

---

# *This looks like that: deep learning for interpretable image recognition*

---

**Chaofan Chen**<sup>1\*</sup>  
cfchen@cs.duke.edu

**Oscar Li**<sup>1\*</sup>  
runliang.li@duke.edu

**Alina Barnett**<sup>1</sup>  
abarnett@cs.duke.edu

**Jonathan Su**<sup>3</sup>  
su@ll.mit.edu

**Cynthia Rudin**<sup>1,2</sup>  
cynthia@cs.duke.edu

<sup>1</sup>Department of Computer Science, Duke University, Durham, NC, USA 27708

<sup>2</sup>Department of Electrical and Computer Engineering, Duke University, Durham, NC, USA 27708

<sup>3</sup>MIT Lincoln Laboratory, Lexington, MA 02421-6426<sup>†</sup>

## Abstract

When we are faced with challenging image classification tasks, we often explain our reasoning by dissecting the image, and pointing out prototypical aspects of one class or another. The mounting evidence for each of the classes helps us make our final decision. In this work, we introduce a deep network architecture that reasons in a similar way: the network dissects the image by finding prototypical parts, and combines evidence from the prototypes to make a final classification. The algorithm thus reasons in a way that is qualitatively similar to the way ornithologists, physicians, geologists, architects, and others would explain to people on how to solve challenging image classification tasks. The network uses only image-level labels for training, meaning that there are no labels for parts of images. We demonstrate the method on the CIFAR-10 dataset and 10 classes from the CUB-200-2011 dataset.

## 1 Introduction

How would you describe why the image in Figure 1 looks like a Florida jay? Perhaps the bird’s head looks like that of a prototypical Florida jay, even though its tail might look like that of either a blue jay or a Florida jay. When we describe how we classify images, we might focus on parts of the image and compare them with prototypical parts of images from a given class. For other images, we might look at them holistically to compare with prototypical objects of similar overall shape. This method of reasoning is commonly used in difficult identification tasks: radiologists compare suspected tumors in X-ray scans with prototypical tumor images for diagnosis of cancer; an art historian can identify a painter by looking at both fine-grained details of painting such as the brush-stroke style, as well as coarse-grained details like subject matter and color palette. Beyond radiology and art, this type of image dissection is used in geology (rock/mineral identification), architecture (style identification), fashion, zoology and entomology. The question is whether we can ask a machine learning algorithm

---

\*Contributed equally

<sup>†</sup>DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Assistant Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Assistant Secretary of Defense for Research and Engineering.

to imitate this way of thinking, in order to explain its reasoning process to humans in a way that they might understand.

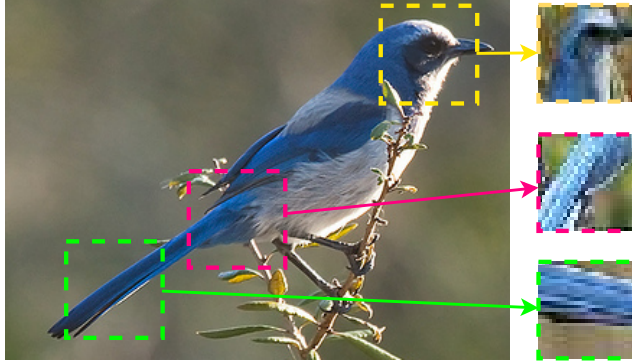


Figure 1: Image of a Florida jay and the learned prototypical parts of a Florida jay used to classify the bird’s species. The smaller images on the right are the prototypical parts of a Florida jay learned by our algorithm. They correspond to parts of a test image (left) as shown.

The goal of this work is to define a form of interpretability in image processing (*this* looks like *that*) that agrees with the way humans describe their own thinking in classification tasks. In this work, we introduce a network architecture that accommodates this definition of interpretability, where the comparison of image parts to learned prototypes is integral to the way our network reasons about new examples. Given an image of a Florida jay in Figure 1, our learning algorithm is able to identify several parts of the image where it thought that *this* identified part of the image looked like *that* prototypical part of training images. The algorithm selects a limited number of prototypical parts for each class that are useful in identifying the class of a new image. It uses an internal notion of distance from a part of the new image to these learned prototypes for providing a predicted class label. In this way, the models introduced here are interpretable, rather than simply explainable, meaning that the reasoning process is actually used by the algorithm, rather than generated afterwards as an explanation.

Our experiments indicate that the accuracy of our interpretable network is comparable with that of analogous standard (non-interpretable) deep networks on datasets of natural images such as CIFAR-10 [Krizhevsky and Hinton, 2009]. The interpretability is gained without losing substantial accuracy.

### 1.1 Related work

Our work relates to those that perform *posthoc* interpretability analysis for trained networks. In posthoc analysis, one interprets a trained network by fitting explanations to how the network performs classification. In this case, a separate modeling effort is required to generate such explanations. A classic approach to understanding networks posthoc is activation maximization (AM), in which the goal is to find an input pattern that maximizes a particular class score [Erhan et al., 2009]. There are other works that perform AM using regularized optimization [Hinton, 2012, Lee et al., 2009, van den Oord et al., 2016, Nguyen et al., 2016, Simonyan et al., 2014, Yosinski et al., 2015], to improve the interpretability of the images from AM. However, the images from regularized AM may not faithfully represent input patterns that maximally activate a network unit, because they are produced by a separate optimization procedure that is not part of training [Montavon et al., 2017]. There is no reason that any network should be inherently interpretable, and so this optimization does not generally lead to meaningful explanations. Alternatives to AM were provided by input-specific (image-specific) posthoc visualization methods. These include deconvolution [Zeiler and Fergus, 2014] and gradient-based saliency visualizations [Simonyan et al., 2014, Sundararajan et al., 2017, Smilkov et al., 2017, Selvaraju et al., 2017]. All of these posthoc visualization methods do not explain the reasoning process of how a network actually makes its decisions. In contrast, our network has a built-in case-based reasoning process, and the explanations generated by our network are actually used during classification and are not created posthoc.

