Classifier Labels as Language Grounding for Explanations

Abstract

Advances in state-of-the-art techniques including convolutional neural networks (CNNs) have led to improved perception in autonomous robots. However, these new techniques make the robot’s decision-making process obscure even for the experts. Our goal is to automatically generate natural language explanations for robot’s state and decision-making algorithms in order to help people understand how they made their decisions. Generating natural language explanations is particularly challenging for perception and other high-dimension classification tasks because 1) we lack a mapping from features to language and 2) there are a large number of features which could be explained. We present a novel approach to generating explanations that first find important features that most affect the classification output and then utilize a secondary detector to label (i.e., generate natural language groundings) only those features. We demonstrate our explanation algorithm’s ability to explain our service robot’s building floor identification classifier.

1 Introduction

Service robots can autonomously generate and execute plans to perform tasks for humans while appropriately handling the uncertainty of their surroundings. In order to achieve these state-of-the-art results, many service robots have adopted deep learning techniques, especially for perception using Convolutional Neural Networks (CNNs). While these algorithms help robots achieve better performance in real-world settings, they are challenging to understand by non-experts.

Because non-experts in the environment may need information about the robot’s perception, state prediction, planning, and execution, our goal is to generate explanations of the robot’s algorithm’s decision. Prior work in explaining robot behavior have focused on either explaining a policy (e.g., Hayes and Shah [2017]) or narrating a particular sequence of execution (e.g.,
While these explanations can potentially help people understand a robot’s actions, they do not help a person determine how the robot determined its current state before performing that action (e.g., using a classifier). One popular approach to generating model-agnostic explanations for a classifier is learning an interpretable model based on the predictions of the original model Ribeiro et al. [2016]. This technique has been very successful but it may still be hard for non-experts to understand the new local classifier. Other approaches explain perception classifiers visually (e.g., Selvaraju et al. [2016]). While a visual representation can be useful, there are times when a robot cannot display an image. Additionally, a visual technique is challenging to extend beyond visual domain.

In this work, we focus on automatically generating natural language explanations that summarize the meaning of the robot’s feature space (e.g., labeling the pixels that a CNN uses to classify the robot’s location). While natural language can be more understandable for a person, it is also challenging to generate because there often is not a grounding that easily translates a classifier’s features to language and it is unclear which features should be explained. Hendricks et al. [2016] uses a pre-defined natural language description of the classes to train a network to produce explanations for classification of different species of birds. However, this process is not practical when there are no descriptions for the classes or when there may be many classes to explain.

Our goal is to create natural language explanations without hand-labeling features. We present a three-part algorithm for explaining a CNN classification that reduces the number of features to explain and does not require a predefined mapping from features to natural language. In the first step, our algorithm makes use of deep visualization techniques to find important pixels in the image. Then, the algorithm makes use of a secondary pre-trained off-the-shelf classifier to label only the important regions of the image. The labels serve as our natural language descriptions of the important features. Finally, the algorithm uses templates to stitch together the labels and their locations within the image for a final explanation.

We demonstrate our explanation algorithm on our robot’s floor identification CNN classifier. After our robot captures an image in our building and classifies the floor it is on, it then uses our algorithm to identify important features in the image that help it determine its floor and produces a natural language explanation. We conclude with practical implementation results that would allow our algorithm to be applied to other vision-based domains as well as non-visual domains where important features can be labeled.

2 Related Work

Finding the important features or performing feature reduction Boger and Guterman [1997]; Blum and Langley [1997] in the context of machine learning is a well-researched area. But the work does not focus on generating an explanation or making those features interpretable.

A popular approach to generating model-agnostic explanations for a classifier is learning an interpretable model based on the predictions of the original model Ribeiro et al. [2016]; Baehrens et al. [2010]; Sanchez et al. [2015]. In Ribeiro et al. the authors explain the classification of any model by learning a separate sparse linear model that locally approximates the original model to explain its decisions. While these models are successful for some domains such as text classification where the explanations are already language-based, it is unclear how to explain these local classifiers to non-experts.

