Street Sign Detection

For our computer vision project, we worked on creating a program that would allow us to identify specific road signs that were found in US streets. The program should be able to take a look at a given image of a random road in the United States and if that picture has common road signs found it should be able to find the road sign and identify what the road sign means. This is the main issue our program is going to be tackling but the code has been created so that it allows for a small amount of editing to increase the number of items the program can identify for example another problem that it could be edited to tackle is reading license plates off a car. This allows the program to have a lot of variety in its applications and can be implemented to tackle many different problems and to be used in many different spaces.

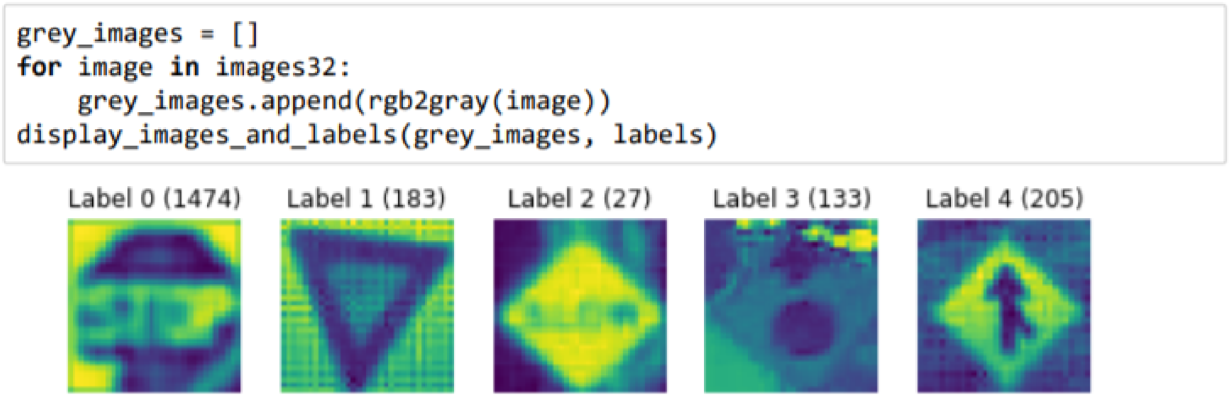
Now, why is this an important problem? We see the issue of being able to have a camera or vehicles recognize specific road signs as a good issue to tackle because in today's cutting-edge technology many companies are working on vehicles that have some type of self-driving feature or autopilot. This type of technology has been increasing throughout the years and seems like it is going to be an extremely large market that everyone will want to get into soon and one of their major issues are going to be how to have their vehicles accurately be able to identify road signs and other important aspects throughout a trip. If a vehicle does not correctly identify specific signs or road hazards throughout its trip it can cause many problems like a vehicular accident or even death. Many companies will be trying to lower there chances of liability by having the technology in their cars and having a vehicle that can correctly identify a road sign even if the conditions are not optimal, like having a sign without color or in a poor angle to the camera that is an important thing that must be considered. The method that we have implemented is a very important thing to consider to because even though the program is very compact it's also is very powerful. In computer vision, it is impossible to get a model that is 100% accurate and some of the best models in the world can only be 95% which is an important thing to consider when creating a program especially in situations where someone's life may be in your hands. Our program cannot be as accurate as something that may be used by Tesla or Ford, but to get the type of accuracy that they are looking for the program must also be relatively large with many different variables and factors to consider. Our program has a respectable rate of accuracy around but taking a look at the code which is relatively small we were able to get a model with a small amount of overhead.

Throughout researching how we were going to solve our problem we found many other people that are working on the same issue or issues that are extremely similar. One of the best examples of someone working on a roadside detection system we were able to find was Waleed Abdulla. He was working on creating a program using TensorFlow that could accurately recognize traffic signals commonly found in Belgium, and we learned some important things from what he was working on for example his program finds the specific road sign in the picture and resized it allowing the program to be much more simple and accurate since it has already found image of the road sign and is specifically only looking at it, reducing the amount of unnecessary information the model has to look at. The program he created was relatively simple due to the fact that the data he used was relatively small and many of the pictures of road signs were very clean and easy for the camera to identify. Another important programmer that we looked at was Alex Staravoitau. He also used TensorFlow to categorize and process a large German traffic sign data set. An interesting aspect he used in his road sign recognition program was to be given an image of a roadside then randomly slip or rotate that image. This allowed his model to be very accurate due to the fact that the images scanned multiple times at different angles and appearances. If a model is flipped in a direction that it shouldn't be and is unrecognizable that image is discarded and if images are in that appear as other images they are categorized separately. After a lot of research and on others who were tackling the same problem we were we were able to create a programmer proud of and is relatively accurate.