Our work relates closely to works that build interpretability into deep neural networks without using posthoc analysis. Attention mechanisms that attempt to identify the most relevant parts of an input

for various tasks have been integrated into neural networks: Pinheiro and Collobert [2015] trained a classification network that highlights important pixels belonging to an object of interest for weakly supervised image segmentation. Zhou et al. [2016] introduced class activation maps that highlight the regions of an image most responsible for classifying the image into a particular class. Both of these works learn class-specific attention maps, which are jointly trained with proposed networks. Xiao et al. [2015] proposed object-level and part-level attention models that select image patches of interest for fine-grained image classification. Lei et al. [2016] proposed a network architecture for natural language processing that extracts important phrases and uses them as rationales for predictions. Attention mechanisms have also been used in deep learning for speech recognition [Chorowski et al., 2015], image captioning [Xu et al., 2015], visual question answering [Chen et al., 2015], and contour estimation [Xu et al., 2017]. All of these works build interpretability into neural networks by learning which parts of an input are important for their respective tasks. Our work differs from all these works in that our model not only identifies parts of images that are important for classification, but also compare those parts to learned prototypical cases during prediction. Knowing which pixels are important (saliency, attention) tells us only which pixels are used and not *how* those pixels are used for reasoning (consider a correct saliency map with a wrong class label for instance, and how challenging that would be to troubleshoot).

Recently there have also been attempts to quantify the interpretability of visual representations learned by a convolutional neural network (CNN). Bau et al. [2017] proposed the network dissection framework that uses the overlap between the receptive field of top activations and regions corresponding to labeled visual concepts as a measure of the interpretability of the convolutional unit. Zhang et al. [2017] used this measure of interpretability and proposed modifications to traditional CNNs to make them interpretable, by introducing template masks into network architecture and adding regularization terms that encourage filters to be activated by a single class and on a single region. These are useful, but the notion of interpretability considered in this work is different. We do not aim to interpret units inside the network, we are looking instead at explanations that are similar to those made by humans to each other. We do not aim to compare everything identified in the image to a known, labeled, visual class. Instead we aim to pinpoint parts of the image that are important, and similar to prototypical parts of images from a class. Our network can automatically identify, for instance, that a prototypical part of a Florida jay’s head is important for identifying it. It can do this without having seen that part labeled on any image of a Florida jay. It is not limited by what labels have been assigned to parts of training images, and does not need any parts labeled at all.

Our work also relates closely to other prototype classification techniques in machine learning [Bien and Tibshirani, 2011, Kim et al., 2014, Priebe et al., 2003, Chenyue Wu and Esteban G. Tabak, 2017]. It relates most closely to Li et al. [2018], who proposed a network architecture that builds case-based reasoning into a neural network. However, their model requires a decoder for visualizing prototypes, and when trained on datasets of natural images such as CIFAR-10, the decoder fails to produce realistically looking prototype images. In contrast, our model does not require a decoder for prototype visualization. It “pushes” the latent representations of prototypes onto the closest latent representations of training image patches, and uses those training image patches for prototype visualization. Unlike Li et al. [2018], whose model requires the prototypes to have exactly the same shape as the latent representations of images, the prototypes in our model can have much smaller spatial dimensions than the latent representations of images in general, which means that our prototypes are prototypical *parts* of images. This allows for more fine-grained comparisons because different parts of an image can now be compared to different prototypes. It improves over Li et al. [2018] also through easier training due to the removal of the decoder, leading to better explanations.

## 2 Methodology

### 2.1 Network architecture

Our network architecture consists of a regular convolutional neural network  $f_{\alpha_1, \beta_1}$  (network filters and biases denoted by  $\alpha_1$  and  $\beta_1$ ), followed by a prototype layer  $g_p$  and a fully connected layer  $h_{\alpha_2}$  (with weight matrix  $\alpha_2$  and no bias). Given an input image  $\mathbf{x}$ , the convolutional layers of our model extract useful features  $f_{\alpha_1, \beta_1}(\mathbf{x})$  for prediction. Let  $H \times W \times D$  be the shape of the convolutional output  $f_{\alpha_1, \beta_1}(\mathbf{x})$ . The network learns  $m$  prototypes  $\{\mathbf{p}_j\}_{j=1}^m$ , whose shape is  $H_1 \times W_1 \times D$  with  $H_1 \leq H$  and  $W_1 \leq W$ . Since the depth of each prototype is the same as that of the convolutional

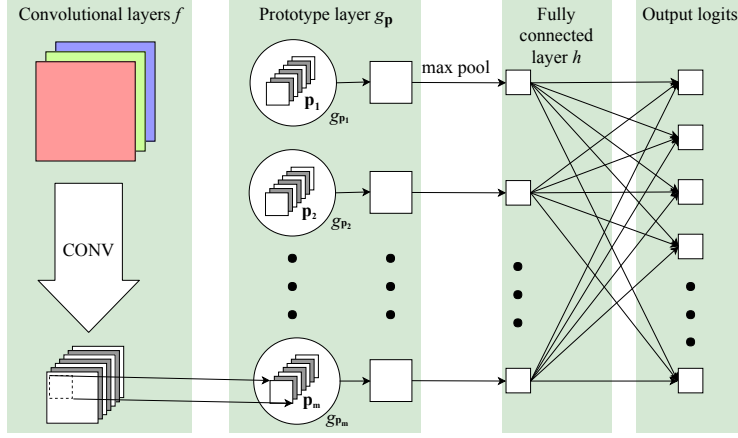


Figure 2: The network architecture.

output but the height and the width of each prototype is less than or equal to those of the convolutional output, each prototype will be used to represent some prototypical activation pattern in a *patch* of the convolutional output, which in turn corresponds to some prototypical patch in the image space. Hence, each prototype  $\mathbf{p}_j$  can be understood as the latent representation of some typical *part* of an image. Given a convolutional output  $\mathbf{z} = f_{\alpha_1, \beta_1}(\mathbf{x})$ , the  $j$ -th prototype unit  $g_{\mathbf{p}_j}$  in the prototype layer  $g_{\mathbf{p}}$  computes the squared  $L^2$  distances between the  $j$ -th prototype  $\mathbf{p}_j$  and all patches of  $\mathbf{z}$  that have the same shape as  $\mathbf{p}_j$ , and transforms the distances into similarity scores using the negative logarithm function. The result is an activation map of similarity scores that preserves the spatial relation of the convolutional output – the upper-left value in the resulting activation map is the similarity score between the upper-left patch of  $\mathbf{z}$  and the prototype. This map of similarity scores is then reduced to a single value using max pooling for each prototype unit  $g_{\mathbf{p}_j}$ . Mathematically, the prototype unit  $g_{\mathbf{p}_j}$  performs the following computation:

$$g_{\mathbf{p}_j}(\mathbf{z}) = \max_{\tilde{\mathbf{z}} \in \text{patches}(\mathbf{z})} -\log(\|\tilde{\mathbf{z}} - \mathbf{p}_j\|_2^2 + \epsilon).$$

If the output of the  $j$ -th prototype unit  $g_{\mathbf{p}_j}$  is large, it means that there is a patch in the convolutional output that is very close to the  $j$ -th prototype in the latent space, and this in turn means that there is a patch in the original input image that has a similar concept to what the  $j$ -th prototype represents.

