Some Approaches to Interpret Deep Neural Networks

Suryabhan Singh Hada
Advisor: Miguel Á. Carreira-Perpiñán

Dept. of Computer Science and Engineering,
University of California, Merced
http://eecs.ucmerced.edu

July 1, 2022

Introduction

Introduction

- Deep neural nets have become the preferred model in several practical problems, such as computer vision, language processing, games, self-driving cars, and other engineering applications.
- The way neural nets are defined and optimized, and the sheer size and complexity of state-of-the-art deep nets, make them very hard to understand in explanatory terms.
- Understanding deep nets has become urgent due to their widespread deployment in sensitive fields such as finance, medical field, or autonomous driving.
- Need explanations that are not limited to researchers but also the end-users.

Types of Explanations

Types of explanations

- Inspecting the structure of neural network.
  - What information parameters of the network encode ?
- Local, instance-level explanations.
  - Explain the network prediction for a given input. These explanations are restricted to only a given input.
- Global explanation.
  - Explanation that is applicable to any instance.

Types of explanations

- Inspecting the structure of neural network.
  - What information parameters of the network encode ?
- Local, instance-level explanations.
  - Explain the network prediction for a given input. These explanations are restricted to only a given input.
- Global explanation.
  - Explanation that is applicable to any instance.

Sampling the “Inverse Set” of a Neuron

Explaining behavior of an individual neuron

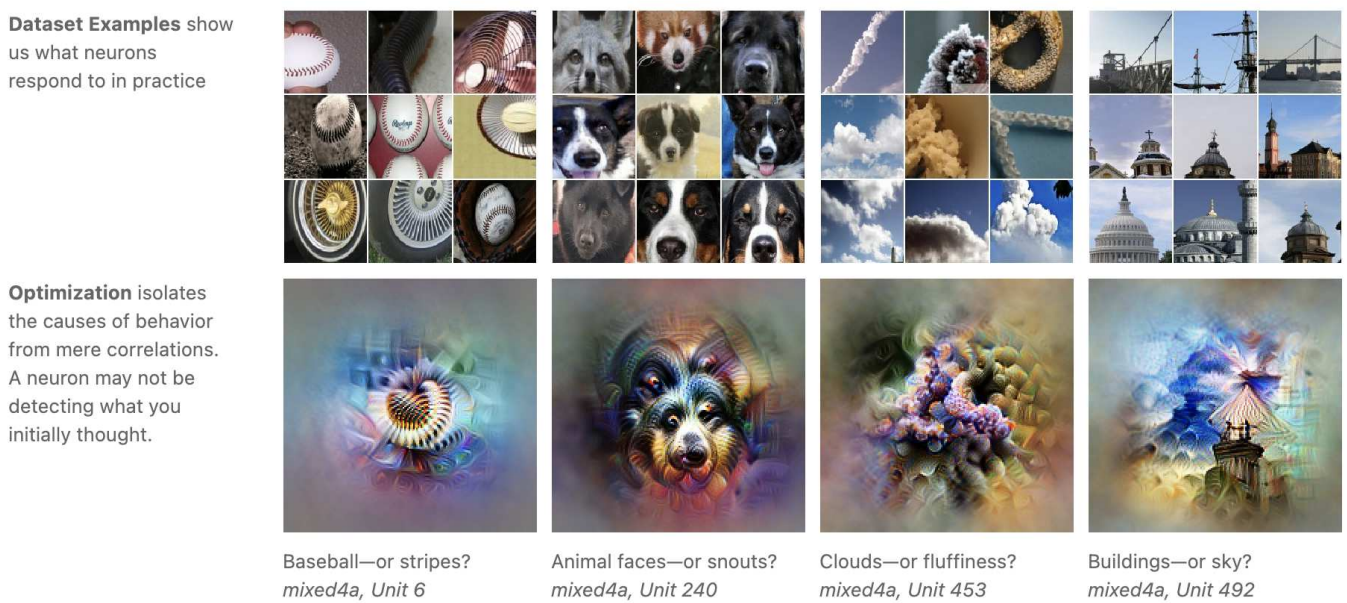
- What concept does a neuron in a deep neural network represents?

Explaining behavior of an individual neuron

- What concept does a neuron in a deep neural network represents?
- Generate an input that maximally activates the neuron of interest: Activation maximization.

Explaining behavior of an individual neuron

- What concept does a neuron in a deep neural network represents?
- Generate an input that maximally activates the neuron of interest: Activation maximization.



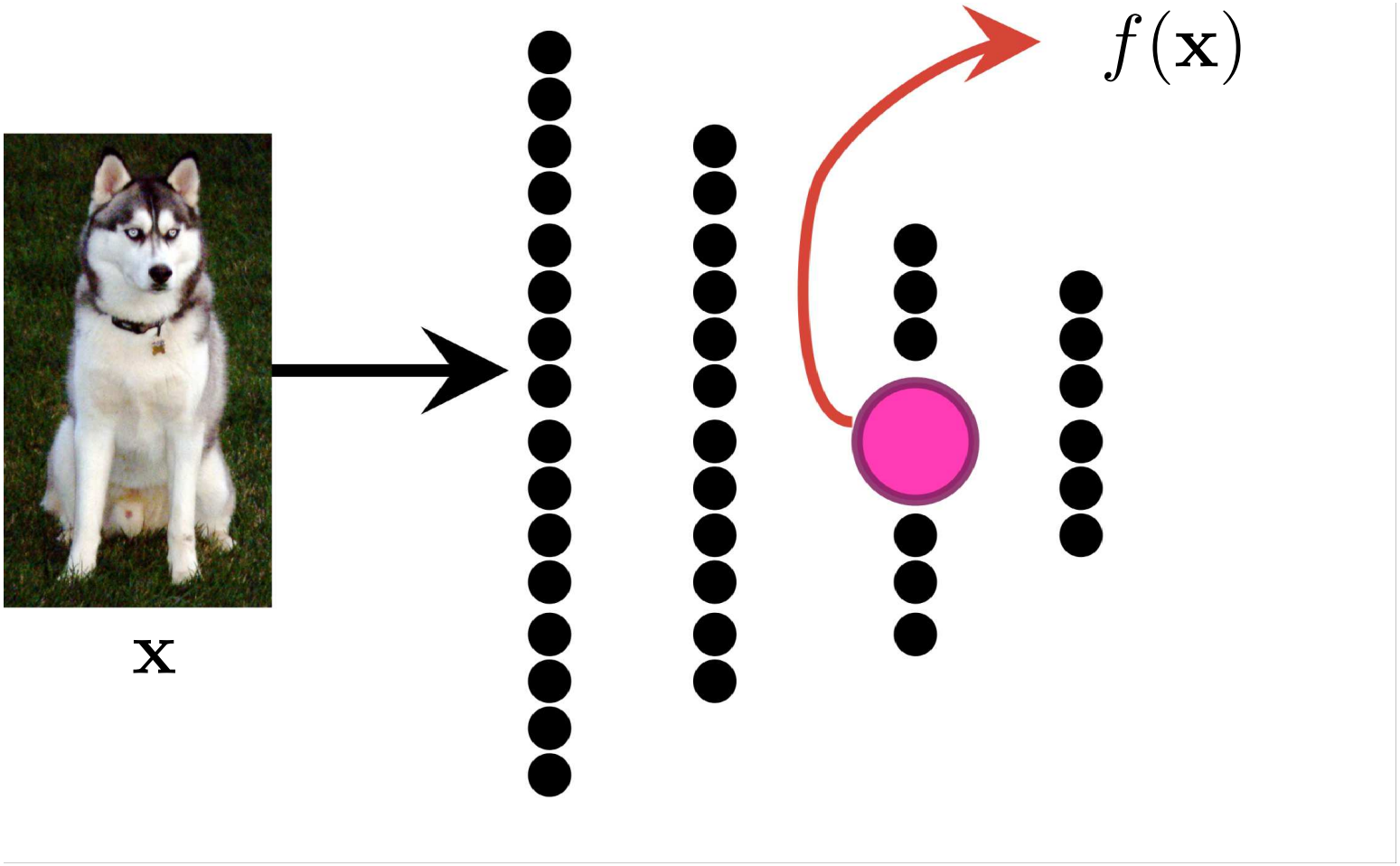
Olah et. al. 2017

Explaining behavior of an individual neuron

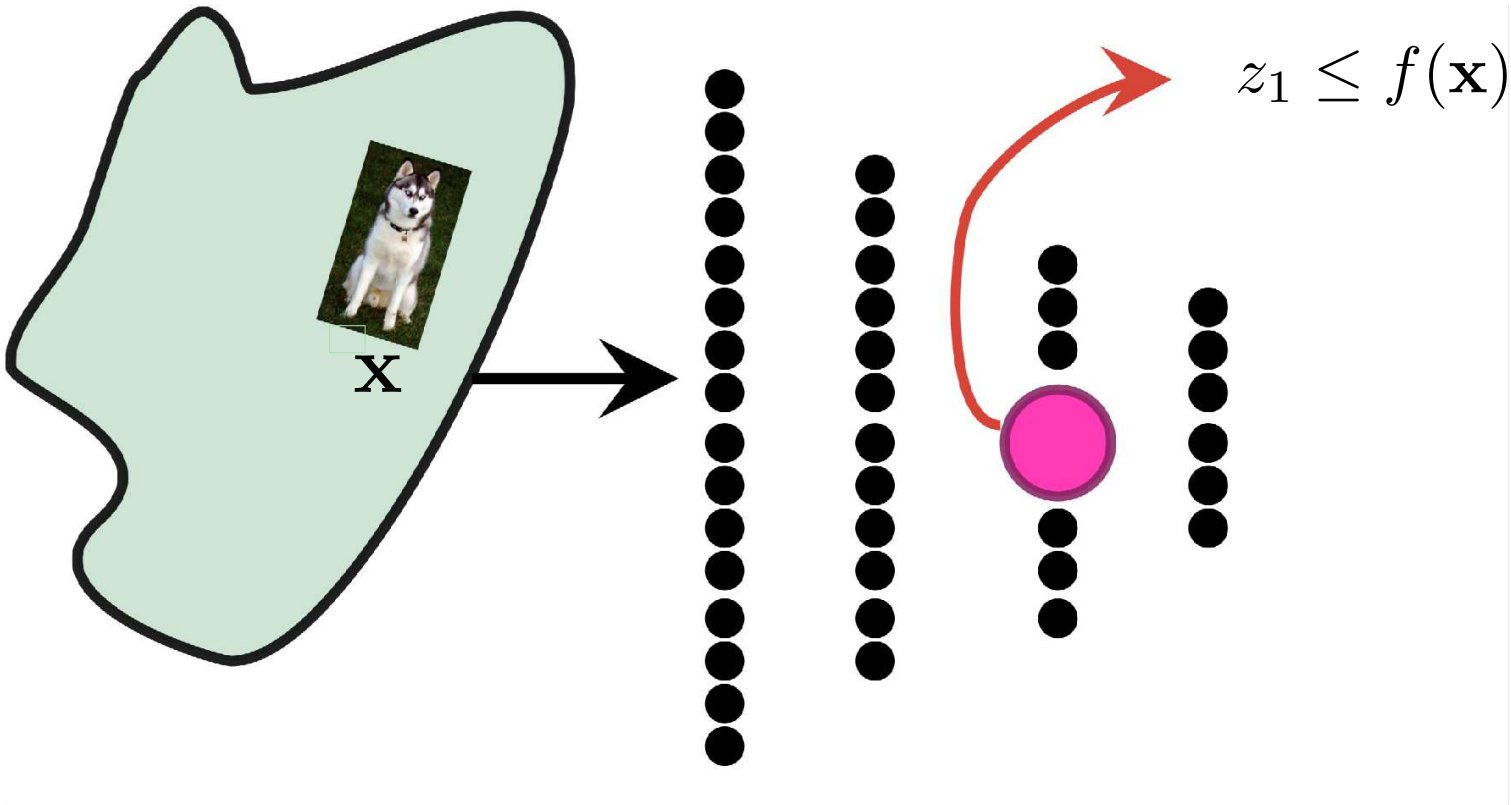
- What concept does a neuron in a deep neural network represents?
- We solve this by characterizing the region of input space that excites a given neuron to a certain level; we call this the **inverse set**.
- This inverse set is a complicated high dimensional object that we explore using an optimization-based sampling approach. Inspection of samples of this set by a human can reveal regularities that help to understand the neuron.

Hada and Carreira-Perpiñán, 2018; Hada and Carreira-Perpiñán, 2021

Activation function of a neuron



Inverse set of neuron



Inverse set definition

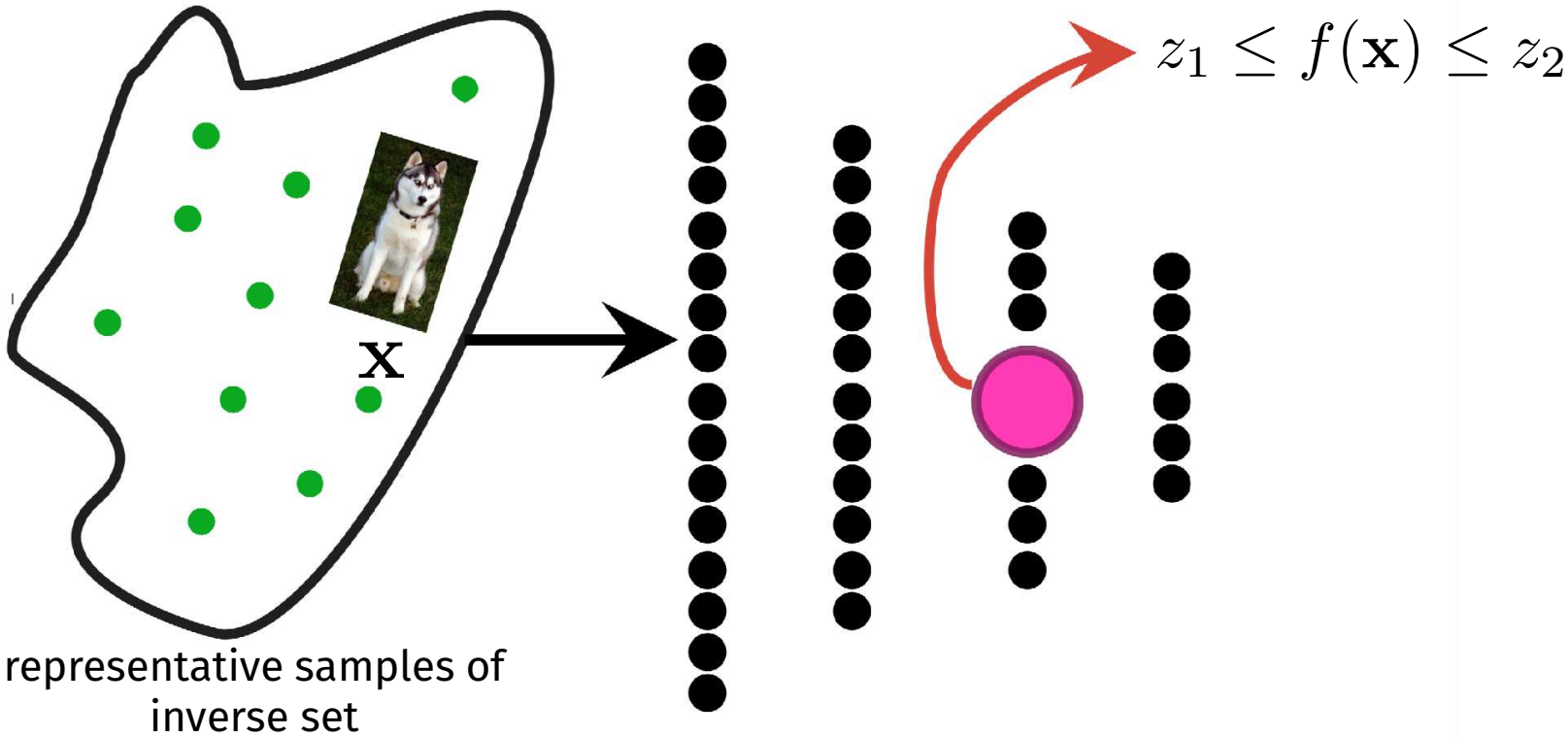
- We say an input $\mathbf{x}$ is in the inverse set of a given neuron having a real-valued activation function f if it satisfies the following two properties:

$$z_1 \leq f(\mathbf{x}) \leq z_2 \qquad \mathbf{x} \text{ is a valid input} \tag{1}$$

  where $z_1, z_2 \in \mathbb{R}$ are activation values of the neuron.

