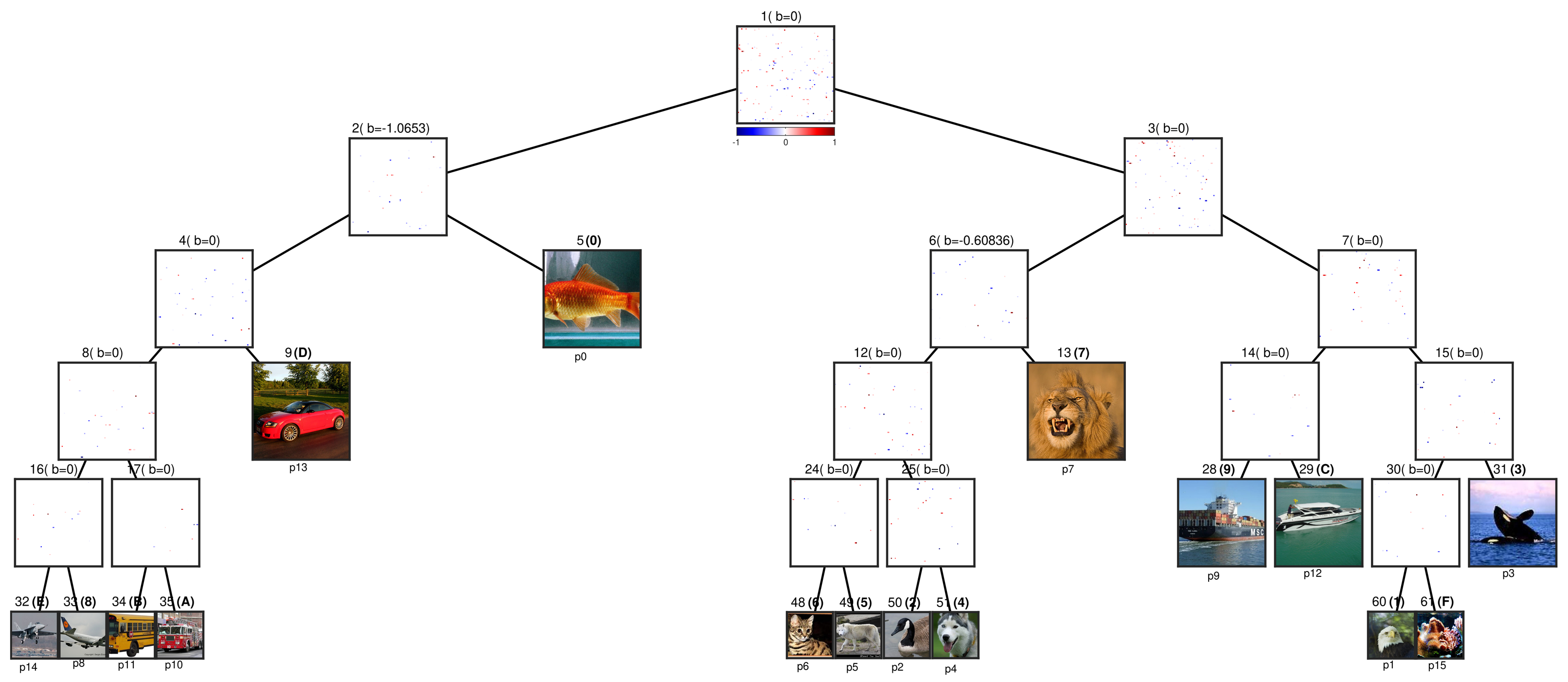
Understanding and Manipulating Neural Net Features Using Sparse Oblique Classification Trees

#3228

Suryabhan Singh Hada and **Miguel Á. Carreira-Perpiñán** and
Arman Zharmagambetov, Dept. CSE, UC Merced

1 Motivation and summary

- Deep neural nets are accurate black-box models.
- Our goal is to understand what internal features computed by the neural net are responsible for a particular class. We achieve this by mimicking the classifier part of the net with a decision tree having sparse weight vectors at the nodes. We can learn accurate enough sparse oblique trees with the tree alternating optimization (TAO) algorithm.
- We found that out of thousands of neurons (in the last layer of feature-extraction part of the net), there is only 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. We show that we can easily manipulate the neural net features in order to make the net predict, or not predict, a given class.
- For VGG16 trained on a subset of ImageNet (16 classes), only 1 366 out of 8 192 (only 17%) neurons are needed to achieve the same performance as the original network. On an average number of neurons associated with a given class is around 200.



⟸ Tree with one class per leaf for VGG16 network trained on subset of ImageNet dataset with 16 classes.

