
Scatterplot View
Jaccard Similarity View
Confusion Hierarchy
Heatmap View

top-1: **bison**

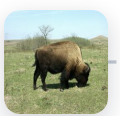
predictions

Details
Predictions
Scatterplot View
Jaccard Similarity View
Confusion Hierarchy
Heatmap View

Model overview

InceptionV3 · last Inception block

- **Layers** as nodes, **weights** as edges
- Inputs **flow** through every layer
- **Details on demand**: open a layer's panels



input

mixed9

Details
Activations
Scatterplot View
Jaccard Similarity View
Confusion Hierarchy
Heatmap View

conv2d_85

Details
Activations
Scatterplot View
Jaccard Similarity View
Confusion Hierarchy
Heatmap View

conv2d_87

Details
Activations
Scatterplot View
Jaccard Similarity View
Confusion Hierarchy
Heatmap View

conv2d_88

Details
Activations
Scatterplot View
Jaccard Similarity View
Confusion Hierarchy
Heatmap View

mixed10

Details

- **Layer** : Concatenate
- **Output** : (8, 8, 2048)
- **Channels** : 2048

Activations



‹ 1 / 86 ›

Scatterplot View
Jaccard Similarity View
Confusion Hierarchy
Heatmap View

top-1: **bison**

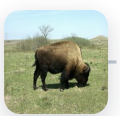
predictions

Details
Predictions
Scatterplot View
Jaccard Similarity View
Confusion Hierarchy
Heatmap View

Model overview

InceptionV3 · last Inception block

- **Layers** as nodes, **weights** as edges
- Inputs **flow** through every layer
- **Details on demand**: open a layer's panels
- Every layer holds the **views**



input

mixed9

Details
Activations
Scatterplot View
Jaccard Similarity View
Confusion Hierarchy
Heatmap View

conv2d_85

Details
Activations
Scatterplot View
Jaccard Similarity View
Confusion Hierarchy
Heatmap View

conv2d_87

Details
Activations
Scatterplot View
Jaccard Similarity View
Confusion Hierarchy
Heatmap View

conv2d_88

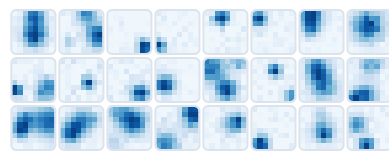
Details
Activations
Scatterplot View
Jaccard Similarity View
Confusion Hierarchy
Heatmap View

mixed10

Details

- **Layer** : Concatenate
- **Output** : (8, 8, 2048)
- **Channels** : 2048

Activations



‹ 1 / 86 ›

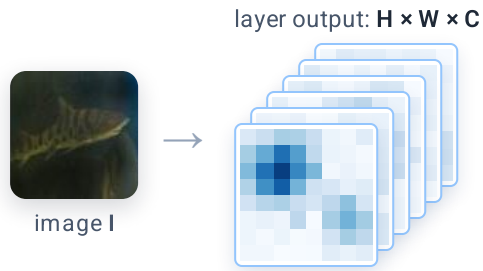
Scatterplot View
Jaccard Similarity View
Confusion Hierarchy
Heatmap View

top-1: **bison**

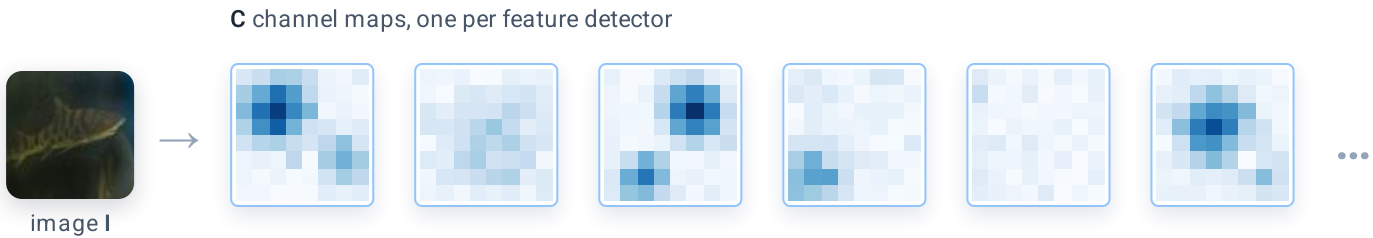
predictions

Details
Predictions
Scatterplot View
Jaccard Similarity View
Confusion Hierarchy
Heatmap View