- For example, consider a linear model with weight vector ($\mathbf{w}$), bias (b), logistic activation function $\sigma(\mathbf{w}^T\mathbf{x} + b)$ and all valid inputs to have pixel values between $[0, 1]$. For $z_2 = 1$ (maximum activation value) and $0 < z_1 < z_2$, the inverse set will be the intersection of the half space $\mathbf{w}^T\mathbf{x} + b \geq \sigma^{-1}(z_1)$ and the $[0, 1]$ hypercube.

Sampling the “Inverse Set” of a Neuron



Inverse set for a neuron in a deep neural network

- For deep neural networks, we approximate the inverse set with a sample that covers it in a representative way.
- A simple way to do this is to select all the images in the training set that satisfy eq. (1), but this may rule out all images.
- Therefore, we need an efficient algorithm to sample the inverse set.

Sampling the inverse set: an optimization approach

- To create a sample $\mathbf{x}_1, \ldots, \mathbf{x}_n$ that covers the inverse set, we transform eq. (1) into a constrained optimization problem:

$$\underset{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n}{\arg\max} \sum_{i,j=1}^{n} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \quad \text{s.t.} \quad z_1 \leq f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n) \leq z_2.$$

- The objective function ensures that the samples are different from each other and satisfy eq. (1).

- It has two issues.
  - The generated images are noisy and unrealistic.
  - The generated images are very sensitive to small changes in their pixels.

Sampling the inverse set: an optimization approach

- We solve the issues in following way:
  - To counter the noisy image issue, we use generator network $\mathbf{G}$ to generate images from a feature vector $\mathbf{c}$.
    - A generative network $\mathbf{G}$ generates realistic images when a feature vector $\mathbf{c}$ is passed as an input.
  - For the second issue, we compute distances on a low-dimensional encoding $\mathbf{E}(\mathbf{G}(\mathbf{c}))$ of the generated images constructed by an encoder $\mathbf{E}$.
    - An encoder $\mathbf{E}$ maps a given image $\mathbf{x}$ to a low dimensional feature vector $\mathbf{c}$.

- Our final formulation for generating n samples.

$$\underset{\mathbf{c}_1,\mathbf{c}_2,\cdots,\mathbf{c}_n}{\arg\max} \sum_{i,j=1}^{n} \|\mathbf{E}(\mathbf{G}(\mathbf{c}_i)) - \mathbf{E}(\mathbf{G}(\mathbf{c}_j))\|_2^2$$

$$\text{s.t.} \quad z_1 \leq f(\mathbf{G}(\mathbf{c}_1)), \ldots, f(\mathbf{G}(\mathbf{c}_n)) \leq z_2.$$

Computation constraints

- Because of the quadratic complexity of the objective function over the number of samples n, it is computationally expensive to generate many samples.
- It involves optimizing all code vectors ($\mathbf{c}$) together; for larger n, it is not possible to fit all in the GPU memory.
- Two approximation:
  - Stop the optimization algorithm once the samples enter the feasible set, as, by that time, the samples are already separated.
  - Create the samples incrementally, K samples at a time (with $K \ll n$).

Faster sampling approach

- Optimize the objective function for the first K samples, initializing the code vectors $\mathbf{c}$ with random values. We stop the optimization once the samples are in the feasible set. These samples are then fixed (called seeds $\mathbf{C}_0$).

- The next K samples are generated by the following equation:

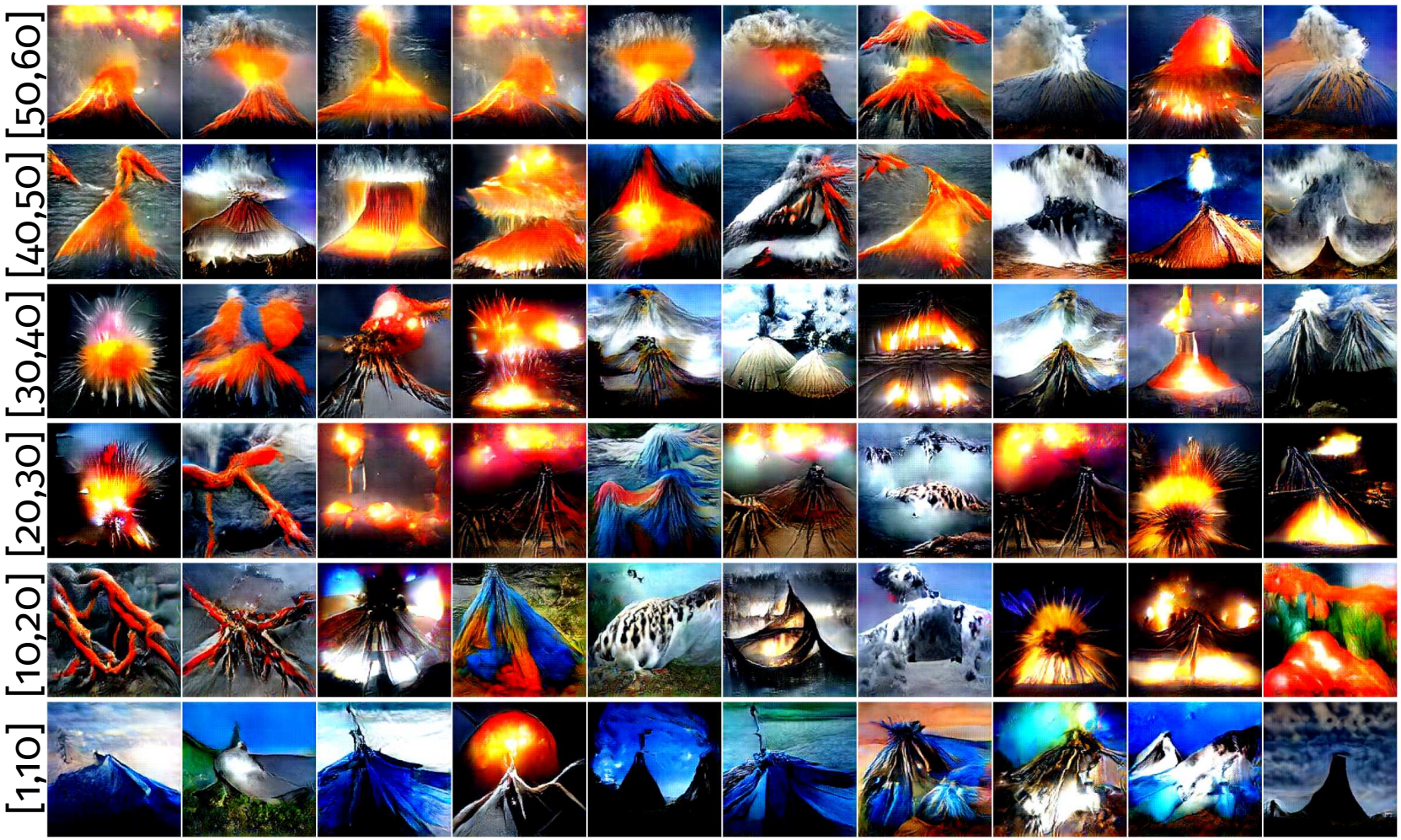
$$\underset{\mathbf{c}_1,\mathbf{c}_2,\cdots,\mathbf{c}_K}{\arg\max} \sum_{i,j=1}^{K} \|\mathbf{E}(\mathbf{G}(\mathbf{c}_i)) - \mathbf{E}(\mathbf{G}(\mathbf{c}_j))\|_2^2 + \sum_{i=1}^{K}\sum_{y=1}^{|\mathbf{C}_0|} \|\mathbf{E}(\mathbf{G}(\mathbf{c}_i)) - \mathbf{E}(\mathbf{G}(\mathbf{c}_y))\|_2^2$$

$$\text{s.t.} \quad z_1 \leq f(\mathbf{G}(\mathbf{c}_1)), \ldots, f(\mathbf{G}(\mathbf{c}_K)) \leq z_2 \text{ and } \mathbf{c}_y \in \mathbf{C}_0.$$

- We initialize them with the previous K samples and take a single gradient step in the feasible region. The resultant samples are the new K samples.

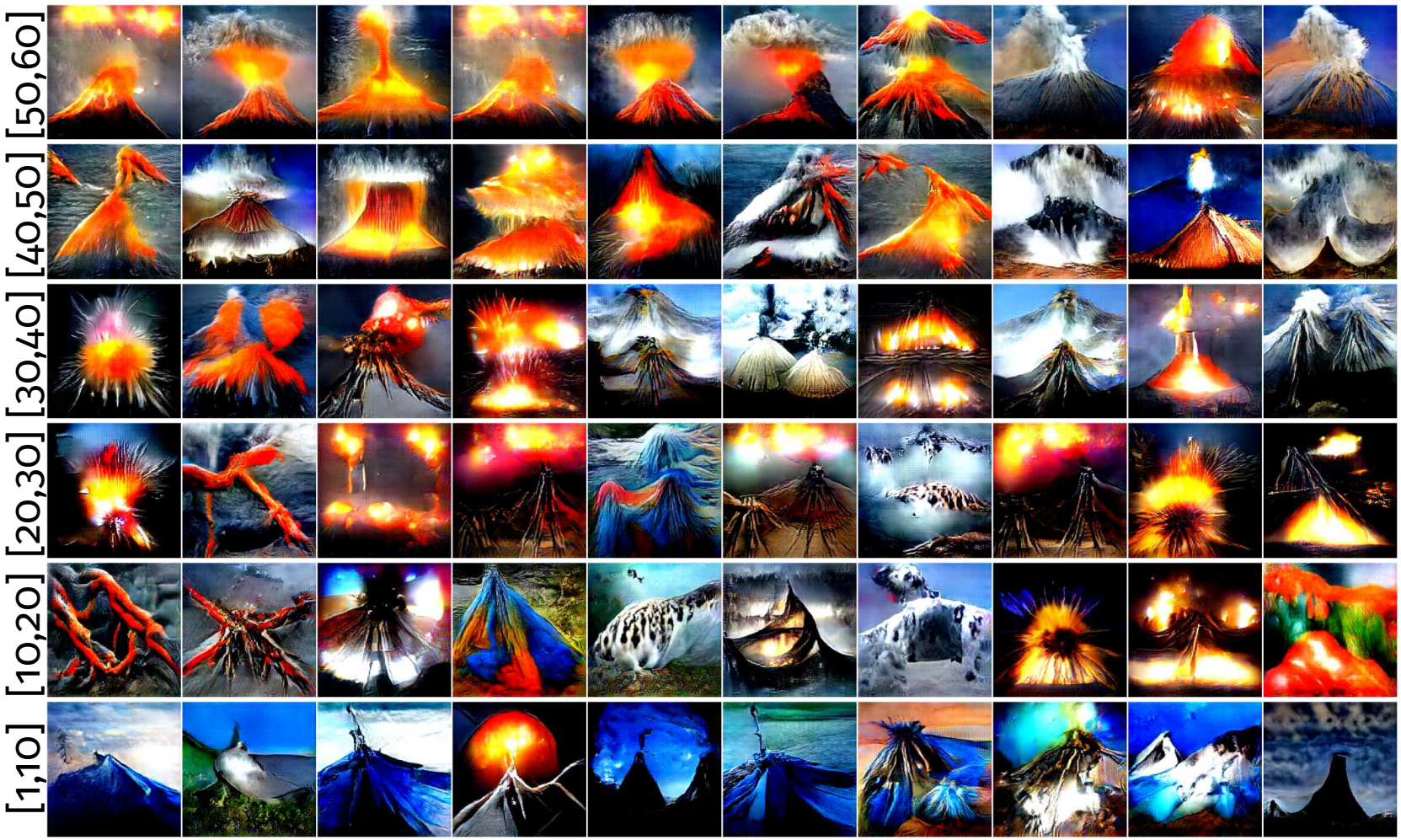
Experiments

- neuron # 981 volcano class.



Experiments

- neuron # 981 volcano class.
- Higher the amount of lava and smoke more the activation.