Hence, given the convolutional output  $\mathbf{z}$ , the prototype layer  $g_{\mathbf{p}}$  produces  $m$  similarity scores between the  $m$  prototypes and the patches of  $\mathbf{z}$  that are most similar to those prototypes. These scores are then multiplied by weight matrix  $\alpha_2$  in the fully connected layer  $h_{\alpha_2}$  to produce the output logits for classification.

## 2.2 Cost function

The cost function for training our network takes into account both classification accuracy and learning interpretable prototypes. Let  $D = [\mathbf{X}, \mathbf{Y}] = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$  be the training set of images, with labels  $y_i \in \{1, \dots, K\}$  for  $i \in \{1, \dots, n\}$ . We use the cross-entropy loss to penalize misclassification on the training data. The optimization problem we aim to solve is as follows:

$$\begin{aligned} \min_{\{\mathbf{p}_i\}_{i=1}^m, \alpha_1, \beta_1, \alpha_2} \quad & \text{CrossEnt}(h_{\alpha_2} \circ g_{\mathbf{p}} \circ f_{\alpha_1, \beta_1}(\mathbf{X}), \mathbf{Y}) + \lambda R_2(\{\mathbf{p}_i\}_{i=1}^m, \mathbf{X}, \mathbf{Y}) + \gamma_1 \|\alpha_1\|_2^2 + \gamma_2 \|\alpha_2\|_1 \\ \text{s.t.} \quad & \min_{i \in \{1, \dots, n\}} \min_{\tilde{\mathbf{z}} \in \text{patches}(f(\mathbf{x}_i))} \|\mathbf{p}_j - \tilde{\mathbf{z}}\|_2^2 = 0 \quad \forall j, \end{aligned}$$

where  $h_{\alpha_2} \circ g_{\mathbf{p}} \circ f_{\alpha_1, \beta_1}(\mathbf{X})$  represents the network's (unnormalized) predictions on training set observations  $\mathbf{X}$ . The cross entropy compares these predictions to the training labels  $\mathbf{Y}$  and encourages accuracy. The term  $R_2$  in our objective is modified from Li et al. [2018]. In their work,  $R_2$  was used to encourage the latent representation of each training image to be close to some prototype. The prototypes in our work are not required to have the same spatial dimensions as latent representations

of images, unlike Li et al. [2018]. In this work, we define  $R_2$  as:

$$R_2(\{\mathbf{p}_j\}_{j=1}^m, \mathbf{X}, \mathbf{Y}) = \frac{1}{n} \sum_{i=1}^n \min_{j \in \{1, \dots, m\}} \min_{\tilde{\mathbf{z}} \in \text{patches}(f(\mathbf{x}_i))} \|\tilde{\mathbf{z}} - \mathbf{p}_j\|_2^2.$$

The minimization of  $R_2$  requires each training image to have some patch whose feature representation is close to at least one prototype. This ensures that the latent space has a clustering structure where the most important patches from the training images will be clustered around the prototypes, which facilitates the  $L^2$  distance based classification performed by our network.

In our cost function, we use  $L^2$  weight decay on the convolutional filters and  $L^1$  regularization on the weight matrix. The use of the  $L^1$  regularization encourages the weight connections between the prototype layer and the output logits to be sparse, so that each prototype contributes to the output logits for only a few classes. This makes it easier for humans to identify the most significant contributions of each prototype to class predictions.

Finally, the constraint in the optimization problem requires each prototype  $\mathbf{p}_j$  to be equal to the latent representation of some training image patch  $\tilde{\mathbf{z}}$ . In this way, each prototype can represent some semantic concept of the corresponding patch in that training image.

### 2.3 Training and prototype visualization

Our optimization technique uses both gradient descent on a relaxed objective (defined shortly) and projection steps. In order to relax the constraint to a differentiable function, we define:

$$R_1(\{\mathbf{p}_j\}_{j=1}^m, \mathbf{X}, \mathbf{Y}) = \frac{1}{m} \sum_{j=1}^m \min_{i \in \{1, \dots, n\}} \min_{\tilde{\mathbf{z}} \in \text{patches}(f(\mathbf{x}_i))} \|\mathbf{p}_j - \tilde{\mathbf{z}}\|_2^2,$$

which is similar to a term of Li et al. [2018], but modified to handle image parts. Adding a multiple of this term to our objective, and removing the explicit constraint, we have formed our relaxed objective that we minimize with gradient descent. The minimization of the relaxed objective allows sufficient exploration of the space as well as ensuring that we do not step too far from the feasible region.

After every few epochs (5 in our experiments) of gradient descent on the relaxed objective, we compute a projection step by projecting  $\mathbf{p}_j$ 's onto the feasible set while only minimally increasing the objective function. To do this, we set each  $\mathbf{p}_j$  to its  $L^2$ -nearest training image patch in the latent space. Since the last iteration is a projection step, the prototypes in the final model are exactly the latent representations of some training image patches. The prototypes can then be visualized by finding the receptive fields of those patches in the pixel space.

## 3 Experiment 1: CIFAR-10

CIFAR-10 is a dataset of  $32 \times 32 \times 3$  color images in 10 classes, with 50,000 training images and 10,000 test images. We use this dataset to demonstrate that our model can achieve comparable test accuracy with that of analogous standard convolutional networks, and it can learn meaningful prototypes that correspond to typical (parts of) objects.