Image-specific visualization techniques seek to find what features CNNs have learned to look for in each image Zeiler and Fergus [2014]; Selvaraju et al. [2016]; Zhang et al. [2016]; K. et al. [2014]; Zintgraf et al. [2016]. Essentially, they serve as functions that find important regions in
an image that most affect the classification and display a heat map representing the relative importance of each pixel over the original image. A variety of different methods have been proposed for determining which features are important, leading to different explanations of the classification predictions of CNNs. Zeiler and Fergus uses an occluding patch to systematically occlude parts of the input image, and use the classifier’s confidence on these images to generate a heat map. The idea behind the approach is if a key feature in an image gets occluded, then the classifier’s confidence will fall upon the occlusion. Gradient visualization technique K. et al. [2014], uses gradients of the classifier’s score with respect to the pixels in the input image as the heat map. The intuition is that the probability scores are more sensitive to the change in values of the important features than others.

While these visual explanations of image classification have been successful, several challenges prevent these techniques from being applied to other domains. First, there are times when either it is not feasible to display a heat map on a robot for an explanation or when a heat map cannot explain the classification. Second, there are many other types of algorithms for non-visual features including neural networks that could also be explained by finding important features. However, the visual heat map does not apply to those algorithms. In this work, we focus on producing natural language explanations of high-dimension data including images.

Hendricks et al. focuses on generating textual explanations for fine-grained classification of 200 bird species using deep network from their images. The algorithm explains ResNet He et al. features extracted from the entire image, and are conditioned both on the image and the class predictions. The explanation generation algorithm is trained on manually-generated bird descriptions. While such an approach might be possible for cases where the vocabulary for explanations are easy to manually define or pre-existing in some other form, it is very tedious to generate natural language for new domains.

Explanations of robot states and actions have similar challenges in mapping features or states to natural language. For explaining robot actions, Hayes and Shah aims to synthesize policy descriptions and respond to queries about why robots do or do not perform particular actions in particular states. They learn a simplified domain model of the environment from demonstrations then use statistics of the planner over the data to create a behavioral model. However, their natural language explanations require manually defined predicates and corresponding language. Rosenthal et al. creates narrations of robot paths and also requires predefined dictionaries mapping states to language. In this work, we utilize existing off-the-shelf classifiers to label important features with natural language, eliminating the need for manual labeling.

3 Explanation Problem and Approach

In this work, we assume that a pre-trained classifier $C$ and an observation (e.g., an image) $I$ with possibly many features are given. Our goal is to provide a good natural language explanation of how $C$ classified $I$ as a given class $y \in Y$. This problem is challenging for two reasons. First, unlike a visual explanation that can produce a heat map overall features, linguistically explaining each of many features is infeasible. Thus, we must determine what are the most important features to explain. The other challenge is that there may not be a mapping from those important features to natural language to use in our explanation. In order to overcome this challenge, we make use of off-the-shelf pre-trained classifiers that can label parts of the feature space. In the case of image-based explanations, we use a multi-object classifier that labels objects within images. We next describe each step in our explanation algorithm:

1. Identify important features for the classification
2. Label the important features, and
3. Generate an explanation using feature labels as natural language groundings.

We describe our algorithm with respect to explaining the classification of images for a mobile service robot application, though the algorithms described are not image-specific.

Step 1: Identifying Important Regions

We assume that a classifier $C$, outputs $p(I = y|w)$, the probability of an image $I \in [0, 1]^{a \times N}$ with $a$ channels (i.e., 3 for R,G,B) and $N$ pixels having classification $y \in Y$ given the trained parameters $w$. For clarity, we will refer to the $ith$ pixel in the image as $I[i]$. Our first step is to find the important pixels that contribute most to the classification of $I$ as $y$.