Inverse set intersection

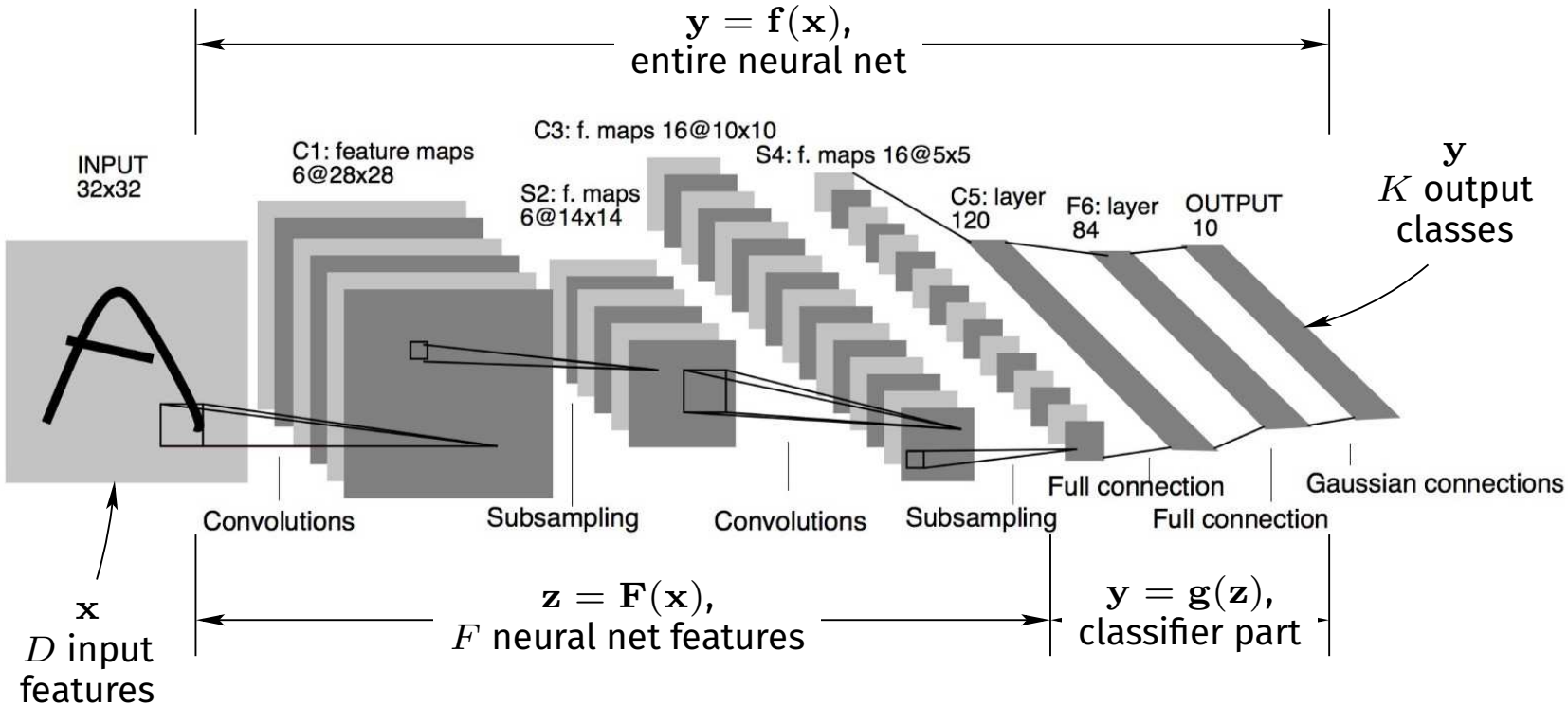
neuron #664 (monastery), [50,60]

neuron #862 (toilet seat), [50,60]

Inverse set Intersection

Interpreting deep neural networks using sparse oblique decision trees

Interpreting deep neural networks using sparse oblique decision trees



Interpreting deep neural networks using sparse oblique decision trees

- Consider a trained deep net classifier:

$$\mathbf{y} = \mathbf{f}(\mathbf{x})$$

- We can write $\mathbf{f}$ as: $\mathbf{f}(\mathbf{x}) = \mathbf{g}(\mathbf{F}(\mathbf{x}))$, where
  - $\mathbf{F}$ represents the features-extraction part ($\mathbf{z} = \mathbf{F}(\mathbf{x}) \in \mathbb{R}^F$).
  - $\mathbf{g}$ represents the classifier part ($\mathbf{y} = \mathbf{g}(\mathbf{z})$).

- The last layer of $\mathbf{F}$ is interesting, as it is associated with the features extracted by $\mathbf{F}$ that goes into $\mathbf{g}$.

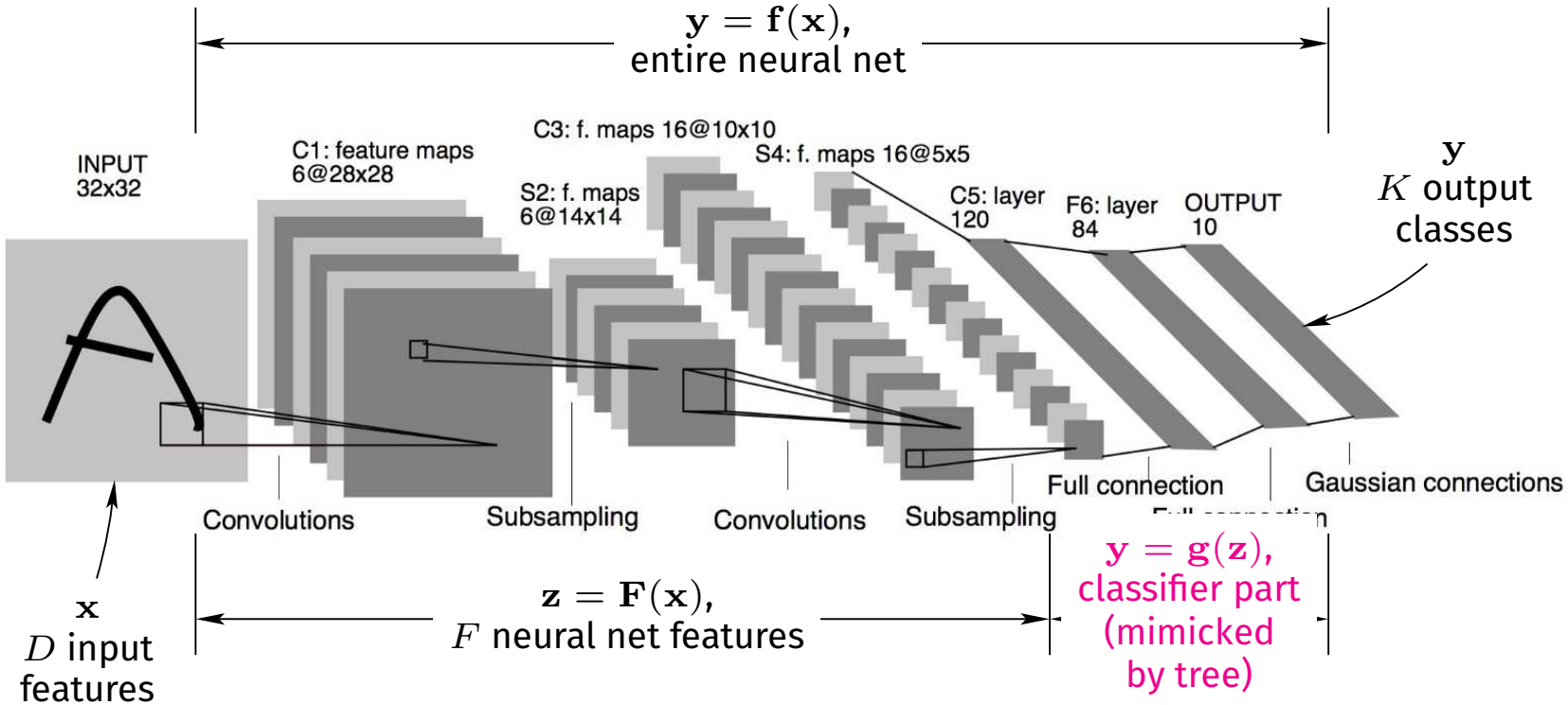
- We want to understand the relationship between neurons in the last layer of $\mathbf{F}$ and the classes.

What we found

- Out of thousands of neurons, there is a small subset of neurons associated with a given class.
- We explore this by introducing a new feature-level adversarial attack via masking a specific set of neurons.
- These attacks include making net to predict or not predict a given class.

Hada et. al. 2021a; Hada et. al. 2021b

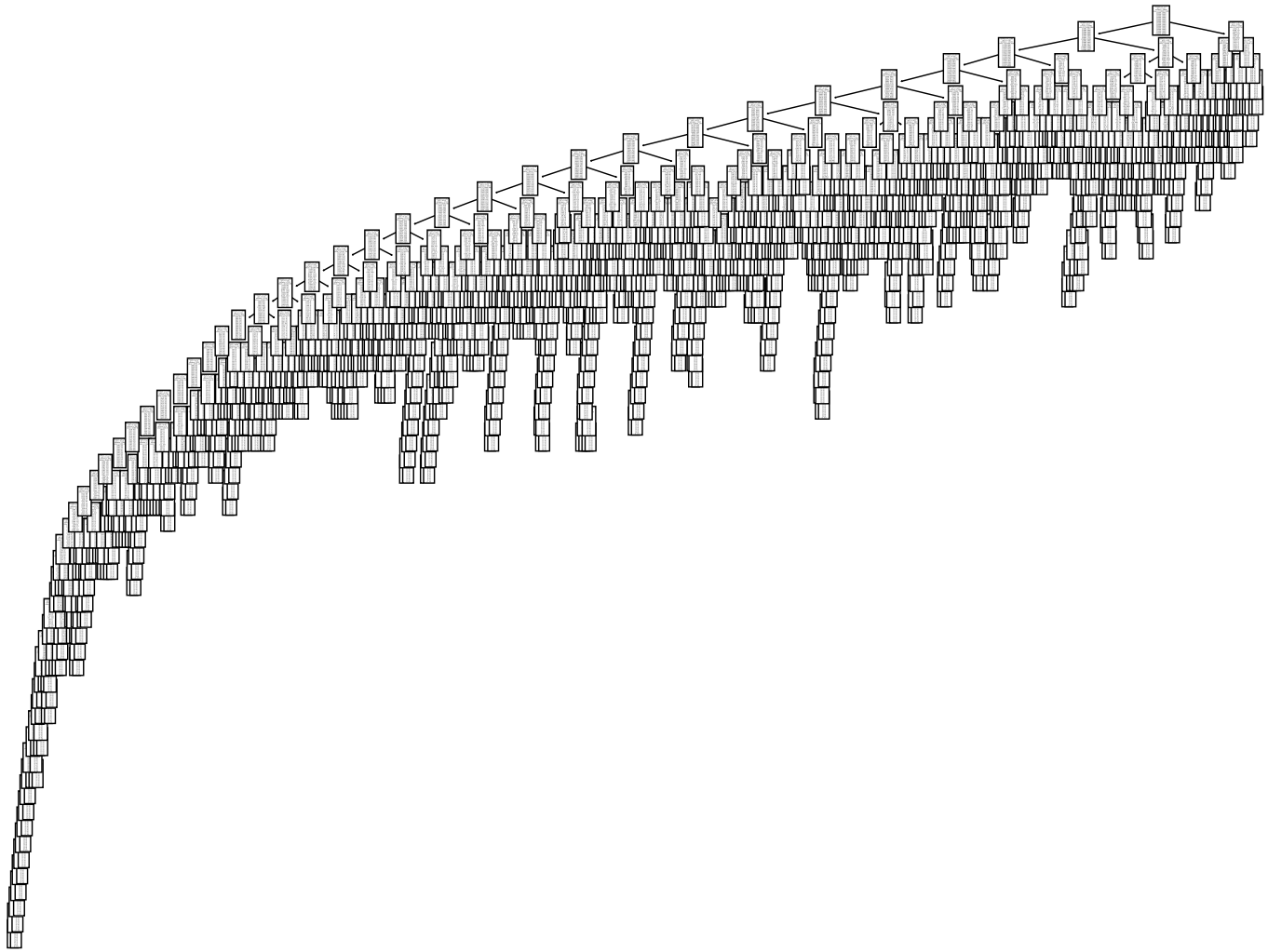
Our approach



Tree mimick with CART

- Deep net test error: 6.79%

Tree test error: 21.24%



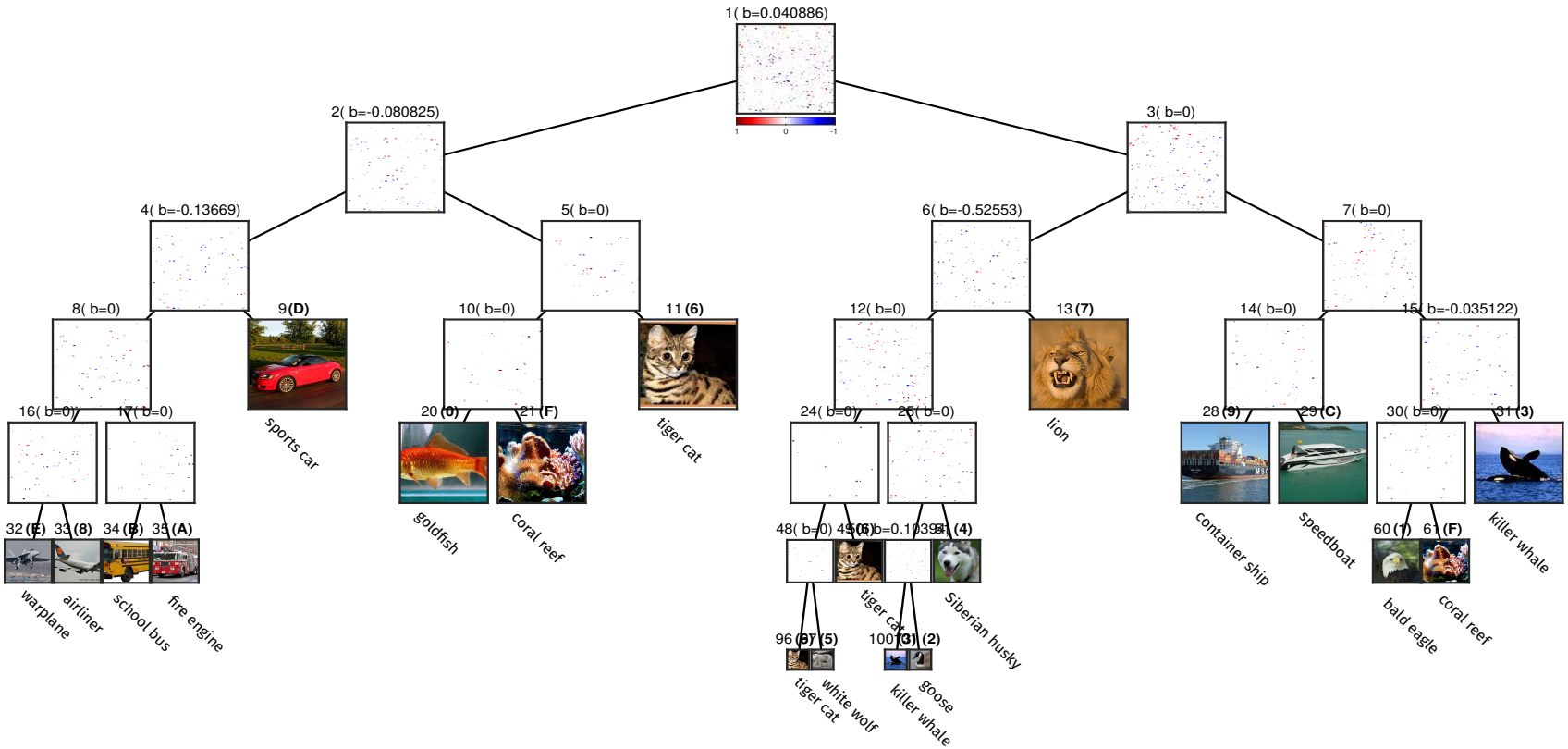
Our approach

- We study the relationship between neurons at the last layer of $\mathbf{F}$ and the classes using sparse oblique trees.
- Overall approach:
  - Train a sparse oblique tree $y = T(\mathbf{z})$ on the training set $\{(\mathbf{F}(\mathbf{x}_n), y_n)\}_{n=1}^{N} \subset \mathbb{R}^F \times \{1, \ldots, K\}$. Choose the sparsity hyperparameter $\lambda \in [0, \infty)$ such that, T mimicks $\mathbf{g}$ very well and is as sparse as possible.
  - Inspect the tree T to create masks.

