Introduction

Over the course of this summer, my research was focused on the fields of Machine Learning and AI. The purpose of my research was to try and explore various classifiers available for image classification, mainly LIME and SHAP. LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive exPlanations) are like detectives for computer images. SHAP counts how much each part of a picture matters to the computer's choice, while LIME tries different picture changes to explain why the computer decided what it did. Both help us see why the computer thinks certain things. My goals over the summer also included exploring current and past works in this topic, exploring different kinds of image datasets available and already used, trying these two classifiers with multiple image datasets and see if there'd be a way to improve the image classifications.

The way these classifications work is if you feed an image to the computer and ask these classifiers for predictions, they will try to predict what the image is about and then find the most important pixels which led the classifiers into making the predictions about the images. For instance, say if there's an image of a baby holding a ball, the classifiers will try to make predictions about the image (like "ball" or a "baby") and then identify the most important pixels in the entire image that lead to its classification.

Results and Accomplishments:

I started the project by exploring the work that's already been done in the field, skimming through the research as recommended by my mentor, and exploring various projects' GIT repositories about the same. I kick-started the working by exploring basic images, faces, number-images, blurred images, and various object-images with both of the LIME and SHAP while also calculating the respective accuracies, precision, recall etc. in order to get an idea about how these classifiers work, how accurate they are, how do they identify the important pixels and if the classifications can be improved by tweaking the parameters in any way.

Once I had a fair idea about how these classifiers work and in which direction can I lead this research into, I decided to mainly focus on LIME for the scope of this fellowship and started trying to improve these classifications. The dataset I mainly focused on in this project is the CIFAR-10 dataset. The CIFAR-10 dataset is a collection of 60,000 32x32 pixels long images categorized into 10 different classes, commonly used for machine learning tasks. Since the images only have 32x32 pixels, these images are quite blurry if you convert them into average sized images which puts the LIME classifier into test. The way my project works is I downloaded the dataset into my local device and used LIME to classify the important pixels for the image, then I prompt the users to choose from 👍 and 👎 options to see if the given explanation of the important pixels by the classifier meets their expectations or is there a scope for improvement and based on the user's feedback the classifications can be improved.

For the project to succeed, I used the tensorflow module, which is quite useful for Machine Learning algorithms, and associated libraries for image classification and explanation. After importing the required libraries, a TensorFlow session is established, and the InceptionV3 model (a common CNN architecture widely used for image classification) is configured based

on the requirements. By initializing the model with a pre-trained checkpoint, the script enables predictions on input images. Then, I fed the computer the directory which had all the CIFAR-10 images stored. I prompt the computer to choose any image at random from the entire database, the chosen image is returned, classified and the pixels important for its classification are prompted and then the users get the option to choose from 👍 and 👎 options as discussed above.

I have two helper functions in my code for random_images() and top_label_explanation(image, n). The random_images() function returns a different random image whenever executed and top_label_explanation(image, n) uses that image to generate the important pixels. By default, the value of n is 1 and the value of n increases by 1 whenever the user selects 👎 as the option for the classification, and as the value of n increases, LIME shows a different classification for the same image and will keep trying to improve the classifications till the user selects the 👍 option. In other words, until the user chooses the 👍 button, LIME will keep trying to improve its classifications for the same image and as soon as the user is satisfied with the classification, they can use the 👍 button and then be prompted by another random image from the directory which will proceed in the same way. To exit the loop, the user just needs to write 'quit' or 'q' in the terminal.