In analyzing our model and how we chose to tackle our problem one of the most important aspects of how we successfully created a traffic sign recognizing model was in how we preprocessed our imaging data. One of the major issues we had to work on how to make sure the model does not factor in the color of the image. The reason this is very important is because out in the real world when a vehicle sees a traffic sign, that sign could be under a lot of different conditions and variables, for example, there could be a large amount of glare that would affect the color, it can be night and dark or the signs them self could different color depending on where they are. One of the ways we tackled this issue was by recoloring every image we got before having it being analyzed by the model. All images were changed from whatever color we got it to a gradient descent of black and white where zero is the whitest an image can get and 255 is the blackest an image can get. Doing this allowed the model not to take color the consideration.

In analyzing our model and how we chose to tackle our problem one of the most important aspects of how we successfully created a traffic sign recognizing model was in how we preprocessed our imaging data. One of the major issues we had to work on how to make sure the model does not factor in the color of the image. The reason this is very important is because out in the real world when a vehicle sees a traffic sign, that sign could be under a lot of different conditions and variables, for example, there could be a large amount of glare that would affect the color, it can be night and dark or the signs them self could different color depending on where they are. One of the ways we tackled this issue was by recoloring every image we got before having it being analyzed by the model. All images were changed from whatever color we got it to a gradient descent of black and white where zero is the whitest an image can get and 255 is the blackest an image can get. Doing this allowed the model not to take color the consideration. In the example, we show we change the picture to have a gradient scale between green and blue that way it would be more appealing and easier for us humans but see but when putting it into the model it was changed to black and white.

.

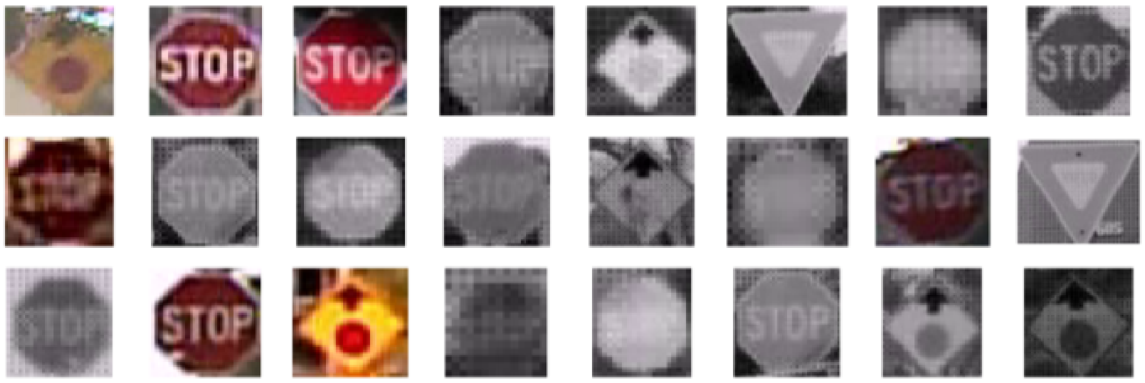


```
grey_images = []
for image in images32:
    grey_images.append(rgb2gray(image))
display_images_and_labels(grey_images, labels)
```

The next issue we worked on was the size and the pixel count of the picture we got. Since all of the images have traffic signs that are different distances away from us the image of the traffic sign itself will be a different size. This would cause a problem because the model would start training itself based on the size of the picture we gave it and also affecting how much detail the model could analyze inside of the picture itself. So to fix this issue every image was resized to be 32 x 32 pixels. This did cause some of the smaller pixels to become extremely blurry and others that were much larger to be very detailed. Fortunately, the model was sophisticated enough to not be affected by how blurry the image was since most of the street signs were relatively different and the model normally had enough information from the shape and how the image inside the sign itself looked to be able to differentiate between different signs.