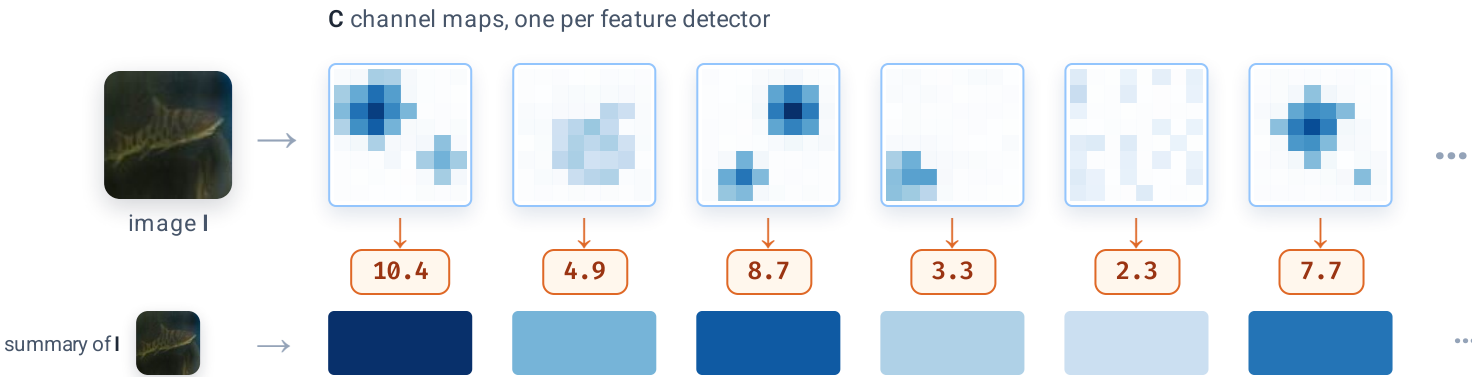
Activation summarization



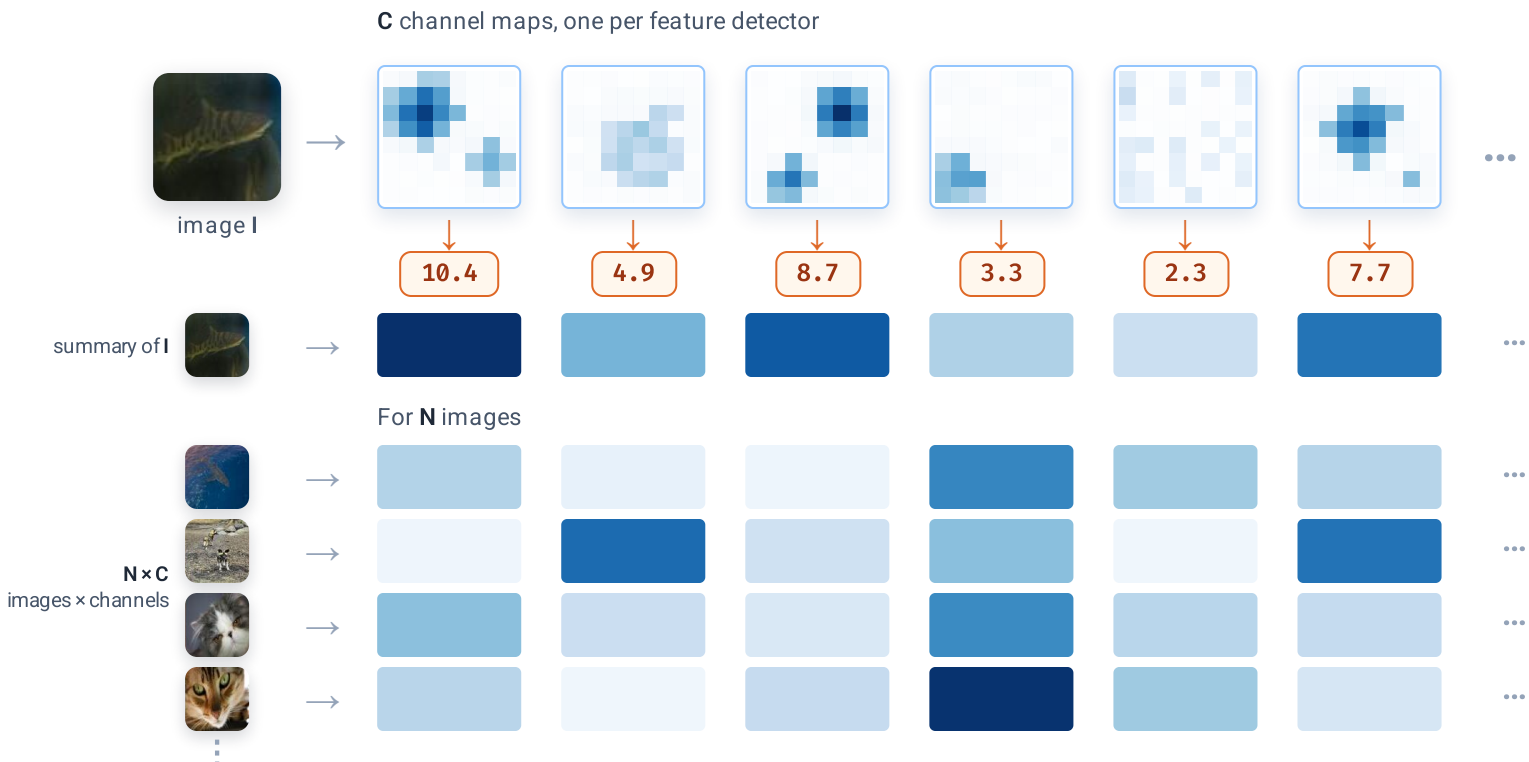
Activation summarization



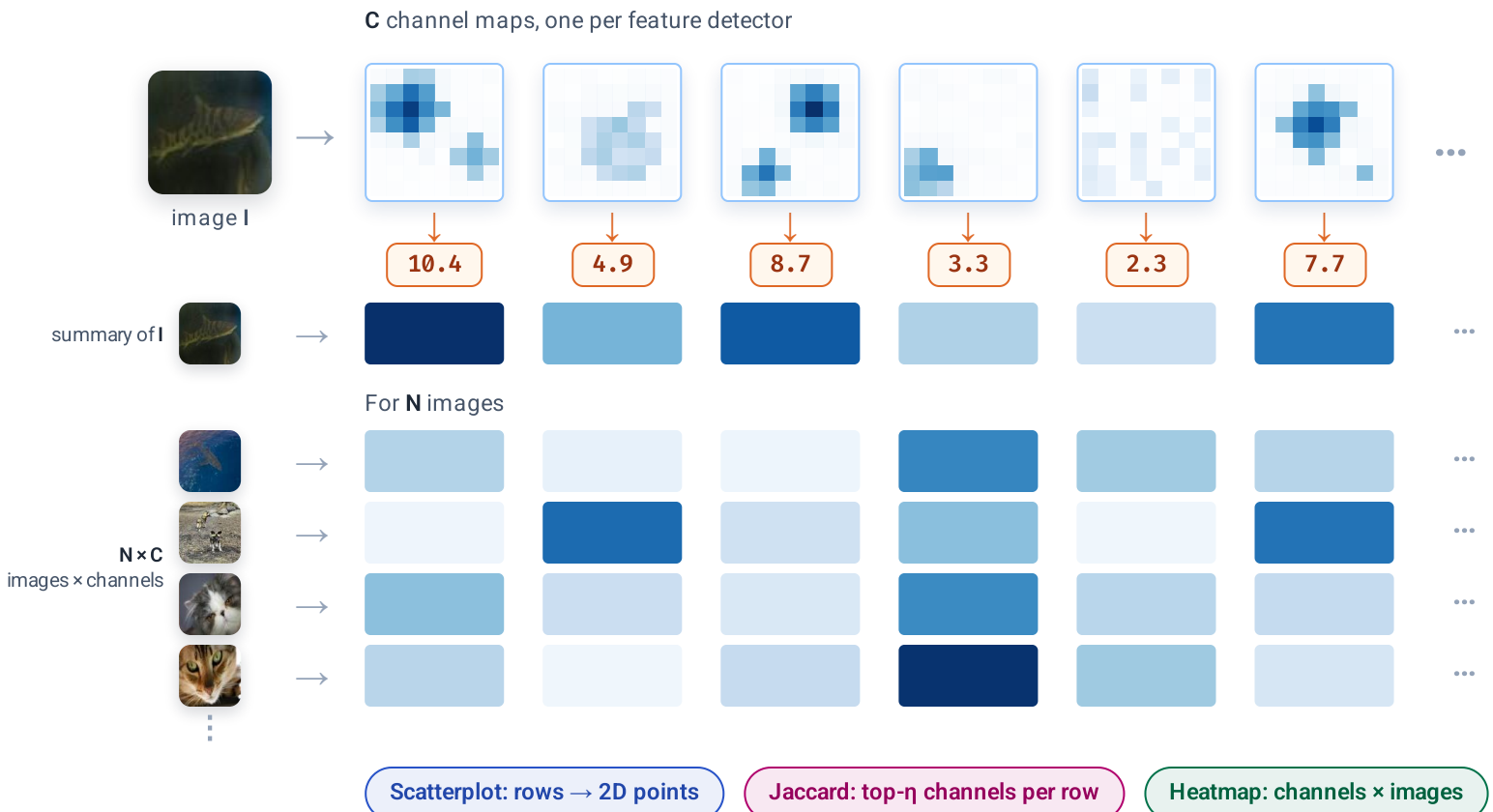
Activation summarization



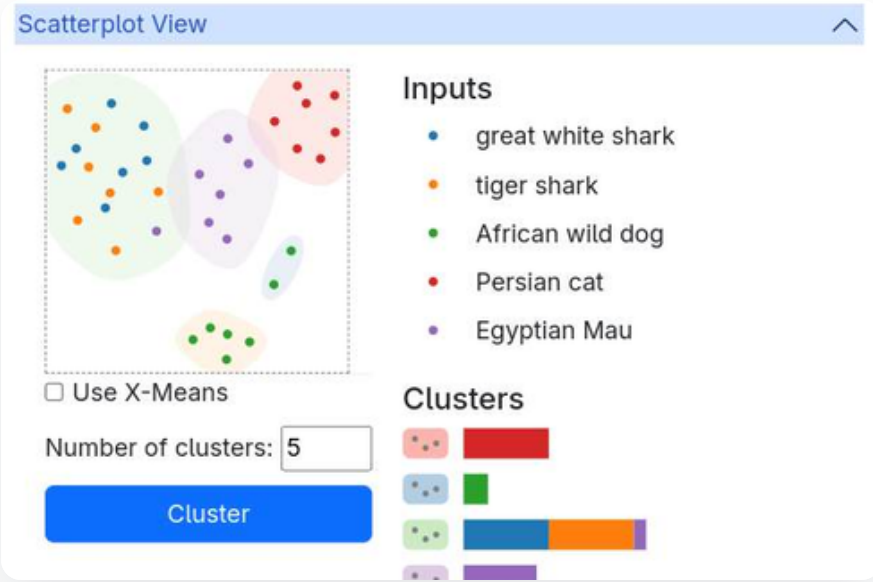
Activation summarization



Activation summarization



Scatterplot View one point per input

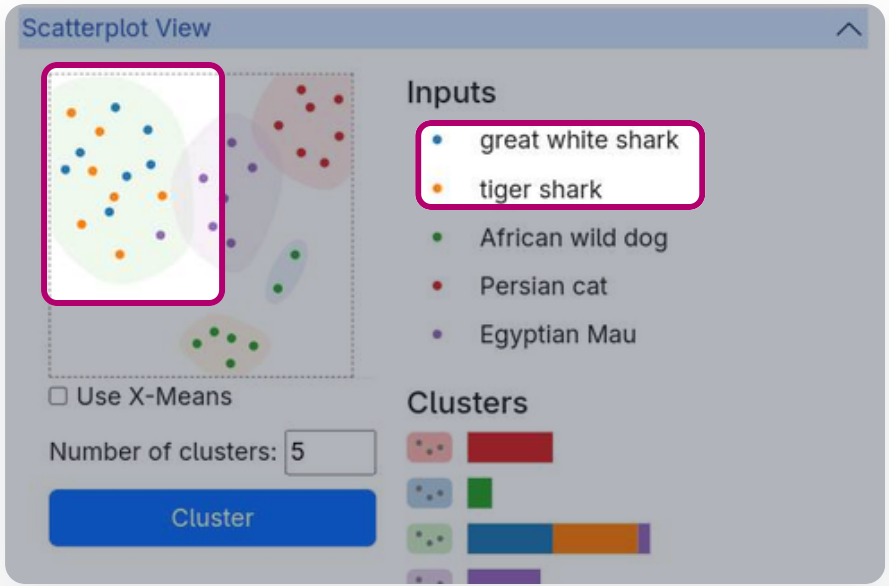


Real view from the tool: five classes, each dot is one image

Classes overlap
the model mixes them: inter-class confusion

One class splits
it has distinct activation patterns: intra-class confusion

Scatterplot View one point per input

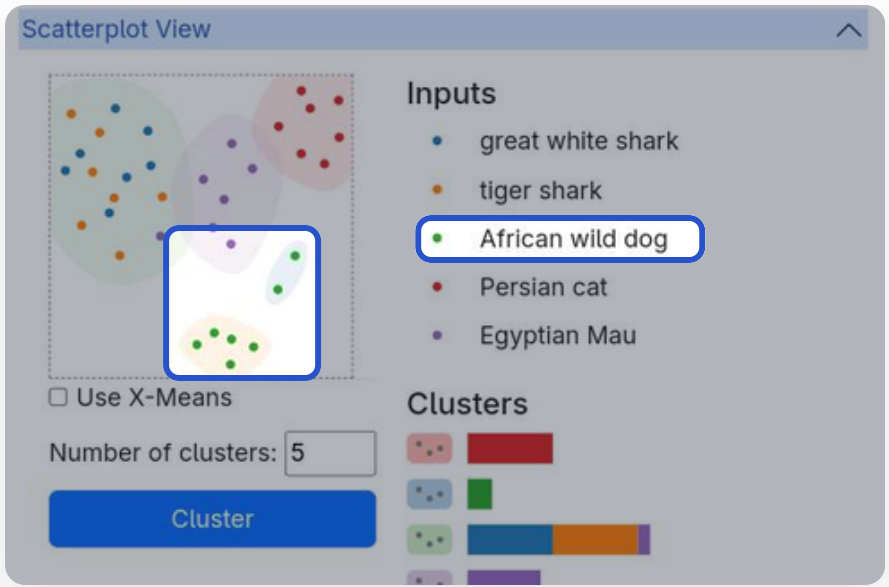


Overlap: white and tiger sharks mix, an inter-class confusion

Classes overlap
the model mixes them: inter-class confusion

One class splits
it has distinct activation patterns: intra-class confusion

Scatterplot View one point per input

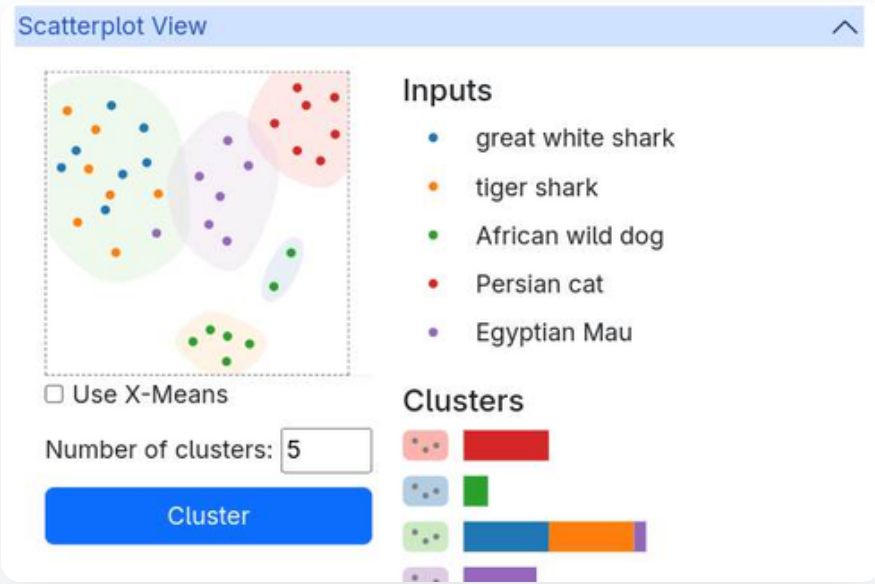


Split: African wild dogs form two clusters, an intra-class confusion

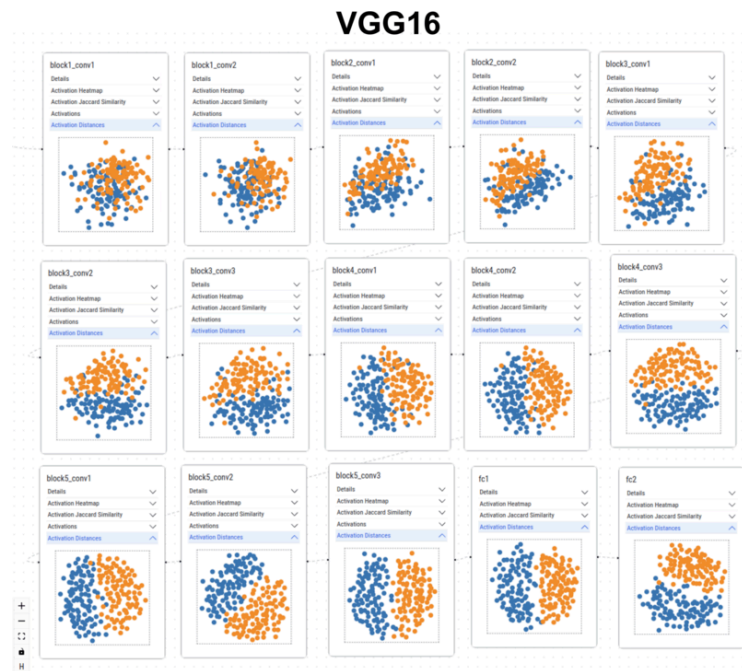
Classes overlap
the model mixes them: inter-class confusion

One class splits
it has distinct activation patterns: intra-class confusion

Scatterplot View one point per input

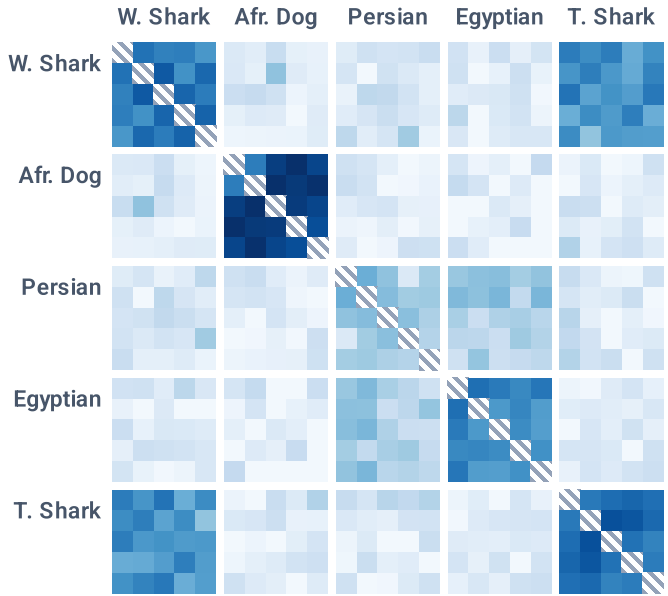


Real view from the tool: five classes, each dot is one image

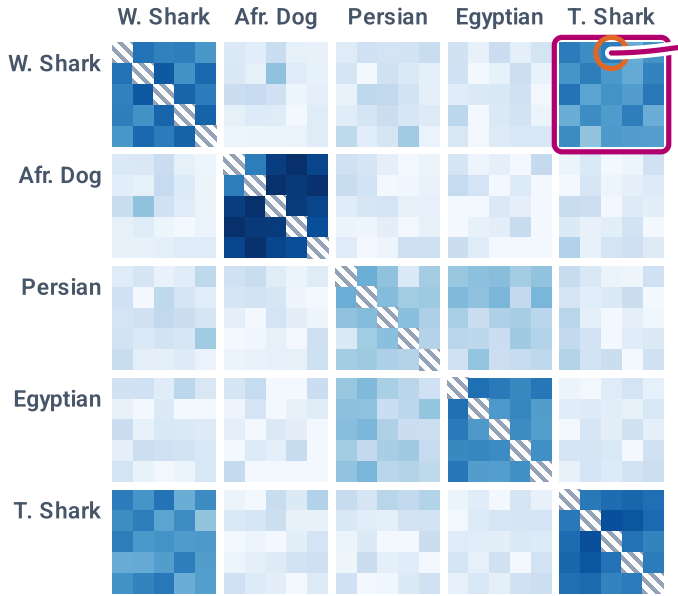


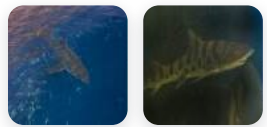
Compare layers side by side: in VGG16, two classes separate gradually with depth

Jaccard Similarity View



Jaccard Similarity View



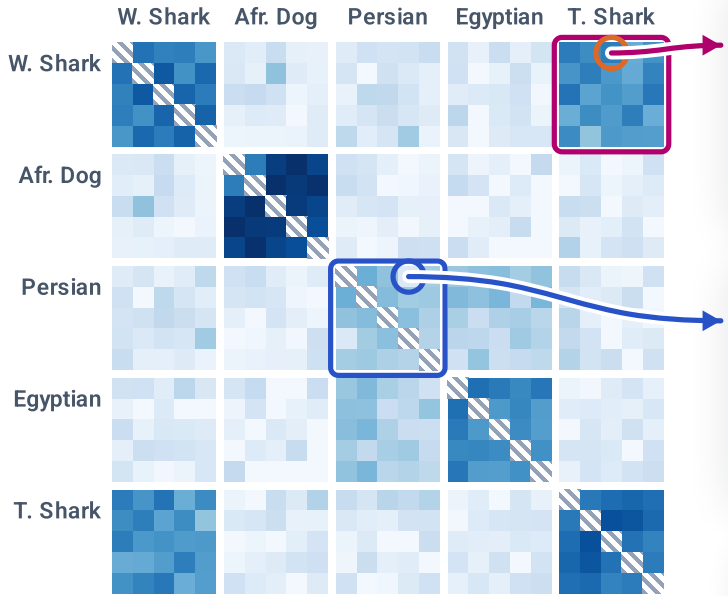


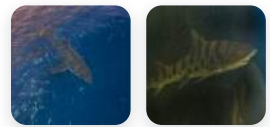
J = **0.60**

white vs. tiger shark

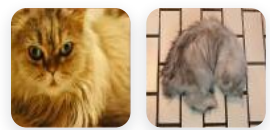
Dark off-diagonal block: inter-class confusion, White ↔ Tiger Shark

Jaccard Similarity View





J = 0.60
white vs. tiger shark

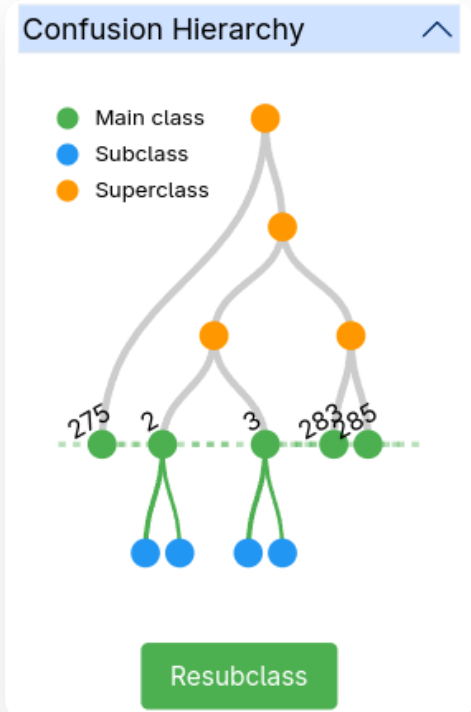


J = 0.12
two Persian cats

Dark off-diagonal block: inter-class confusion, White ↔ Tiger Shark

Light diagonal block: intra-class confusion, Persian Cat

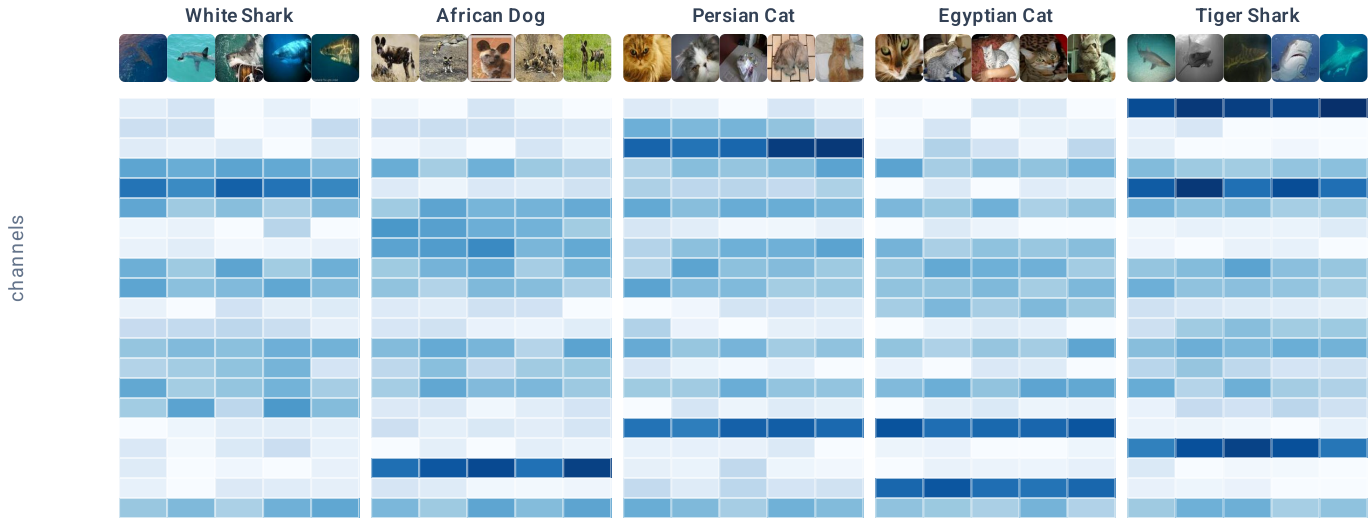
Confusion hierarchy built automatically for every layer



Confusion hierarchy, built per layer: superclasses by agglomerative clustering of class similarity, subclasses by clustering within a class

Heatmap View

which channels identify a class



columns: images, grouped by class

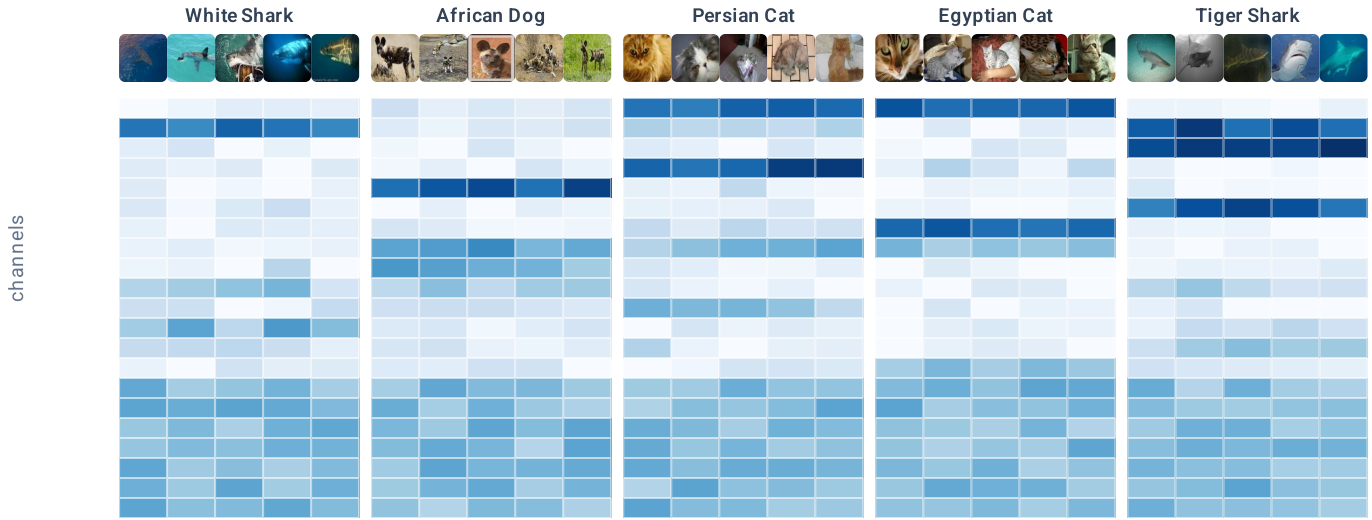
rows: activation channels

rows ordered by variance, ζ, or edge weight

thresholded at the 10th percentile

Heatmap View

which channels identify a class



Stripes: dark across one class, so the channel detects class-identifying features