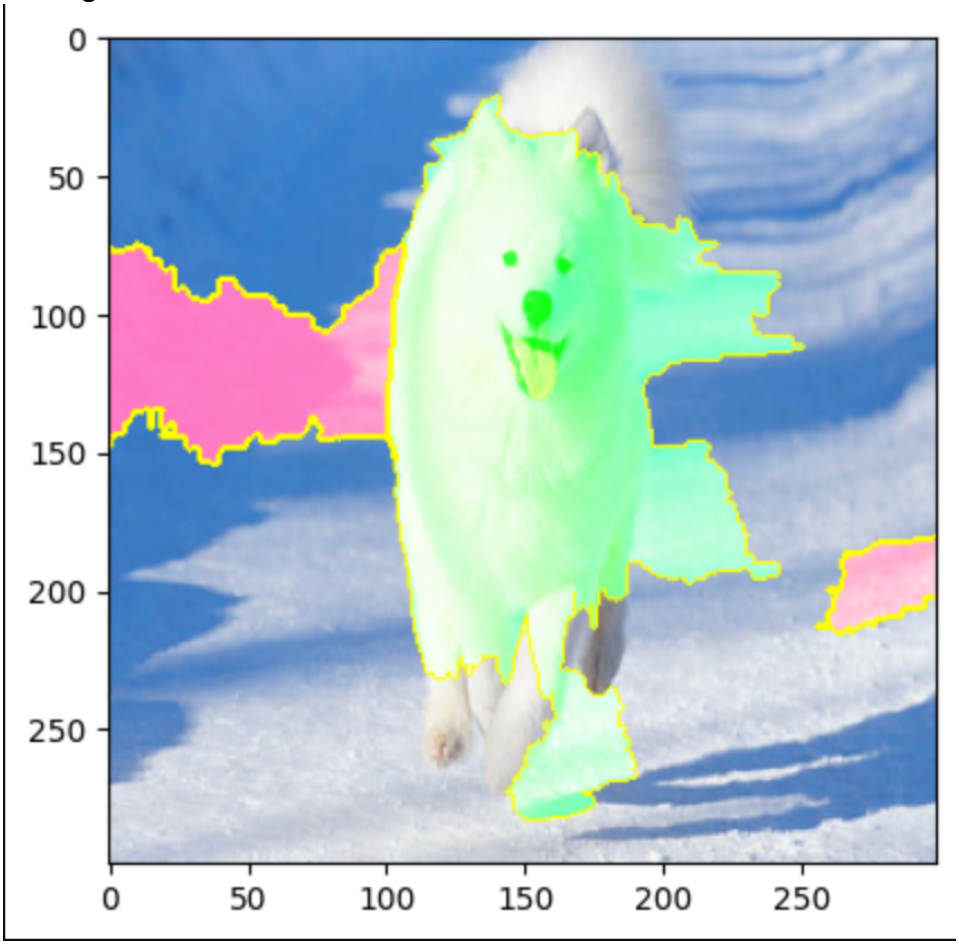
To illustrate the idea, let's take a sample image:

It's the picture of a dog, specifically the Samoyed. When fed into the LIME classifier, it provides us with 5 descriptions of what this image can be, ranging from most likely to least likely. Here are the 5 results I received for the image:

