Understanding Convolutional Networks with APPLE : Automatic Patch Pattern Labeling for Explanation

Sandeep Konam, Ian Quah, Stephanie Rosenthal, Manuela Veloso
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213 USA
{skonam,itq}@andrew.cmu.edu,srosenthal@sei.cmu.edu,veloso@cs.cmu.edu

Abstract

With the success of deep learning, recent efforts have been focused on analyzing how learned networks make their classifications. We are interested in analyzing the network output based on the network structure and information flow through the network layers. We contribute an algorithm for 1) analyzing a deep network to find neurons that are "important" in terms of the network classification outcome, and 2) automatically labeling the patches of the input image that activate these important neurons. We propose several measures of importance for neurons and demonstrate that our technique can be used to gain insight into, and explain how a network decomposes an image to make its final classification.

Introduction

Deep learning models have been shown to improve accuracy in a variety of application domains including image classification (Krizhevsky, Sutskever, and Hinton 2012; Szegedy et al. 2015), object detection ((Girshick 2015; He et al. 2016)), and even robotics (e.g., localization (Yang et al. 2016), navigation (Zhu et al. 2016), motion planning (Wulfmeier, Wang, and Posner 2016) and manipulation (Zhang et al. 2015)). Despite its success, there is still little insight into the internal operation and behavior of deep network models, or how they achieve such good performance (Zeiler and Fergus 2014).

Many different algorithms have been proposed with the goal of explaining deep learning models, particularly for convolutional neural networks (CNNs) which analyze images. For example, Class Activation Maps (CAM) explain the output classification by visualizing a CNN's most discriminative pixels in the input image (Zhou et al. 2015). Local Interpretable Model-agnostic Explanations (LIME) aims to learn a more interpretable model locally around the prediction and highlight the super-pixels with positive weight towards a specific class (Ribeiro, Singh, and Guestrin 2016). While these approaches have been successful in visualizing important pixels for classification, they do not explain how the network used those features to make the prediction.

To address the challenge of determining how a network uses image features, Fergus & Zeiler (Zeiler and Fergus 2014) used deconvolutional networks (Zeiler, Taylor, and Fergus 2011) to visualize patterns that activate each neuron in the network (e.g., facial features such as the eyes or nose, other defining components of animals such as fur patterns, or even whole objects). However, their analysis of individual neurons and their image patch patterns was accomplished largely manually, which is impractical given the size of today's state-of-the-art networks as well as the number of images that are tested.

Building on this prior work, we contribute an algorithm to automatically label the features of an image that the network focuses on in order to explain why the network made its prediction. We accomplish this by analyzing the neurons that are most important to the output classification of an image as well as the patterns that activate those neurons. Our proposed approach, APPLE (Automatic Patch Pattern Labeling for Explanation), first analyzes the signal propagation through each layer of the network in order to find neurons that contribute highly to the signal in subsequent layers. Then, it deconvolves the important neurons at each layer to determine the parts or patches of the image that these neurons use as their input. Finally, our algorithm automatically labels the image patches with attributes that describe the object in the image using a separately-trained classifier.

We contribute several measures of neuron importance within a network and we demonstrate that our algorithm is able to use the measures to identify neurons that focus on important attributes of recognized objects. Beyond simply highlighting important pixels for visualization purposes, our image patch labels can be used to explain an image classification without requiring a human to decipher the image or manually probe the reasoning behind the prediction. On image classification tasks, we demonstrate that our APPLE algorithm reduces manual coding, finds important features of images, and automatically classifies those features for human interpretability.

Related work

We divide the prior research pertaining to understanding a CNN's predictions into two categories: weakly supervised localization and network structure analysis.

In weakly supervised localization, the objective is to highlight the object features (pixels) within an image. For example, Class Activation Mapping (CAM) for CNNs use global

average pooling to visualize the discriminative object parts detected by the CNN (Zhou et al. 2015). Similar methods use other pooling techniques (e.g., global max pooling (Oquab et al. 2015) and log-sum-exp pooling (Pinheiro and Collobert 2015)), or reduce the network structure requirements that CAM imposes (Selvaraju et al. 2016). Local Interpretable Model-agnostic Explanations (LIME) is another weakly supervised localization technique (Ribeiro, Singh, and Guestrin 2016). In contrast to CAM which highlights discrimitive pixels, LIME finds a sparse linear approximation to the local decision boundary of a given black-box ML system including CNNs. Its visualization on an image allows a human operator to inspect how the classification depends locally on the most important input features. While these techniques use different measures for determining which pixels in the image are most important for classification, they do not analyze how the pixel features are propagated through the network to arrive at a prediction.