No stripes: sporadic everywhere, likely little contribution

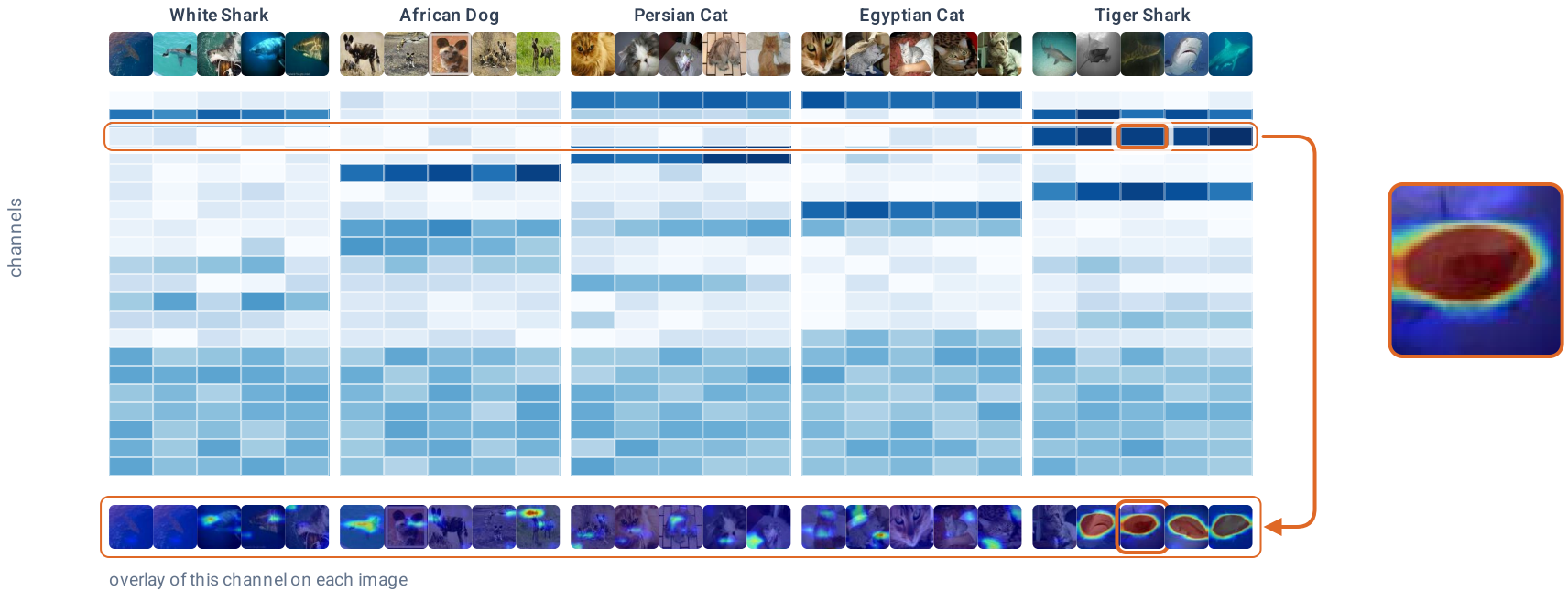
columns: images, grouped by class

rows: activation channels

rows ordered by variance, ζ, or edge weight

thresholded at the 10th percentile

Heatmap View which channels identify a class



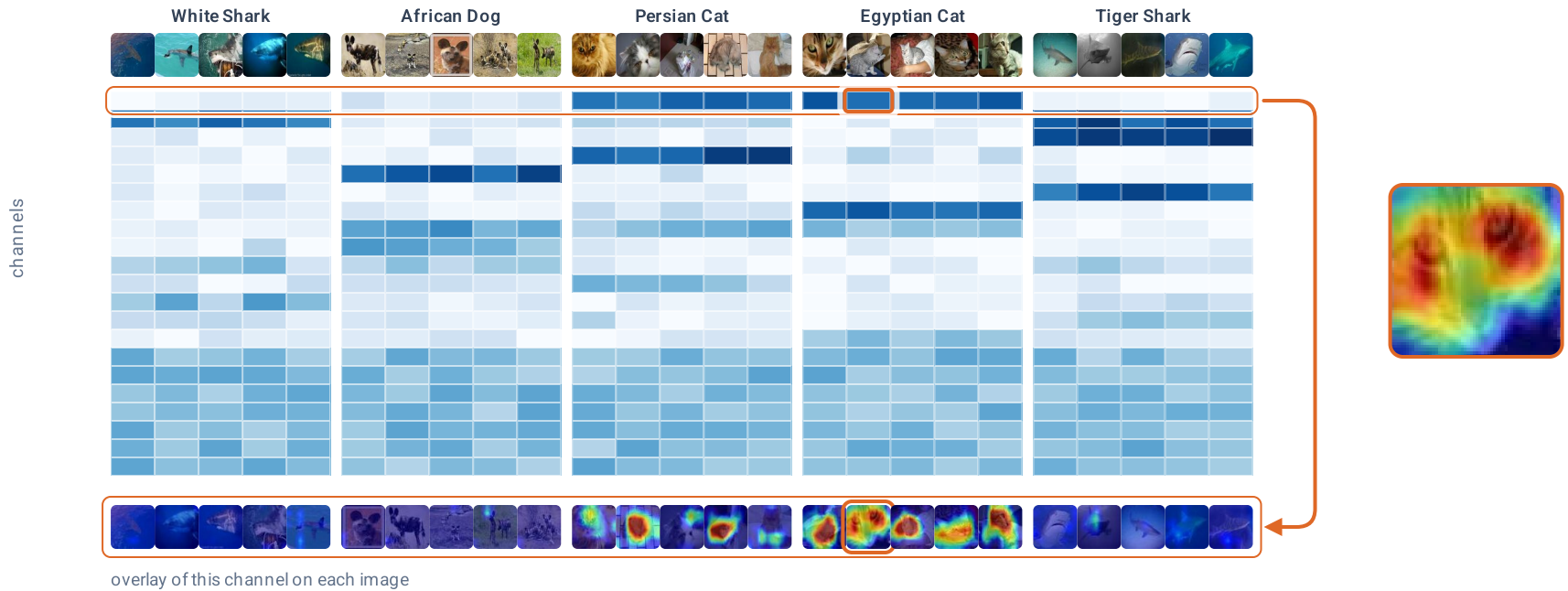
columns: images, grouped by class

rows: activation channels

rows ordered by variance, ζ, or edge weight

thresholded at the 10th percentile

Heatmap View which channels identify a class



columns: images, grouped by class

rows: activation channels

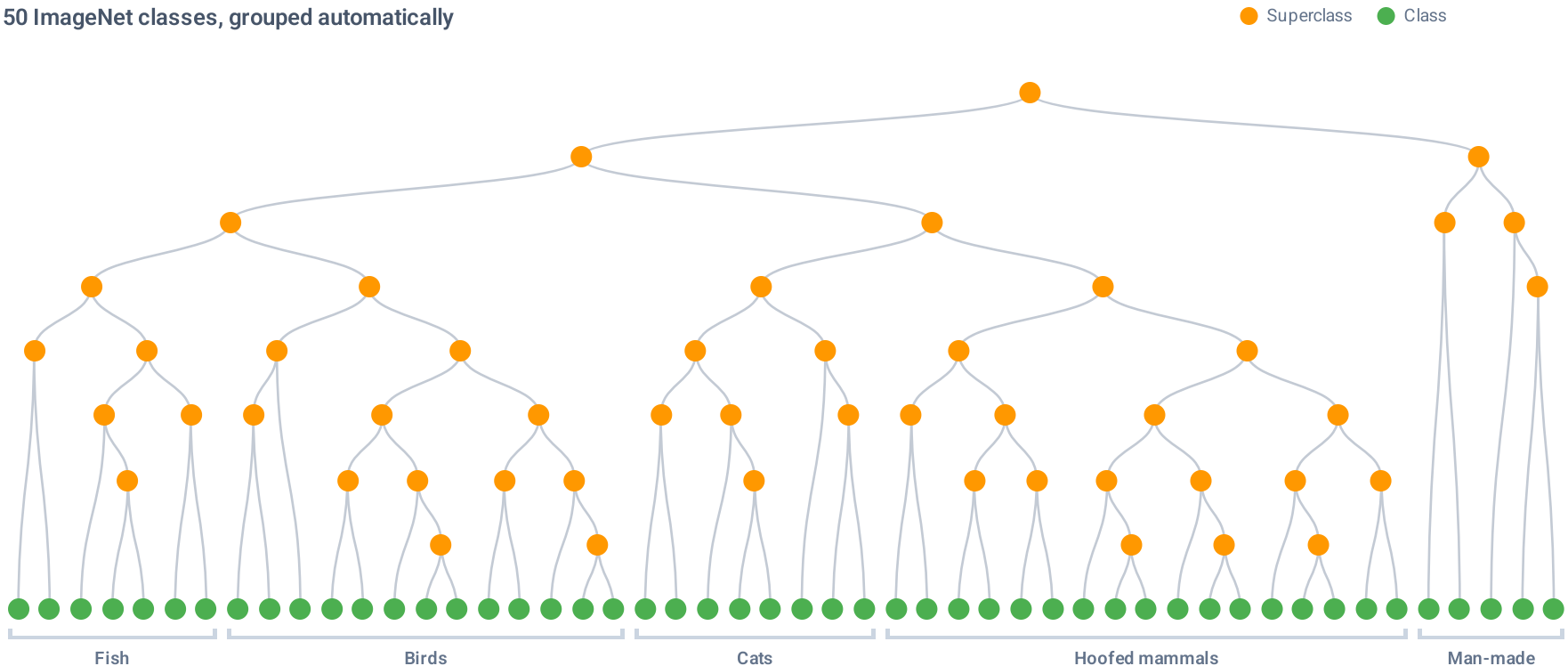
rows ordered by variance, ζ, or edge weight