Importance Functions

Given $C$ and $I$, an importance function $\text{importance}(I, C)$, ranks the pixels in the image based on their impact on the classification of the image as $y$. The importance functions takes as input $I$ and $C$, and outputs a heat map $H \in [0, 1]^N$ that contains a measure of relevance of each pixel $I[i]$ to the class $y$. The heat map will have higher values for those pixels, which are considered important and lower values otherwise. A variety of importance functions, each with their own heat map, have been proposed for explaining the classification predictions of CNNs. For example, Figure 1(b) shows the visualizes the heat map for image in Figure 1(a). In this work we use the Gradient Visualization Technique K. et al. [2014] though there are others including Zeiler and Fergus [2014] and Zhang et al. [2016].

3.0.1 Gradient Visualization Technique

For the gradient visualization technique, $H$ represents the magnitude $m$ of the derivative of the classification confidence with respect to the image. The magnitude of $ith$ pixel $m_i$ represents
Figure 2: Example for generating $E_M$. Our algorithm takes an original image (a) and creates a binary mask of the important features (b). It also uses a secondary classifier to label regions of the image (c) and discretizes the image into 9 grid cells (d) to use in the explanation.

the sensitivity of the network’s prediction to the change in that pixel’s value and is equal to the derivative of the classification probability $p(I = y|w)$ with respect to $I[i]$. We expect the classifier accuracy to be more sensitive to the change in values of the important features than others. Note that since the gradients are pixel-wise importance values for the image, the heat map is generally of high entropy and thus lacks continuous important image regions.

**Importance Mask**

While the heat map is useful for visualization, we propose the use of a binary mask – for example the red mask in Figure 1(c) – to signify whether a feature (pixel) is included in the important region or not. To find the important pixel features from a heat map, many different segmentation techniques have been proposed – region-based Chen and Chen [2009], threshold-based Karthikeyan et al. [2012] or model-based Lehmann [2011]. Although model-based segmentation tends to perform better, for simplicity, we use a simple threshold-based segmentation to find the important regions in this work. A binary mask $M \in \{0, 1\}$ is created such that each pixel $i$ takes value:

$$M[i] = \begin{cases} 1 & \text{if important} \\ 0 & \text{otherwise.} \end{cases}$$

We find the important features for the classification by segmenting the heat map using thresholding to compute a binary importance mask $M$. We generate the mask by segmenting top $\rho\%$ of the pixels from $H$. For example, Figure 1(c) is the importance mask obtained by thresholding the heat map shown in Figure 1(b) with $\rho = 50\%$ of pixels.

**Step 2: Labeling Important Regions using a Secondary Classifier**

Using the mask $M$ of important features in $I$ (e.g., Figures 1(a) and 1(c) and also Figures 2(a) and (b)), our algorithm then generates natural language groundings for the important regions in the image masked by $M$, by using a secondary classifier. In our application, we use an image-based multi-object detector, but any secondary classifier over features in $I$ would work.
Extracting Explainable Features of an Image

The detector identifies a set of objects as potential explainable features $E \{o_0, ... o_l\}$ in the image and their corresponding bounding boxes $\{(xa_0, xb_0, ya_0, yb_0), ... (xa_l, xb_l, ya_l, yb_l)\}$ as shown in Figure 2(c). The bounding boxes represent the features that pertain to the object. To generate explainable features for the important regions in the image $E_M$, we remove objects in $E$ with less than a threshold ratio $t$, of their bounding box overlapping with the importance mask $M$. We also filter unrelated features in $E$ by removing tuples containing objects present in a rejection corpus $R$. The filtering is done to remove erroneous detections from the multi-object detector. The rejection corpus is constructed using object names that would not be found in the robot’s environment.

To summarize, our algorithm automatically generates a list of explainable features $E_M$ for a particular image $I$ and importance mask $M$ by applying filters:

$$E_b = \{ e_i | \frac{\text{sum}(M[xa_i : xb_i, ya_i : yb_i])}{(xa_i - xb_i) * (ya_i - yb_i)} > t, \ \forall e_i \in E\}$$

$$E_M = E_b \{ e_i | o_i \in R, \ \forall e_i \in E\}$$

where $M[x1:x2, y1:y2]$ is a submatrix of $M$ with $(y2-y1)$ rows and $(x2-x1)$ columns, it containing elements from $M$ in $x1$ to $x2$ columns and $y1$ to $y2$ rows.