### 3.1 Accuracy

Our network for CIFAR-10 has 8 convolutional layers before the prototype layer and the fully connected layer. The 8 convolutional layers have the same architecture as the first 8 layers in the ALL-CNN-C model described in Springenberg et al. [2014]. We replaced the last convolutional layer and the global average pooling layer in the ALL-CNN-C model with our prototype layer with 30 prototypes of shape  $1 \times 1 \times 192$  and a fully connected layer. During training, we used only three simple data augmentation techniques – vertical and horizontal shift of pixels by at most 10% of height and width, respectively, and random horizontal flip, to improve the generalizability of our network. The highest test accuracy we achieved using our model is 89.58% after the prototypes have been pushed onto the nearest patches of training images in the latent space. We also performed an ablation study by training a standard convolutional network that has a similar architecture to our model. In particular, we replaced the prototype layer with a convolutional layer, which uses 30 convolutional

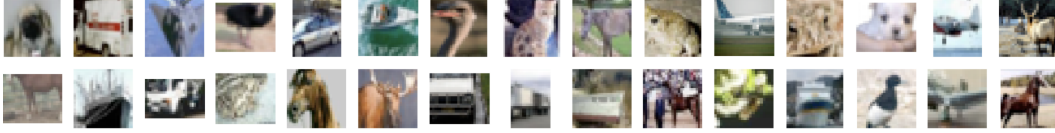


Figure 3: The learned prototypes.

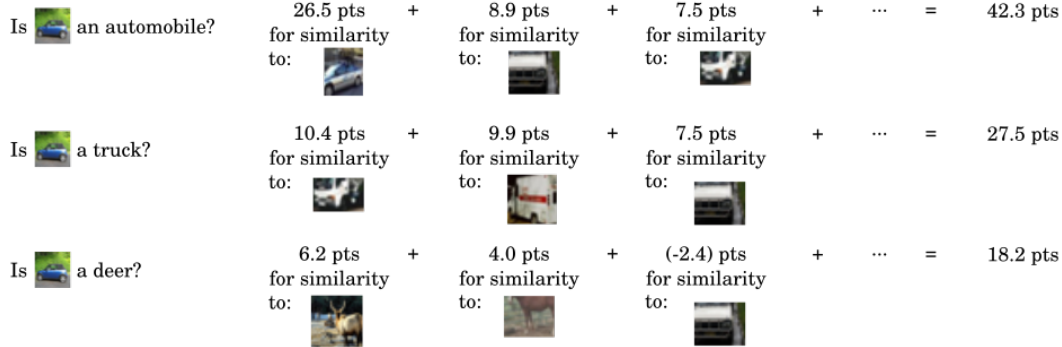


Figure 4: Classifying a new test image (blue car on the left): The class scores for every class (e.g. automobile, truck, deer) are calculated from the similarity to each prototype. The final prediction is the argmax of the class scores; for this example it is “automobile.”

filters of shape  $1 \times 1 \times 192$ . The highest test accuracy achieved by this standard convolutional network in our experiment is 89.30%, with the same data augmentation techniques.

Thus, our network has 89.58% accuracy, whereas the non-interpretable analogy of our method has 89.30%; we did not lose accuracy to gain interpretability.<sup>3</sup>







### 3.2 Visualization and analysis

Figure 3 shows the 30 prototypes visualized in the pixel space. Since the learned prototypes are precisely the latent representations of some patches from training images, we can visualize the prototypes by mapping them back to the receptive fields of the corresponding training images.

To understand how much each prototype contribute to class predictions, we analyze the weight connections between the prototype layer and the output logits. Table 2 shows part of the weight matrix from our trained model on the CIFAR-10 dataset. The full weight matrix can be found in the supplementary material. The leftmost column in the table displays 6 of the prototypes learned by our model, and the remaining columns show the contributions of each prototype unit to the prediction scores of the 10 classes. A positive entry in the weight matrix means that a high similarity to the corresponding prototype contributes positively to the prediction score of the corresponding class – given an image, if it has a patch whose latent representation is very close to the prototype, the corresponding prototype unit will be highly activated, and when the output of this prototype unit is multiplied by a positive value in the weight connection, the result is a positive contribution to the class score. On the other hand, a negative entry in the weight matrix means that a high similarity to the corresponding prototype makes a negative contribution to the prediction score of the corresponding class. Thus, a dog-face prototype should contribute positively to the prediction score of a dog class, but it should contribute negatively to the prediction score of a bird class, for example, because the presence of a dog face should reduce the possibility that the image is that of a bird. This is exactly the case in our weight matrix – Prototype 2 in Table 2 shows the face of a dog, and it contributes positively to the prediction score of the dog class, but it contributes negatively to the prediction score of the bird class. An interesting observation is that this dog-face prototype also contributes somewhat positively to the prediction scores of a cat – this should not be too surprising because dogs look

<sup>3</sup>Note that there is some difference between the best test accuracy we achieved for this data for *any* deep network we tried (89.58%) and the best previously reported accuracy for this dataset (92.75%), reported on the ALL-CNN-C model by Springenberg et al. [2014]; however no source code is available, and we were not able to replicate this level of accuracy.

Table 1: A subset of the weight matrix showing how similarity to each prototype contributes to class score. Values are rounded to the nearest thousandth. The full weight matrix can be found in the supplementary material.

Prototype		Class label									
		air-plane	auto-mobile	bird	cat	deer	dog	frog	horse	ship	truck
0		0	3.302	0	0	0	0	0	0	0	0
1		0	0	0	2.598	-0.012	0.243	0	0	0	-0.260
2		0	0	-0.383	0.920	0	1.356	0.040	0	0	0
3		0	1.078	0	0	0	0	0	0	0	1.491
4		0	1.212	0	0	-0.321	0	0.886	0	0	1.027
5		0	0	0	0	0	0.001	0	2.562	0	0

more similar to cats than to birds. In our weight matrix, there is a significant number of entries with values very close to 0, which means that prototype has no contribution to the prediction score of the corresponding class. This shows the effectiveness of  $L^1$  regularization in achieving the sparsity of the weight matrix.