We build on prior work that aims to understand how information propogates through a network (e.g., (Erhan et al. 2009; Springenberg et al. 2014; Mahendran and Vedaldi 2015; Zeiler and Fergus 2014)). For example, (Erhan et al. 2009) find the optimal stimulus for each neuron by performing gradient descent in image space to maximize the neuron's activation. Our work uses deconvolutional networks, which are used to visualize which patterns activate each neuron (Zeiler, Taylor, and Fergus 2011; Zeiler and Fergus 2014). Although most of the network structural analyses, including (Zeiler and Fergus 2014), provide insights into a CNN's classification at the neuron-level, they require human intervention to manually analyze the activations or the important image patches to interpret how a network made a prediction. This manual process does not scale as the network gets bigger or as the number of images to analyze increases.

There is another category of work related to neural caption generation, where models learn to generate textual justifications for classifications of the primary neural model. (Hendricks et al. 2016) use an LSTM caption generation model with a loss function that encourages class discriminative information to generate justifications for the image classification of a CNN. (Park et al. 2016) produce both textual justification and a visual attention map. (Vedantam et al. 2017) produce captions that are locally discriminative, in the context of other images. However, all of these works use natural language descriptions at a large scale, collecting which has a prohibitive cost, compared to our work which only uses labels corresponding to important attributes of the objects.

Our Automatic Patch Pattern Labeling for Explanation (APPLE) algorithm is built on top of Fergus & Zeiler (Zeiler and Fergus 2014), but does not require any manual probing to understand the image patches that activate individual neurons, and we eliminate the need to analyze all of the neurons. In particular, we focus only on analyzing neurons that are important to the network. By assessing the ranked image patches corresponding to important neurons, APPLE simultaneously localizes the object within the image while also automatically labeling those patches with object feature attributes (e.g., eyes, nose, paws as attributes of the object class polar bears).

Automatic Patch Pattern Labeling for Explanation (APPLE)

Our goal is to explain the output of a CNN image classifier by ranking the neurons based on importance (i.e., most contribution to the final classification) and automatically labeling their corresponding patches of the image. For example, when classifying images of polar bears, some neurons within the network are activated more than others and it is likely that the most active neurons are detecting important attributes of polar bears (e.g., their eyes, nose, or paws). We produce a secondary classifier which takes as input the patches which correspond to the important or most active neurons and automatically labels them based on a predefined list of bear attributes. The combination of the important image patches and corresponding predicted attribute labels of those patches provide a qualitative understanding of what the CNN is using to classify polar bears based on what appears in its important regions in the image.

In order to accomplish this goal, we propose our APPLE algorithm that:

1. finds high importance neurons within the CNN,
2. deconvolves the network to determine the patch of the image that each important neuron looks at, and
3. automatically labels those patches using a secondary classifier to determine object features that the patch contains.

High importance Neurons

We base our importance functions on the deep network structure. In that structure, a neuron η in row j' and column k' in layer l of a deep network takes input signal $X_{c',j',k'}$ on channel c' (i.e., in images, r, g, and b each represent a channel). Neurons in layer l are connected to layer $l+1$ by weights, W. A neuron's weight matrix $W_{\eta',\eta,m,n}$ weighs the activations of neuron η in layer l and the rectangle of $m \times n$ neurons surrounding η that connect to neuron η' in layer $l+1$. Convolving W and X produces output Z where element $Z_{c,j,k}$ is the value of the output neuron within channel c at row j and column k. The activation $A_{c,j,k}$ is a function of Z, $\phi(Z_{c,j,k})$. Summarizing the definition formally, the forward propagation step as a CNN is described based on (Goodfellow, Bengio, and Courville 2016) as:

$$Z_{c,j,k} = \sum_{c,m,n} X_{c',j'+m-1,k'+n-1} W_{\eta',\eta,m,n} \qquad (1)$$

$$A_{c,j,k} = \phi(Z_{c,j,k}) \qquad (2)$$

The propagation of information through the network can be considered a function of the post-activation output of a neuron, $A_{c,j,k}$ and the weights $W_{\eta',\eta,m,n}$.

