Statistical Learning Individual Write-up
Hunter Kempf

Table of Contents

Define the Questions

Binary Classification requires two classes to be selected or made out of the dataset. In this case the CIFAR-10 dataset has 10 potential classes which can be individually chosen or combined to make a binary classification problem. When thinking about the types of images in the dataset our team chose to look at the problem of airport incidents between aircraft and automobiles. This classification algorithm would sit as part of an airport tracking system that would determine in real time where the automobiles and planes are on the various runways and other airport roads. A system like this could be much cheaper than an IOT system that tracks all the automobiles belonging to the airport and can provide similar accuracy. This system could reuse existing airport cameras to do the analysis. Our piece of the solution will assume that there is a preprocessing step that is able to find objects and send a 32x32 pixel image to be classified as either a plane or automobile. With that as the motivation for the problem we moved into the next step of Getting the Data.

Get the Data

The CIFAR-10 dataset comes presplit with 50,000 images for training and 10,000 images for testing. Since there are 10 classes in the CIFAR dataset that means that there are 5,000 images of training and 1,000 images of testing for each class. Since our usecase is only focused on automobiles and airplanes, our total dataset size is 12,000 images broken down into 10,000 training images (5,000 for both planes and automobiles) and 2,000 testing images (1,000 for both planes and automobiles). Before analyzing and making models on the data we split an additional 2,000 images out of the training set to create a validation set. This was done so that the test set could be made up of images that the model had never seen during the optimization phase. Our final dataset split was 50/50 between automobiles and planes put into 8,000 images of training, 2,000 images in validation and 2,000 images in testing. Since the original Data had Airplanes labeled as 0 and Automobiles labeled as 1 we did not have to make any changes for our binary classification to work.

Explore the Data

Our team started looking at the data and a few things quickly became apparent. First the data was made up of 32x32 pixel images. This means that the contents of those images were often grainy, pixelated and hard to decipher even for humans. Second the dataset included images taken from multiple angles. The different angles would mostly be a good thing for our usecase since automobiles would be driving around the airport and wouldn't be always recorded from the same angle. The issue is that some of the times these images would be zoomed in on a bumper or a fender which could cause the model to be confused more easily. Third the dataset often had images of different types or colors of planes and cars. This would be

good to catch a variety of car and plane types that may be at the airport. One drawback is that the model may train on and get really good with a certain type of plane or car that will never be at the airport. Finally images had additions in them such as humans sitting on the hood of a car. This would be good for our case because humans (airport staff) will be on and around cars and planes at our airport and it would be bad if a human in an image caused the plane or car to be misclassified. In an ideal world we would collect our training, validation and testing datasets from the airport cameras which would control for most of these potential issues and give us more "real world" data that would work for our situation.

From the perspective of useful features uncovered in our EDA, a few things are clear. Wings and propellers are clear signs that an image is a plane. Wheels can be in both cars and planes but may be visible more often in cars so they would probably be a feature with a slight lean towards an image being a car. Backgrounds can be deceiving since most planes are photographed on the ground there is no significant predictor that an image is a car or a plane based on the background color of the image. Based on the variety of shapes of planes (especially military style planes) classification techniques may have problems with the fact that there is no single feature that every plane in the dataset has.

Preprocessing

Our team decided to use a CNN model. This is an algorithm that usually does better without any preprocessing in front of the convolutional layers. The only real caveat to that is for data that is not on the same scale. In our case all of the data was on the same scale from 0-255 which means that a CNN model could take the input in a relatively raw format. We also compared this with a min max scaler which scales the min of the dataset to 0 and the max to 1

which I will talk about further in the Model Optimization stage. The benefits of getting data into a 0-1 range is that it will run through the training phase faster. A preprocessing step that we normally would have to do would be converting an image to an array. Since we were using the CIFAR-10 dataset this step was not needed because it can be imported from the Keras library.

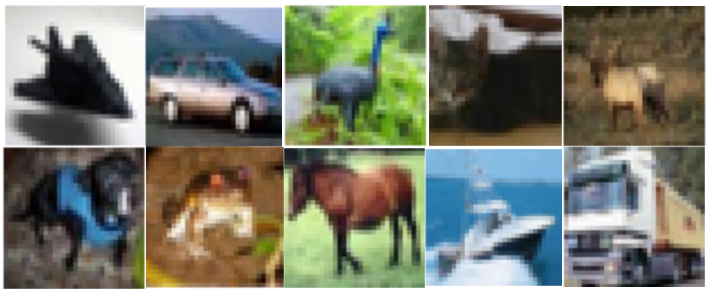
Initial Model

We chose a Convolutional Neural Net model since it is a model that works well with image and audio data and is able to select features from the image in a 2d window. The initial model we built was a basic CNN with 8 filters in a single convolutional layer and a 2x2 window. After the convolutional layer we added a single pooling layer with a 2x2 window size then flattened the outputs and passed it to a hidden layer with 32 nodes and a single output layer with a sigmoid activation function. This very basic model was trained over 10 epochs with a batch size of 100. We decided to optimize our model using binary cross entropy which is a loss metric that will penalize uncertainty in the model (a .51 will have a higher loss than a .99 if the correct class is 1) and accuracy for the evaluation metric. Accuracy is a good metric for more balanced cases and since our case was a 50/50 split accuracy is suited well to show the results. Our base model results were pretty impressive for a first attempt, the accuracy was 92.73 percent on the training set and 88.95 percent on the validation set. Note we did not test the initial model on the training set because we did not want to include it in the model selection process and we only tested our final model on it. This model showed us that our Convolutional Neural Net was the right path forward for this use case.