We now look at how our model reaches a classification decision on a test image of automobile shown in the left of Figure 4. Given this image, our model computes similarity scores of it towards each prototype. The most similar prototypes are prototype 0 in Table 2 (car prototype), prototype 4 in Table 2 (the front of the car), and prototype 3 in Table 2 (the truck prototype), with similarity scores of 8.05, 7.33, and 6.97. For each prototype, the similarity score is then multiplied with the row of the weight matrix associated with that prototype to get the class score contributions from that prototype. Prototype 0 in Table 2 has a weight of 3.302 for automobile and 0 for all other classes. The contribution to the automobile class score is 26.5 points, as shown in Figure 4. The interpretability of our model comes from both the distance-based similarity scores towards meaningful prototypes and an understanding of how these similarity scores contribute to the final class prediction. This model resembles a classification scoring system used by humans, where each score is produced by a sum of similarities to semantic concepts, weighted by the importance of those concepts in classification. In contrast, a standard convolutional neural network can only produce class scores that have no explicit explanations of where these scores come from.

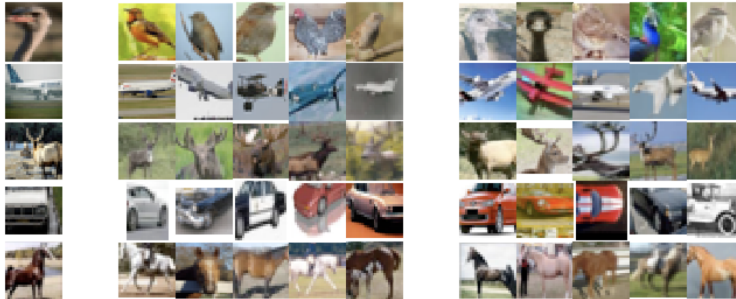


Figure 5: *First column:* 5 prototypes learned from the CIFAR-10 dataset. *Second to sixth columns:* the 5 closest ( $L^2$  distance in the latent space) patches from the training set for each prototype (excluding the patch itself). *Seventh to eleventh columns:* the 5 closest patches from the test set for each prototype. The closest patches for every prototype can be found in the supplementary material.

To understand the cluster structure of the latent space, we find the 5 closest training and test image patches to each prototype in the latent space. Figure 5 shows the closest training and test patches to 5 of the learned prototypes. The closest patches to all of the 30 prototypes can be found in the supplementary material. As shown in Figure 5, each prototype is surrounded by image patches of the



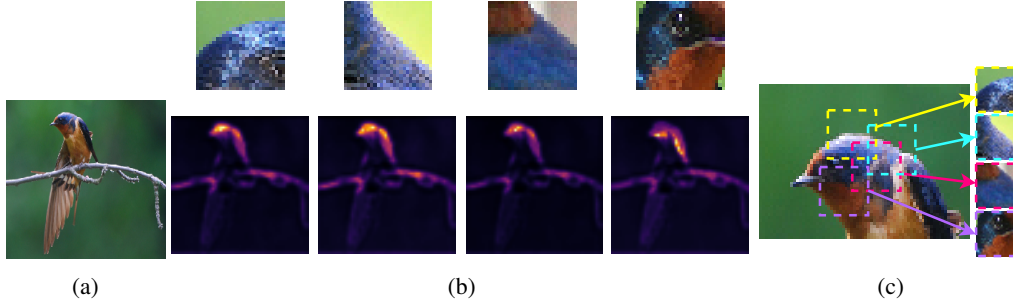


Figure 6: (a) A test image of barn sparrow. (b) *Top*: four learned prototypes from our network. *Bottom*: for each prototype, the corresponding heat map shows where in the test image the prototype is activated. Yellow shows high activation, black shows low activation. (c) The most activated patches of the test image for each of the four learned prototypes in (b).

same semantic concept in the latent space. For example, the (side-facing) horse prototype shown in Figure 5 is surrounded mostly by training and test image patches of side-facing horses in the latent space. We also observe that the nearest patches for each prototype come from distinct object instances with somewhat different viewing angles and colors. This shows that our network is able to learn highly invariant representations that capture the high-level semantic concepts for clustering in the latent space.

## 4 Experiment 2: Bird Identification

CUB-200-2011 is a dataset of color images of 200 bird species [Wah et al., 2011]. In our second experiment, we used the same network architecture as we did on CIFAR-10, and trained our network on 10 bird species from CUB-200-2011: parakeet auklet, indigo bunting, cardinal, gray catbird, Florida jay, song sparrow, barn sparrow, cedar waxwing, downy woodpecker, and common yellowthroat, with 300 training images and 289 test images. Despite the fact that the dataset has a small number of training images (30 per class) and we trained our network from scratch, we were able to achieve 76.12% test accuracy. We are using this experiment to demonstrate the potential of our network in learning prototypical representations of parts of birds that are important for distinguishing different species, and in comparing these prototypes with the relevant parts of an unseen test image.

In Figure 6, we show how our trained network classifies a test image of a barn sparrow. The top of Figure 6(b) displays the learned prototypes from the barn sparrow class, and the bottom of the same figure displays corresponding heat maps that highlight where in the test image the prototypes are activated. Figure 6(c) shows the patches in the original image that produced the highest activations (i.e. had the smallest distances in the latent space) for the learned prototypes. For example, the first prototype shows the crown of the head of a training image, and the crown of the head of the test image is highlighted in the first heat map. The last prototype centers around the eye and the throat of the bird, and the corresponding heat map has the highest activation at the throat and eye of the bird.

Even though we did not use any bounding boxes in training our network, neither did we constrain where the network can select the prototypes, it is striking that our network is able to learn meaningful prototypical representations of relevant parts of birds, and that it can perform comparisons based on these relevant parts.

## 5 Conclusion

There are challenges in designing image classifiers that provide explanations faithful to what the network computes, and similar to those a human might provide. The supplementary material illustrates how humans typically analyze images to help each other with challenging classification tasks. The explanations produced by our network agree with this reasoning style. These networks could be more useful than previous approaches in high-stakes applications, troubleshooting by human-machine teams in challenging image classification tasks, and training humans to identify objects images. The



accuracy provided by our network is comparable with that of analogous but standard (uninterpretable) deep networks; there was nothing sacrificed to gain interpretability.