We propose four measures of importance based on the weights and activations of each neuron (j,k) in layer l. Note that the weights and activations are a function of the matrix size $m \times n$ that surround each neuron, and therefore all of the importance measures of a neuron vary over indices $(j+\hat{m}, k+\hat{n})$ where $\hat{m} = (\frac{-m}{2}, \frac{m}{2})$ and $\hat{n} = (\frac{-n}{2}, \frac{n}{2})$.

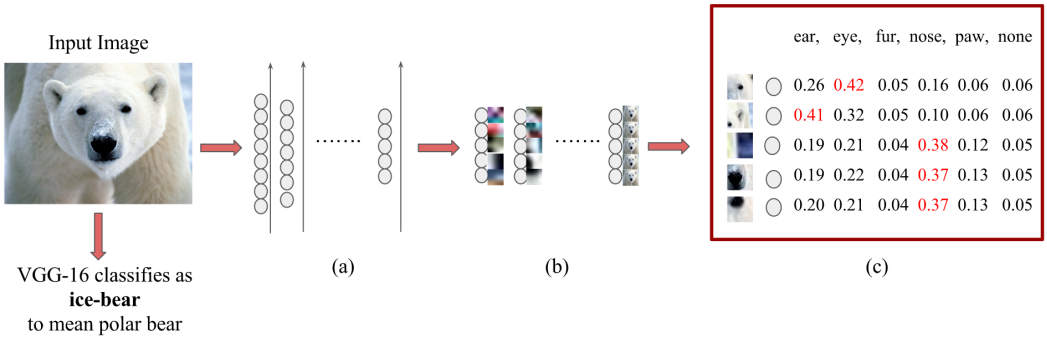


Figure 1: The APPLE algorithm a) ranks neurons based on importance, b) identifies image patches corresponding to the top 5 neurons from (a), and then c) labels the patches using a secondary classifier to determine important object features.

- **Activation Matrix Sum:** The sum of all values in the post-activation output $A_{c,j,k}$:

$$\sum_{\hat{m},\hat{n}} A_{c,j+\hat{m},k+\hat{n}} \tag{3}$$

- **Activation Matrix Variance:** The variance of all values in the post-activation output $A_{c,j,k}$:

$$\sigma^2_{\hat{m},\hat{n}} A_{c,j+\hat{m},k+\hat{n}} \tag{4}$$

- **Weight Matrix Sum:** The sum of all values in the weight matrix $W_{\eta',\eta,m,n}$:

$$\sum_{\hat{m},\hat{n}} W_{\eta',\eta,m,n}[\hat{m}][\hat{n}] \cdot \varphi, \text{ where } \varphi = \begin{cases} 1, & \text{if } A_{c,j,k} \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

- **Weight Matrix Variance:** The variance of all values in the weight matrix $W_{\eta',\eta,m,m}$:

$$\sigma^2_{\hat{m},\hat{n}} \; W_{\eta',\eta,m,n}[\hat{m}][\hat{n}] \cdot \varphi, \text{ where } \varphi = \begin{cases} 1, & \text{if } A_{c,j,k} \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

Once the importance measure is computed for each neuron, they can be sorted and ranked to find the top neurons for each layer. These top neurons are used in the next step: patch extraction.

Extraction of Patches corresponding to neuron

Given the ranked neurons, we are interested in identifying the image patches that they convolve (Figure 1b). The APPLE algorithm determines the image patches by deconvolving the network using a multi-layered Deconvolutional Network (deconvnet) as in Zeiler and Fergus ((Zeiler and Fergus 2014)). A deconvnet can be thought of as a convnet (CNN) model that uses the same components (filtering, pooling) but in reverse, so instead of mapping pixels to features does the opposite.

To examine a given convnet activation, APPLE sets all other activations in the layer to zero and pass the feature maps as input to the attached deconvnet layer. Then APPLE successively (i) unpools, (ii) rectifies and (iii) filters to reconstruct the activity in the layer beneath that gave rise to the chosen activation. This is then repeated until the input pixel space, referred to as a patch is reached.