To simplify references to the explainable features (i.e., in our case references are locations in the image), we discretize the image into $g$ grid cells (e.g., Figure 2d with 9 cells), and use the grid index $gId$ of the detected object’s center as a proxy to their location in the image rather than the xy location. The discretization can be utilized to non-vision data as well. For example, rather than using full timestamps in temporal data, it is possible to discretize events by morning, afternoon, evening, and night. In the end, our explainable features are labeled objects that appear in the most important regions and their relative location within an image.

Extracting Explainable Attributes of a Class

Though we have a list of many explainable features for an image, we would like to identify the explainable features that occur commonly in the class and that could be described as an attribute of the class rather than anomalous objects. Given a training set of images prior to providing any explanations, our algorithm computes the explainable features as above for each training image and then computes the union (probabilistic combination can also be used) of $E_M$s as the explainable attributes for class $y \in Y$ as $E_y$.

Note that we use a small training set of 5 so the union is still a small number of explainable class attributes. Other methods such as the intersection or a threshold may be used to reduce the size of the class attributes if it is large. We assume that the classifier, importance function, and multi-object detector are good enough to produce unique explainable features for each class, i.e., all the classes are distinguishable using only their explainable feature $E_y$. $E_y$ represents the dictionary of attributes defining the class.

Step 3: Generating Explanations

Finally, our algorithm computes the intersection $E_{M,y}$ of features in $E_M$ (representing the important regions of the image) and $E_y$ (representing the attributes of class $y$) to find the explainable features that the two have in common. There may be features in the image that are not attributes of the class or vice versa that should not be explained. For example, if a plant
is detected in a hallway but that plant is not commonly found there, it should not be reported
as a reason that the image was classified as the hallway.

We use language grounding and template-based natural language generation to convert
$E_{M,y}$ to natural language explanations. To ground the grid locations, we related grid location
to a corresponding spatial location. For example, $gLd= (0,2)$ is ‘top right’ and $gLd= (1,1)$ is
‘center.’ For grounding explainable features, the multi-object detector’s object labels are a
proxy for naming the objects in the image. We combine all the natural language groundings
from $E_{M,y}$ with a simple natural language template to create an explanation. Our algorithm
demonstration section includes the template used in our work.

Rather than explaining all pixels or specific pixels, the algorithm focuses on important
pixels and finds natural labels for those pixels using a secondary classifier. It also focuses on
explaining the important attributes of the class rather than all objects that could be labeled
in the important regions. This explanation is a summary of the objects or explainable features
that are representative of the class.

4 Algorithm Demonstration

When a robot navigates across many floors of a building or multiple buildings, one major
challenge that it has is localizing itself to determine which floor it is currently on. We collected
a Building-Floor dataset and trained a CNN scene recognition classifier for determining which
floor our robot is on. We then tested our algorithm’s ability to explain why it classified our
robot’s images as particular floors of the building. We used an off-the-shelf multi-object detector
to label explainable features like chairs, plants, and tables in our hallways that distinguish one
floor from another. Our results show that the robot’s explanations are accurate yet concise
given the number of features in the images.

4.1 Building-Floor Dataset

We collected a Building-Floor dataset in one of our buildings. Each image contains the scene
just outside the elevator from six different floors of the building, ref Figure 3. The goal of
the classifier $C$, trained on this dataset is to find which floor the image is taken from. Since
the robot cannot change floors other than by taking the elevator, the elevators are the only
locations where we need to classify the robot’s location.