## References

- David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network Dissection: Quantifying Interpretability of Deep Visual Representations. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 3319–3327. IEEE, 2017.
- J. Bien and R. Tibshirani. Prototype Selection for Interpretable Classification. *Annals of Applied Statistics*, 5(4):2403–2424, 2011.
- Kan Chen, Jiang Wang, Liang-Chieh Chen, Haoyuan Gao, Wei Xu, and Ram Nevatia. Abc-cnn: An attention based convolutional neural network for visual question answering. *arXiv preprint arXiv:1511.05960*, 2015.
- Chenyue Wu and Esteban G. Tabak. Prototypal analysis and prototypal regression. *CoRR*, abs/1701.08916, 2017.
- Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-Based Models for Speech Recognition. In *Advances in neural information processing systems*, pages 577–585, 2015.
- D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing Higher-Layer Features of a Deep Network. Technical Report 1341, University of Montreal, June 2009. Also presented at the ICML 2009 Workshop on Learning Feature Hierarchies, Montreal, Canada.
- Henry Gray, Peter L. Williams, and Lawrence H. Bannister. *Gray’s anatomy*. Edinburgh: Churchill Livingstone, 1995.
- Geoffrey E Hinton. A Practical Guide to Training Restricted Boltzmann Machines. In *Neural networks: Tricks of the trade*, pages 599–619. Springer, 2012.
- Been Kim, Cynthia Rudin, and Julie Shah. The Bayesian Case Model: A Generative Approach for Case-Based Reasoning and Prototype Classification. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1952–1960, 2014.
- Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.
- Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pages 609–616, 2009.
- Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. Rationalizing Neural Predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.
- Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep Learning for Case-Based Reasoning through Prototypes: A Neural Network that Explains Its Predictions. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 2018.
- Melissa Mayntz. House sparrow identification. <https://www.thespruce.com/house-sparrow-identification-385983>, 2016. Posted: 2017-11-16, Accessed: 2018-05-17.
- Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for Interpreting and Understanding Deep Neural Networks. *CoRR*, abs/1706.07979, 2017. URL <http://arxiv.org/abs/1706.07979>.
- Newburyport: Preservation Trust. Newburyport: A showplace of early american domestic architecture. =<http://www.nbptpreservationtrust.org/newburyportarchitecture>. Accessed: 2018-05-17.
- A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 3387–3395, 2016.

- Pedro O Pinheiro and Ronan Collobert. From Image-Level to Pixel-Level Labeling With Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1713–1721, 2015.
- Carey E Priebe, David J Marchette, Jason G DeVinney, and Diego A Socolinsky. Classification Using Class Cover Catch Digraphs. *Journal of classification*, 20(1):003–023, 2003.
- Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.
- Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising Image Classification Models and Saliency Maps. In *International Conference on Learning Representations (ICLR) Workshop*, 2014.
- Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. SmoothGrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.
- Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for Simplicity: The All Convolutional Net. *CoRR*, abs/1412.6806, 2014.
- Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL <http://proceedings.mlr.press/v70/sundararajan17a.html>.
- A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel Recurrent Neural Networks. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1747–1756, 2016.
- C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.
- Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. The Application of Two-Level Attention Models in Deep Convolutional Neural Network for Fine-grained Image Classification. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 842–850. IEEE, 2015.
- Dan Xu, Wanli Ouyang, Xavier Alameda-Pineda, Elisa Ricci, Xiaogang Wang, and Nicu Sebe. Learning Deep Structured Multi-Scale Features using Attention-Gated CRFs for Contour Prediction. In *Advances in Neural Information Processing Systems*, pages 3964–3973, 2017.
- Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.
- Jason Yosinski, Jeff Clune, Thomas Fuchs, and Hod Lipson. Understanding Neural Networks through Deep Visualization. In *In ICML Workshop on Deep Learning*, 2015.
- Matthew D Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 818–833, 2014.
- Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable Convolutional Neural Networks. *arXiv preprint arXiv:1710.00935*, 2017.
- Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 2921–2929. IEEE, 2016.

## Supplementary material

### A Diagrams created by people to explain classifications to other people

Classification of images is often done by hand using prototypical parts. These prototypical parts are labeled with arrows, and descriptions or figures are provided to compare with prototypical cases. For instance, the book Gray’s Anatomy [Gray et al., 1995] has diagrams showing which parts of the image one should consider to diagnose a specific disease, in the case of Figure 7(left) it is a Chiari I Malformation. Another example, in Figure 7(right) is from the field of architecture, illustrating prototypical aspects of a Victorian house.

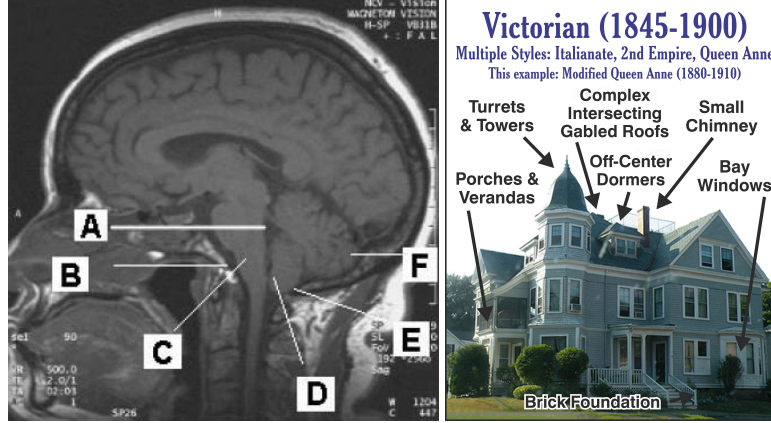


Figure 7: *Left*: Figure reproduced from Gray et al. [1995], illustrating prototypical parts useful for demonstrating how a human would classify a Chiari I Malformation. The descriptions of A-F are in the book. *Right*: An image where prototypical parts of a Victorian house are labeled for purposes of genre identification. Image reproduced from Newburyport: Preservation Trust.

The types of images produced by the method in this paper are also similar to those produced by ornithologists. Figure 8 illustrates an image of a bird with prototypes, as well as a labeled image produced by ornithologists.

### B More detailed results

Table 2 shows the weight matrix from our algorithm on the CIFAR-10 dataset. Each entry shows how similarity to a prototype contributes to class score. If the score is positive, looking like the prototype increases class score for that class. If the score is negative, looking like that prototype decreases the class score of that class.