Patch classifier

In order to label the object attributes in each high importance patch, we construct and train a secondary classifier. Given a set of object attributes as classifier labels (e.g., eyes, nose, ear, fur, and paws for polar bears), we crop image patches as training data for each of these labels. We also include a 'none' label which represents our background scene and parts of the object that may be difficult to distinguish.

Once the patch classifier is trained, it can be run on the important image patches in order to determine the attribute label. Because we use a multi-class classifier, APPLE ranks the likelihood of each label on each patch (Figure 1c) to determine the most likely label. Compared to (Zeiler and Fergus 2014) which requires manual probing to understand patches that activate neurons, our patch classifier automatically determines the *labels* that can be used to explain what important areas of the image the network focused on.

Putting it all together

Given an image (Figure 3a) and a CNN model, our APPLE algorithm forward propagates the image to determine its classification. If APPLE has an attribute labeler for that class, it then automatically analyzes the CNN to find the top N most important neurons and constructs a list of image patch regions to label. Figure 3b shows the 15 image patches selected by the Activation Matrix Sum measure (top 5 neurons from the three layers - layers 5-7 - of interest) for polar bears. APPLE runs the patch classifier on the image patches to determine the most likely attribute labels (Figure 4a). Labeled patches are then further sorted based on the maximum confidence. The important patches are a visual representation of the explanation of how the network determined the image's classification. The labels represent a semantic representation of the same explanation without requiring humans to interpret the image.

Experiments

In order to demonstrate the ability of our APPLE algorithm to find important neurons and corresponding image patches and then automatically label them, we evaluated its use on two different datasets and three different object recognition tasks. In particular, we tested APPLE's ability to find the same 5 attributes of polar bears and dogs - eyes, ears, fur, nose,

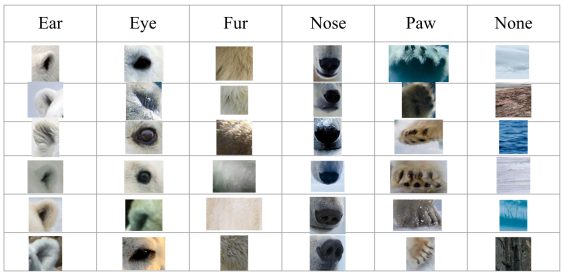


Figure 2: Sample training data used to train the Patch classifier for the polar bear class

and paw - as well as its ability to find attributes of people - head, torso, hand, leg, and foot. We consider animal images particularly challenging as the background of the images often contains similar colors as the animals themselves, and because features that a person would consider to be strong distinguishers - black eyes and black nose - could easily be confused with each other and with rocks present in the environment.

In this section, we describe our experimental setup using the VGG-16 classifier, our own trained patch classifer and the results of our experiments to evaluate the ability of our importance metrics to find regions of the object and the accuracy of our patch labels.

CNN Classifier

For the purpose of our experiment we chose the VGG-16 architecture (Simonyan and Zisserman 2014) because it is a deep network with enough intermediate layers to gradually decrease the component granularity (to study increasing feature composition and its affect on final confidence). The VGG-16 architecture consists of 13 convolution layers followed by 3 dense layers, with max pooling after the 2^{nd}, 4^{th}, 7^{th}, 10^{th} and 13^{th} layer. Despite the number of layers available for us to analyze, we only evaluated the image patches for neurons between layers 5 and 7 for two reasons. First, the image patches in the first few layers were too small to train a patch classifier for. Furthermore, results from Ranzato et al.(Ranzato et al. 2006) already show that the first layers learn stroke-detectors and Gabor-like filters. Additionally, image patches corresponding to the last few layers include a majority of the image and contain too many of the object attributes to accurately label just one.

In order to evaluate APPLE on different image classes and its importance functions on different weights within the VGG-16 architecture, we trained the VGG-16 architecture in two ways. To test our approach on polar bears and dogs, we used pre-trained weights for ImageNet(Russakovsky et al. 2015) as provided in (Chollet and others 2015). To test our approach on people, we trained VGG-16 on the INRIA dataset (Dalal and Triggs 2005).