There was another factor we had to consider when feeding our model images. We did not want our model to accidentally be dependent on the order we gave it images so before the model started looking at the images and training we would randomize the order. This allowed every image that all got to be in a different place in the model not to be too dependent on how it receives the information. We first realize problem when we were training the model on a data set that we did not randomize in the model came so accurate that there had to be an issue. The model started to see the order of the image it was getting and every time start categorizing it based on that information, which caused us to overtrain our model to that specific dataset.

Throughout working in testing on our program we used a few different data sets. Originally we were using a data set that was completely composed of random images taken from Google maps. Unfortunately, the problem with this data set was even though the data set was extremely large and we had a lot to choose from most of the images were not specific to the problem that we were tackling. There would be a lot of images that do not have road signs that would be able to be used or just didn't have road signs in general. Then when we were researching we were able to find a Belgium data set. The Belgium dataset was pretty small so we didn't get a lot of information from would be used it to test out a few of the programs were doing in the beginning because it allowed us to quickly run through a model, see if it were and check how accurate it might be. Lastly, we landed on the Laboratory for Intelligent and Safe Automobile traffic sign dataset which luckily had just come out. This data set specifically included 47 different US traffic signs types. All of the images had been annotated and the images themselves were of very high quality. This is the current dataset we are using to train and test our model. Another great reason why the LISA data set was so useful to us was that it had just come out no one was working on this data set are ready so we could find unity in different ways to tackle the problem of US traffic signs recognition. Here are some of the data sets we decided to use our final model.



After finding the LISA dataset, we found out that it contains multiple frames that were extracted from a video of a car driving through US roads. These frames contained street signs such as Stop, Yield, TurnLeft and so on. The dataset also included a CSV file that had all the image names, annotations and coordinates of the sign in the frames. We continued by cropping out the street signs that were found in those frames and organized them in properly labeled files using python scripts. We split the main LISA database, that is composed of around seven thousand images, into two distinct training and testing sets. We applied a filter to make the cropped images have a black-and-white color. We also resized all the cropped images into a uniform size of 32x32. The dataset was also equalized which limits variations in photos and allow our model to get more accurate reading. This process also helped us deal with the glare that was present in some frames and made it hard to recognize the street signs. After that, we normalize all the pixels again so that each individual pixel will only hold a value from 0 to 255. In order to avoid the model developing a certain tendency of predictions, we randomized model input which makes the data less predictable when using multiple batches of the same file.

After processing the data set and doing the alterations that we discussed earlier, we discovered that our accuracy mainly varied and depended on the existence of speed limit signs. Whenever there were no speed limit signs in the testing dataset, our model was able to get almost 97% accuracy; However, when speed limits do exist in the testing data set, our model's accuracy drops down to 85% which need further investigation.

In conclusion, building this project required us to do a significant amount of research from the software perspective. This project also required us to have a good grasp of what a neural network is and all of its inner components like layers and nodes. However, with a very simple neural network, we were able to get a decent amount of accuracy and would've definitely yielded greater results if more data was provided.

Team Roles

Luke was responsible for training and testing the model.

Nadim was responsible for setting up the dataset.

Thiago was responsible for the final report.

We all worked in a team assisting each other and also did our research together

References

BelgiumTS - Belgian Traffic Sign Dataset, btsd.ethz.ch/shareddata/.

Abdulla, Waleed. “Traffic Sign Recognition with TensorFlow – Waleed Abdulla – Medium.” Medium, Medium, 17 Dec. 2016, medium.com/@waleedka/traffic-sign-recognition-with-tensorflow-629dffc391a6.

“Laboratory for Intelligent and Safe Automobiles - CVRR - UCSD.” Computer Vision and Robotics Research Laboratory, cvrr.ucsd.edu/LISA/lisa-traffic-sign-dataset.html.

Staravoitau, Alex. “Traffic Signs Classification with a Convolutional Network.” Alex Staravoitau's Blog, 15 Jan. 2017, navoshta.com/traffic-signs-classification/.