Tree mimick with a sparse oblique tree

- We train these trees using Tree alternating optimization (TAO) algorithm.
- Deep net test error: 6.79%

Tree test error: 7.9%



Carreira-Perpiñán and Tavallali 2018

Tree alternating optimization (TAO)

- Tree alternating optimization (TAO) is a recently proposed algorithm that can train highly accurate oblique or axis-aligned trees.
- To train the sparse oblique it optimize following objective function:

$$E(\boldsymbol{\Theta}) = \sum_{n=1}^{N} L(y_n, T(\mathbf{x}_n; \boldsymbol{\Theta})) + \lambda \sum_{i \in \mathcal{L}} \|\boldsymbol{\theta}_i\|_1$$

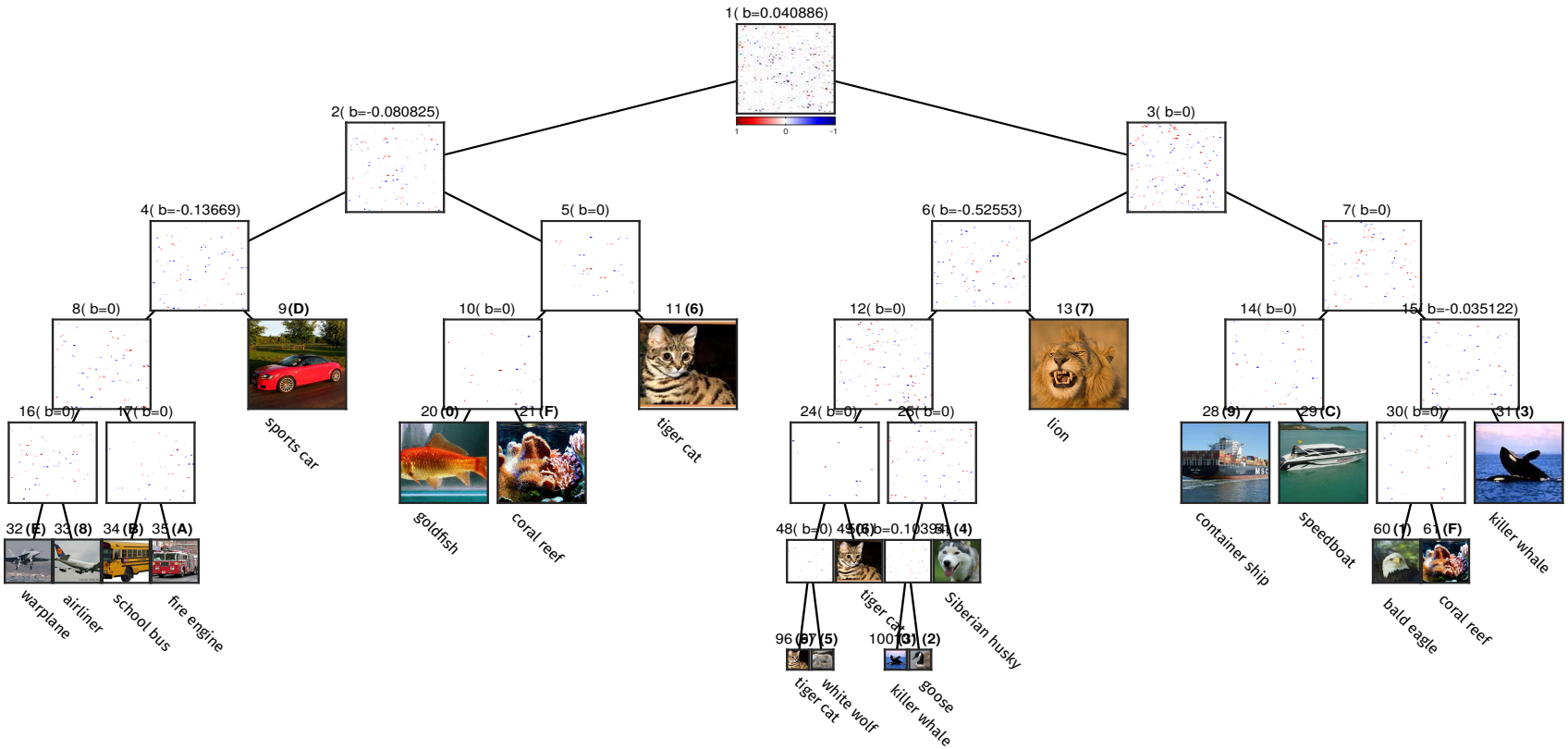
  - $\boldsymbol{\Theta}$ represents parameters of the tree.
  - $L(.,.)$ is 0/1 loss function.
  - λ is ℓ_1 penalty on the parameters of the each decision node.
- TAO relies on two theorems to train the tree:
  - Objective function separates over any subset of non-descendant nodes.
  - Optimizing over the parameters of a single node i simplifies to a binary classification problem that decides if incoming instance goes to left child or the right child.

Carreira-Perpiñán and Tavallali 2018

Tree mimick with a sparse oblique tree

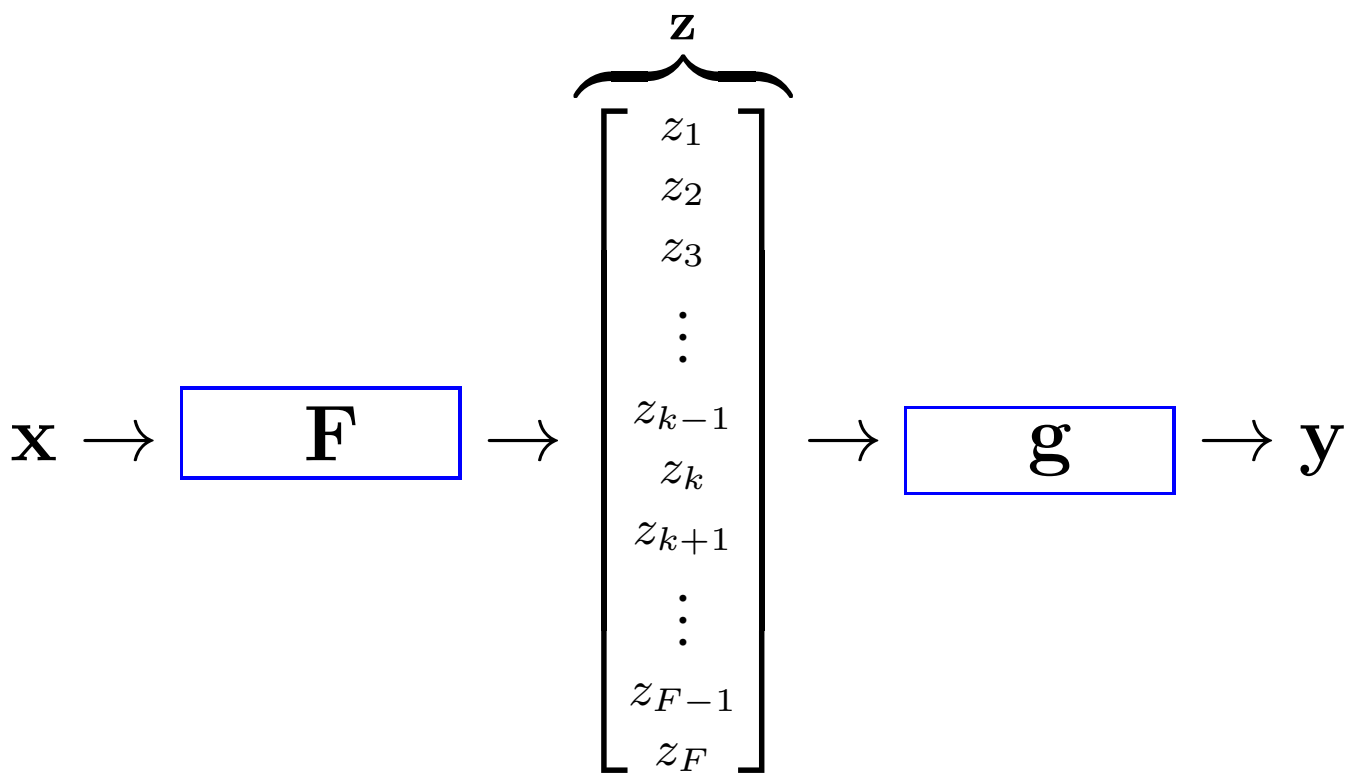
- We train these trees using Tree alternating optimization (TAO) algorithm.
- Deep net test error: 6.79%

Tree test error: 7.9%



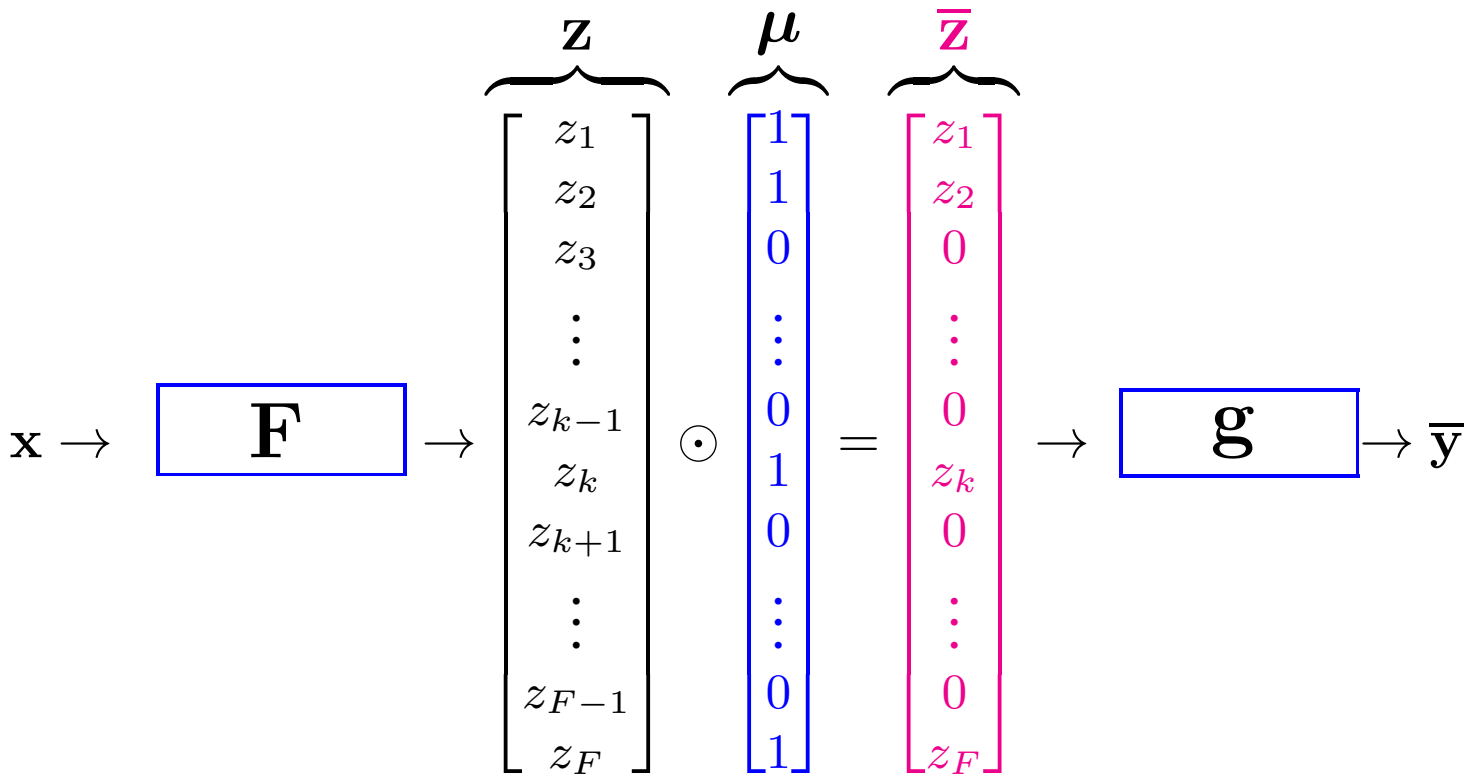
Carreira-Perpiñán and Tavallali 2018

Masking of deep net features



Masking of deep net features

- Mask μ is created using the tree mimic of g.



Masking of deep net features

- Objective is to control the behavior of network prediction by manipulating deep net features ($\mathbf{z} = \mathbf{F}(\mathbf{x}) \in \mathbb{R}^F$), without modifying the $\mathbf{F}$ and $\mathbf{g}$.
- Original net: $\mathbf{y} = \mathbf{f}(\mathbf{x}) = \mathbf{g}(\mathbf{F}(\mathbf{x}))$.
- Original features: $\mathbf{z} = \mathbf{F}(\mathbf{x})$.
- Masked net: $\overline{\mathbf{y}} = \overline{\mathbf{f}}(\mathbf{x}) = \mathbf{g}(\boldsymbol{\mu}(\mathbf{F}(\mathbf{x})))$
- Masked features: $\overline{\mathbf{z}} = \boldsymbol{\mu}(\mathbf{F}(\mathbf{x})) = \boldsymbol{\mu}(\mathbf{z})$.
- $\overline{\mathbf{z}} = \boldsymbol{\mu}(\mathbf{z}) = \boldsymbol{\mu}^{\times} \odot \mathbf{z} + \boldsymbol{\mu}^{+}$.
  - $\boldsymbol{\mu} = \{\boldsymbol{\mu}^{\times}, \boldsymbol{\mu}^{+}\}$
    - where, $\boldsymbol{\mu}^{\times} \in \{0, 1\}^F$ is the *multiplicative mask*.
    - $\boldsymbol{\mu}^{+} \geq 0$ is the *additive mask*.