thresholded at the 10th percentile

Use case: discovering a class hierarchy

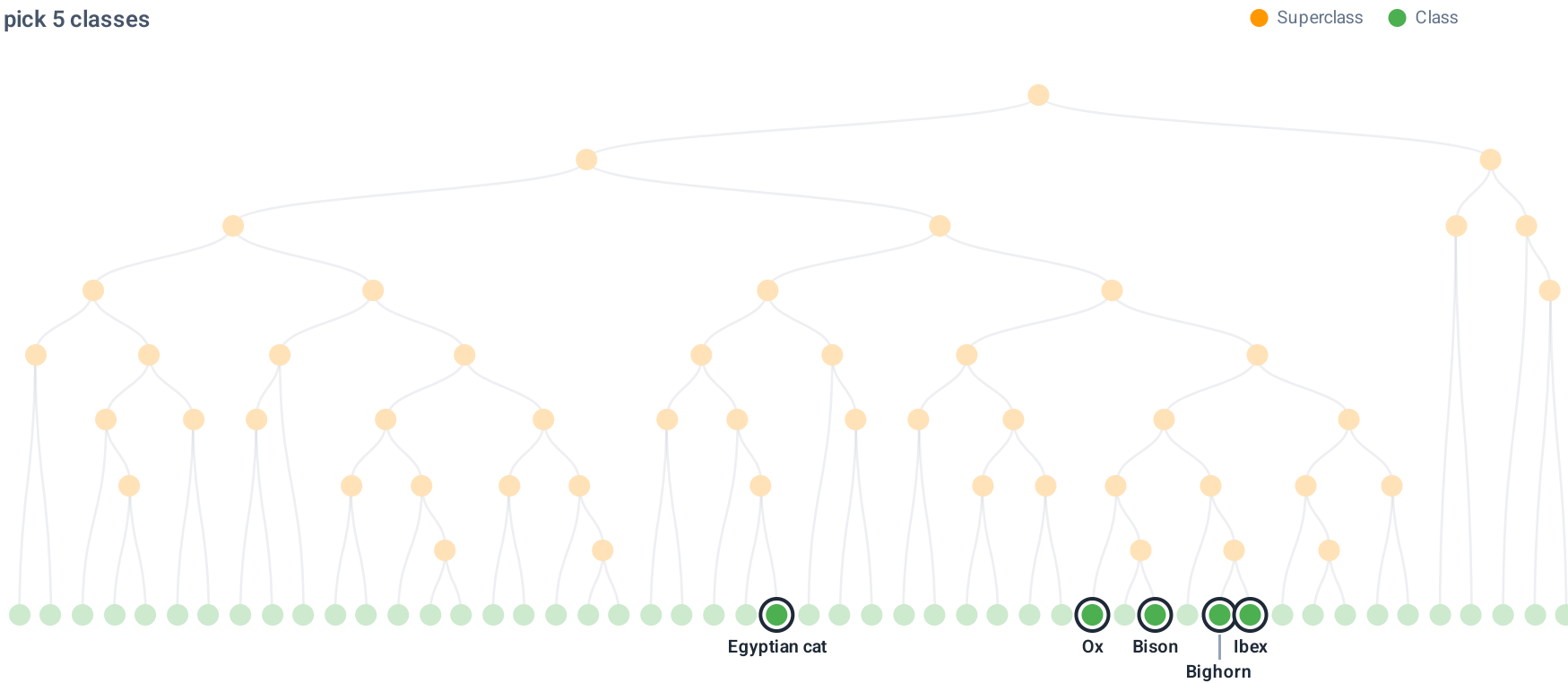
InceptionV3 · mixed10 · ImageNet

50 ImageNet classes, grouped automatically



Use case: discovering a class hierarchy

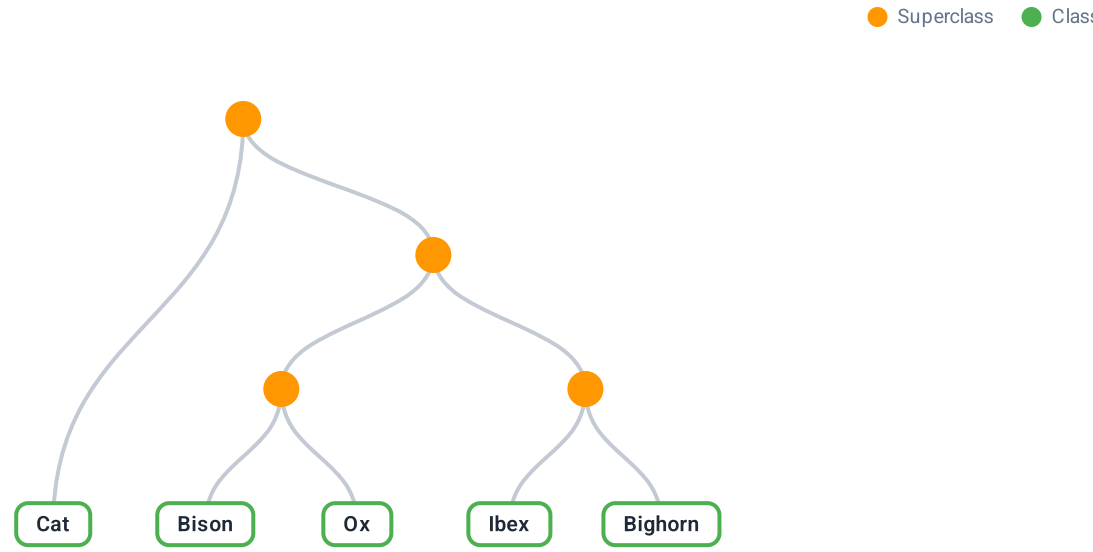
InceptionV3 · mixed10 · ImageNet



Use case: discovering a class hierarchy

InceptionV3 · mixed10 · ImageNet

rebuild the hierarchy for these 5



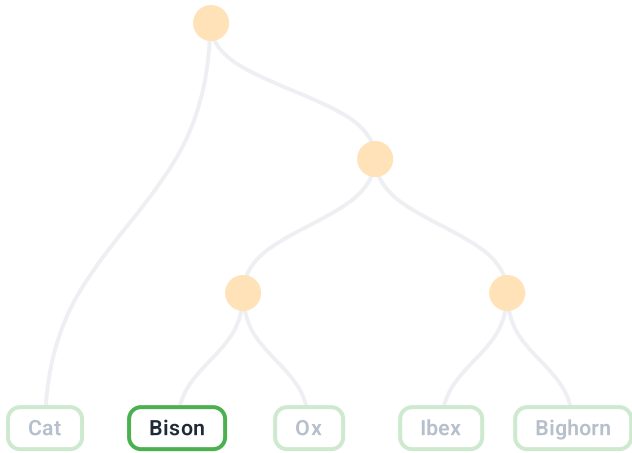
Use case: discovering a class hierarchy

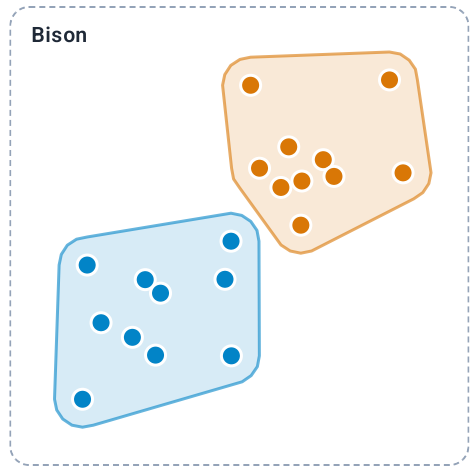
InceptionV3 · mixed10 · ImageNet

Scatterplot View

Bison forms two clusters

Superclass Class



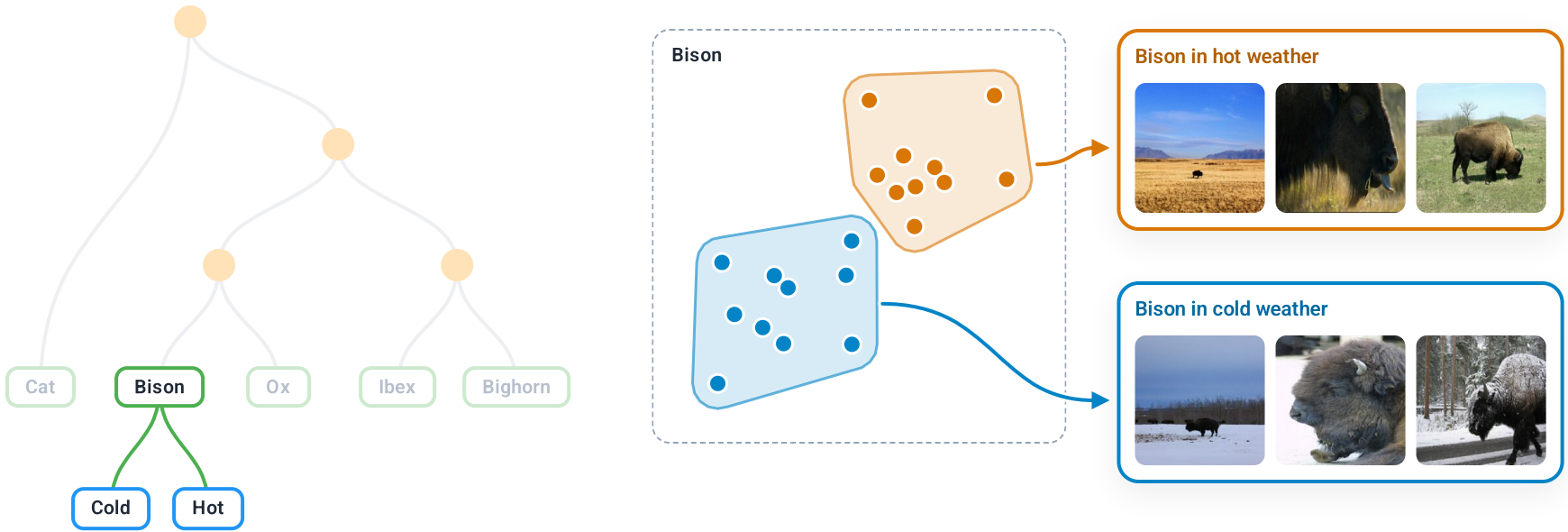


Use case: discovering a class hierarchy

InceptionV3 · mixed10 · ImageNet

Scatterplot View

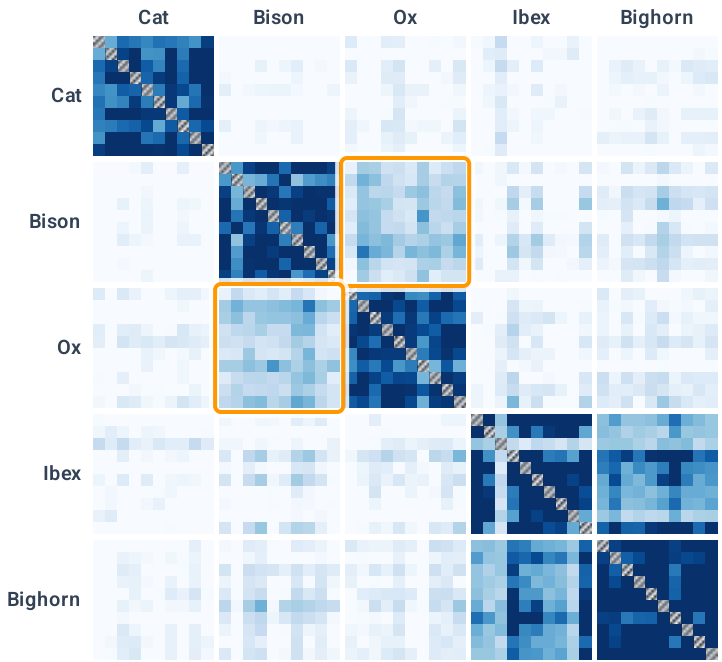
cold vs. hot weather: 2 subclasses

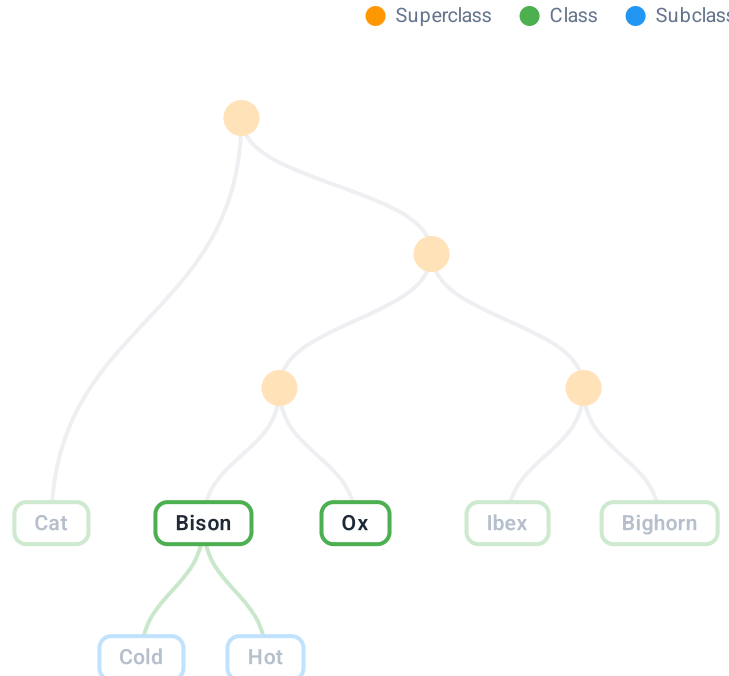


Use case: discovering a class hierarchy

InceptionV3 · mixed10 · ImageNet

Jaccard Similarity View Bison and Ox share many channels

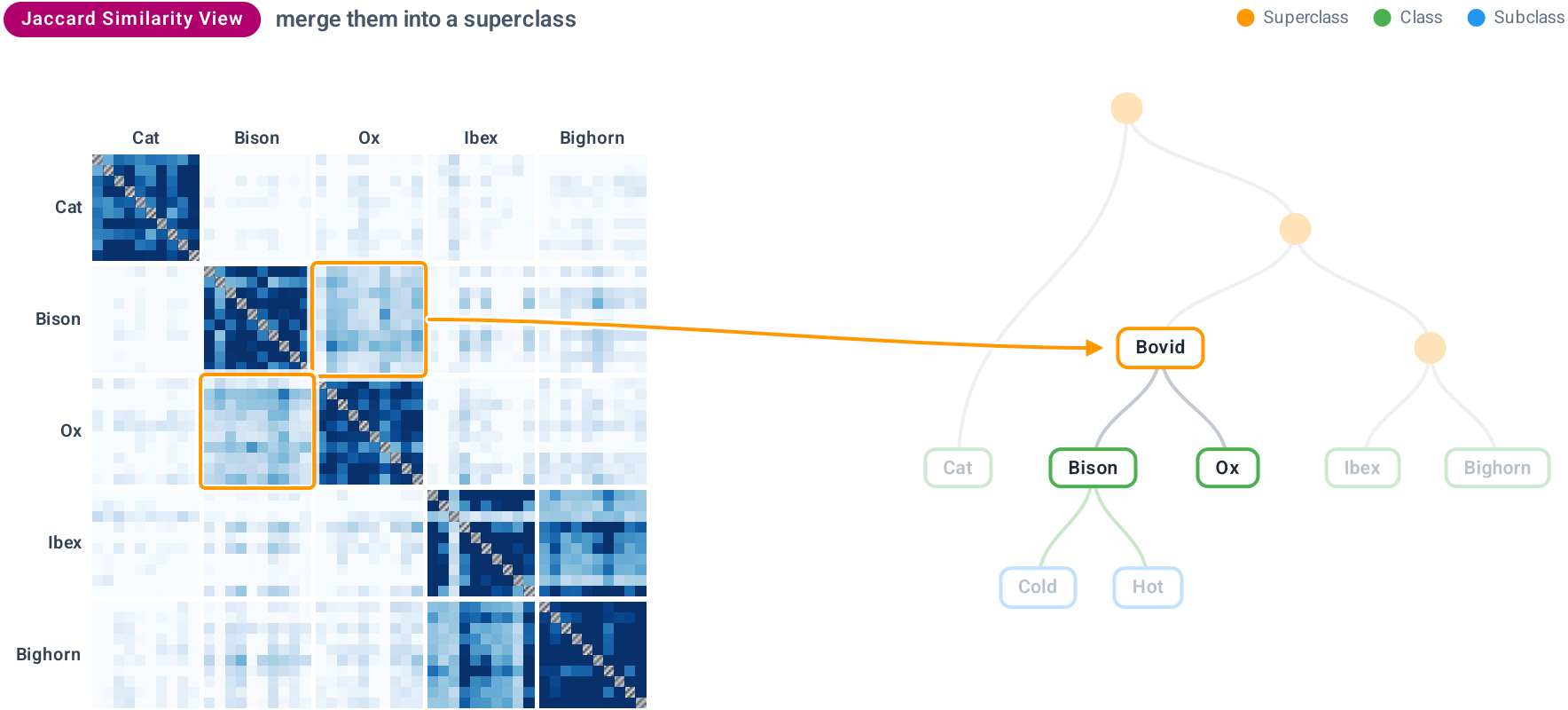




Use case: discovering a class hierarchy

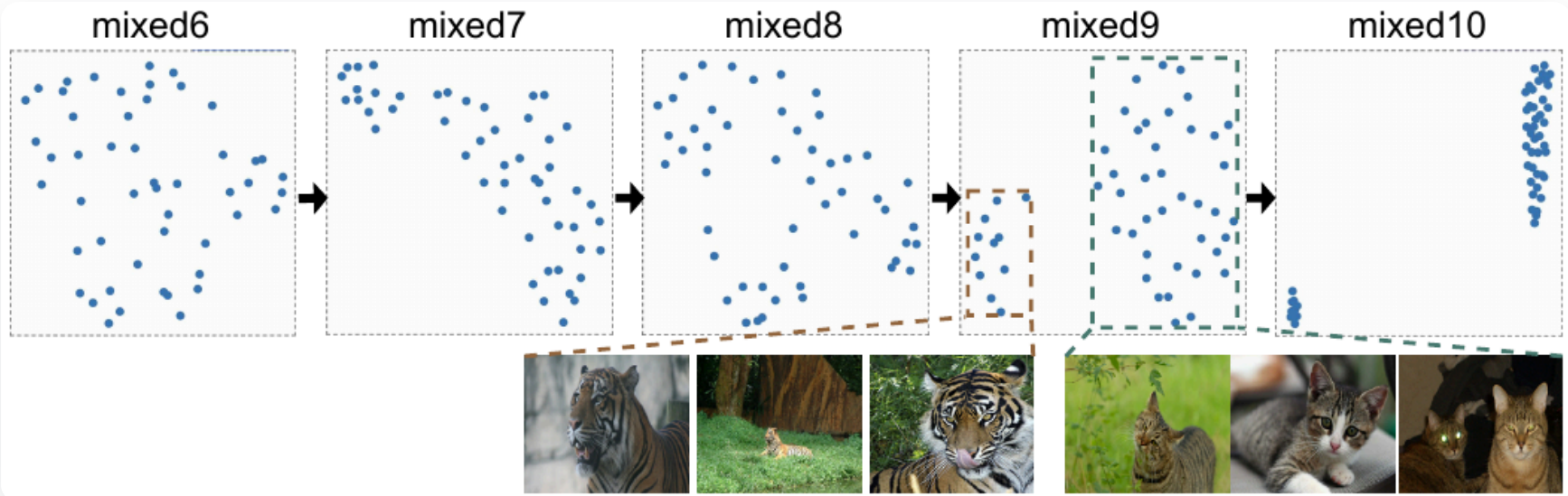
InceptionV3 · mixed10 · ImageNet

Jaccard Similarity View merge them into a superclass



Use case: finding mislabeled images

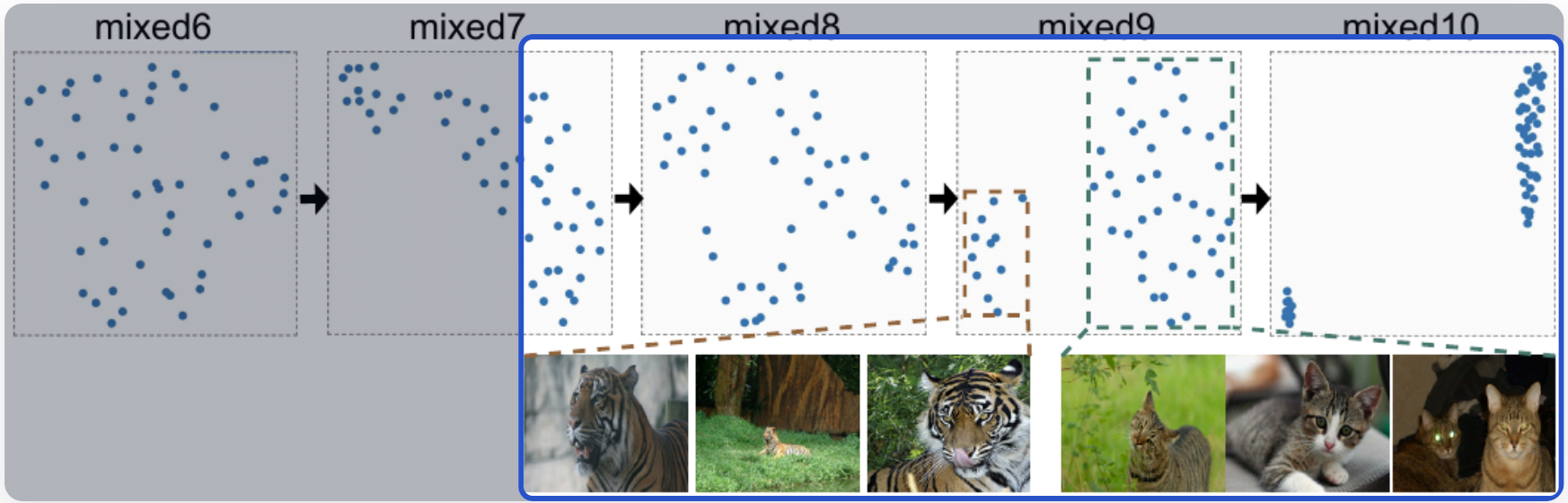
InceptionV3 · 50 Tiger Cat images



> The hierarchy shows two subclasses, and from `mixed8` on a small group separates

Use case: finding mislabeled images

InceptionV3 · 50 Tiger Cat images

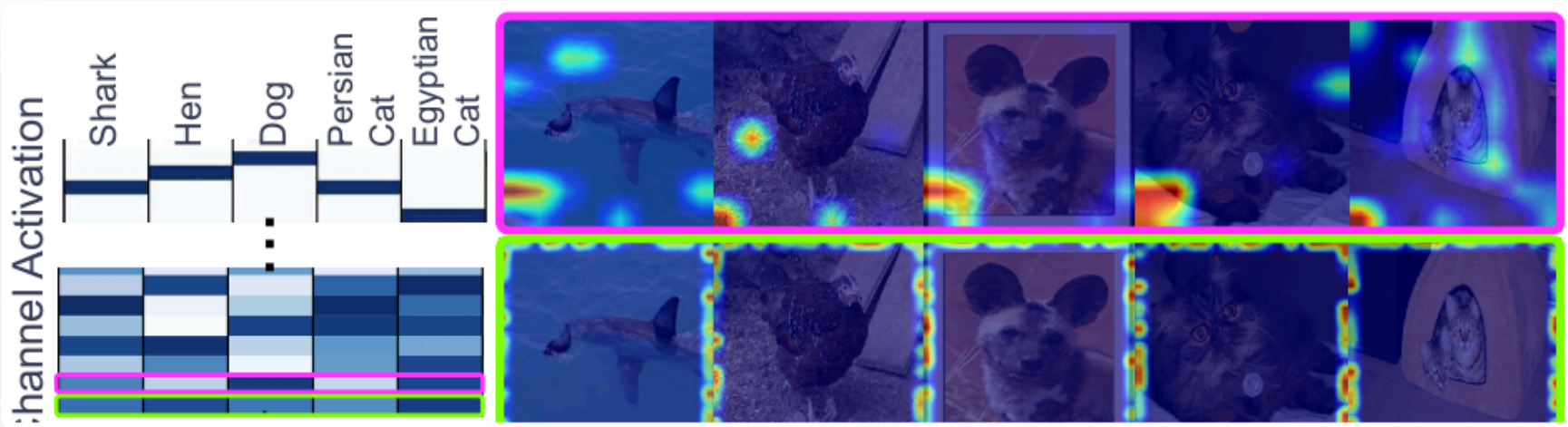


The hierarchy shows two subclasses, and from `mixed8` on a small group separates

Those points are **tigers**, likely mislabeled: deeper channels fire for one group and stay silent for the other

Use case: pruning channels

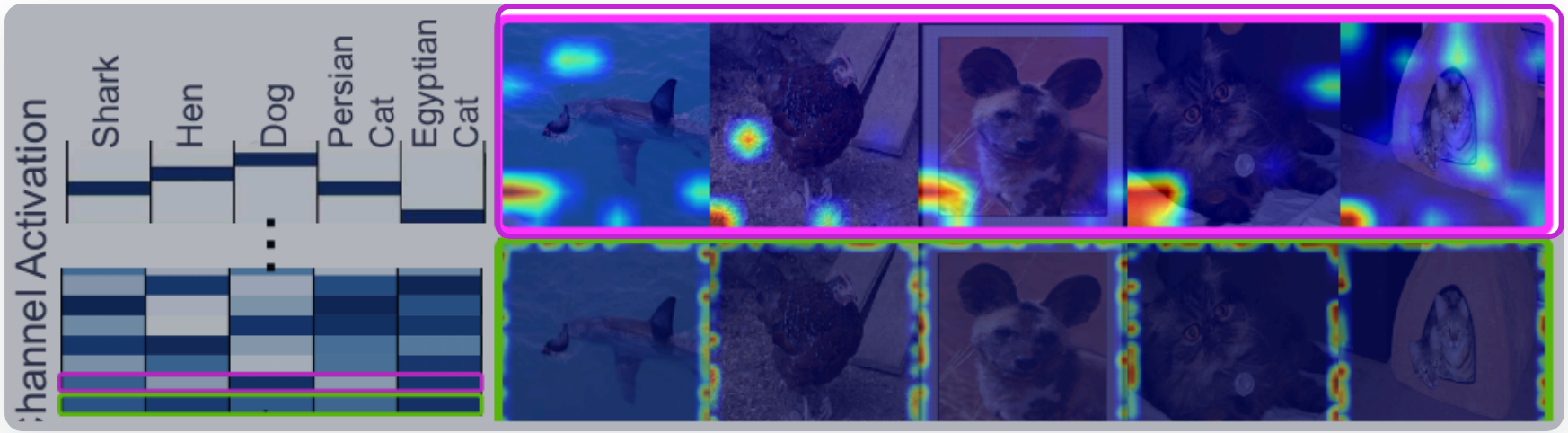
InceptionV3 · mixed10 · bottom rows of the Heatmap View



Channels ranked lowest by class-pairwise distance fire **sporadically** across every class

Use case: pruning channels

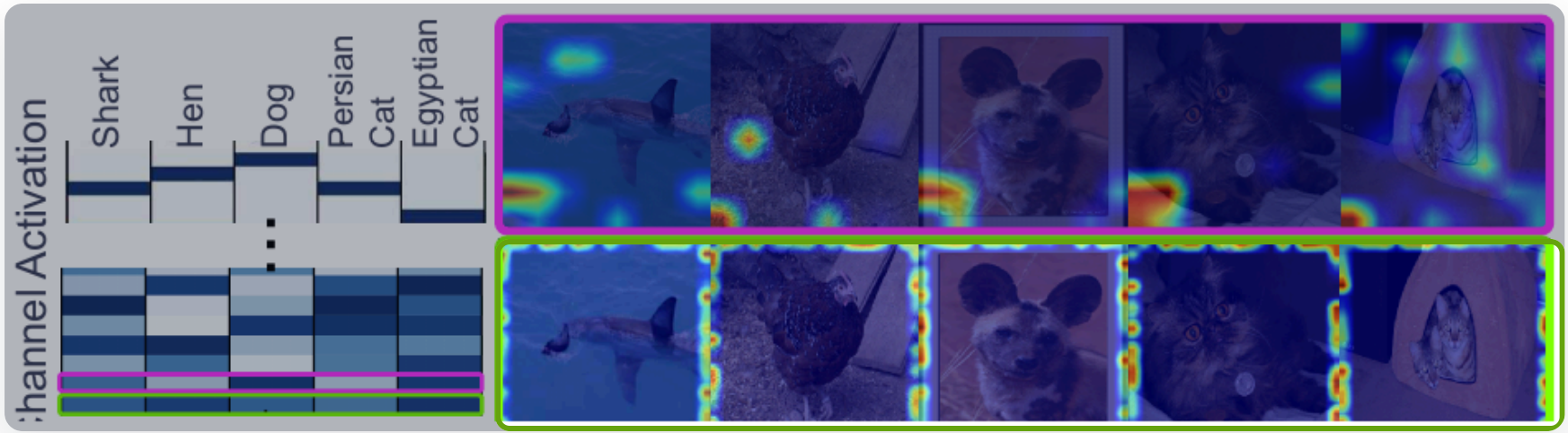
InceptionV3 · mixed10 · bottom rows of the Heatmap View



Corner channel: always fires in the bottom-left corner

Use case: pruning channels

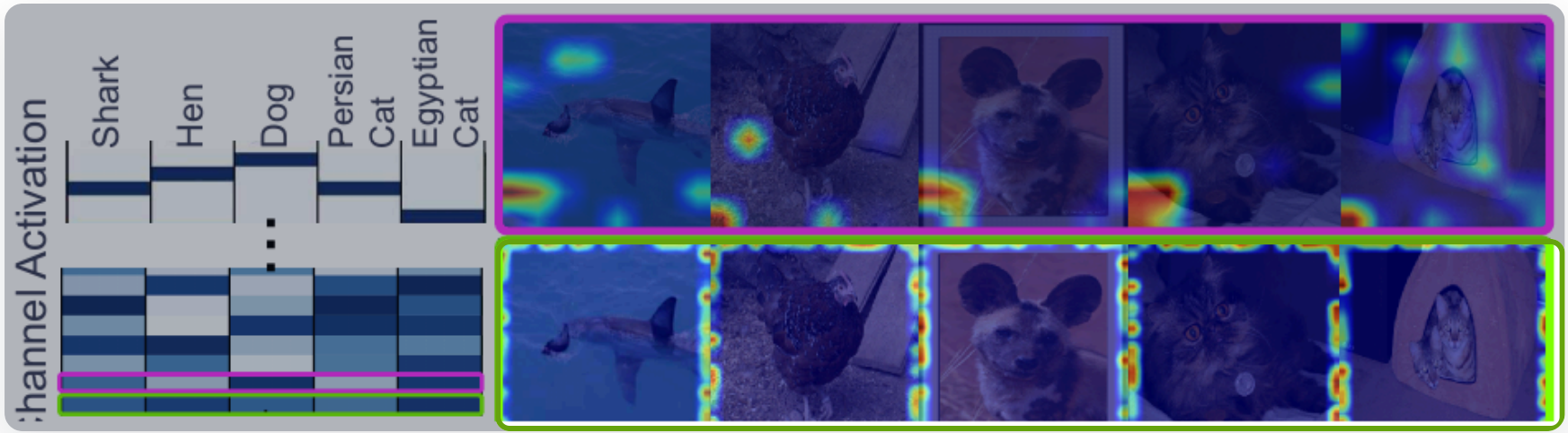
InceptionV3 · mixed10 · bottom rows of the Heatmap View