Patch Classifier

In order to train our patch classifier, we listed distinguishing attributes of polar bears and great pyranees or dogs - ears, eyes, nose, fur, and paws, and that of human beings - head, torso, hand, leg, and foot and manually cropped those important features from images in the Imagenet dataset (Russakovsky et al. 2015) and INRIA dataset (Dalal and Triggs 2005) respectively. For the 'none' class, we collected background patches from the same images. We had around 80 images representing each attribute, totalling the training data size to be 480 for each class (polar bear, dog, person). Each training image was 128x128 pixels. Figure 2 contains sample patches from our classifier dataset for polar bear.

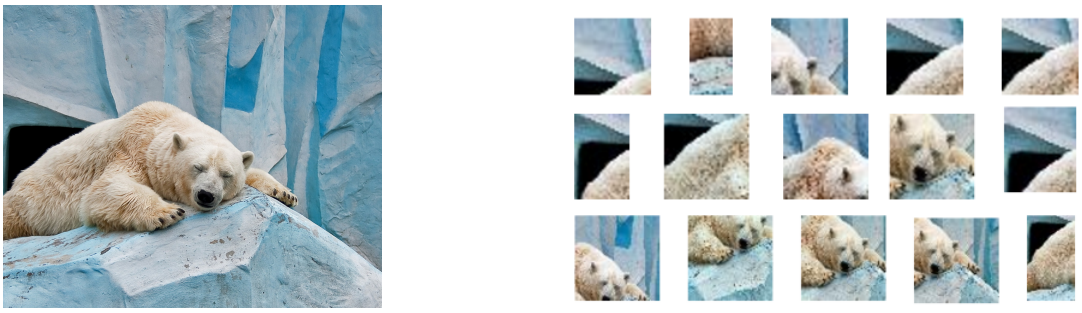
We used a multi-class Support Vector Classifier (AdaBoost-SVM (Li, Wang, and Sung 2008)) with c=0.771, and γ = 0.096, with an rbf kernel, determined by running K-fold cross validation on the patch data. The classifier obtained an average test set accuracy of 80% when the data was split into 80% training- 20% test. SVMs were chosen for the task because of their ability to handle high dimensional complex data: an RBF kernel was used as described in (Li, Wang, and Sung 2008). Our patch classifiers demonstrate the applicability of APPLE to a wide variety of image classes, and we note that we did not spend a large amount of time to acheive our attribute label results. It is possible to train a more accurate classifier on a larger dataset for even better results.

Evaluation Setup

Our experiments were conducted on 360 images, with 120 images per each class (polar bears, great pyrenees and persons). We selected the test images by searching for specific class on the Internet. Because our goal is to demonstrate the ability of APPLE to find important patches in any image and because the available labeled datasets for these classes is small, we chose to manually search for images of the classes in a variety of environments, poses, and conditions. The selection process involved criteria such as pose, number of classes visible in the image, lighting, and image resolution. By varying the different settings we could determine how different conditions influence the results of both our high importance neuron ranking method, as well as our patch classifier. We evaluate APPLE based on two measures of success: object localization and patch label precision.

Object Localization Measure: We first evaluated APPLE's ability to find image patches that were localized on the object of interest that the CNN is classifying (i.e., polar bears, dogs, and people in our experiments). *Object localization* is measured as the ratio of number of patches containing pixels of the object (polar bear, dog, person) to the total number of patches. We analyzed three different types of objects to understand whether our importance functions are successful for different trained CNN classes, and compare against CAM's ability to find the important attributes of the object as well.

To do this, we outlined APPLE's important patches using the *Activation matrix sum* measure on top of the original image. Figure 5 illustrates the localization abilities of our APPLE algorithm (red boxes indicate patches belonging to layers 5 - 7). We manually evaluated each image patch to determine whether it contains the object. For example, Figure 3b shows 15 high importance patches picked by our heuristic. Each of these patches contains pixels belonging to polar bear, and hence they are all deemed as correct patches. In total,



(a) (b)

Figure 3: (a): Input image. (b): High Importance Neuron Patches selected by APPLE algorithm.