2 Masking of deep net features

- 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 ($\mathbf{y} = \mathbf{g}(\mathbf{z})$).
- Train a sparse oblique tree $y = T(\mathbf{z})$ on the training set $\{(\mathbf{F}(\mathbf{x}_n), y_n)\}_{n=1}^N \subset \mathbb{R}^F \times \{1,\dots,K\}$. Choose the sparsity hyperparameter $\lambda \in [0,\infty)$ such that, T mimicks $\mathbf{g}$ very good and is as sparse as possible. Next, inspect the weights of the decsion nodes to create masks.
- Our masking operation is as follows:
  - 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* and $\boldsymbol{\mu}^+ \geq 0$ is the *additive mask*.
- We show three masks:
  - ALL TO CLASS k: Let $k \in \{1,\dots,K\}$. Classify all instances $\mathbf{x}$ as class k.
  - ALL CLASS k_1 TO CLASS k_2: Let $k_1 \neq k_2 \in \{1,\dots,K\}$. 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 $k \in \{1,\dots,K\}$. For any instance originally classified as k, classify it as any other class. For any other instance, do not alter its classification.

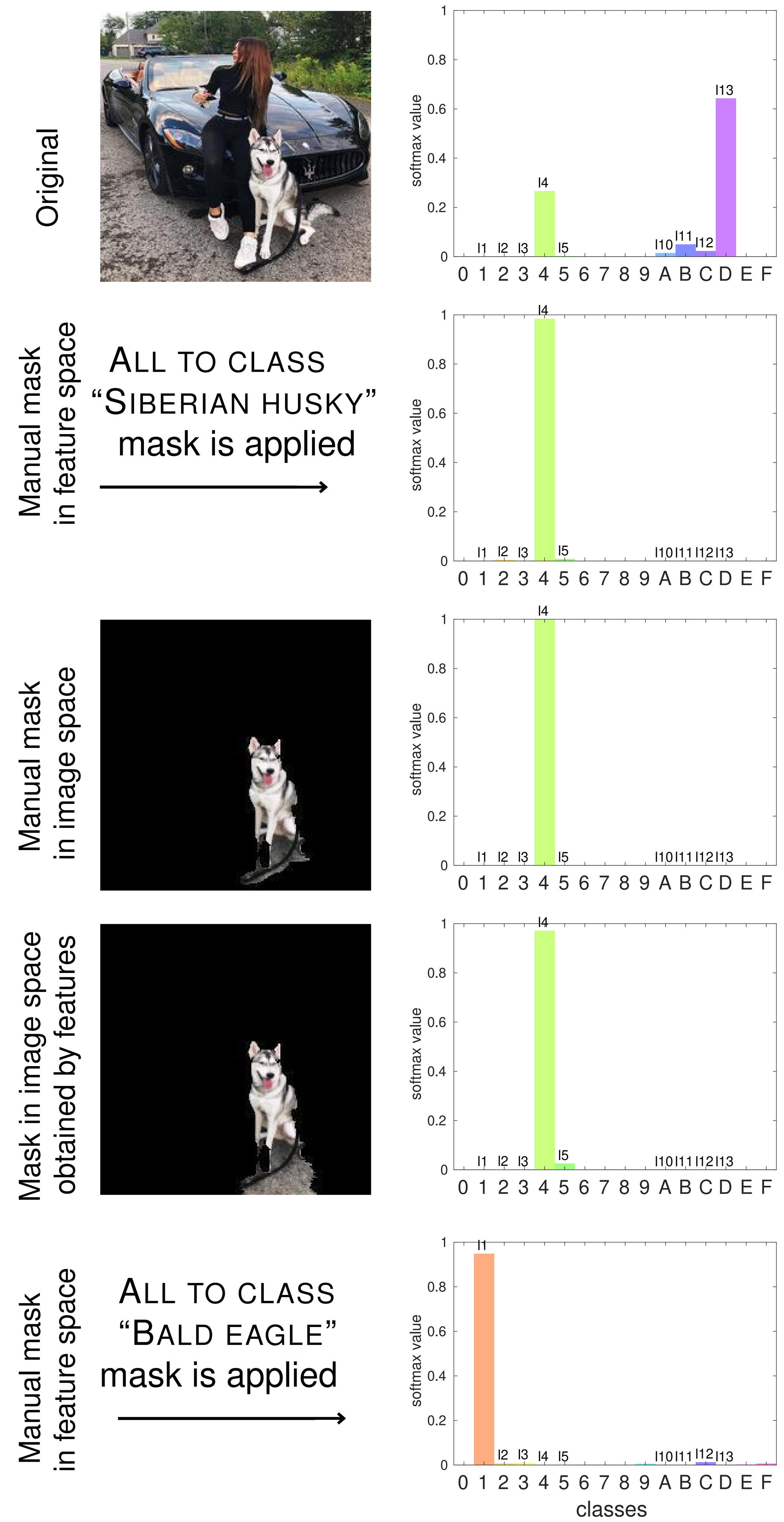


Illustration of masks for a particular image in VGG16. Column 1 shows the image masks (when available). Column 2 shows the histogram of corresponding softmax values. Row 3 shows a mask manually cropped in the image, whose features resemble those of row 2. Row 4 shows a mask in feature space obtained by finding the top-3 superpixels whose features most resemble those of the masked features of row 2.