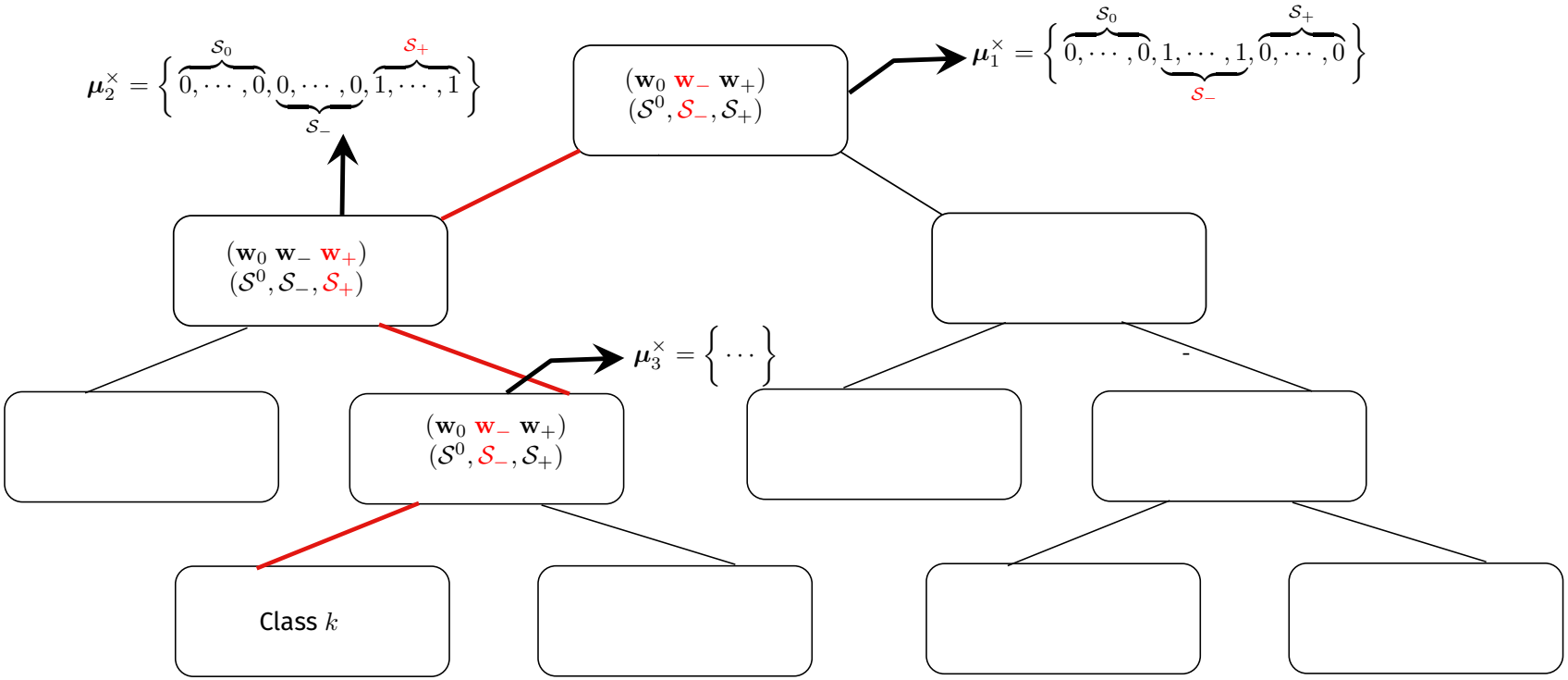
Types of masks

- ALL TO CLASS k.
  - Let $k \in \{1, \ldots, K\}$. Classify all instances $\mathbf{x}$ as class k. Location of 1's in $\mu^{\times}$ is the location of neurons associated with class k.
- ALL CLASS k_1 TO CLASS k_2
  - Let $k_1 \neq k_2 \in \{1, \ldots, K\}$. For any instance originally classified as k_1, classify it as k_2. For any other instance, do not alter its classification.
- NONE TO CLASS k
  - Let $k \in \{1, \ldots, K\}$. For any instance originally classified as k, classify it as any other class. For any other instance, do not alter its classification

All to class k mask

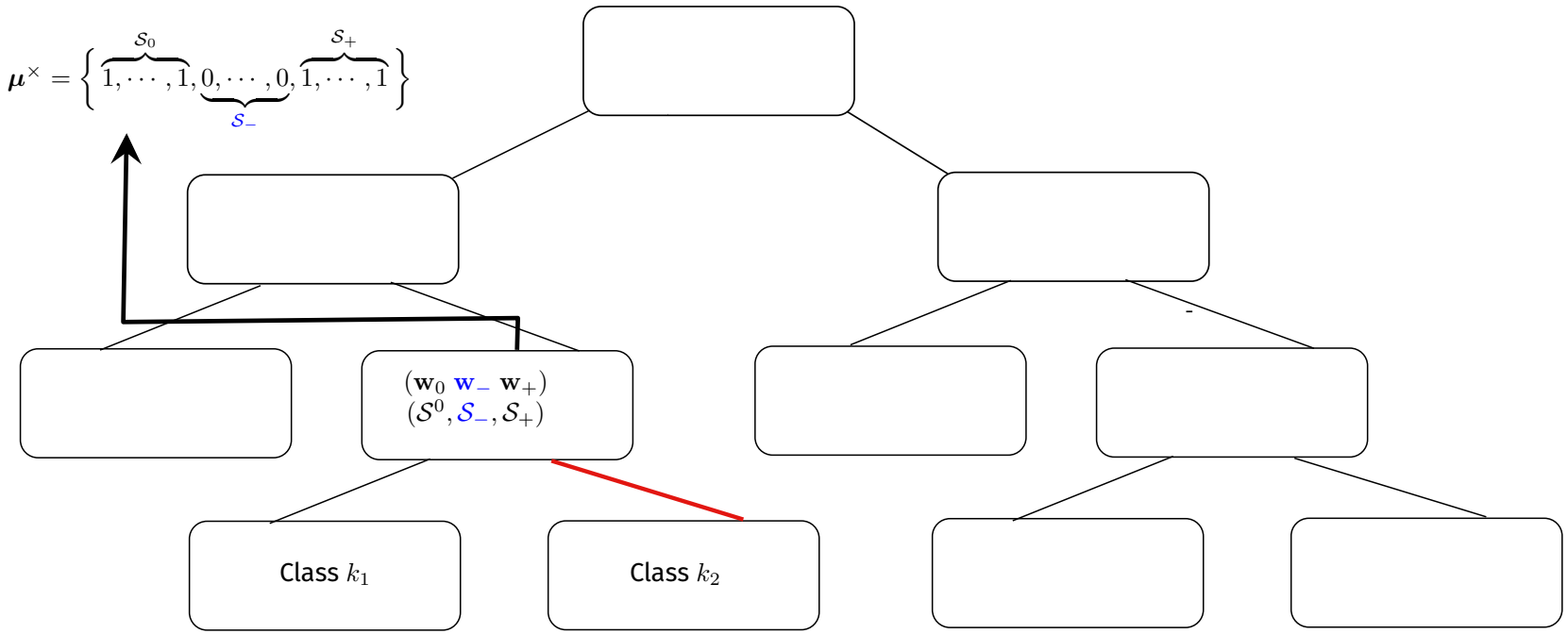
- The main idea is to create mask by following the path from root to the target leaf. If it works in the tree, it should work in the same way in the network.
- Consider a decision node i in T with decision rule:
  - "if $\mathbf{w}_i^T \mathbf{z} + b_i \geq 0$ then go to right child, else go to left child"
    - $\mathbf{w}_i \in \mathbb{R}^F$ is the weight vector
    - $b_i \in \mathbb{R}$ is the bias.
- Write $\mathbf{w}$ as: $\mathbf{w} = (\mathbf{w}_0 \ \mathbf{w}_- \ \mathbf{w}_+)$
  - $\mathbf{w}_0 = 0$.
  - $\mathbf{w}_- < 0$.
  - $\mathbf{w}_+ > 0$.
- Call $\mathcal{S}_0$, $\mathcal{S}_-$ and $\mathcal{S}_+$ the corresponding sets of indices in $\mathbf{w}$.

All to class k mask



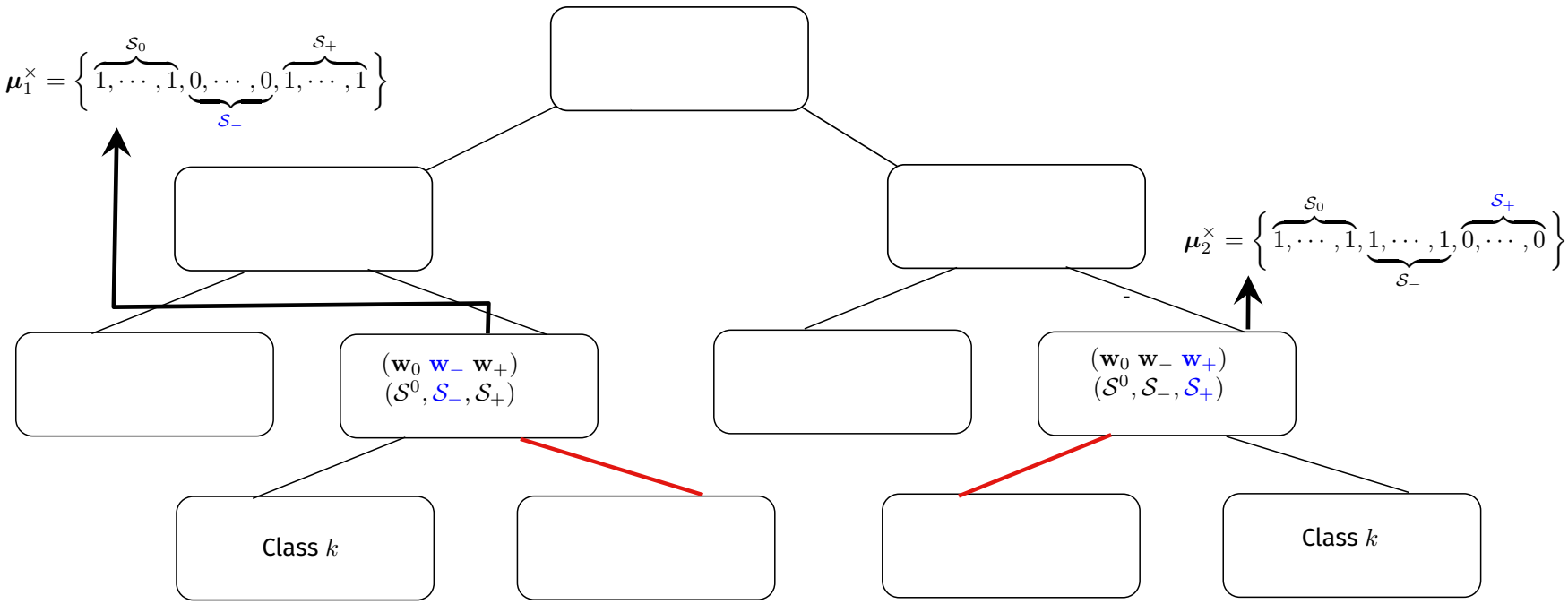
- $\boldsymbol{\mu}^\times = \boldsymbol{\mu}_1^\times \wedge \boldsymbol{\mu}_2^\times \wedge \boldsymbol{\mu}_2^\times$.
- $\boldsymbol{\mu}^+ = \epsilon\boldsymbol{\mu}^\times$, where ϵ is a small positive number.
- This represents the neurons associated with class k

All class k_1 to class k_2 mask



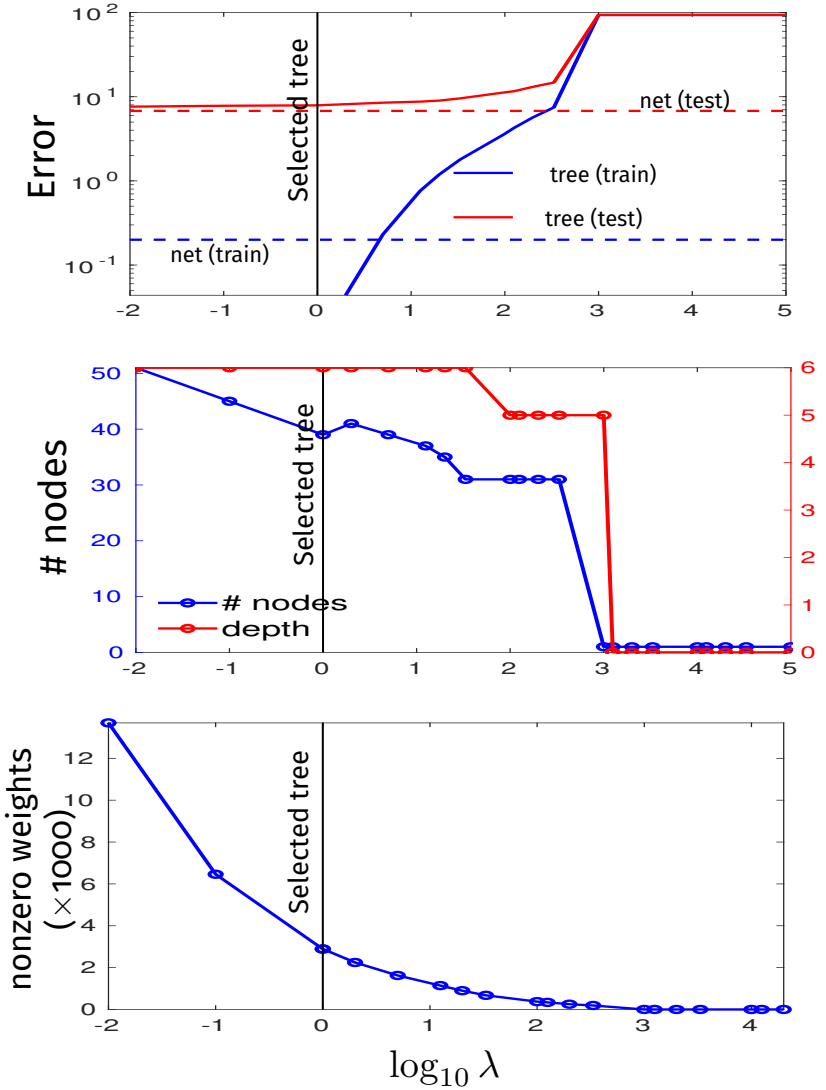
- $\boldsymbol{\mu}^{+} = \{\overbrace{\epsilon,\cdots,\epsilon}^{\mathcal{S}_0}, \underbrace{0,\cdots,0}_{\mathcal{S}_-}, \overbrace{\epsilon,\cdots,\epsilon}^{\mathcal{S}_+}\}$, where ϵ is a small positive number.