Classification : ice_bear

Layer	Neuron	Patches	Predictions: ear, eye, fur, nose, paw, none
Layer : 7	Neuron : 2		0.56 0.27 0.06 0.02 0.04 0.05
Layer : 7	Neuron : 3		0.20 0.51 0.05 0.08 0.06 0.10
Layer : 7	Neuron : 4		0.08 0.22 0.50 0.03 0.10 0.07
Layer : 6	Neuron : 4		0.08 0.25 0.50 0.02 0.10 0.05
Layer : 6	Neuron : 1		0.08 0.24 0.49 0.03 0.10 0.06

(a)

Classification : Person
Action : Stop

Layer	Neuron	Patches	Predictions: head, torso, hand, leg, foot, None
Layer : 7	Neuron : 4		0.77 0.09 0.06 0.01 0.03 0.04
Layer : 6	Neuron : 3		0.12 0.74 0.04 0.03 0.04 0.03
Layer : 6	Neuron : 4		0.73 0.08 0.04 0.04 0.08 0.03
Layer : 5	Neuron : 4		0.70 0.12 0.06 0.03 0.04 0.05
Layer : 6	Neuron : 2		0.12 0.23 0.07 0.49 0.04 0.05

(b)

Figure 4: APPLE sorts the labeled patches by confidence to present to a human in order to explain the CNN's image classification. Two example images are shown with their important patches selected using *Activation matrix sum*.

15 important patches (top 5 patches across 3 layers) were evaluated as to whether they contained portions of the object (using our input image as reference).

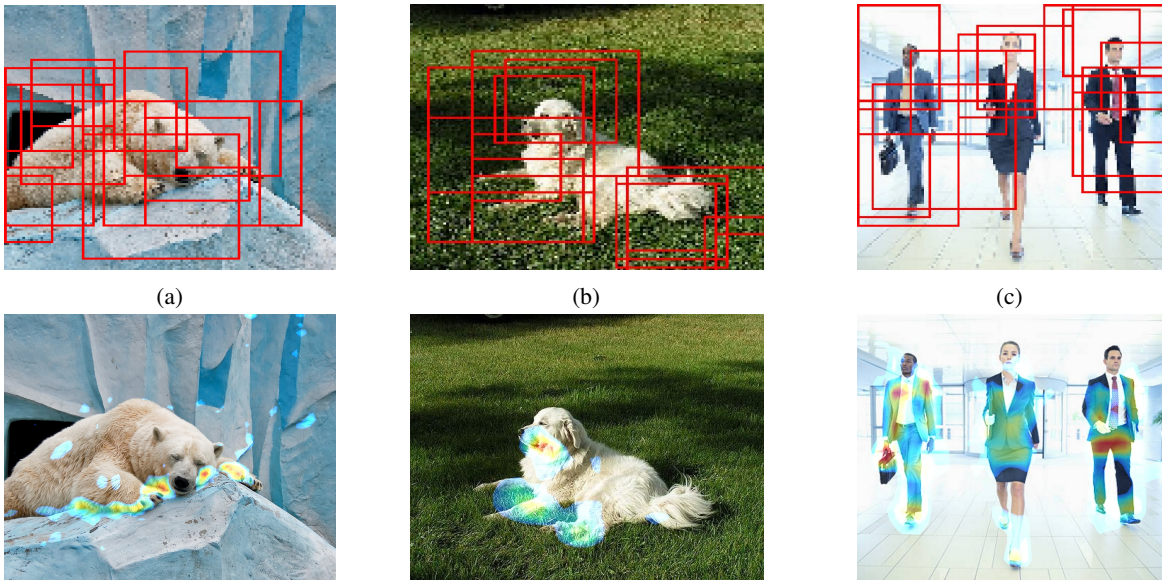
Patch Label Precision Measure: We evaluated patch classifier's precision at labeling the image patches. *Patch label precision* is measured as the ratio of correctly classified patches to the total number of patches. A patch classification is considered correct if the top label output by the patch classifier matches our manually labeled ground-truth.

Results

Table 1 shows the precision of each of our evaluation measures on all 360 of the test images. We first note that our results were very similar for all four proposed importance metrics indicating that they are all successful and that any suggested measure could be used for the purpose of extracting neurons and thereby important patches. We compare the four-proposed metrics to a fifth metric which randomly picks 15 neurons (5 neurons across 3 layers). For clarity, we will report Activation Matrix Sum results which correspond to the image patches in Figure 5.