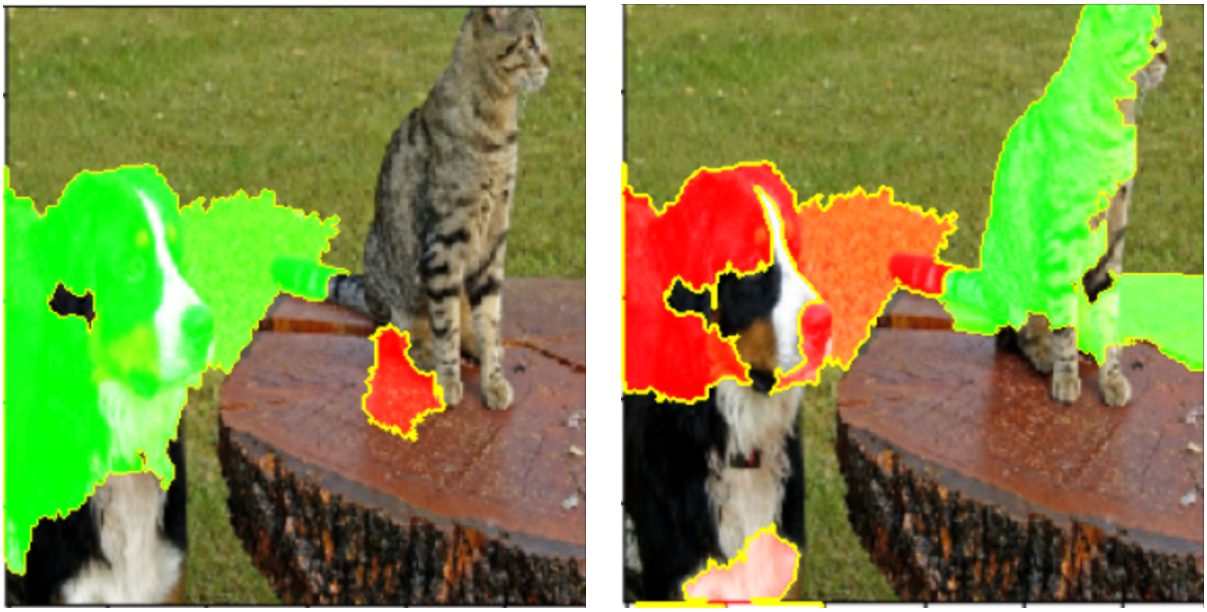
```
259 Samoyed, Samoyede 0.8532087
280 Arctic fox, white fox, Alopex lagopus 0.006204973
260 Pomeranian 0.0021916789
271 white wolf, Arctic wolf, Canis lupus tundrarum 0.0019501925
262 keeshond 0.0014903763
```

All of them either related to a specific breed of a dog or any animal that might look close enough (like white fox). As one can see, the correct dog breed that's listed on the top, Samoyed is the most likely and the correct description for the image with a confidence score of 0.8532087 out of 1.

Then, when I try to return classify the most significant pixels leading to this description, I receive this image:

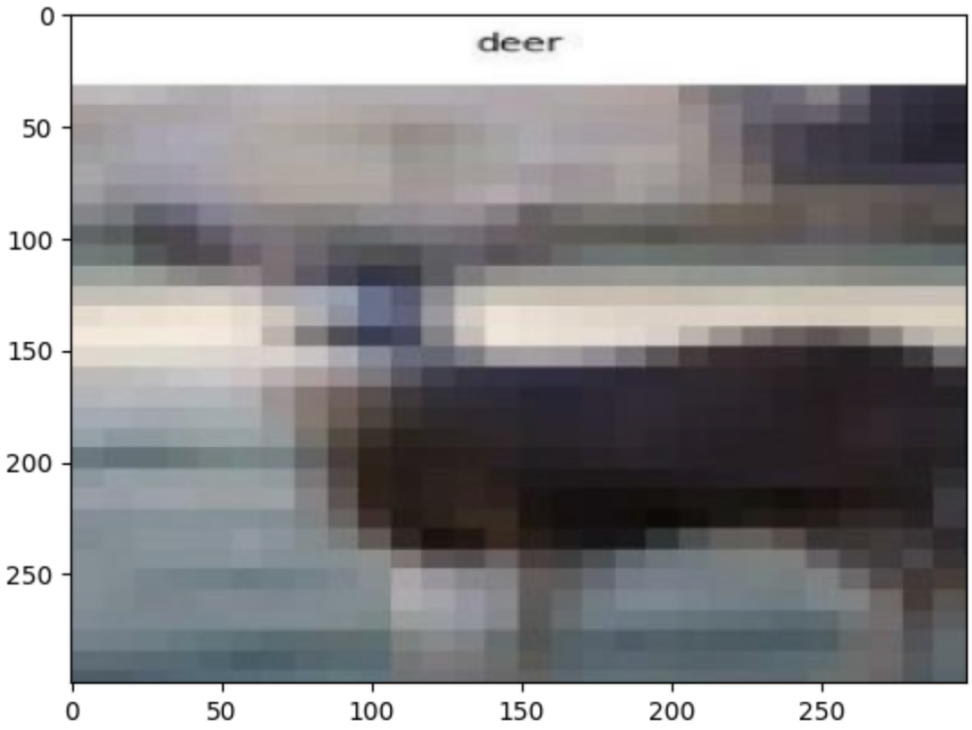


Green selections mean that these are the most significant pixels for the description of the image as a “Samoyed” breed for the dog while the red pixels were the one’s not as significant for the classification. It kind of makes sense that the green pixels almost cover the entire dog and the red pixels are the ones in the background, not contributing anything to the image of the dog. Next, I fed an image of a dog with a cat and here are the results:

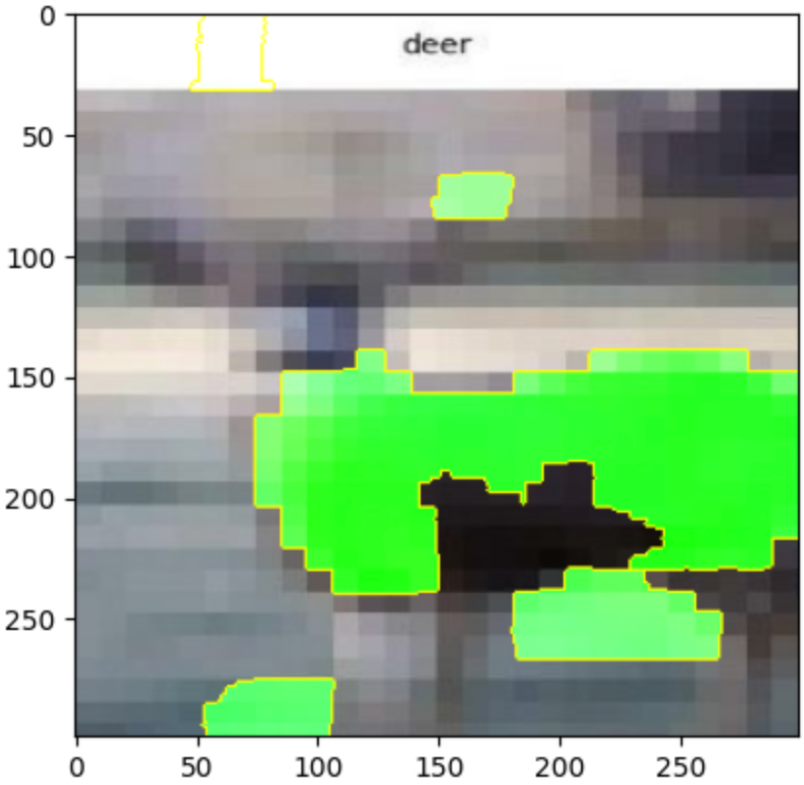


Important pixels for the classification as a dog v/s as a cat.

Now let's run through my program and see how it works. As I mentioned before, I have a directory filled with blurry cifar-10 images and when I run my code, an image is chosen at random with the pixels important for its classification colored.



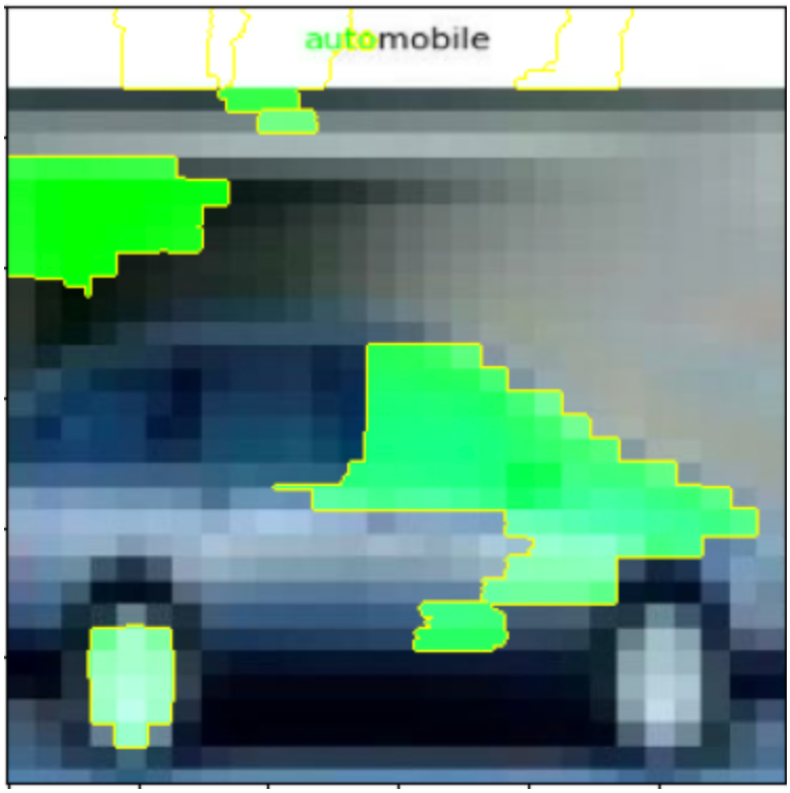
The image above can be seen as an extremely blurry image of a deer. While running the classifier, the following result is obtained:




```
Provide feedback (👍 or 👎):
```