None to class k mask



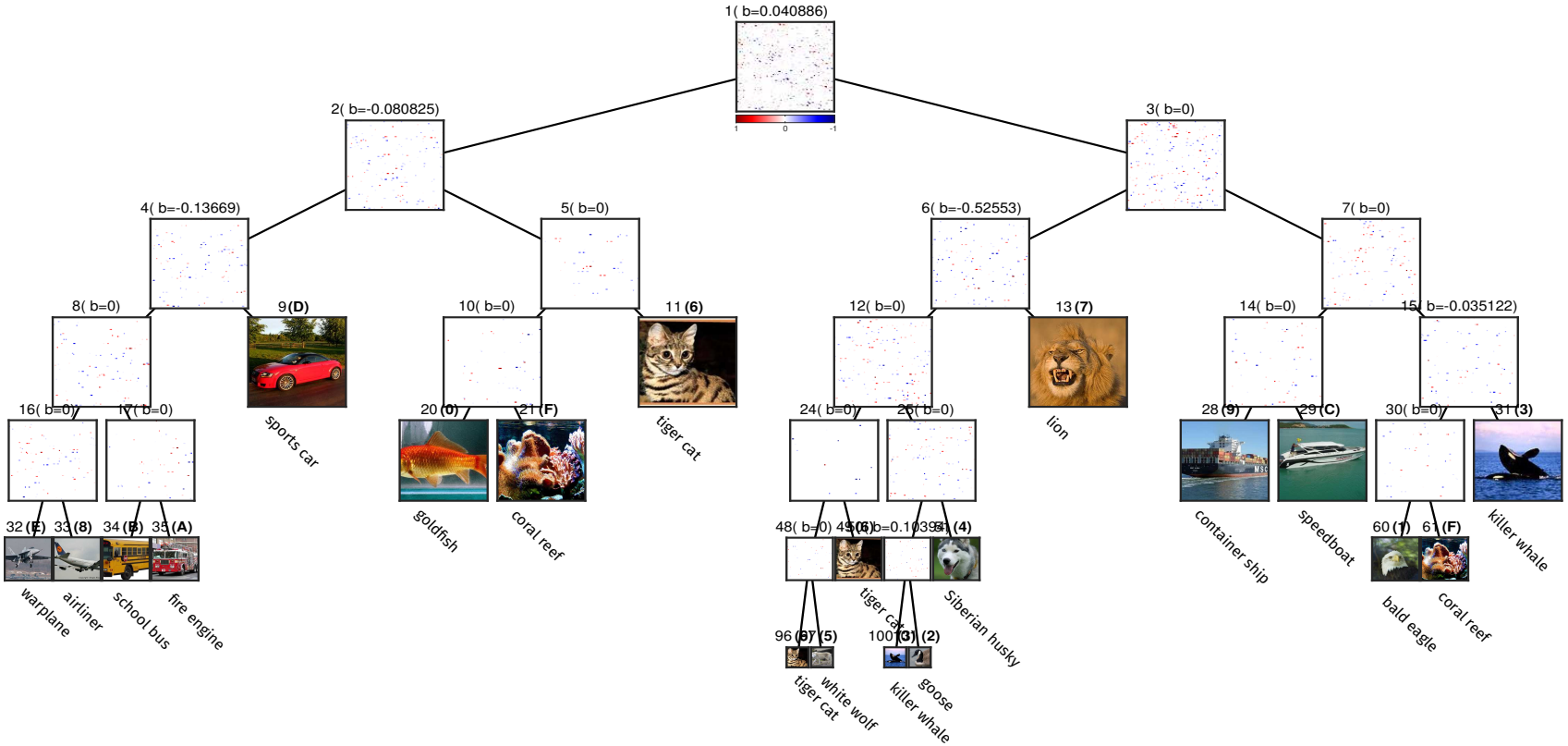
- $\boldsymbol{\mu}^{\times} = \boldsymbol{\mu}_1^{\times} \vee \boldsymbol{\mu}_2^{\times}$.
- $\boldsymbol{\mu}^{+} = \epsilon \boldsymbol{\mu}^{\times}$, where ϵ is a small positive number.

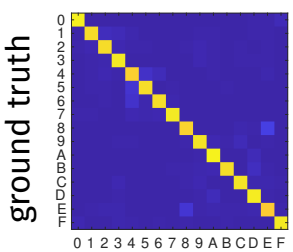
Experiments

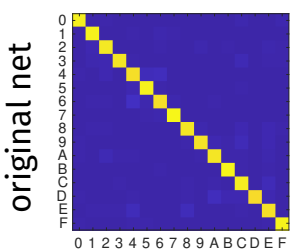


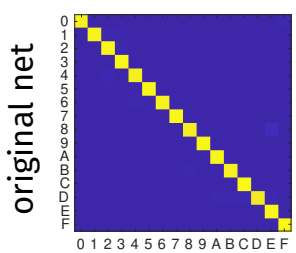
- We use VGG16 network, trained over a subset of 16 classes from ImageNet.
  - Training error: 0.2%
  - Test error: 6.79%
  - $\mathbf{z} \in \mathbb{R}^{8192}$
- We use the tree with $\lambda = 1$.
  - Training error: 0%
  - Test error: 7.9%
  - # nodes: 39
  - features used: 1366 out of 8192 (only 17%)

T with $\lambda = 1$





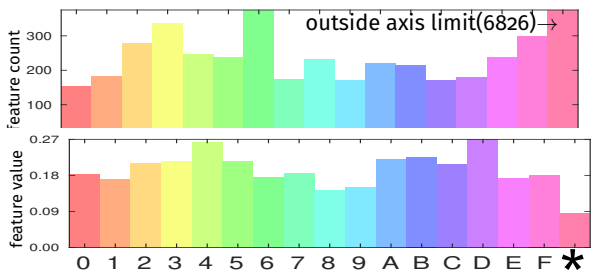


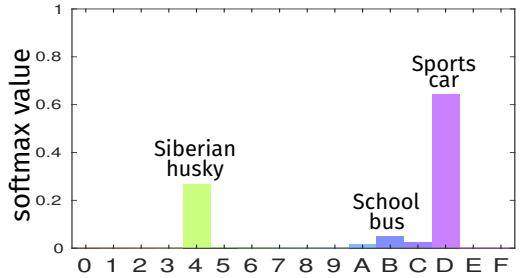


83% neurons masked

Mask on a single image

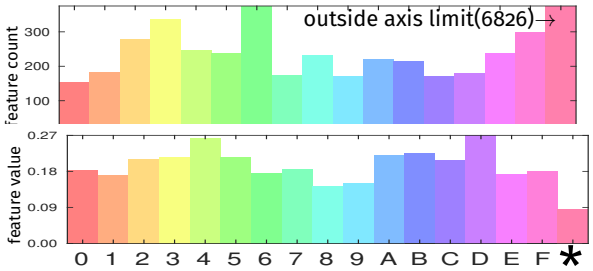
Original

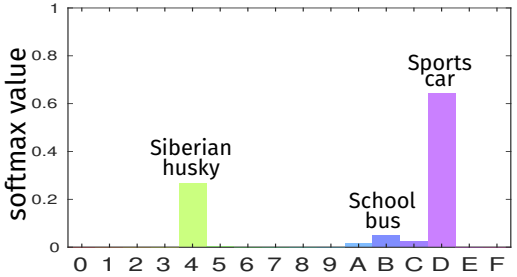




Mask on a single image

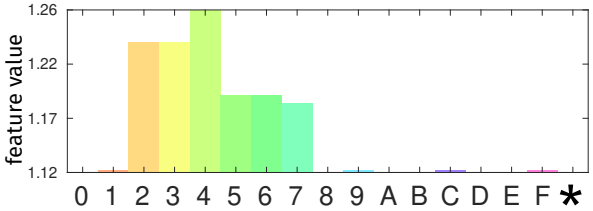
Original

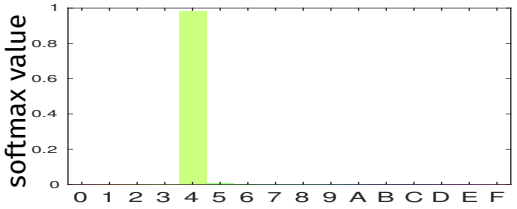




Mask in feature space

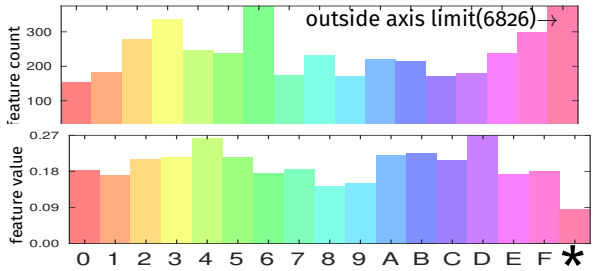
ALL TO CLASS "SIBERIAN HUSKY" mask is applied

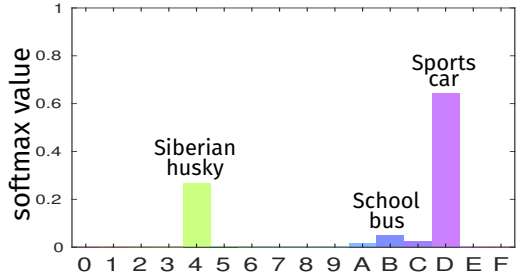




Mask on a single image

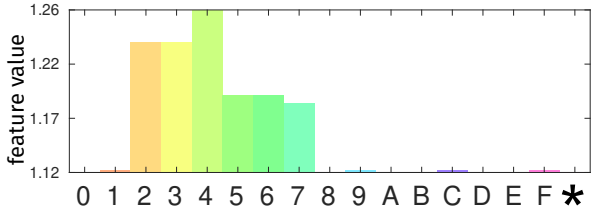
Original

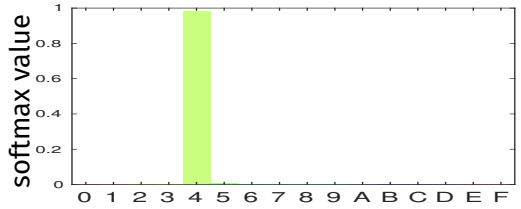




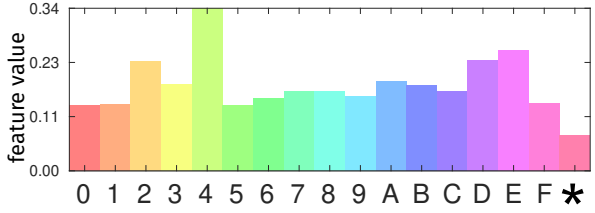
Mask in feature space

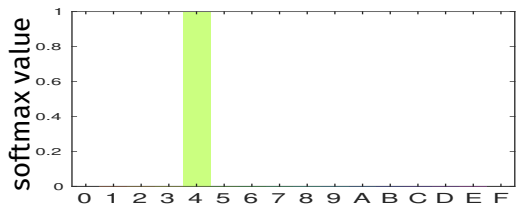
ALL TO CLASS "SIBERIAN HUSKY" mask is applied





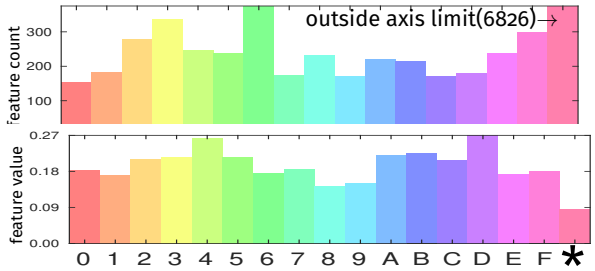
Manual mask in image space

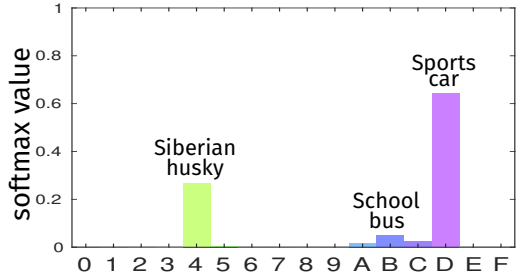




Mask on a single image

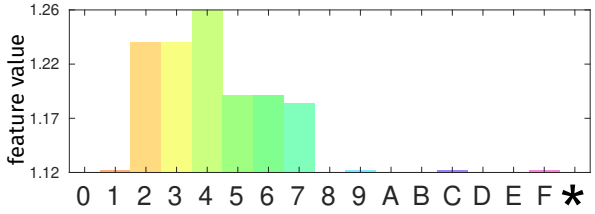
Original

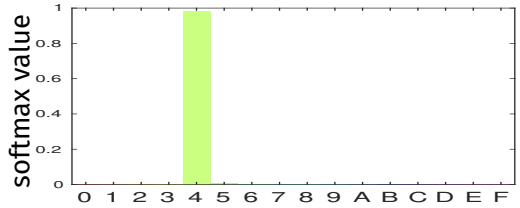




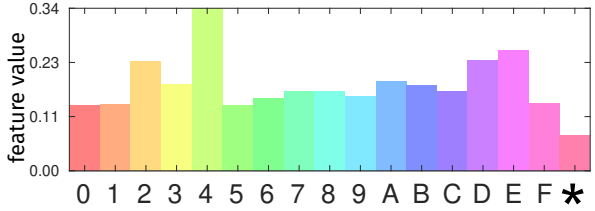
Mask in feature space

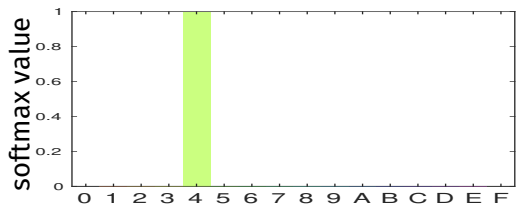
ALL TO CLASS "SIBERIAN HUSKY" mask is applied



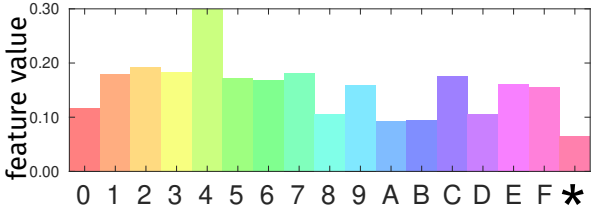


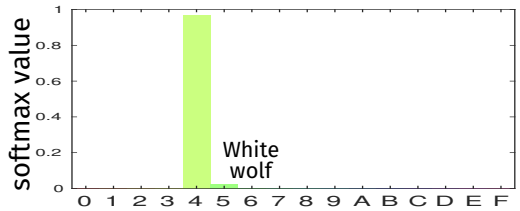
Manual mask in image space





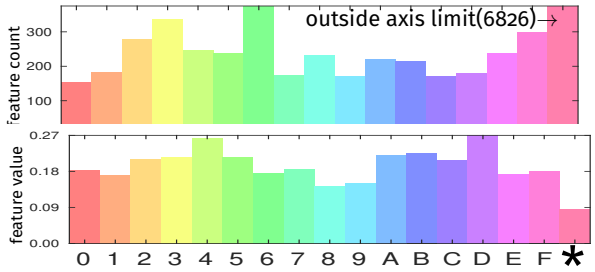
Mask in image space obtained by features