For each of the floors in the building, five images were taken at a particular location that
our robot stops at after exiting the elevator. To simplify the analysis, all the images were taken
at the same time of the day, and the effects of people moving around in the building are not
considered. The training data consists of three images, and the remaining two images form
the verification dataset. As the scene does not change a lot in theory, one image is enough to
be able to train and obtain good performance. In practice, this is not true so we use three
images to avoid overfitting and capture some variation in the scene. Testing data collected
separately consists of five images from each of the floor taken at similar daytime and settings
as the training dataset. The newly collected testing dataset was obtained while our robot was
performing multi-floor navigation tasks.

4.2 Network Architecture

The deep learning network for our floor identification classifier $C$, consists of nine layers following
AlexNet Krizhevsky et al. [2012] in a modified Siamese architecture as proposed in Sun et al.;
Figure 3: Sample of images from the Floor detection dataset. Each image belongs to a different floor.

Zheng et al., which combined the identification—Softmax, and the verification loss—Contrastive, for better performance, refer Figure 4. The approach we have taken to classify each of the floors using a DNN based network is currently a popular method and can be easily scaled to accommodate more classes. Our main reason for combining identification and verification loss with a pre-trained network is to reduce overfitting which could happen when the complexity of network is higher than the data. During training, the first seven layers of our network were initialized from Places205-AlexNet which was trained in the Places205-Standard dataset and provided by the authors Zhou et al. [2014]. The remaining two layers were trained from scratch. During training, the contrastive loss was utilized in the eighth layer which is a dense layer of 1000 units, while the softmax loss was employed in the ninth layer.

The classifier $C$ can classify all the images in the testing dataset correctly. We believe the high performance of our network is because the training and testing images were taken in similar setting and that the inter-class variation is quite high in the dataset. Another important reason for the good performance of our floor identification module is the use of the Siamese architecture during training. Siamese architecture by its nature increases the network’s ability to maximizes inter-class variability while minimizing intra-class variability.

4.3 Demonstration Setup

We implemented our explanation generation algorithm to describe the deep learning floor classifier. Given an image that was classified, the algorithm first finds the important regions of the image. Because the heat map generated from the gradient visualization technique is generally of high entropy and lacks continuous important image regions, we dilated the heat map twice with a 3x3 kernel. Dilating smoothens the heat map and improves the continuity of important regions. The algorithm then generates the importance mask $M$, by segmenting top $\rho=50\%$ pixels from the dilated heat map.

In our demonstration, our algorithm used Yolo9000 Redmon and Farhadi [2016] as the multi-
Figure 4: Architecture of modified Siamese network used for training the floor identification classifier.

Yolo9000 uses a hierarchical tree classification which is built using WordNet Miller et al. [1990] concepts. For example, ‘vehicle’ would be above ‘car’ and ‘bike’. For each image, Yolo9000 outputs a set of bounding boxes, each box has a probability attached to it and a label tree. We accept boxes with a probability greater than 0.1, and we set a threshold of 0.7 to determine which label in the tree we use (we prefer more general yet accurate labels to specific but uncertain ones).

The hierarchical nature of the classification leads to some objects receiving labels near the top of the tree. For example, some objects would be labeled instrumentality, things or matter rather than a more specific label. Our rejection corpus $R$ is also constructed to remove these labels. For our building-floor domain, $R = \{\text{‘home appliance’, ‘living thing’, ‘container’, ‘artifact’, ‘person’, ‘conveyance’, ‘bottle’, ‘instrumentality’, ‘whole’, ‘deep-freeze’, ‘electric refrigerator’, ‘machine’}\}$. At least $t = 0.5\%$ of the pixels in the explainable feature bounding box must be in the importance mask for our algorithm to include it in $E_M$.

We use the same building-floor training and validation dataset to determine the explainable class attributes. The attributes are presented in the results section below. Additionally, our algorithm discretizes the image into $g=9$ grid squares. We defined groundings like ‘top’ for $(0,1)$, ‘top right’ for $(0,2)$, and ‘center’ for $(1,1)$ $gIds$.

