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 (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) for CNNs 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 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
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 \( \eta \) 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 \( W_{\eta',\eta,m,n} \) weights the activations of neuron \( \eta \) in layer \( l \) and the rectangle of \( m \times n \) neurons surrounding \( \eta \) that connect to neuron \( \eta' \) 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:

\[
Z_{c,j,k} = \sum_{c',m,n} X_{c',j'+m-1,k'+n-1} W_{\eta',\eta,m,n}
\]

(1)

\[
A_{c,j,k} = \phi(Z_{c,j,k})
\]

(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 \((j+m,k+n)\) where \(m = (\frac{-m}{2}, \frac{m}{2})\) and \(n = (\frac{-n}{2}, \frac{n}{2})\).
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 \( A_{c,j,k} \):
  \[
  \sum_{m,n} A_{c,j+m,k+n}
  \]
  \( (3) \)

- **Activation Matrix Variance**: The variance of all values in the post-activation output \( A_{c,j,k} \):
  \[
  \sigma^2_{m,n} A_{c,j+m,k+n}
  \]
  \( (4) \)

- **Weight Matrix Sum**: The sum of all values in the weight matrix \( W_{\eta',\eta,m,n} \):
  \[
  \sum_{m,n} W_{\eta',\eta,m,n}[\hat{m}][\hat{n}] \cdot \varphi, \text{ where } \varphi = \begin{cases} 1, & \text{if } A_{c,j,k} \geq 0 \\ 0, & \text{otherwise} \end{cases}
  \]
  \( (5) \)

- **Weight Matrix Variance**: The variance of all values in the weight matrix \( W_{\eta',\eta,m,n} \):
  \[
  \sigma^2_{m,n} W_{\eta',\eta,m,n}[\hat{m}][\hat{n}] \cdot \varphi, \text{ where } \varphi = \begin{cases} 1, & \text{if } A_{c,j,k} \geq 0 \\ 0, & \text{otherwise} \end{cases}
  \]
  \( (6) \)

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 the 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 3c). 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,
we used pre-trained weights for ImageNet (Russakovsky et al. 2015) on VGG-16 architecture, we trained the VGG-16 architecture consisting of 13 convolution layers following attributes of polar bears and great pyrenees or dogs - Patch Classifier dataset (Dalal and Triggs 2005). 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).

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 2nd, 4th, 7th, 10th and 13th 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).

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 [3] 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,
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 ability 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 3, 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
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.

<table>
<thead>
<tr>
<th>Evaluation Measure</th>
<th>Object Localization</th>
<th>Label Precision (Top-1)</th>
<th>Label Precision (Top-2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight Sum</td>
<td>0.91</td>
<td>0.610</td>
<td>0.810</td>
</tr>
<tr>
<td>Weight Variance</td>
<td>0.92</td>
<td>0.632</td>
<td>0.784</td>
</tr>
<tr>
<td>Activation Sum</td>
<td>0.93</td>
<td>0.697</td>
<td>0.832</td>
</tr>
<tr>
<td>Activation Variance</td>
<td>0.91</td>
<td>0.641</td>
<td>0.871</td>
</tr>
<tr>
<td>Random</td>
<td>0.54</td>
<td>0.614</td>
<td>0.782</td>
</tr>
</tbody>
</table>

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