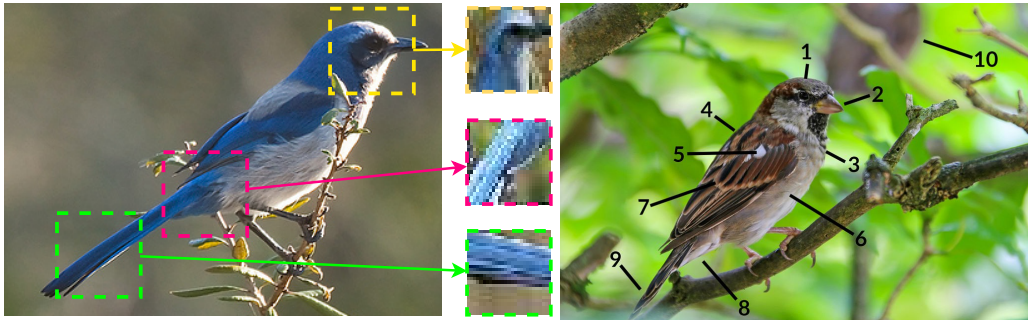


Figure 8: *Left*: Image of a Florida jay and the learned prototypical parts of a Florida jay used to classify the bird’s species. The smaller images on the right are the prototypical parts of a Florida jay learned by our algorithm. They correspond to parts of a test image (left) as shown. *Right*: Human-labeled image of a sparrow, reproduced from Mayntz [2016].

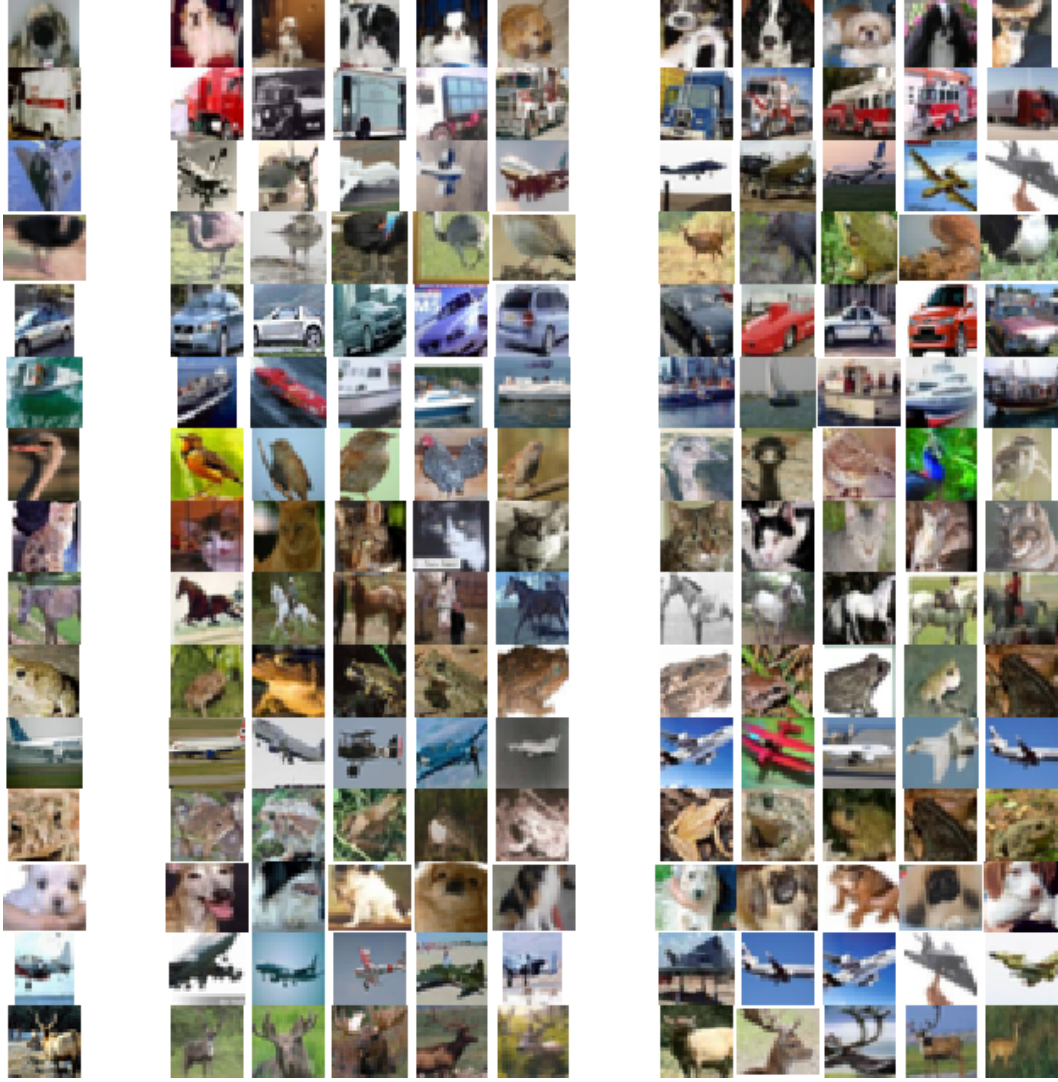
































Figure 9: *First column:* the first 15 prototypes learned from the CIFAR-10 dataset. *Second to sixth columns:* the patches from the training set which are closest to the prototype (excluding the prototype itself). In order of distance where the leftmost image is closest to the prototype (smallest  $L^2$  distance in the latent space). *Seventh to eleventh columns:* the patches from the test set which are closest to the prototype. In order of distance where the leftmost image is closest to the prototype.

Figures 9 and 10 show the prototypes learned from the CIFAR-10 dataset in the left-hand column. For each prototype, the five closest (smallest  $L^2$  distance in the latent space) patches from the training set are shown in the middle and the five closest patches from the test set are shown on the right.

Table 2: The weight matrix showing how similarity to each prototype corresponds to class score. Values are rounded to the nearest thousandth.