Border channel: fires at the image border, whatever the content

Use case: pruning channels

InceptionV3 · mixed10 · bottom rows of the Heatmap View



Border channel: fires at the image border, whatever the content

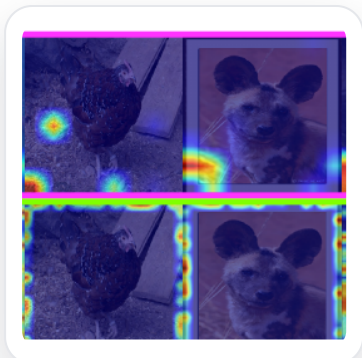
74%
of mixed10 channels removed

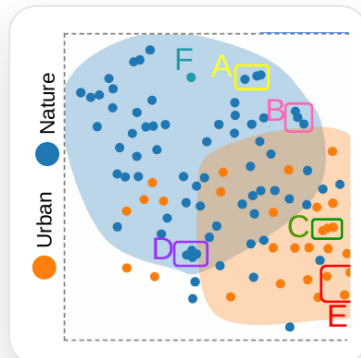
0.500 → 0.498
mAP on 50,000 ImageNet validation images, 5 runs each

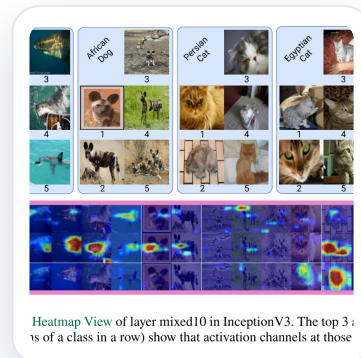
12.1%
faster CPU inference

≈ 50%
faster GPU inference after ~70% removed; memory drops linearly

Final discussion

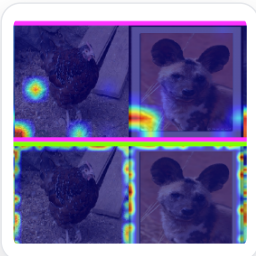


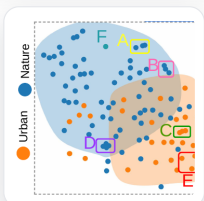


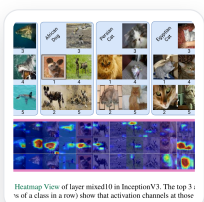


Heatmap View of layer mixed10 in InceptionV3. The top 3 a
ns of a class in a row) show that activation channels at those

Final discussion



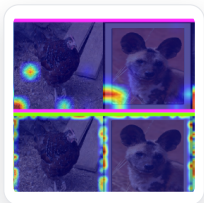




Localize, then fix

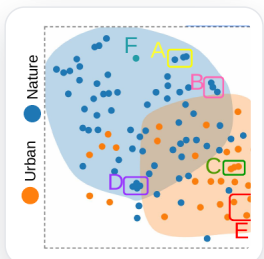
Separability failures in any deep network trace to a few **layers**, or even single **channels**. That enables targeted fixes, such as new subclasses, relabeling, or pruning, instead of retraining the whole model.

Final discussion



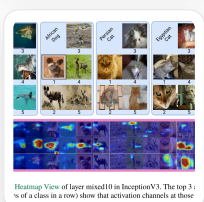
Localize, then fix

Separability failures in any deep network trace to a few **layers**, or even single **channels**. That enables targeted fixes, such as new subclasses, relabeling, or pruning, instead of retraining the whole model.

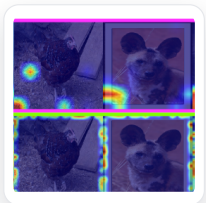


Limitations

Only **image-based layers** for now, iterative models such as Diffusion only **one timestep at a time**, and no automatic mapping of **activation semantics**.

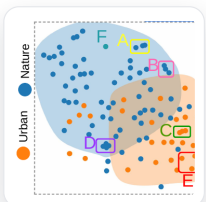


Final discussion



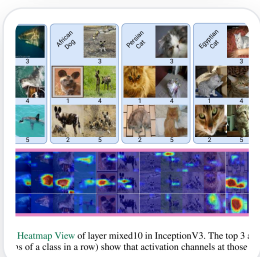
Localize, then fix

Separability failures in any deep network trace to a few **layers**, or even single **channels**. That enables targeted fixes, such as new subclasses, relabeling, or pruning, instead of retraining the whole model.



Limitations

Only **image-based layers** for now, iterative models such as Diffusion only **one timestep at a time**, and no automatic mapping of **activation semantics**.



Future work

Already in progress: **attention and Vision Transformers**, with AI labeling channel and activation semantics automatically, opening the way to **LLMs**, GPT models, and other **multimodal models**.

Thank You!



channelexplorer.netlify.app

ChannelExplorer: Exploring Class Separability Through Activation Channel Visualization

Authors

Rahat Zaman

Bei Wang

Paul Rosen

Resources

DOI 10.1109/TVCG.2026.3669148

arXiv 2505.04647

rahatzamancse/ChannelExplorer

PyPI pip install channelexplorer

Presenter

Rahat Zaman

rahat.zaman@utah.edu

rahatzaman.me

Other Backup Slides

Implementation

open source, a few lines around your own model

`channelexplorer` on PyPI

backends wrap your own model and dataset

TensorFlow PyTorch

Analysis

pluggable summaries, t-SNE · PCA · MDS · UMAP, X-Means, confusion hierarchy

scikit-learn · umap-learn · pyclustering · SciPy

FastAPI server + Redis cache

long computations run as background tasks, results stay cached

Next.js frontend

React Flow node-link overview, D3 views, Redux state

```python
from channelexplorer import ChannelExplorer_TF, metrics

server = ChannelExplorer_TF(
    model=model,                # any keras.Model
    dataset=dataset,            # tf.data of (image, label)
    label_names=labels,
    preprocess=preprocess,
    preprocess_inverse=preprocess_inv,
    summary_fn_image=metrics.summary_fn_image_l2,
    log_level="info",
)
server.run(host="localhost", port=8000)
```

Stable Diffusion U-Net · ≈ 860M params

EfficientNet · 160+ image layers

200 inputs in 16 GB RAM

```
pip install channelexplorer[all]
```

Implementation

open source, a few lines around your own model

`channelexplorer` on PyPI

backends wrap your own model and dataset

TensorFlow · PyTorch

Analysis

pluggable summaries, t-SNE · PCA · MDS · UMAP, X-Means, confusion hierarchy

scikit-learn · umap-learn · pyclustering · SciPy

FastAPI server + Redis cache

long computations run as background tasks, results stay cached

Next.js frontend

React Flow node-link overview, D3 views, Redux state

```python
from channelexplorer import APAnalysisTorchModel

server = APAnalysisTorchModel(
    model=model,                # any torch.nn.Module
    input_shape=(1, 3, 224, 224),
    dataset=dataset,            # (image, label) pairs
    label_names=labels,
    log_level="info",
)
server.run(host="localhost", port=8000)
```

Stable Diffusion U-Net · ≈ 860M params

EfficientNet · 160+ image layers

200 inputs in 16 GB RAM

```
pip install channelexplorer[all]
```

Implementation

open source, a few lines around your own model

`channelexplorer` **on PyPI**

backends wrap your own model and dataset

TensorFlow PyTorch

Analysis

pluggable summaries, t-SNE · PCA · MDS · UMAP, X-Means, confusion hierarchy

~~scikit-learn · umap-learn · pyclustering · SciPy~~

FastAPI server + Redis cache

long computations run as background tasks, results stay cached

Next.js frontend

React Flow node-link overview, D3 views, Redux state

```python
import numpy as np
from channelexplorer import ChannelExplorer_TF

# (batch, H, W, C) → (batch, C)
def summary_fn_image_mean(x):
    return np.abs(x).mean(axis=(1, 2))

server = ChannelExplorer_TF(
    model=model,
    dataset=dataset,
    label_names=labels,
    summary_fn_image=summary_fn_image_mean,
)
```

Stable Diffusion U-Net · ≈ 860M params

EfficientNet · 160+ image layers

200 inputs in 16 GB RAM

`pip install channelexplorer[all]`

Implementation

open source, a few lines around your own model

`channelexplorer` **on PyPI**

backends wrap your own model and dataset

TensorFlow PyTorch

Analysis

pluggable summaries, t-SNE · PCA · MDS · UMAP, X-Means, confusion hierarchy

scikit-learn · umap-learn · pyclustering · SciPy

FastAPI server + Redis cache

long computations run as background tasks, results stay cached

Next.js frontend

React Flow node-link overview, D3 views, Redux state

```python
import numpy as np
from channelexplorer import ChannelExplorer_TF

# (batch, H, W, C) → (batch, C)
def summary_fn_image_mean(x):
    return np.abs(x).mean(axis=(1, 2))

server = ChannelExplorer_TF(
    model=model,
    dataset=dataset,
    label_names=labels,
    summary_fn_image=summary_fn_image_mean,
)
```

Stable Diffusion U-Net · ≈ 860M params

EfficientNet · 160+ image layers

200 inputs in 16 GB RAM

```
pip install channelexplorer[all]
```

Implementation

open source, a few lines around your own model

`channelexplorer` **on PyPI**

backends wrap your own model and dataset

TensorFlow PyTorch

Analysis

pluggable summaries, t-SNE · PCA · MDS · UMAP, X-Means, confusion hierarchy

scikit-learn · umap-learn · pyclustering · SciPy

FastAPI server + Redis cache

long computations run as background tasks, results stay cached

Next.js frontend

React Flow node-link overview, D3 views, Redux state

```python
import numpy as np
from channelexplorer import ChannelExplorer_TF

# (batch, H, W, C) → (batch, C)
def summary_fn_image_mean(x):
    return np.abs(x).mean(axis=(1, 2))

server = ChannelExplorer_TF(
    model=model,
    dataset=dataset,
    label_names=labels,
    summary_fn_image=summary_fn_image_mean,
)
```

Stable Diffusion U-Net · ≈ 860M params

EfficientNet · 160+ image layers

200 inputs in 16 GB RAM

`pip install channelexplorer[all]`

Use case: Stable Diffusion U-Net mid block · timestep 49 of 50



100 LLM-written scene prompts, as user-defined classes **Nature** and **Urban**

Use case: Stable Diffusion U-Net mid block · timestep 49 of 50



100 LLM-written scene prompts, as user-defined classes **Nature** and **Urban**

The two classes separate clearly

Use case: Stable Diffusion

U-Net mid block · timestep 49 of 50



100 LLM-written scene prompts, as user-defined classes **Nature** and **Urban**

The two classes separate clearly

Nearby points share concepts: misty open space, cliffs, crowds, large trees

Use case: Stable Diffusion

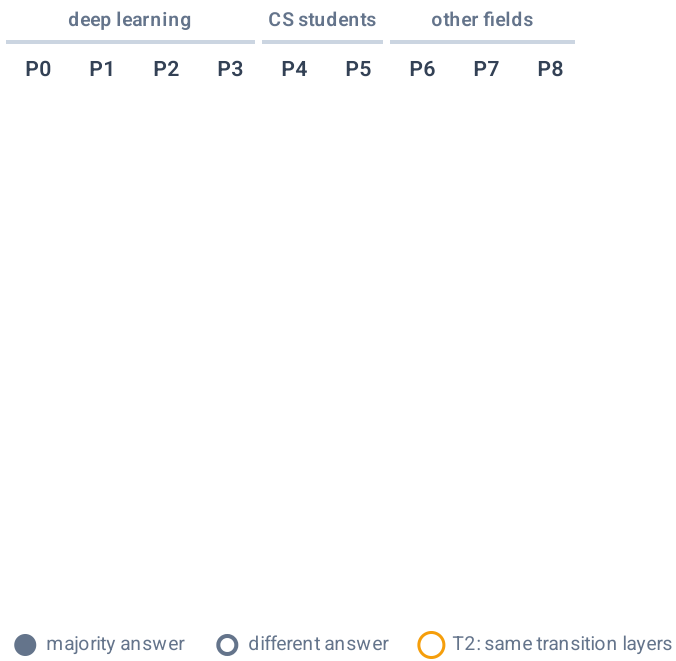
U-Net mid block · timestep 49 of 50



- 100 LLM-written scene prompts, as user-defined classes **Nature** and **Urban**
- Nearby points share concepts: misty open space, cliffs, crowds, large trees
- The two classes separate clearly
- Ambiguous prompt F, *"A large open space during day"*, lands near **Nature**. Caveat: some neighbors share nothing, so projections can mislead

User evaluation

9 graduate students · InceptionV3 · 25 images from 5 classes · 1 hour each

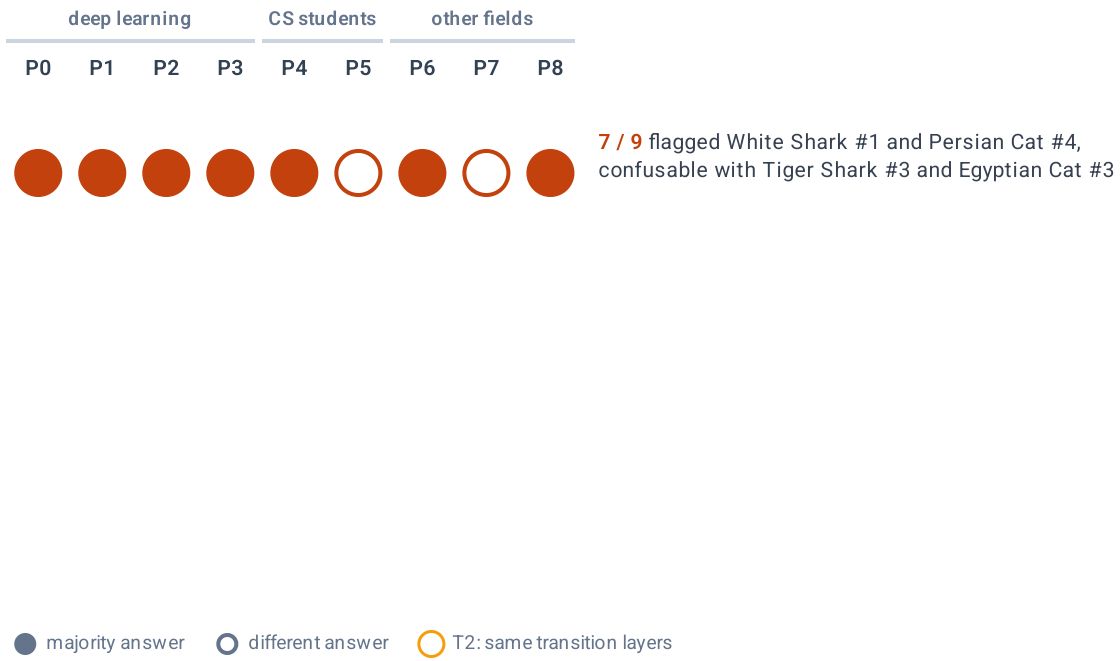


User evaluation

9 graduate students · InceptionV3 · 25 images from 5 classes · 1 hour each

T1 Dataset View

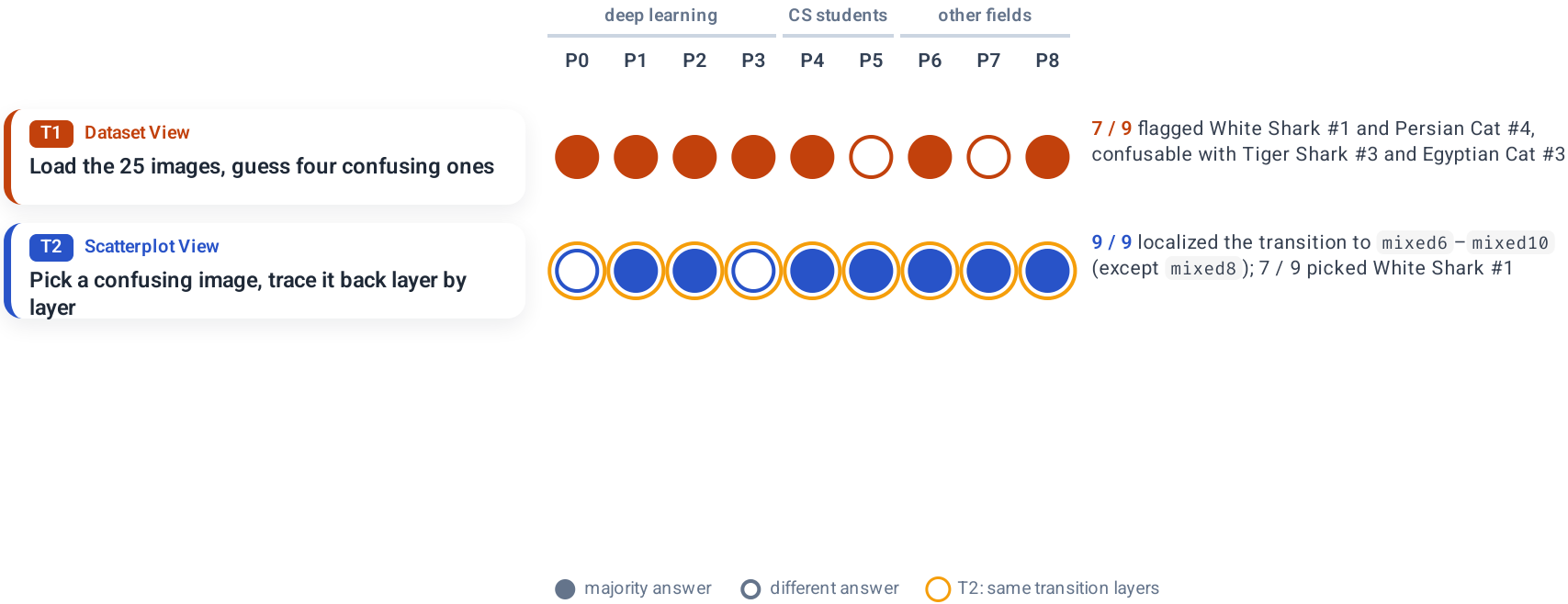
Load the 25 images, guess four confusing ones