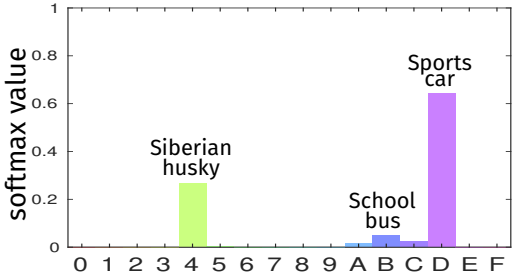




Mask on a single image

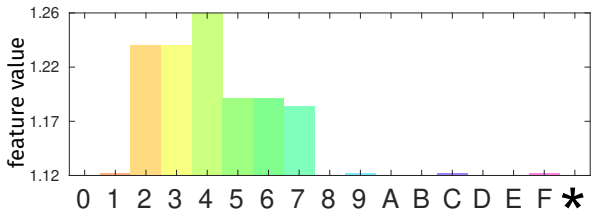
Original

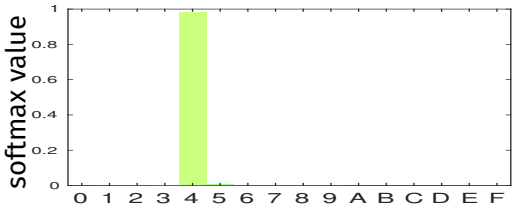




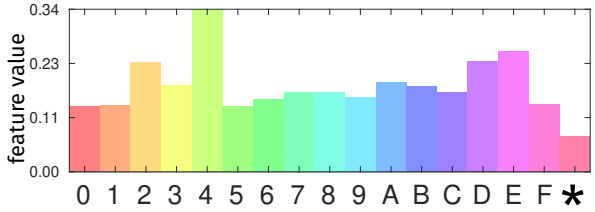
Mask in feature space

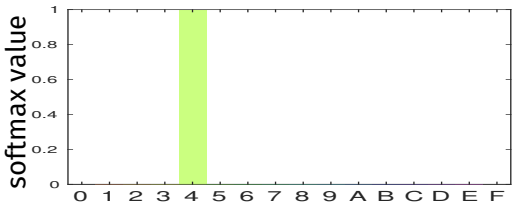
ALL TO CLASS "SIBERIAN HUSKY" mask is applied →



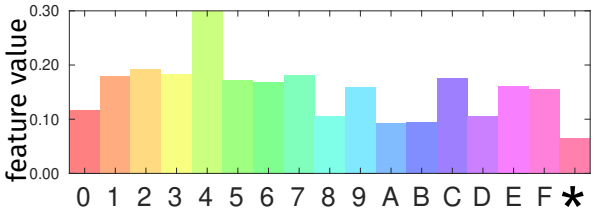


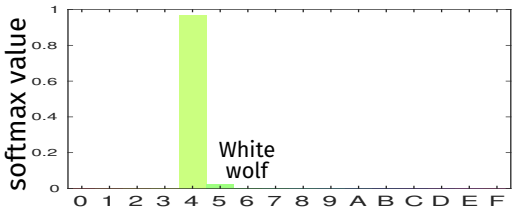
Manual mask in image space





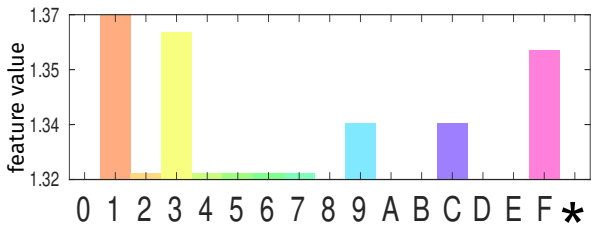
Mask in image space obtained by features

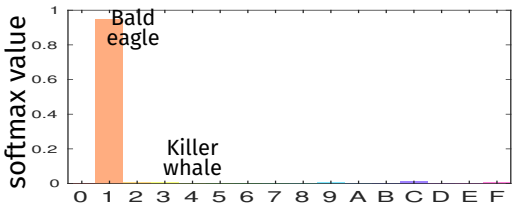




Mask in feature space

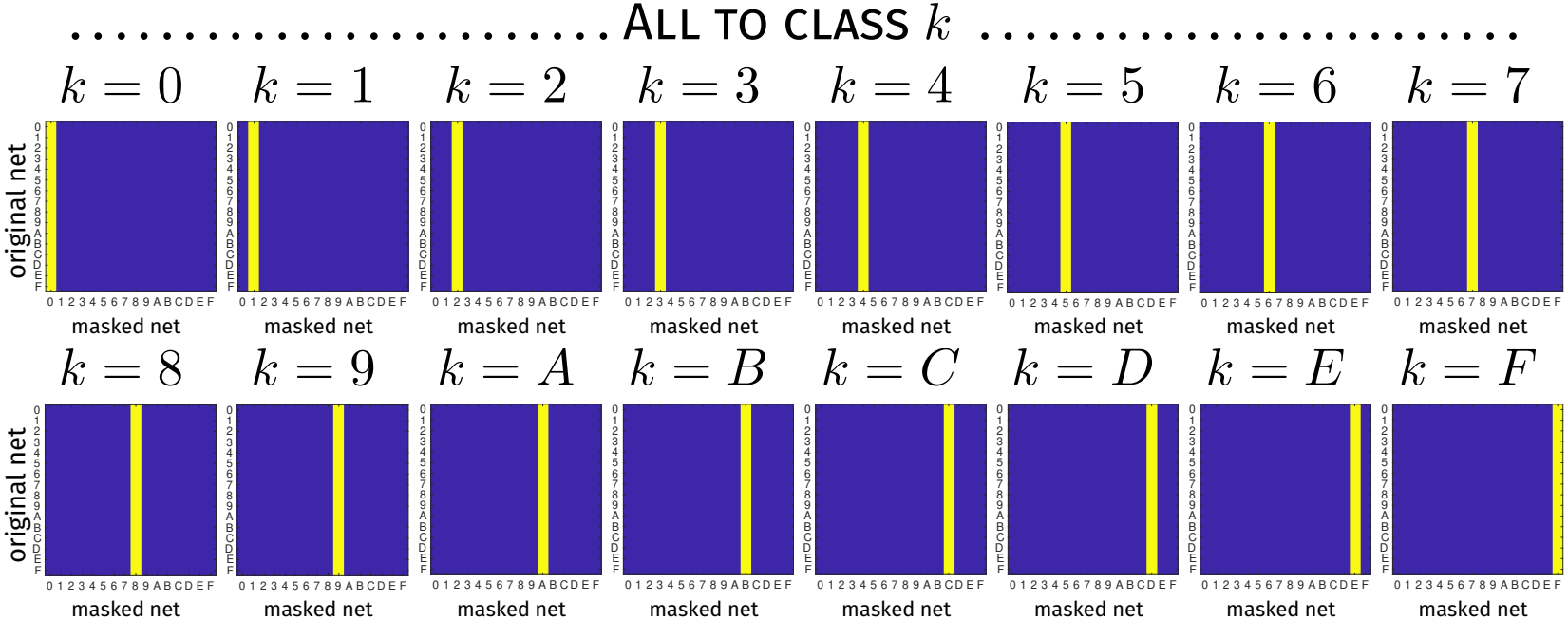
ALL TO CLASS "BALD EAGLE" mask is applied →



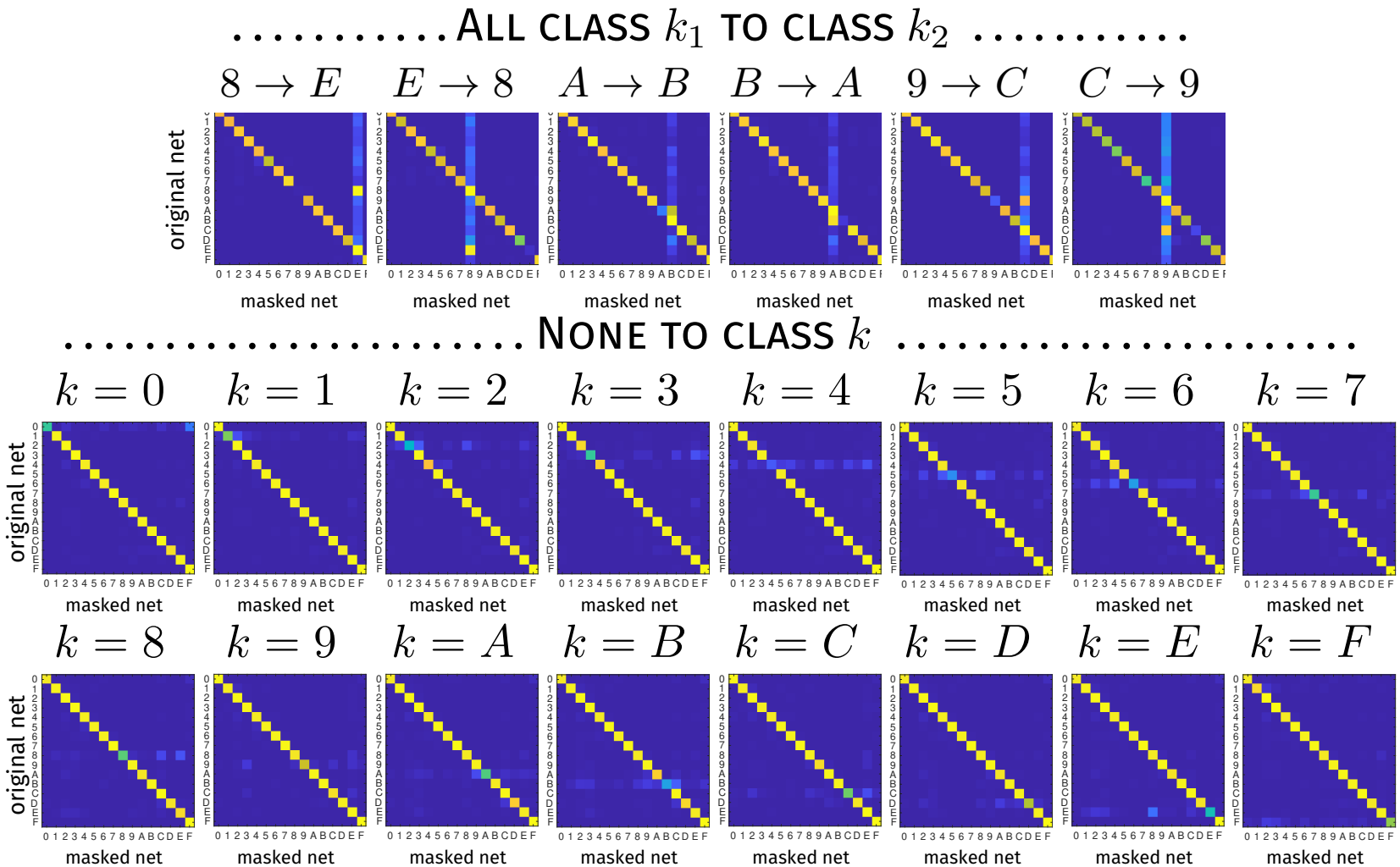


Mask results on the test set

- The masks created in the tree translate almost perfectly to the deep net.



Mask results on the test set

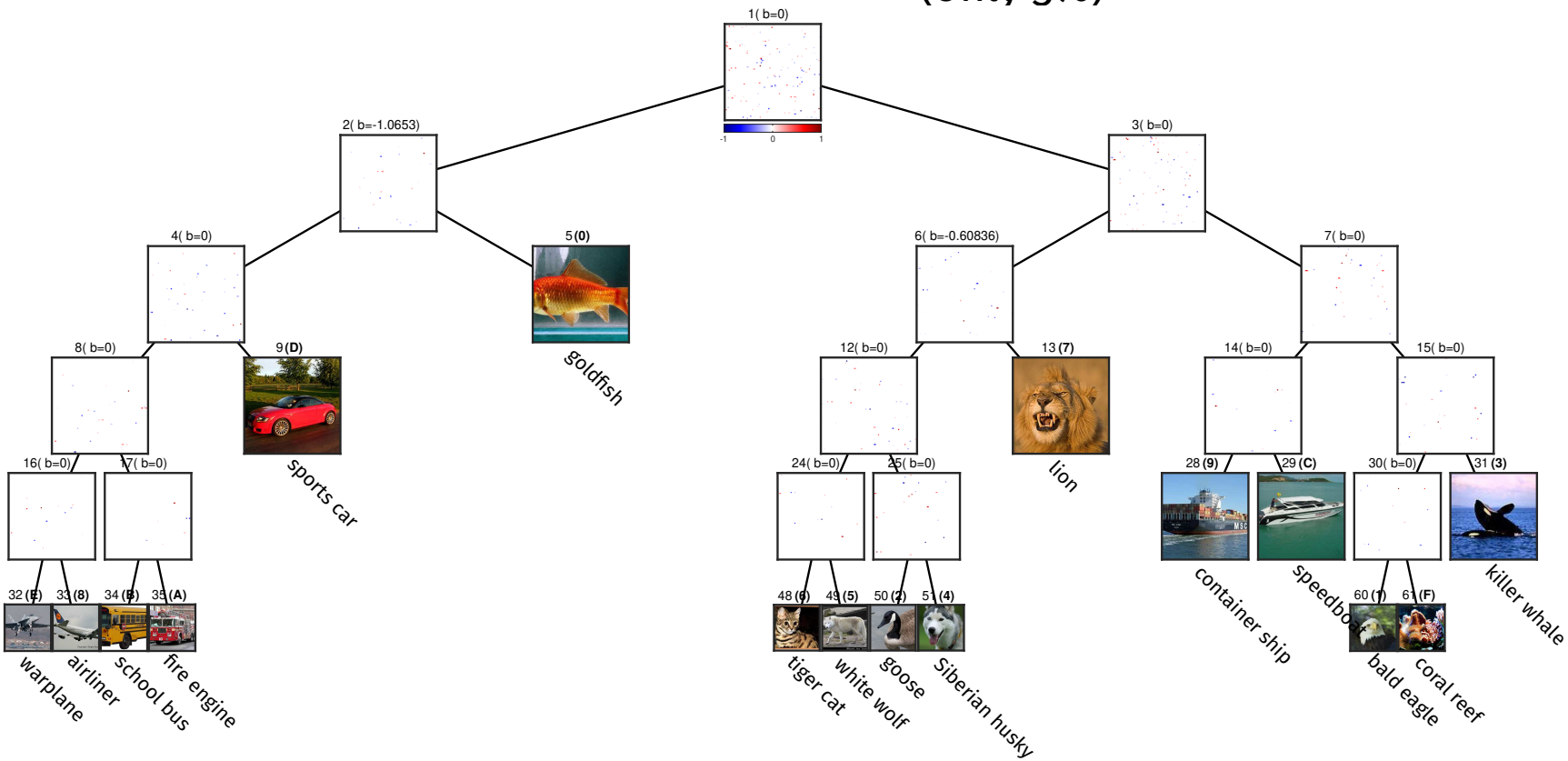


Hierarchy of classes

- An intuitive hierarchy of classes that is primarily related to the background or surroundings of the main object in the image.