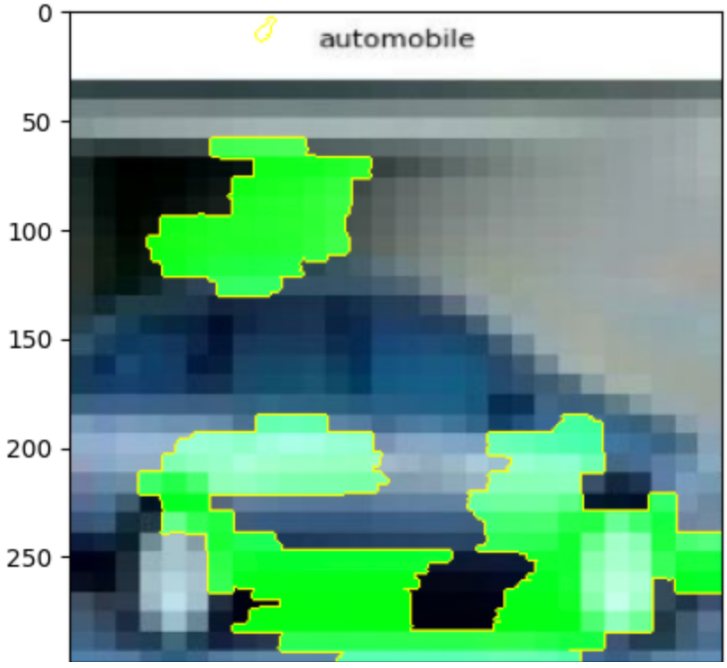
Then, it asks for the feedback based on the classification. If the user chooses 👍, it'll proceed to another image within the directory and repeat the process but if the user chooses 👎, it'll attempt to better the classification and return a different set of classifications and will keep doing that till the user selects 👍.
Next, after choosing 👍, I received the image of an automobile:

And here's the classification:



Now, one may argue that the most widely known pixels for an automobile classification should be its wheels. This classification only selects one wheel so there might be a scope for improvement. Here, if I choose the 👍 button with respect to the classification, the following result comes:

```
Provide feedback (👍 or 👎):  👎
100% 300/300 [00:10<00:00, 28.08it/s]
11.274767875671387
```



Which does select a part of both the wheels.

And once you're done with the classifications, you can just type in quit and exit the loop:

```
Provide feedback (👍 or 👎):  quit
Do you want to restart? (yes/no):  no
```

Now, I believe there's a lot of scope for improvement in this field. One major drawback I have in my project is these user feedbacks and improved classification only comes into place while classifying the same image. LIME would try to improve the classifications for the same image as long as the user keeps selecting the 👎 button, but as soon as the user is satisfied and selects the 👍 button, the program starts afresh. It doesn't learn from other previous classifications and feedback. So incorporating this one thing can be a major source of challenge. Further, though LIME works quite decent already with normal images, as can be seen above, it's not as great when it comes to blurry CIFAR images so that might be something that can be worked upon in the future as well.