Putting all of the information together, our algorithm outputs natural language text based on our template:

$$I \text{ am in Location}(y), \text{ because I see Label}(O_0) \text{ at GridLabel}(gId_0), \text{ Label}(o_1) \text{ at GridLabel}(gId_1), \ldots \text{ and Label}(o_n) \text{ at GridLabel}(gId_n).$$
Table 1: Explanations for three floors of our building.

4.4 Robot Explanation Demonstration

We tested our algorithm’s ability to explain each floor of our building. The intermediate and final outputs are provided for three example floors in Table 1. For the first image representing Floor 3, the multi-object detector finds six explainable features that have more than $t = 0.5$ of their area included in the importance mask – $E_{M,3} = [(0,1), \text{pot'}, (1,1), \text{chair'}, (2,1), \text{furnishing'}, (2,1), \text{chair}']$. However, when compared to the class attributes, only four of the features are included in $E_{M,3}$ as shown in Table 1. The natural language explanation for the Floor 3 image is: ‘I am in floor 3, because I see pot at left top, chair at center, furnishing at right center and chair at right center.’

5 Practical Implementation Results and Discussion

In implementing the explanation algorithm, we observed several practical requirements and made several design decisions that affect applicability to new domains.

First, it is important to choose a secondary classifier that can actually label the features in the domain and ideally produce unique attributes per class. Two of our floors (2 and 7) have
very few features in our training images. In fact, the classifier was unable to find any features in any training images for Floor 2. Since no other class attribute set was empty, the lack of explainable features was considered unique to that class. In our testing, Yolo9000 could not detect any objects for two images from ‘Floor 7’ either, since the image contained nothing other than a brown wall (sometimes there was a plant in view). The inability to distinguish Floors 2 and 7 on those images were the only two errors in our classifier. Training Yolo9000 to detect walls in general (it labeled walls as ‘electric refrigerator’ in some images as shown in Figure 2c) and specifically the difference between brown and white walls would have improved this result. However, overall the algorithm was very robust to our classifier and threshold choices.

Next, we made a decision to include only the explainable features in the image that also occurred within the class attributes. While taking an intersection to generate $E_{M,c}$ provides a good explanation in our domain, we also considered other ways of determining which features to include. In one alternative, we included all objects from $E_M$. As a result for Floor 3, the algorithm explains seven features [(0,1), pot’, (1,1), pot’, (1,1), chair’, (1,2), seat’, (2,1), person’, (2,1), furnishing’, (2,1), chair’] rather than four. In practice, we found that removing features that were not important for the class (e.g., seat) reduced some of the confusion about whether the feature was important for the image or for the classification. Another alternative was to use the explainable features that represented the difference between $E_{M,c}$ and $E_{c'} \forall c' \in C \& c' \neq c$. This algorithm explains three features [(0,1), pot’, (1,1), pot’, (2,1), person’] which uniquely differentiate it from all other classes. In practice, this selection algorithm often produced too few and/or empty feature lists because all the classes contain a similar set of objects but in different positions. Choosing to explain $E_{M,c}$ was a good balance between including enough features to be useful yet limiting the features that were not important in the classifier.

Finally, our implementation built on top of the extensive prior work in finding important regions of images. To extend this work to classifiers operating on non-image data, we suggest using other existing importance functions such as PCA or a modified version of the deep visualization technique. We used an image-based multi-object detector, but any classifier over subsets of features would be applicable. Even the discretization of our image for an explanation can be adapted to non-vision data. For example, rather than using full timestamps in temporal data, it is possible to discretize events by morning, afternoon, evening, and night.

6 Conclusion

We contribute an algorithm for automatically generating an explanation for classifiers using natural language. In order to avoid explaining all features, our algorithm first finds the important regions of the image. Then, using a secondary pre-trained classifier, it finds the explainable features that makeup only the important regions of the image and uses the labels as the natural language description. By relying on a separate multi-object detector, our algorithm can label and explain the images without manual labelings. The algorithm then determines which of the image’s features are common to the class and generates an explanation of only those features. We demonstrate our algorithm on a real CNN classifier used on our robot and discuss practical implementation details that affect its applicability to new domains.
References