- Training error: 1.79%
- Test error: 9.56%

- # nodes: 31
- features used: 408 out of 8192 (only 5%)



Interpreting image datasets using sparse oblique decision trees

Interpreting image datasets

- Interpreting the image datasets is a difficult task, as each image contains a lot of irrelevant data. This makes it hard to understand what part of the image is important or what common concept defines a particular category of the class.
- We do not know how the classes are distributed in the input space; is one class closer to another class, or how two classes or group of classes differentiate with each other; or if there is any sub-groups exist in a given class, if yes what is the difference them; or the selected features are optimal for a class, or sub-group of a class, or even a single instance?
- We address these issues by using sparse oblique trees as a tool to understand the given image dataset.

Extracting features using weights of decision nodes

- Consider a decision node i in T with decision rule:
  - "if $\mathbf{w}_i^T \mathbf{x} + b_i \geq 0$ then go to right child, else go to left child"
    - $\mathbf{w}_i \in \mathbb{R}^D$ is the weight vector.
    - $b_i \in \mathbb{R}$ is the bias.
    - $\mathbf{x} \in \mathbb{R}^D$ is the input.
- Write $\mathbf{w}$ as: $\mathbf{w} = \begin{pmatrix} \mathbf{w}_0 & \mathbf{w}_- & \mathbf{w}_+ \end{pmatrix}$
  - $\mathbf{w}_0 = 0$.
  - $\mathbf{w}_- < 0$.
  - $\mathbf{w}_+ > 0$.
- Call $\mathcal{S}_0$, $\mathcal{S}_-$ and $\mathcal{S}_+$ the corresponding sets of indices in $\mathbf{w}$.

Extracting features using weights of decision nodes

- Consider input $\mathbf{x} \in \mathbb{R}^D$:
  - If $\mathbf{x}$ goes right, we represent the feature selected as a binary vector $\mu_+ \in \{0,1\}^d$, containing ones only at $\mathcal{S}_+$.
  - If $\mathbf{x}$ goes left, we represent the feature selected as a binary vector $\mu_- \in \{0,1\}^d$, containing ones only at $\mathcal{S}_-$.
- We call μ^+ and μ^- the **Node-Features**, where location of one represents features selected by $\mathbf{w}$.

Interpret dataset using **Node-Features**

- For each decision node **Node-Features** represents the features related to left and right subtree. By using **Node-Features**, we can understand what set of features separate a group of classes.

- Features associated with a class k: for each node in the path from the root to leaf for class k collect **Node-Features**, and at the end take logical OR of all **Node-Features**. If there is more than one leaf for class k, take the union of all the features selected.

- For features specific to a given an input $\mathbf{x}$, repeat the process as above, but only for the leaf containing the input $\mathbf{x}$. Next, keep only those features that are active in the $\mathbf{x}$.

Since the data is greyscaled images, we can plot these features to visualize what concept is captured by these features.

Hada and Carreira-Perpiñán, 2022

Interpret dataset using **Node-Features**

- For each decision node **Node-Features** represents the features related to left and right subtree. By using **Node-Features**, we can understand what set of features separate a group of classes.

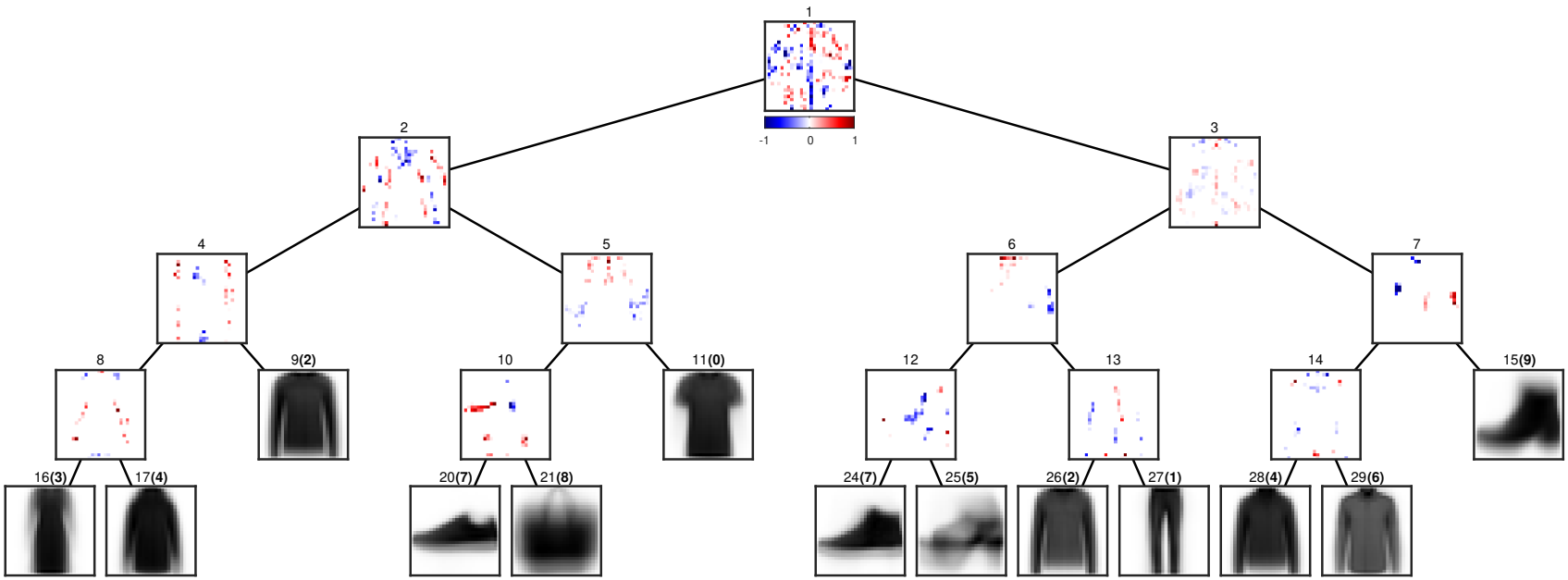
- Features associated with a class k: for each node in the path from the root to leaf for class k collect **Node-Features**, and at the end take logical OR of all **Node-Features**. If there is more than one leaf for class k, take the union of all the features selected.

- For features specific to a given an input $\mathbf{x}$, repeat the process as above, but only for the leaf containing the input $\mathbf{x}$. Next, keep only those features that are active in the $\mathbf{x}$.

Since the data is greyscaled images, we can plot these features to visualize what concept is captured by these features.

Hada and Carreira-Perpiñán, 2022

Fashion-MNIST dataset



- Sub classes within the sneaker class in node # 20 and # 24.

Difference between pair of classes

Decision node # 8

dress class leaf # 16

coat class leaf # 17

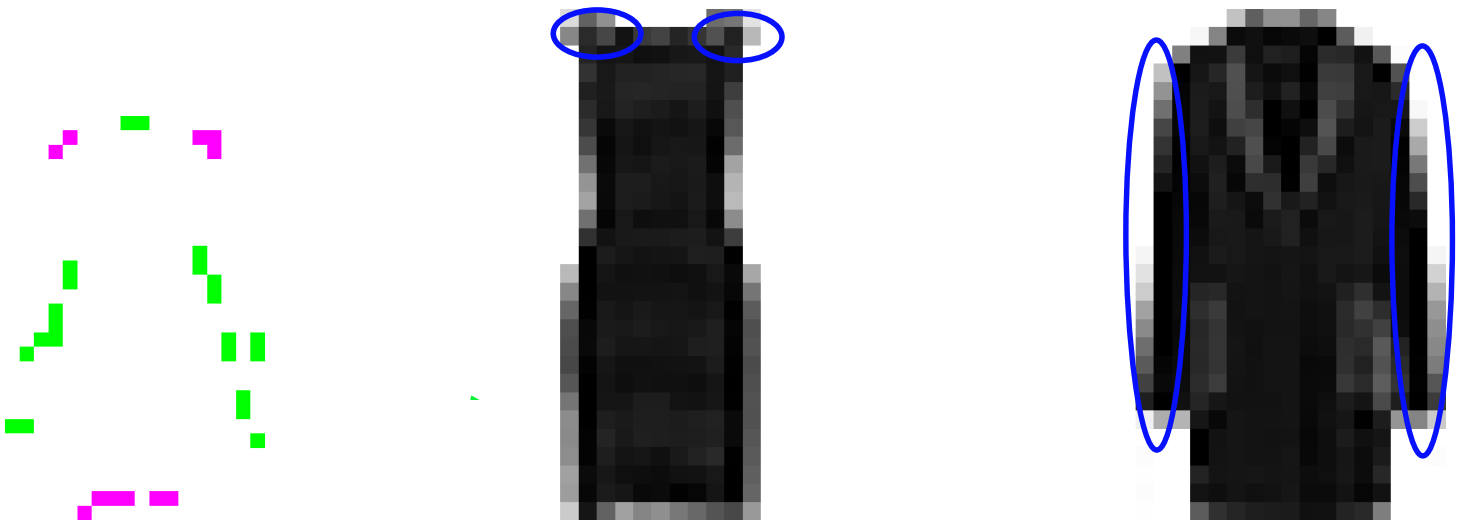


Figure: Magenta color represents negative weights (left child) and green color represents positive weights (right child).

Difference among group of classes

left subtree

right subtree

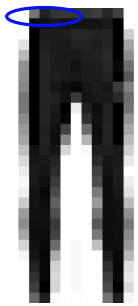
Decision node # 6

sneaker class leaf # 24

sandal class leaf # 25

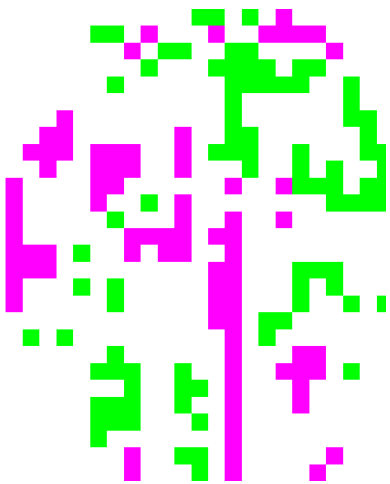
pullover leaf # 26

trouser leaf # 27



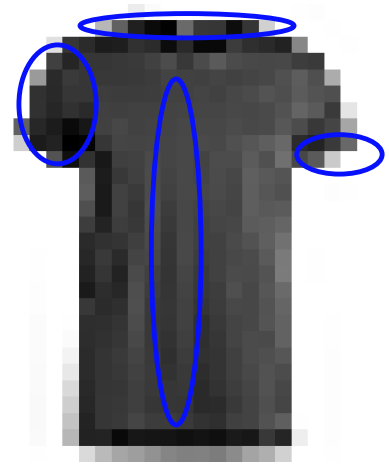
Feature selection for a given instance

Decision node weight



weights along path node # 1, left child

image part selected



Feature selection for a given instance

Decision node weight

weights along path
node # 2, right child

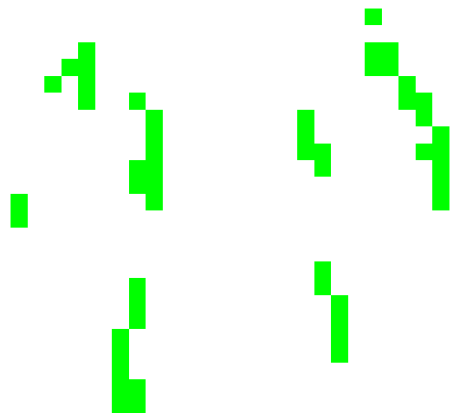
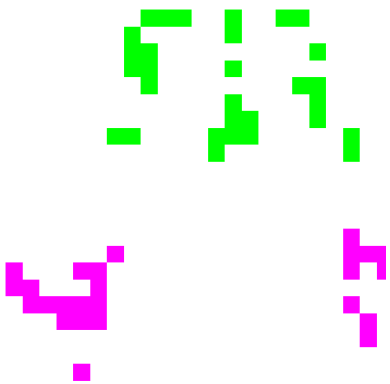


image part selected

Feature selection for a given instance

Decision node weight



weights along path
node # 5, right child

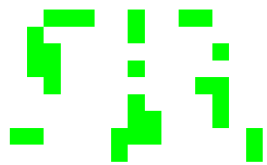


image part selected

Conclusion

Conclusion

- Deep neural nets are accurate but black-box models. In this talk we present two novel approaches to interpret deep nets.
- Sampling the “Inverse Set” of a Neuron
  - By characterizing a neuron’s preference by a diverse set of examples, we can explain this preference in a more detailed manner.
- Interpreting deep neural networks using sparse oblique decision trees
  - Sparse oblique decision trees can be used as a powerful “microscope” to investigate the behavior of deep nets, by learning interpretable yet accurate trees that mimic the classifier part of a deep net.
  - The tree mimick not only allows us to find a group of neurons associated with a class but also provides an overview of how the deep net differentiates between classes.
  - Sparse oblique decision trees can also be used to interpret image datasets.

Future work

- Interpreting NLP datasets
  - Convert raw text data into TF-IDF features.
  - Train sparse oblique trees to find the interesting features similar to image dataset case.
  - Map selected features to corresponding words.

- Extracting class-specific concepts
  - Find the class-specific neurons.
  - Apply “inverse-set” approach to project the neurons into the input space
  - Fine-tune results by setting different activation ranges for different neurons.
  - Generate multiple inverse sets with varying activation ranges to visualize what concepts in the input change the class probability.

Thank you !!

References

- [Hada and Carreira-Perpiñán 2018] S. S. Hada and M. Á. Carreira-Perpiñán: Sampling the "Inverse Set" of a Neuron: An Approach to Understanding Neural Nets. arXiv:1910.04857.
- [Hada and Carreira-Perpiñán 2021] S. S. Hada and M. Á. Carreira-Perpiñán: Sampling the "Inverse Set" of a Neuron: An Approach to Understanding Neural Nets. in proceedings of IEEE International Conference on Image Processing (ICIP) 2021.
- [Hada 2021a] S. S. Hada and M. Á. Carreira-Perpiñán and A. Zharmagambetov: Sparse Oblique Decision Trees:A Tool to Understand and Manipulate Neural Net Features. arXiv:2104.02922.
- [Hada 2022] S. S. Hada and M. Á. Carreira-Perpiñán: Understanding and manipulating neural net features using sparse oblique classification trees. in proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2022.
- [Hada 2021b] S. S. Hada and M. Á. Carreira-Perpiñán and A. Zharmagambetov: Understanding and Manipulating Neural Net Features Using Sparse Oblique Classification Trees in proceedings of IEEE International Conference on Image Processing (ICIP) 2021b.
- [Olah 2017] C. Olah and A. Mordvintsev and L. Schubert: Feature Visualization. Distill, 2017.
- [Carreira-Perpiñán and Tavallali 2018] M. Á. Carreira-Perpiñán and P. Tavallali: Alternating optimization of decision trees, with application to learning sparse oblique trees in NEURIPS 2018.