7 / 9 flagged White Shark #1 and Persian Cat #4, confusable with Tiger Shark #3 and Egyptian Cat #3

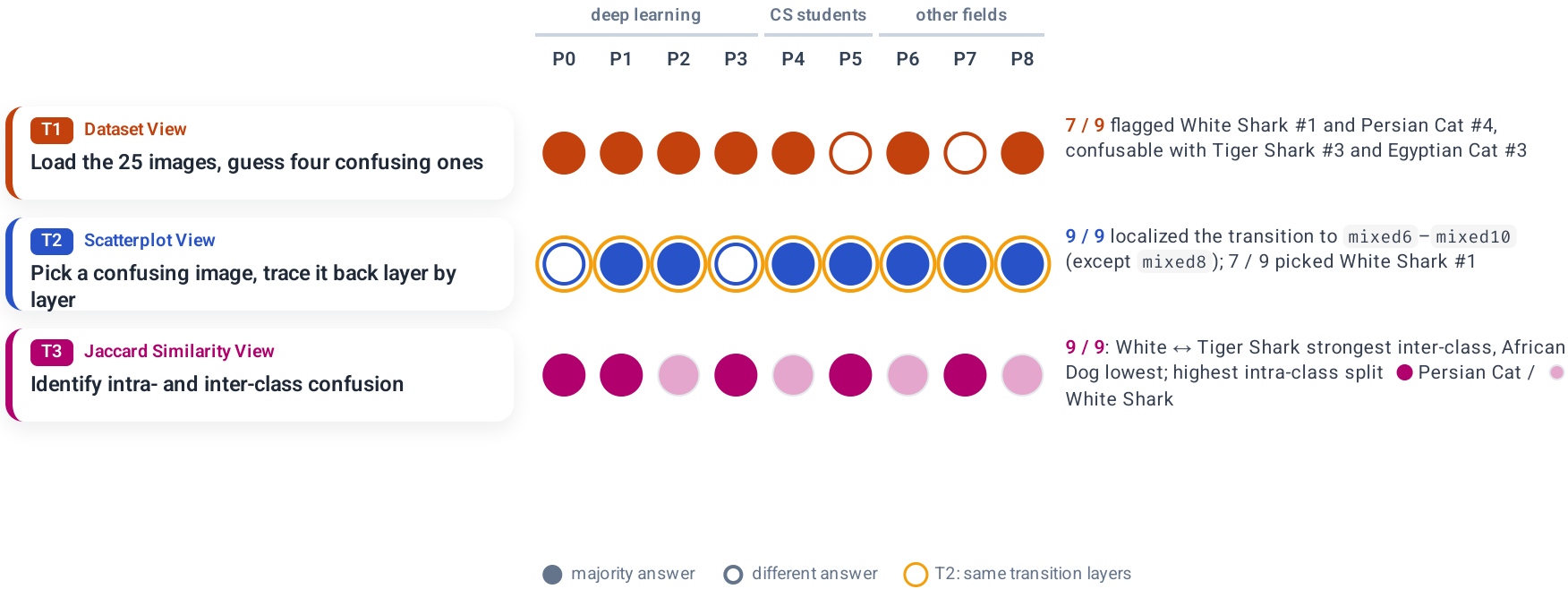
User evaluation

9 graduate students · InceptionV3 · 25 images from 5 classes · 1 hour each



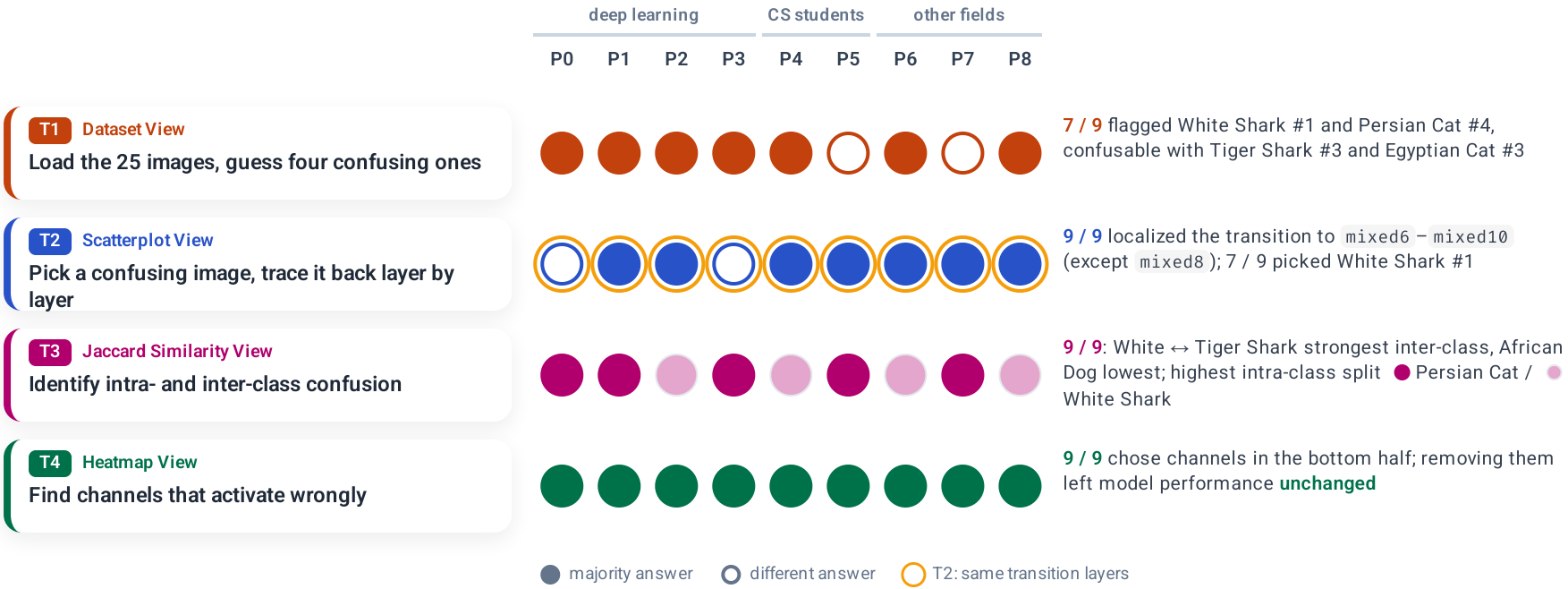
User evaluation

9 graduate students · InceptionV3 · 25 images from 5 classes · 1 hour each



User evaluation

9 graduate students · InceptionV3 · 25 images from 5 classes · 1 hour each



User evaluation

9 graduate students · InceptionV3 · 25 images from 5 classes · 1 hour each



T1 Dataset View

Load the 25 images, guess four confusing ones

7 / 9 flagged White Shark #1 and Persian Cat #4, confusable with Tiger Shark #3 and Egyptian Cat #3

T2 Scatterplot View

Pick a confusing image, trace it back layer by layer

9 / 9 localized the transition to `mixed6` – `mixed10` (except `mixed8`); 7 / 9 picked White Shark #1

T3 Jaccard Similarity View

Identify intra- and inter-class confusion

9 / 9: White ↔ Tiger Shark strongest inter-class, African Dog lowest; highest intra-class split ● Persian Cat / ● White Shark

T4 Heatmap View

Find channels that activate wrongly

9 / 9 chose channels in the bottom half; removing them left model performance **unchanged**

● majority answer ○ different answer ◯ T2: same transition layers

Findings

Participants consistently localized low-separability layers, compared confusion, and found pruning targets. Changing the projection or clustering did not change their decisions.

Limitations

The tool needs a solid understanding of neural networks, and it highlights regions to inspect rather than giving diagnoses.

study details
osf.io/8zusf

Analyzed Labels

- [x] tench
- [x] English springer
- [x] cassette player
- [x] chain saw

Input Images

External Images

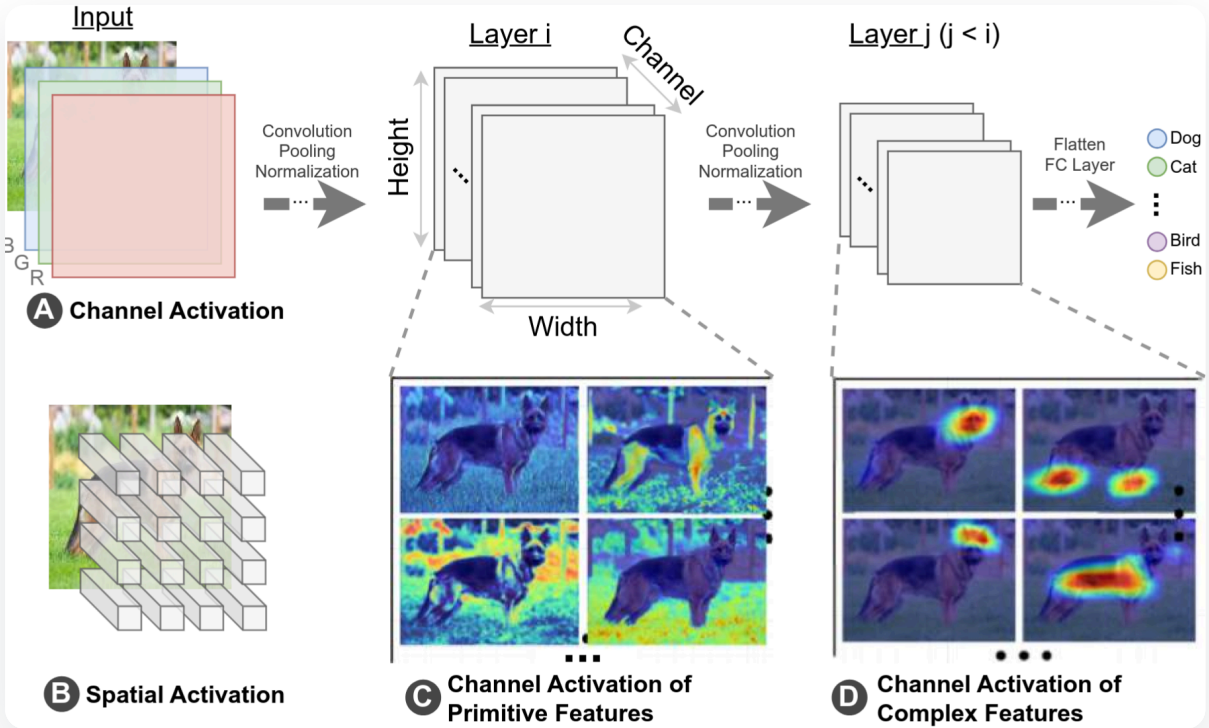
Choose Files

Class: tench




Inside a layer: activation channels

what an image-based layer produces

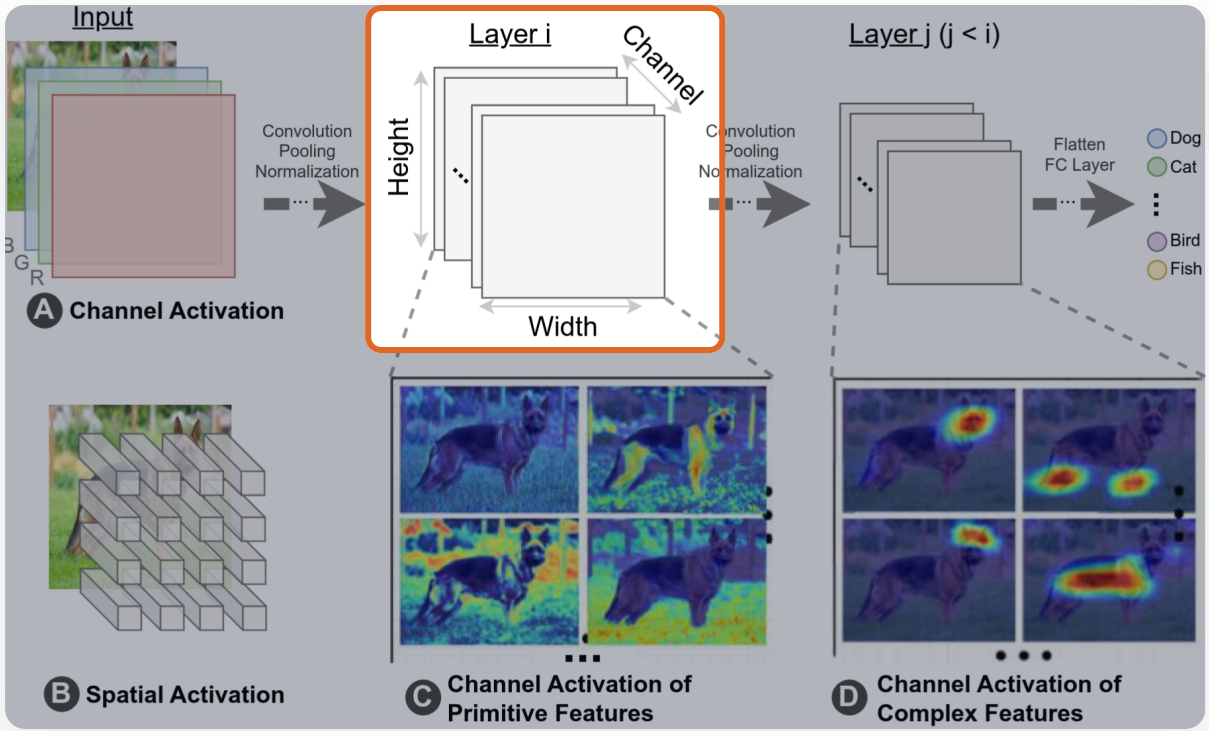


1. A layer outputs an **H × W × C** tensor
2. **Channel activation (A)**: one H × W map per filter. Spatial activation (B): one C-vector per pixel
3. **Early layers (C)**: primitive features such as edges, grass, and texture
4. **Later layers (D)**: complex, semantic parts such as a head or a body

Each channel is a feature detector we can map back onto the image

Inside a layer: activation channels

what an image-based layer produces

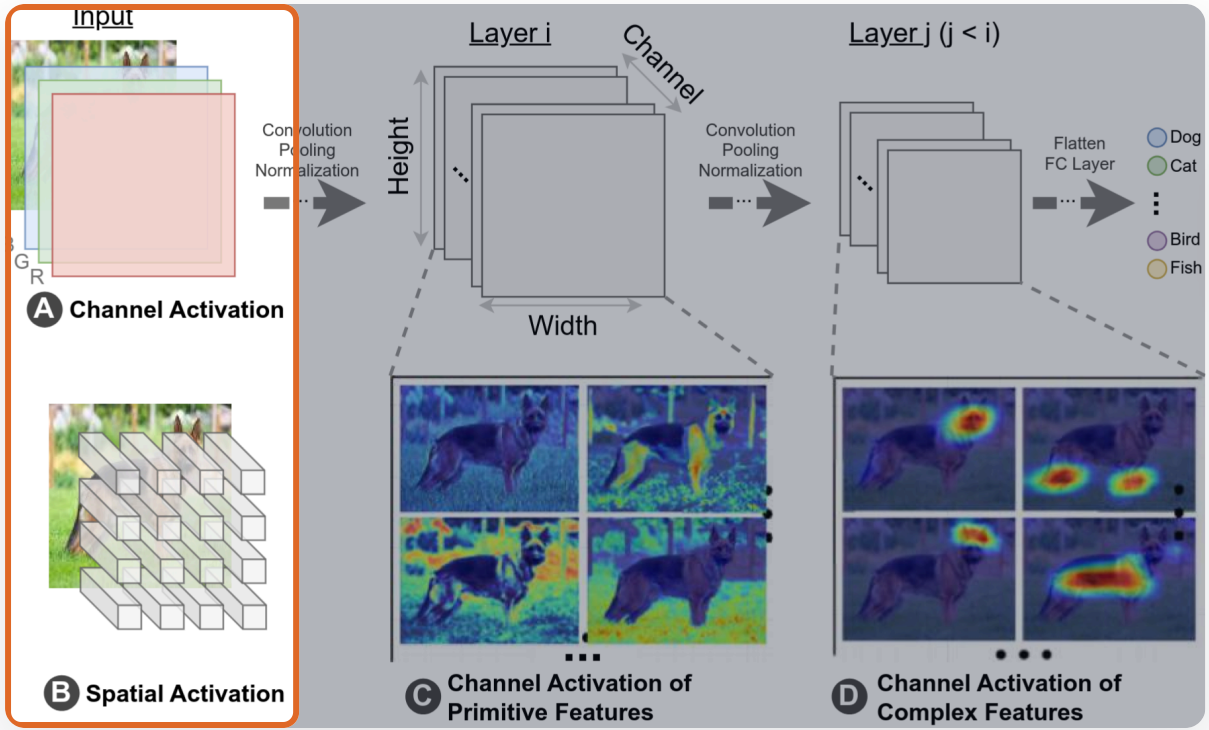


1. A layer outputs an **H × W × C** tensor
2. **Channel activation (A)**: one H × W map per filter. Spatial activation (B): one C-vector per pixel
3. **Early layers (C)**: primitive features such as edges, grass, and texture
4. **Later layers (D)**: complex, semantic parts such as a head or a body