We first evaluated the abilty of our algorithm to localize objects in the image. As seen from Table 1, the four proposed importance metrics excel at object localization compared to the metric which randomly picks neurons. As seen from Figure 5, most (93%) of the red boxes are centered around the object, indicating that the important patches that APPLE finds are contributing highly to the classification. If we take into account the entire outline of our patches, they encompass the entire object, always more of the object than CAM does (Figure 5d,e,f). This is significant because CAM requires modifications to the CNN architectures, whereas our approach works without any network modifications. Additionally, APPLE labels those important patches with object attributes, further automating the process. Because the patches are accurate for each of the three object classes and for different weights within the VGG-16 architecture, we conclude that APPLE's importance functions successfully and accurately find important patches across a variety of objects (polar bear, dog, and person) and datasets (ImageNet and INRIA).

Our next set of experiments focused on the accuracy of our patch labels. Table 1 shows the patch label precision of the AdaBoost SVM on the important image patches for all 360 images. Our results show that we are able to label the image patches much more accurately than random guessing. However, the top-1 precision is less than 0.7 likely due to the choice of patch classifier and the quality of training data provided. For example, we only obtained 80 patches per label and the low-resolution of the images led to high confusion between the components (eyes and nose as well



(a) (b) (c)

(d) (e) (f)

Figure 5: APPLE and CAM each find important regions of images. In APPLE (a,b,c), red boxes indicate layers 5 - 7, and they encompass the entire object. On the CAM images (d,e,f), the heatmap visualizes its important pixels.

Table 1: Evaluation of important patches averaged over all image classes.

Evaluation Measure	Object Localization	Label Precision (Top-1)	Label Precision (Top-2)
Weight Sum	0.91	0.610	0.810
Weight Variance	0.92	0.632	0.784
Activation Sum	0.93	0.697	0.832
Activation Variance	0.91	0.641	0.871
Random	0.54	0.614	0.782

as fur and background (None)). We evaluated whether the correct label is in the top-2 predicted labels and found that the precision jumps to as high as 0.87. This result indicates that our trained patch labeler has high confusion with pairs of classes but more training data would help improve the precision further. It should be noted that results for the metric which randomly picks neurons are comparable to the four proposed metrics, which is unsurprising as 'none' is one of the classes considered while training the patch classifier.

Despite the challenges in building object attribute classifiers, our results demonstrate the patches identified as important by APPLE can be labeled accurately to help a human understand the CNN information propagation. This is the case even in the challenging tasks of labeling attributes of animals who's shapes and colors match the image backgrounds. We found that the results are the same across each of our three object classes.

Conclusion

APPLE helps people gain a deeper understanding of what deep neural networks learn at the intermediate layers, and why they make the conclusions that they do. While prior work has focused on identifying important pixels that contribute to classification, little work has explored the impact of information propagation through the network.In this work, we contribute our algorithm, APPLE, to analyze the neuron-level information propagation and to facilitate the understanding of regions of interest. We contributed four different measures of neuron importance, such as sum and variance across the activation matrix and weight matrix of neurons.

We demonstrated that in image classification tasks, our algorithm is able to use the measures to identify neurons within the CNN that focus on important attributes of the recognized object (i.e., body parts of animals). In particular, we demonstrated that all of APPLE's importance measures find regions of the images that contain the object of interest. We then used a patch classifier to label the attributes of the object, although it did confuse similar looking features of the bear. Currently, our approach uses one patch classifier for each class under consideration. At first glance, it might occur that our approach doesn't scale easily with increased number of classes, however it would be possible to automatically generate the list of features for each object (e.g., using web search) as well as crop attributes from images (e.g., using crowd-sourcing). Manual evaluation can also be automated by blocking out important patches and evaluating if the importance correlates with the change in class score. Future work could include constructing a universal patch dataset, improving patch classifier performance using a better classifier, more training data for a patch classifier and including context while classifying the patch.

References

[Chollet and others 2015] Chollet, F., et al. 2015. Keras. https://github.com/fchollet/keras.

[Dalal and Triggs 2005] Dalal, N., and Triggs, B. 2005. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, 886–893. IEEE.

[Erhan et al. 2009] Erhan, D.; Bengio, Y.; Courville, A.; and Vincent, P. 2009. Visualizing higher-layer features of a deep network.

[Girshick 2015] Girshick, R. 2015. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, 1440–1448.

