**Paul Tluczek**
Automotive Classification within Neural Networks

There are very many problems that have been encountered with the production of the initial couple of stages implementing the “first” project.

**Initial Data:**
There was no data. I tasked myself with creating the data from photo-shopping out vehicle body parts from high resolution images. I then created batches of these images by copying them hundreds of times. Transformations were done with the ImageDataGenerator function from Keras where every picture was randomly transformed via scale, rotation, skew, affine, color shift, etc. This would create a nice training set where the Neural Network would be constantly being fed random samples of augmentation from the ImageDataGenerator. This is ideal as the training phase would never become over-fit. Ideally I would have liked to do this in OpenCV where the range of options is much greater, and I could really get an Object Detection project going by introducing randomly generated backgrounds for the vehicle body-parts.

**Initial Data Baseline:**
Random Forests was the initial baseline machine learning function used to predict classification. A smaller subset of the batch of images was used and permanently augmented in random ways. This was my way in trying out OpenCV. However for the smaller subset of data, these augmentative randomizations were too vast for the Machine Learning model, so I just used a small rotation and reflection for the images. This was enough for the Random Forests function to predict the bodypart in the image 70% of the time.

**Neural Net Learning Data:**
The convolutional neural network that tried to learn the data performed poorly and would rarely ever get over 30%. Having experimented with different architectures for a two dimensional, convolutional neural network, this was enough to start thinking about going a different direction.

**Final Data:**
However if I wanted to be able to predict the bodyparts in an image, on an automobile, multiple times, there had to be much more augmentations to be done. However in the limited scope of time that I had, I realized I did not have the time to learn OpenCV to the extent that I wanted. This alone would be the factor in changing my idea into something else.

The final dataset was an imageset from Stanford, uploaded to imagenet. It was a dataset of 10,000 images classified by make, model, and type. I decided to extract the type, and use that solely as a more difficult scenario to predict. This prediction would include looking at the general shapes in the car as well as smaller details that could only be made evident from proper training. Technically these smaller details would be the same kind looked for in what my original project was going to be, so this dataset was still in line with that general vision.

Modeling included various architectures that ranged from multiple consecutive convolutions, to no drop layers, to a VGG-16 architecture with and without relu and sigmoid activations. None of these models raised more of an accuracy than around 30%.

Eventually RESnet was implemented, which is a “residual” type of neural network, that takes the input, passes it through consecutive layers in the block, while simultaneously bypassing them as well. The output into the next block of layers is summed together. This allows for features to be compared with the initial image. Since the architecture of the neural network is not always deemed to be perfect, this allows for the outputed layers to be compared with what the initial input is. Importance between all of this data can then be compared before entry into the next block with an activation function.

**More Time:**
With an extra two weeks, I would like to read up more about OpenCV and all of its image augmentation functions. Applying that to my original data set would allow me to create a much larger data set with much more non-linearity and entropy for a Neural Network to learn on.

Additionally I would like to implement the RESnet model onto my original dataset. RESnet's architecture allows for a much more efficient way in looking for important features within a complicated image. This would have progressed the initial project much further than the original stages. Being able to predict the parts of an automobile are the first stage in later being able to detect damage.

**Does it Scale:**
This kind of neural net can be used for any kind of image set that requires a highly complex amount and kind of features. RESnet is more useful than a regular convolutional network, because it doesn't guess with its architecture, but actually compares with the original image if RESnet is instituted in the very beginning.

**Data Product:**
Websites like cars.com and any kind of website that lists possible vehicle type as an attribute is a possible candidate for this kind of software. Instead of manual labor being instituted where an individual goes through every single image and assigns an attribute, the weights from the neural net will be already able to assign all of these vehicles.

**Beating the Baseline:**
Beating the Random Forests model is not difficult because Random Forests isn't able to learn features the same way that a neural network does. If a neural network encounters an image, it will look for these features. However not much is known about what the Machine Learning algorithm picks up in terms of training, and most likely when a new image dataset is introduced, it will perform very poorly.

**Tradeoffs:**
As I explained above, the tradeoffs are that not much is known about the machine learning algorithm and what the features are that it actually learns during the training process. The mathematical algorithm works in a way where it takes random features at every step in the branch between the trees that make it up. So saying that it comes to one conclusion every time the model is trained, would be a very naïve assumption. It is mostly a non-deterministic and stochastic process that the data goes through within the Random Forests algorithm. It cannot be relied on to produce meaningful data trained from a sample to learn the entire population.

**Generalization:**
Th evidence that can be presented for sufficient generalization is depicted by the test set that is evaluated upon at the end of the model. Since the entire dataset is completely non-linear and not much similarity exists in much of the images, the test data that is tested upon which has never been shown to the statistical model is a perfect way of measuring performance.

**Significance:**
There is definite significance in the model since it has predicted the test set accurately. Since the distribution of balance has been normalized and made optimum, and the performance of the model is good, there is little to prove that the model is not significant. Different augmentations could most likely be worked on to improve performance further.

**Hyperparameters:**
During the grid search, it showed that none of the hyperparameters really mattered in making a distinguishable difference in the model accuracy. This is most likely because of the model architecture not being completely accurate in the first place. There are many ways of combining these different layers in creating a convolutional model. However, the hyperparameters may be adjusted in the RESnet model, since the architecture alone delivered credible results. This means that the hyperparameters may actually show a measurable difference.

**Accuracy:**
Accuracy was used as the final metric since the classes served no imbalance. There was no use for other metrics as accuracy was alone able to tell the progress of the model. The loss function also depicted credibility of the model and was shown to decrease over time as accuracy went up.