Prototype		Class label									
		air-plane	auto-mobile	bird	cat	deer	dog	frog	horse	ship	truck
0		0	0	0	-0.092	0	3.381	0	0	0	0
1		0	-0.036	0	1.282	0	0	0	0	0	2.480
2		1.092	0	0	0	0.188	0	0	0	-0.003	0
3		0	0	0.990	0	0.380	0	0	0	0	0
4		0	3.302	0	0	0	0	0	0	0	0
5		0	0	0	0	0	0	0	0	1.000	0
6		0	0	2.846	0	0	0	0	0	0	0
7		0	0	0	2.598	-0.012	0.243	0	0	0	-0.260
8		0	0	0	0	0	0.183	0	1.450	0	0
9		0	0	0	0	0	0	2.034	0	0	0
10		1.125	0	0	0	0	0	0	0	0	0
11		0	0	0	0	0	0	1.058	0	0	0
12		0	0	-0.383	0.920	0	1.356	0.040	0	0	0
13		1.541	0	0	0	0	0	0	0	0	0
14		0	0	0	0	2.030	0	0	0	0	0
15		-0.085	0	0	0	1.161	0.359	0	0.150	0	0
16		0	0	0	0	-0.001	0	0	0	0.716	0
17		0	1.078	0	0	0	0	0	0	0	1.491
18		0	0	0	0	0	0	0.764	0	0	0
19		0	0	0	0	0	0.001	-0.013	0.380	0	0
20		0	0	0	0	1.800	0	0	0	0	0
21		0	1.212	0	0	-0.321	0	0.886	0	0	1.027
22		0.399	-0.105	0.015	-0.137	0	-0.205	-0.027	0	0.555	0.147
23		0	0	0	0	-0.055	0	0	0	0.532	0
24		0	0	0	0	0.433	0.004	0	0.707	0	0
25		0	0	0	0.162	0	0	0.933	0	0	0
26		0	0	0	0	0	0	0	0	2.015	0
27		0	0	1.397	0	-0.233	0.126	0	0.081	0	0
28		1.032	0	0	0	0	0	0	0	0	0
29		0	0	0	0	0	0.001	0	2.562	0	0



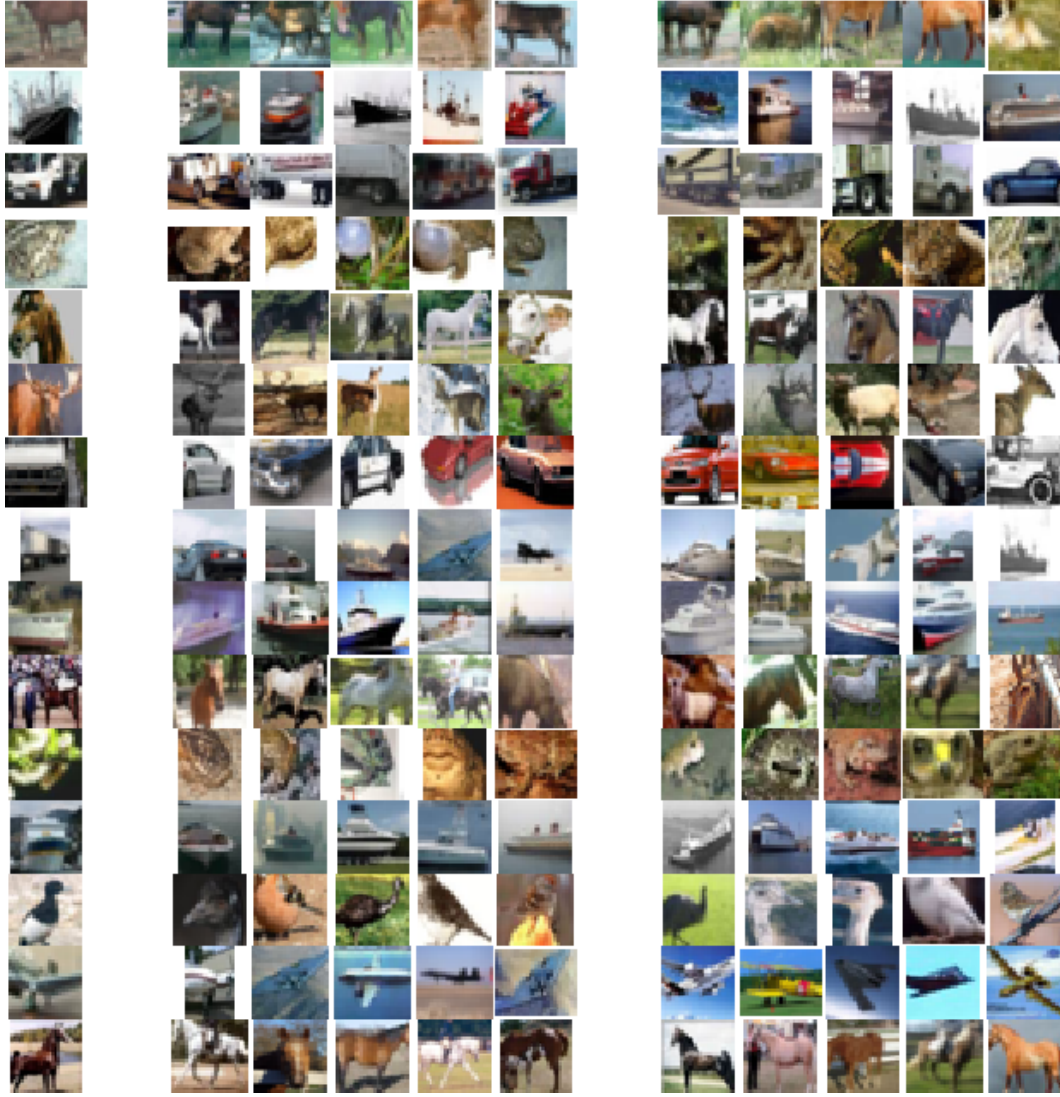


Figure 10: *First column:* the second 15 prototypes learned from the CIFAR-10 dataset. *Second to sixth columns:* the patches from the training set which are closest to the prototype (excluding the prototype itself). In order of distance where the leftmost image is closest to the prototype (smallest  $L^2$  distance in the latent space). *Seventh to eleventh columns:* the patches from the test set which are closest to the prototype. In order of distance where the leftmost image is closest to the prototype.