[Goodfellow, Bengio, and Courville 2016] Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[He et al. 2016] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

[Hendricks et al. 2016] Hendricks, L. A.; Akata, Z.; Rohrbach, M.; Donahue, J.; Schiele, B.; and Darrell, T. 2016. *Generating Visual Explanations*. Cham: Springer International Publishing. 3–19.

[Krizhevsky, Sutskever, and Hinton 2012] Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

[Li, Wang, and Sung 2008] Li, X.; Wang, L.; and Sung, E. 2008. Adaboost with svm-based component classifiers. *Engineering Applications of Artificial Intelligence* 21(5):785–795.

[Mahendran and Vedaldi 2015] Mahendran, A., and Vedaldi, A. 2015. Understanding deep image representations by inverting them. In *2015 IEEE conference on computer vision and pattern recognition (CVPR)*, 5188–5196. IEEE.

[Oquab et al. 2015] Oquab, M.; Bottou, L.; Laptev, I.; and Sivic, J. 2015. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 685–694.

[Park et al. 2016] Park, D. H.; Hendricks, L. A.; Akata, Z.; Schiele, B.; Darrell, T.; and Rohrbach, M. 2016. Attentive explanations: Justifying decisions and pointing to the evidence. *CoRR* abs/1612.04757.

[Pinheiro and Collobert 2015] Pinheiro, P. O., and Collobert, R. 2015. From image-level to pixel-level labeling with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1713–1721.

[Ranzato et al. 2006] Ranzato, M.; Poultney, C.; Chopra, S.; and LeCun, Y. 2006. Efficient learning of sparse representations with an energy-based model. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, 1137–1144. Cambridge, MA, USA: MIT Press.

[Ribeiro, Singh, and Guestrin 2016] Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "why should I trust you?": Explaining the predictions of any classifier. *CoRR* abs/1602.04938.

[Russakovsky et al. 2015] Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115(3):211–252.

[Selvaraju et al. 2016] Selvaraju, R. R.; Das, A.; Vedantam, R.; Cogswell, M.; Parikh, D.; and Batra, D. 2016. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR* abs/1610.02391.

[Simonyan and Zisserman 2014] Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR* abs/1409.1556.

[Springenberg et al. 2014] Springenberg, J. T.; Dosovitskiy, A.; Brox, T.; and Riedmiller, M. 2014. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

[Szegedy et al. 2015] Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–9.

[Vedantam et al. 2017] Vedantam, R.; Bengio, S.; Murphy, K.; Parikh, D.; and Chechik, G. 2017. Context-aware captions from context-agnostic supervision. *CoRR* abs/1701.02870.

[Wulfmeier, Wang, and Posner 2016] Wulfmeier, M.; Wang, D. Z.; and Posner, I. 2016. Watch this: Scalable cost-function learning for path planning in urban environments. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, 2089–2095. IEEE.

[Yang et al. 2016] Yang, S.; Song, Y.; Kaess, M.; and Scherer, S. 2016. Pop-up slam: Semantic monocular plane slam for low-texture environments. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, 1222–1229. IEEE.

[Zeiler and Fergus 2014] Zeiler, M. D., and Fergus, R. 2014. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, 818–833. Springer.

[Zeiler, Taylor, and Fergus 2011] Zeiler, M. D.; Taylor, G. W.; and Fergus, R. 2011. Adaptive deconvolutional networks for mid and high level feature learning. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2018–2025. IEEE.

[Zhang et al. 2015] Zhang, F.; Leitner, J.; Milford, M.; Upcroft, B.; and Corke, P. 2015. Towards vision-based deep reinforcement learning for robotic motion control. *arXiv preprint arXiv:1511.03791*.

[Zhou et al. 2015] Zhou, B.; Khosla, A.; Lapedriza, A.; Oliva, A.; and Torralba, A. 2015. Learning deep features for discriminative localization. *arXiv preprint arXiv:1512.04150*.

[Zhu et al. 2016] Zhu, Y.; Mottaghi, R.; Kolve, E.; Lim, J. J.; Gupta, A.; Fei-Fei, L.; and Farhadi, A. 2016. Target-driven visual navigation in indoor scenes using deep reinforcement learning. *arXiv preprint arXiv:1609.05143*.