Each channel is a feature detector we can map back onto the image

Inside a layer: activation channels

what an image-based layer produces

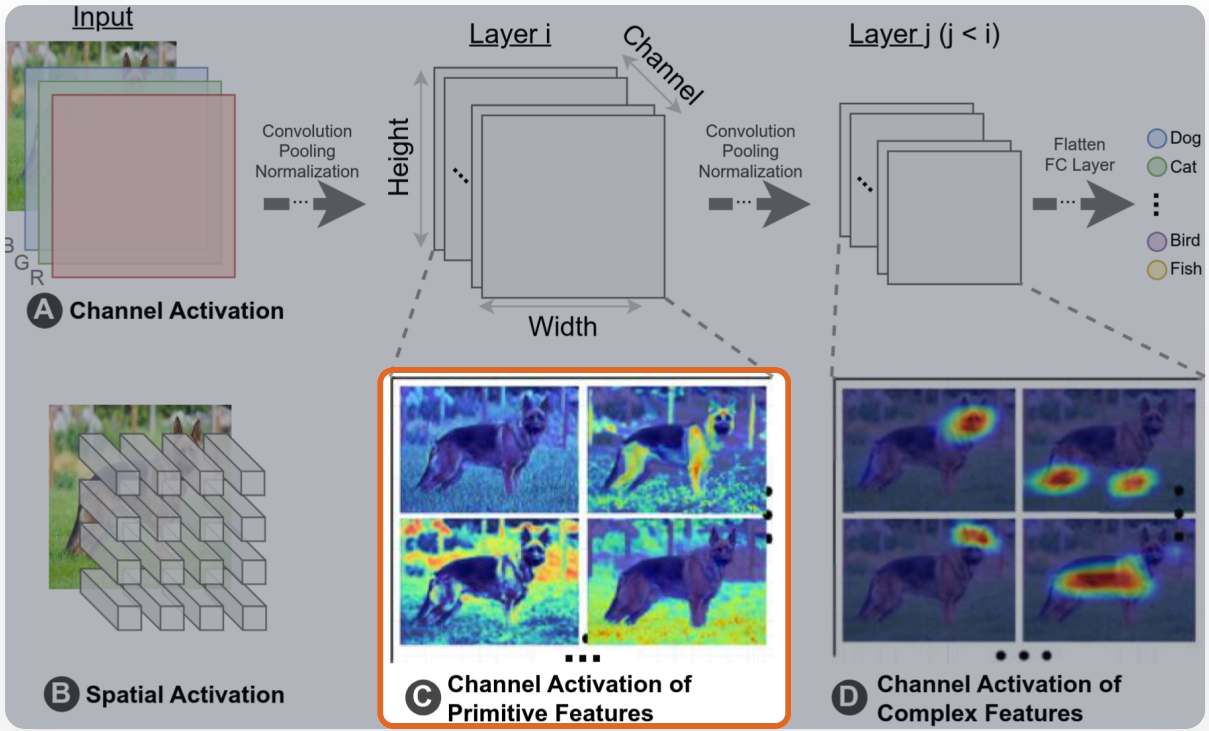


1. A layer outputs an **H × W × C** tensor
2. **Channel activation (A)**: one H × W map per filter. Spatial activation (B): one C-vector per pixel
3. **Early layers (C)**: primitive features such as edges, grass, and texture
4. **Later layers (D)**: complex, semantic parts such as a head or a body

Each channel is a feature detector we can map back onto the image

Inside a layer: activation channels

what an image-based layer produces

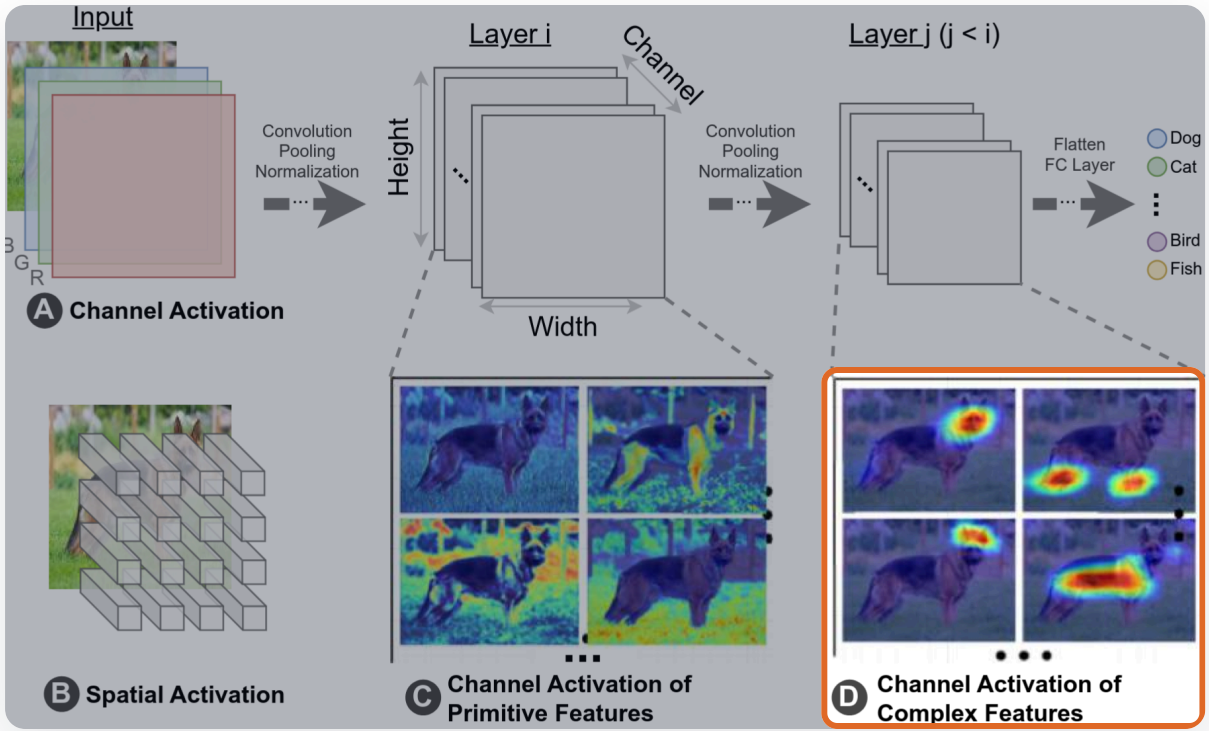


1. A layer outputs an **H × W × C** tensor
2. **Channel activation (A)**: one H × W map per filter. Spatial activation (B): one C-vector per pixel
3. **Early layers (C)**: primitive features such as edges, grass, and texture
4. **Later layers (D)**: complex, semantic parts such as a head or a body

Each channel is a feature detector we can map back onto the image

Inside a layer: activation channels

what an image-based layer produces



1. A layer outputs an **H × W × C** tensor
2. **Channel activation (A)**: one H × W map per filter. Spatial activation (B): one C-vector per pixel
3. **Early layers (C)**: primitive features such as edges, grass, and texture
4. **Later layers (D)**: complex, semantic parts such as a head or a body

Each channel is a feature detector we can map back onto the image

Positioning against prior work

Metrics and confusion

ConfusionFlow · InstanceFlow · CNNComparator · EnsembleMatrix · Neo · Blocks

Scores and confusion matrices miss low-confidence predictions and overlapping activations, and ignore what happens inside the layers

Raw channels and concepts

DeepVis · TensorFlow Playground · Quiver · TCAV · ACE · ConceptExplainer · ActiVis

Channels shown one layer at a time, or concepts and class averages, without the granularity of individual channels and inputs

Projections, aggregation, synthesis

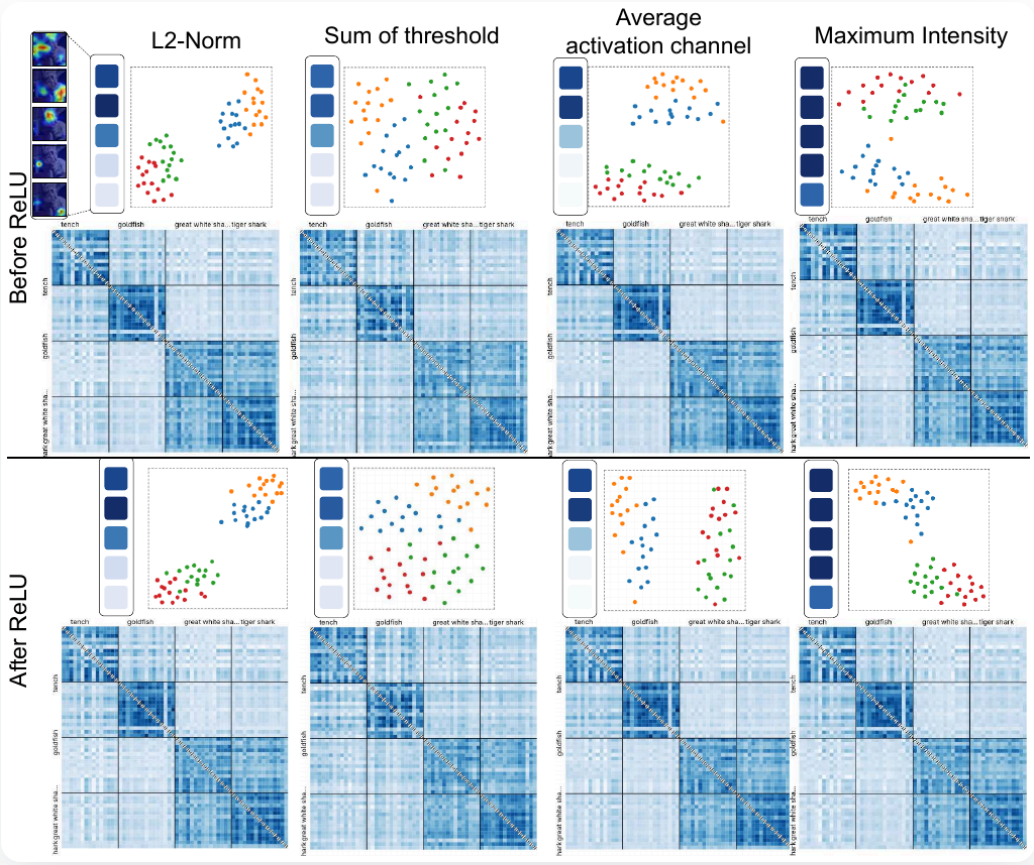
Activation Atlas · t-SNE of CNNs · HSNE · Summit · DeepDream · OpenAI Microscope

Average activations, pathways, or synthesized features help interpretation, but cannot show individual images or layers for debugging

ChannelExplorer summarizes every channel of every input, per layer, and compares layers side by side through "Details Inside" nodes, for any model with image-based layers.

Summarization functions and nonlinearities

Fig. 5



L2-norm

$$S(I) = \sqrt{\sum_{x,y} I(x,y)^2}$$

Sum of threshold

$S(I) = \sum_{x,y} I'(x,y)$, with $I' = |I|$ above Otsu's T

Average

$$S(I) = \sum_{x,y} |I(x,y)| \,/\, (w \cdot h)$$

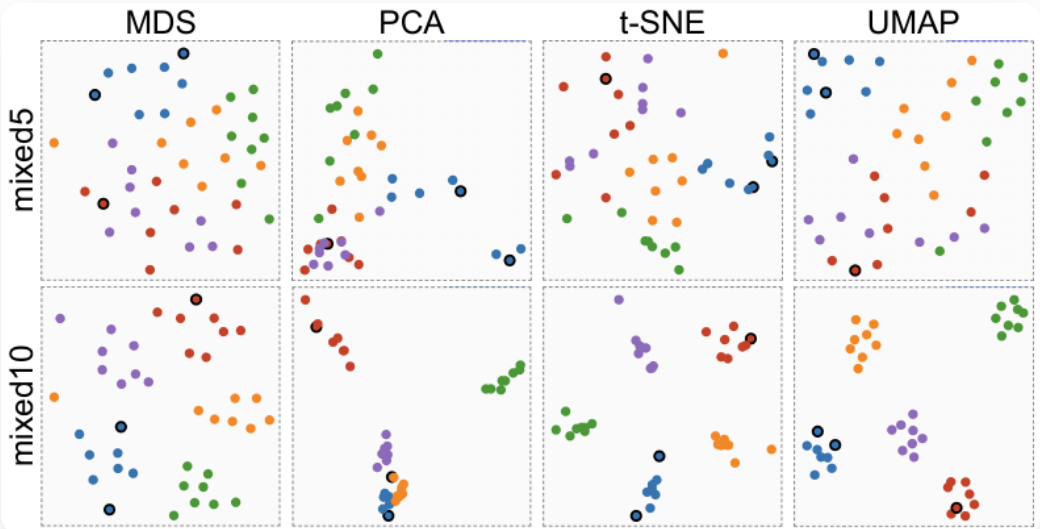
Maximum

$$S(I) = \max_{x,y} |I(x,y)|$$

- Functions 1–3 behave the same in the Scatterplot and Jaccard views; sum of threshold gives the clearest Heatmap stripes, so it is the default
- VGG16 is almost non-negative, so summaries match before and after ReLU
- In InceptionV3, ≈ 19% of channels have negative values, yet the views show no qualitative difference
- Default: pre-nonlinearity, absolute values; an API option summarizes after

Projection methods, and why t-SNE is the default

Fig. 6



InceptionV3, five classes. At `mixed10`, UMAP and t-SNE keep the global distances between classes; MDS and PCA emphasize local structure and variance

t-SNE by default, because it is

1. the most widely adopted projection in DNN visual analytics
2. good at well-separated local neighborhoods for cluster inspection
3. fast enough to stay interactive

Also available

PCA: cheap and insensitive to initialization, for quick debugging. **MDS**: preserves pairwise distances and global structure. **UMAP**. Reseeding checks whether a pattern is stable.

Distance and clusters

L2 on summary vectors was the most consistent. **X-Means** finds clusters (k-means when K is set); DBSCAN's density estimates were unreliable in high dimensions.

Heatmap View: ordering and overlay Eqs. 2–3

Ordering channel i of layer l over n images and $\mathcal{C}$ classes

$$\sigma_{il}^2 = \frac{1}{n}\sum_{j=1}^{n}\left(S(C_{ij}^l) - \mu_{il}\right)^2$$

$$\zeta_{il} = \sum_{u=1}^{\mathcal{C}-1}\sum_{v=u+1}^{\mathcal{C}} d\left(\mathbf{C}_{ui}^l, \mathbf{C}_{vi}^l\right)$$

$\mathbf{C}_{ui}^l$ holds channel i's summaries for class u, and d is Euclidean. ζ is the default: it gives the most stripes at the last layer of accurate models. Edge weights are the L2 norms of each filter.

Overlay of channel C_{ij}^l on image j

Normalize to $\tilde{C}_{ij}^l$ in $[G_{\min}^l, G_{\max}^l]$, the global range of layer l, then resize bilinearly to the input size.

$$O_c(x,y) = \alpha\, M_c\left(\tilde{C}_{ij}^l(x,y)\right) + (1-\alpha)\, f_c(x,y)$$

M_c is the Jet colormap per RGB channel c, f_c is the input image, and $\alpha = 0.6$. Heatmap values below the **10th percentile** are cut off before coloring.

Scalability

Computation

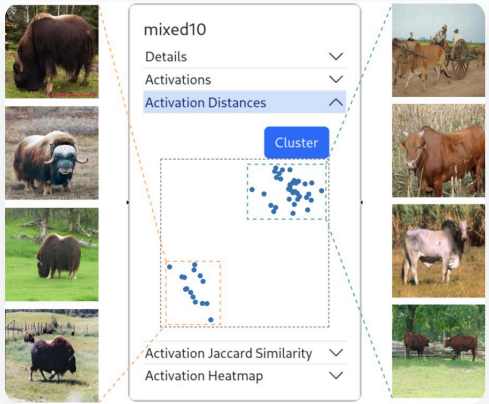
≈ 860M	parameters, 23 residual blocks: Stable Diffusion's U-Net stays interactive
160+	image-based layers in EfficientNet (≈ 66M parameters)
200	inputs in 16 GB of RAM; memory and compute grow linearly with inputs and model size

Visualization

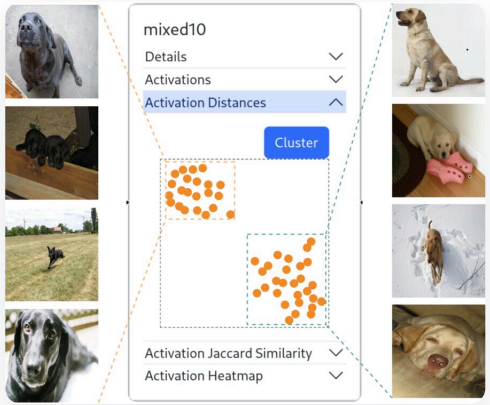
≈ 500	samples readable in the Dataset, Scatterplot, and Confusion Hierarchy views, with representative selection
≈ 50	inputs before the dense Jaccard and Heatmap views get hard to read
≤ 5	representatives per class suffice: activation distributions rarely show more than four modes

More intra-class splits

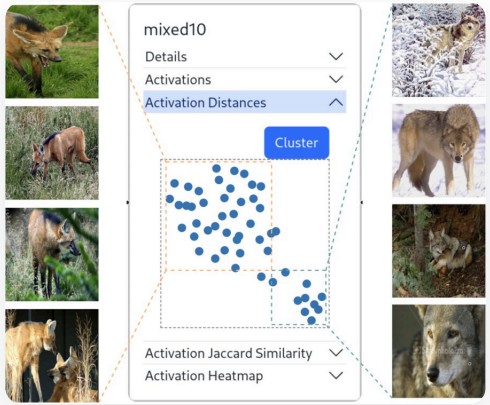
InceptionV3 · mixed10 · one class, two clusters



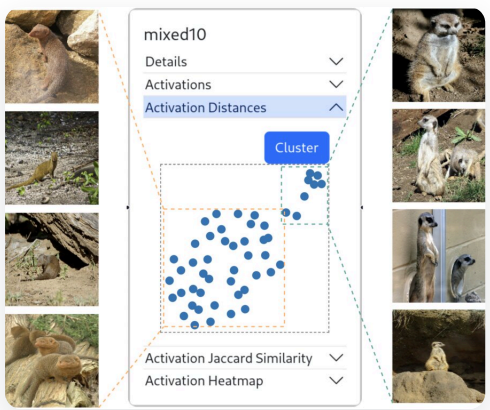
Ox: MuskOx vs. cattle



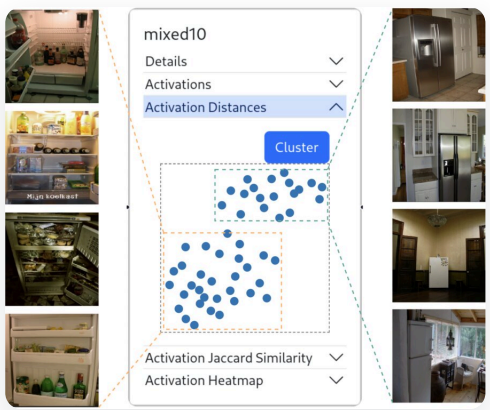
Labrador: black vs. yellow



Red wolf: reddish vs. grey



Mongoose: mongooses vs. meerkats



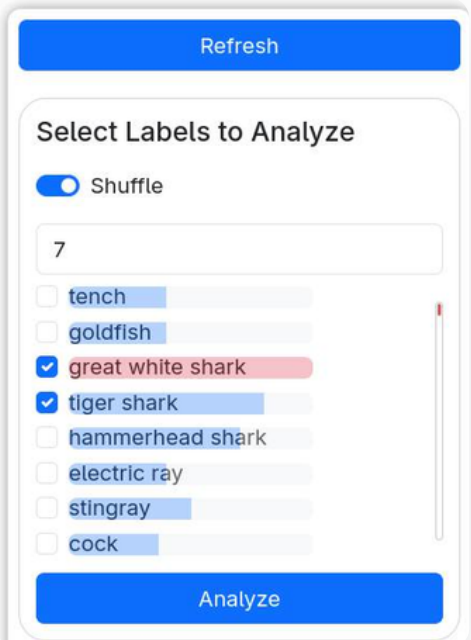
Refrigerator: open vs. closed



Harmonica: being played vs. alone

Dataset View and representative inputs

DATASET VIEW



REPRESENTATIVE INPUTS

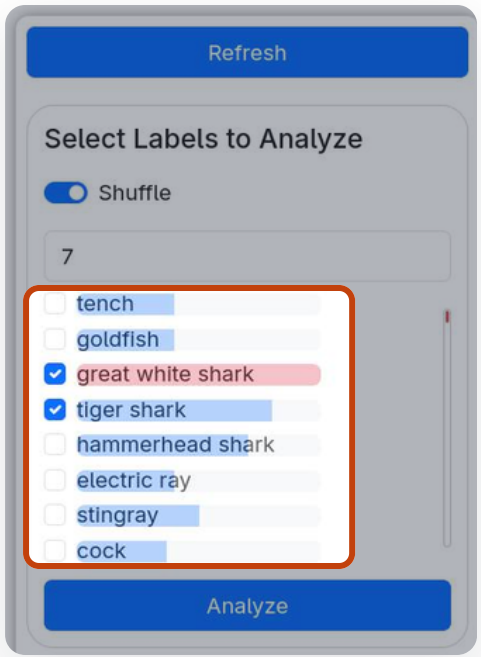
Class 1 Class 2 Class 3

1 **Cluster** each class with X-Means on its activation summaries across all layers, or set K yourself

2 **Sample** one exemplar per cluster, so a few images cover every activation pattern

Dataset View and representative inputs

DATASET VIEW



Bars show **model-wide activation similarity** to a reference class. Classes can also be user-defined groups of inputs.

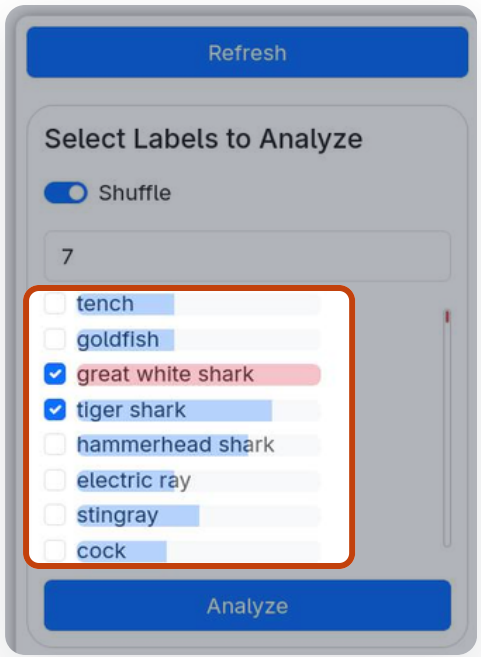
REPRESENTATIVE INPUTS

Class 1 Class 2 Class 3

1. **Cluster** each class with X-Means on its activation summaries across all layers, or set K yourself
2. **Sample** one exemplar per cluster, so a few images cover every activation pattern

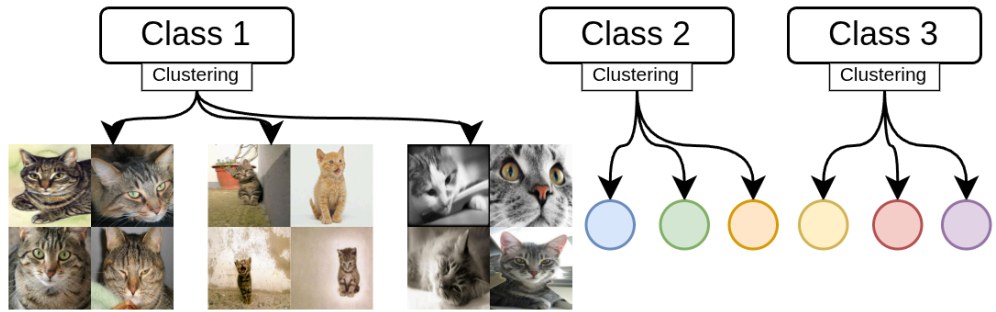
Dataset View and representative inputs

DATASET VIEW



Bars show **model-wide activation similarity** to a reference class. Classes can also be user-defined groups of inputs.

REPRESENTATIVE INPUTS

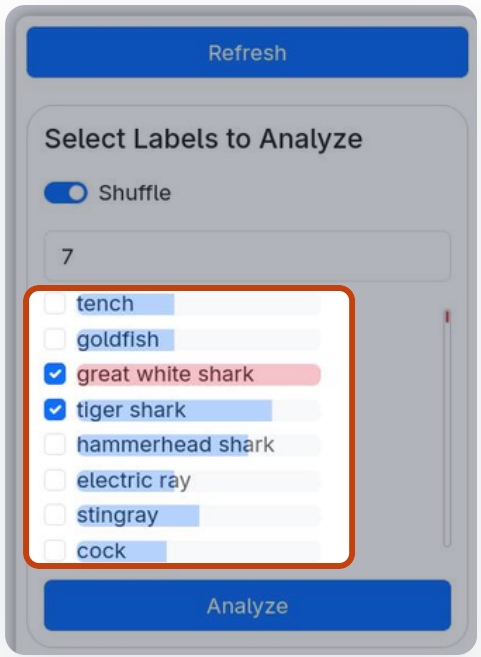


1. **Cluster** each class with X-Means on its activation summaries across all layers, or set K yourself

2. **Sample** one exemplar per cluster, so a few images cover every activation pattern

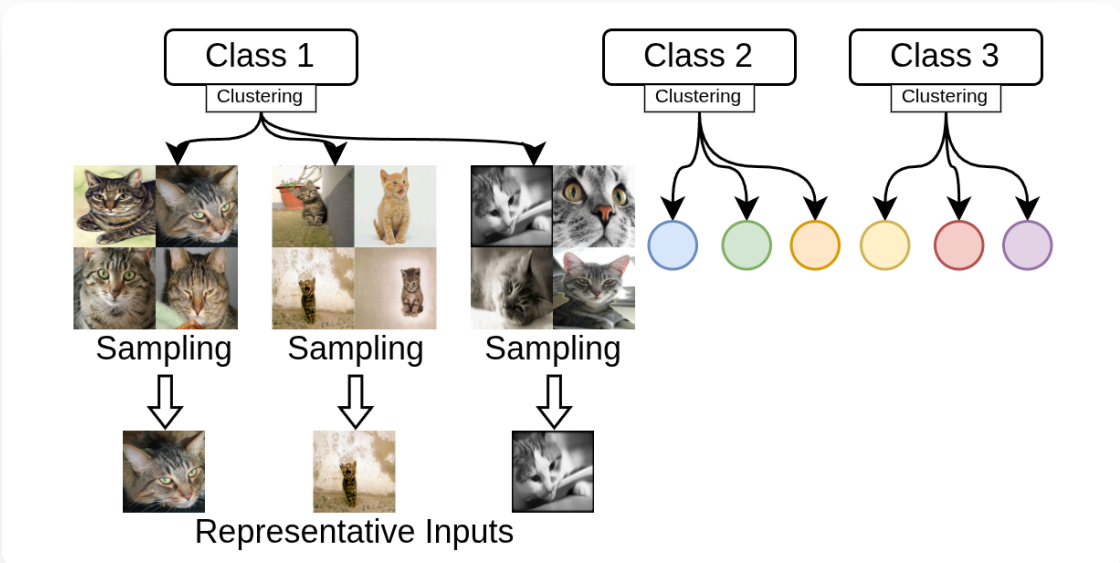
Dataset View and representative inputs

DATASET VIEW



Bars show **model-wide activation similarity** to a reference class. Classes can also be user-defined groups of inputs.

REPRESENTATIVE INPUTS



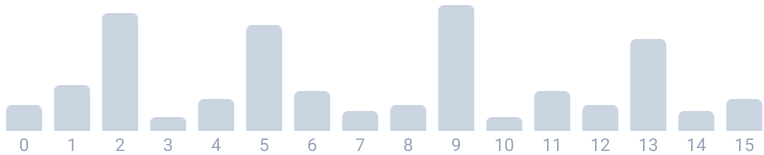
1. **Cluster** each class with X-Means on its activation summaries across all layers, or set K yourself

2. **Sample** one exemplar per cluster, so a few images cover every activation pattern

Jaccard Similarity View

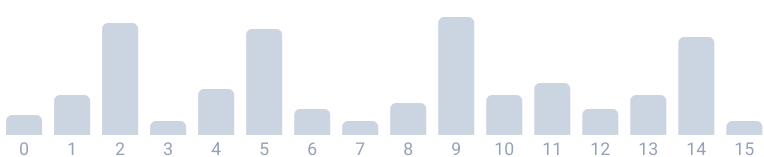


White Shark #1





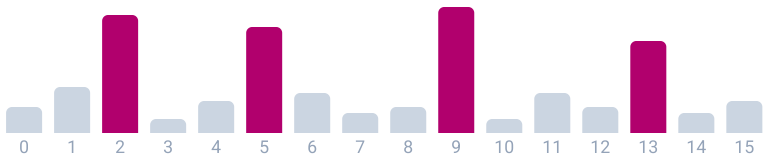
Tiger Shark #3



Jaccard Similarity View



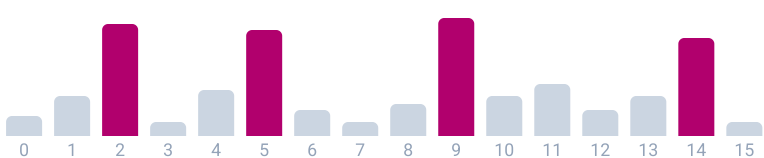
White Shark #1



$S_i = \{ 2, 5, 9, 13 \}$



Tiger Shark #3



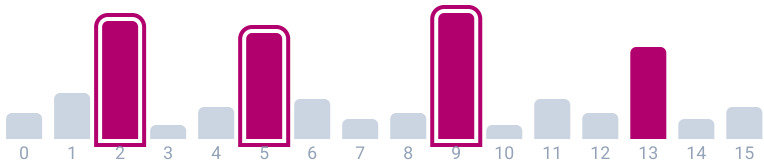
$S_j = \{ 2, 5, 9, 14 \}$

fully activated = top $A_\eta = \lceil \eta \, k \rceil = \lceil 0.25 \cdot 16 \rceil =$ **4** channels

Jaccard Similarity View



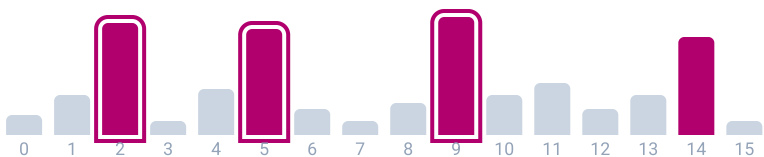
White Shark #1



$S_i = \{ 2, 5, 9, 13 \}$



Tiger Shark #3



$S_j = \{ 2, 5, 9, 14 \}$

fully activated = top $A_\eta = \lceil \eta \, k \rceil = \lceil 0.25 \cdot 16 \rceil =$ **4** channels

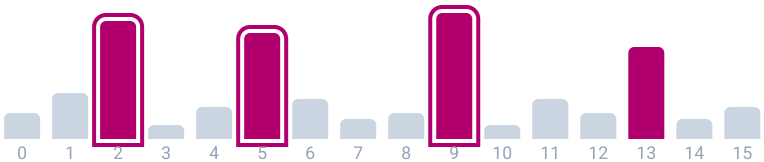
$$J_{ij} = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} = \frac{|\{2, 5, 9\}|}{|\{2, 5, 9, 13, 14\}|} = \frac{3}{5} = \mathbf{0.60}$$

channels active in **both** images ÷ channels active in **at least one**

Jaccard Similarity View



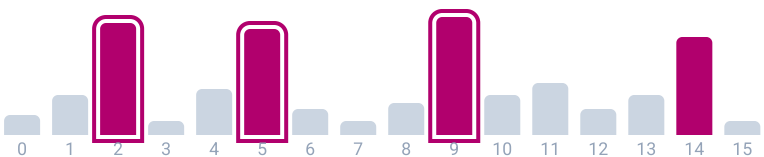
White Shark #1



$S_i = \{ 2, 5, 9, 13 \}$



Tiger Shark #3

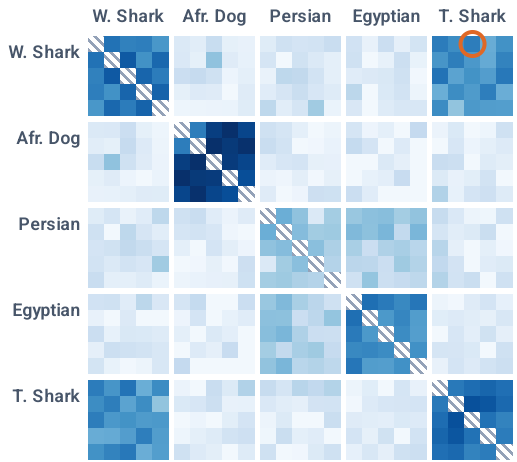


$S_j = \{ 2, 5, 9, 14 \}$

fully activated = top $A_\eta = \lceil \eta \, k \rceil = \lceil 0.25 \cdot 16 \rceil =$ **4** channels

$$J_{ij} = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} = \frac{|\{2, 5, 9\}|}{|\{2, 5, 9, 13, 14\}|} = \frac{3}{5} = \mathbf{0.60}$$

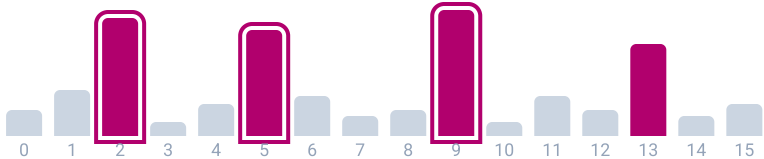
channels active in **both** images ÷ channels active in **at least one**



Jaccard Similarity View



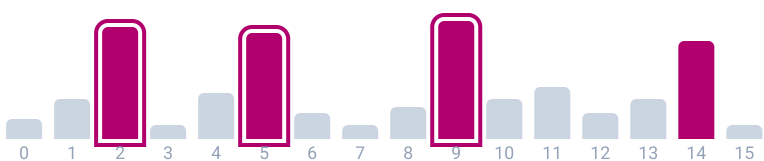
White Shark #1



$S_i = \{2, 5, 9, 13\}$



Tiger Shark #3

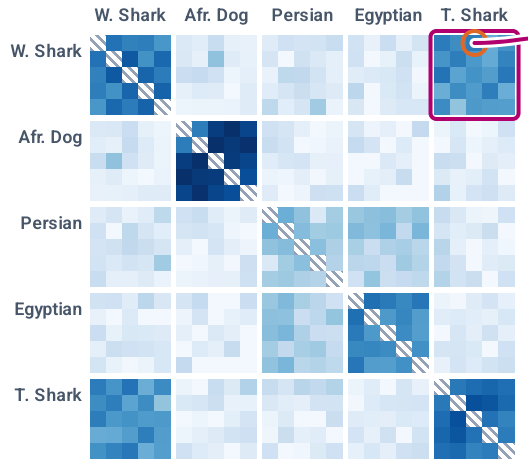


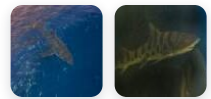
$S_j = \{2, 5, 9, 14\}$

fully activated = top $A_\eta = \lceil \eta\, k \rceil = \lceil 0.25 \cdot 16 \rceil = 4$ channels

$$J_{ij} = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} = \frac{|\{2, 5, 9\}|}{|\{2, 5, 9, 13, 14\}|} = \frac{3}{5} = 0.60$$

channels active in **both** images ÷ channels active in **at least one**





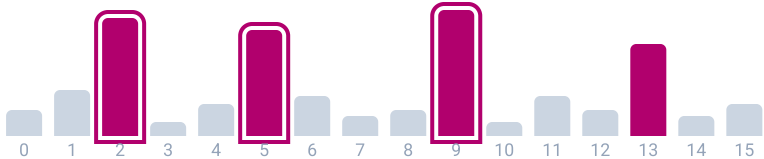
$J = 0.60$
white vs. tiger shark

Dark off-diagonal block: inter-class confusion, White ↔ Tiger Shark

Jaccard Similarity View



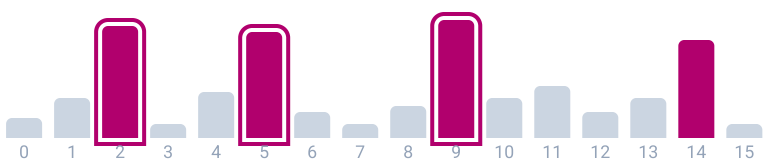
White Shark #1



$S_i = \{2, 5, 9, 13\}$



Tiger Shark #3

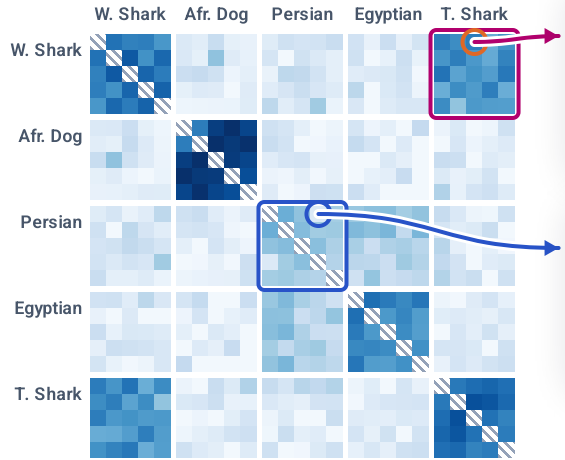


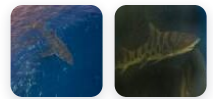
$S_j = \{2, 5, 9, 14\}$

fully activated = top $A_\eta = \lceil \eta\, k \rceil = \lceil 0.25 \cdot 16 \rceil = 4$ channels

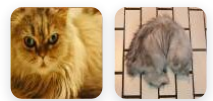
$$J_{ij} = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} = \frac{|\{2, 5, 9\}|}{|\{2, 5, 9, 13, 14\}|} = \frac{3}{5} = 0.60$$

channels active in **both** images ÷ channels active in **at least one**





$J = 0.60$
white vs. tiger shark



$J = 0.12$
two Persian cats

Dark off-diagonal block: inter-class confusion, White ↔ Tiger Shark

Light diagonal block: intra-class confusion, Persian Cat