Model Optimization

Finally we arrived at the Model Optimization stage. This is the part of Neural Nets that is much more of an art than a science because there are so many potential parameters to tune. The tuning parameters we tried were: window sizes (both for the convolutional and pooling layers), number of filters in the convolutional layer, l2 regularization penalty, dropout, activation function, number of convolution layers, number of pooling layers, number of dense hidden layers and number of nodes in each dense hidden layer. While that list may seem like a lot really it is just a subsample of the types of parameters that can be tuned in a neural net. Some of the biggest gains we achieved with our final model over our baseline model were related to increasing the window size from (2,2) to (3,3) basically this allows for a bigger area to search for features in and can allow bigger or more complex features. Another area of improvement came from adding more convolutional blocks, we ended up with 3 convolutional layers each one followed by a pooling layer which was followed by a dropout layer. The last big area of improvement was to add more dense hidden layers, we ended up with 3 dense hidden layers after the convolutional blocks and flatten. Our Final Model had a score of 97% accuracy on the held out test set which matched the 97% accuracy on the validation set. This gives me confidence that the model isn't overfit and would perform similarly given new data. We did try our final model on the min max scaled data and it ran in about 2 seconds per epoch vs 15 seconds per epoch that it took to train without the scaled data but the performance was much lower at around 92.8% and the training speed improvement in this case did not outweigh the accuracy difference.

Figures



- 0 → airplane
- 1 → automobile
- 2 → bird
- 3 → cat
- 4 → deer
- 5 → dog
- 6 → frog
- 7 → horse
- 8 → ship
- 9 → truck

Figure 1. EDA: Example of each image class in the CIFAR-10 Dataset

Figure 2. EDA: Challenges in the CIFAR-10 Dataset

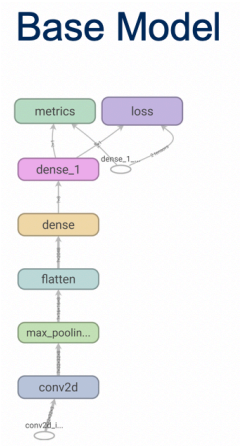
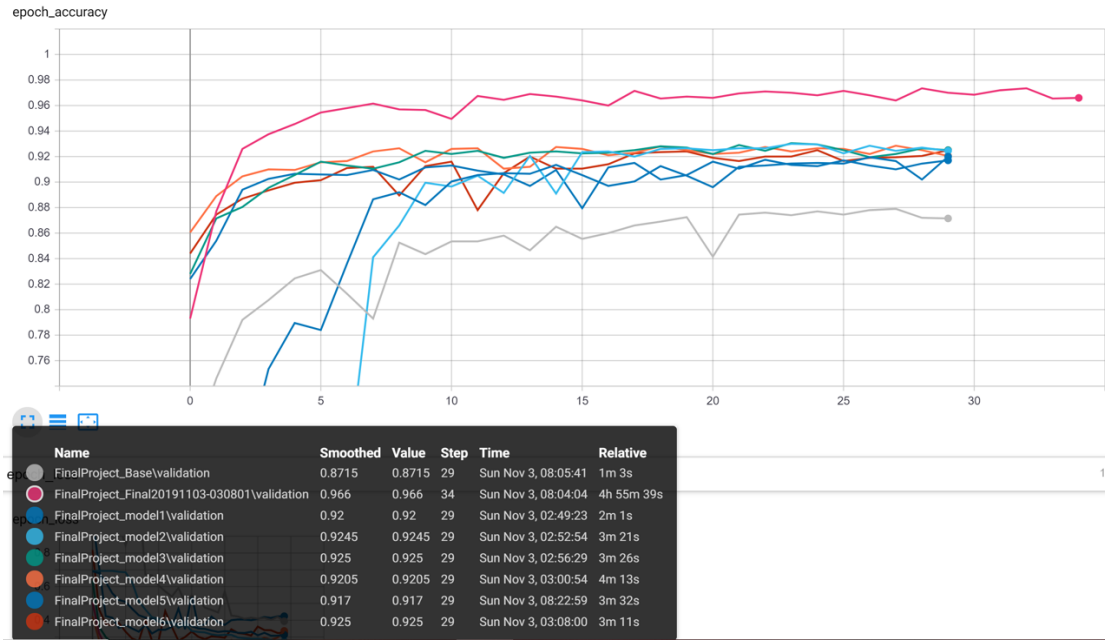


Figure 3. Base Model: Structure



Name	Smoothed	Value	Step	Time	Relative
FinalProject_Base\validation	0.8715	0.8715	29	Sun Nov 3, 08:05:41	1m 3s
FinalProject_Final20191103-030801\validation	0.966	0.966	34	Sun Nov 3, 08:04:04	4h 55m 39s
FinalProject_model1\validation	0.92	0.92	29	Sun Nov 3, 02:49:23	2m 1s
FinalProject_model2\validation	0.9245	0.9245	29	Sun Nov 3, 02:52:54	3m 21s
FinalProject_model3\validation	0.925	0.925	29	Sun Nov 3, 02:56:29	3m 26s
FinalProject_model4\validation	0.9205	0.9205	29	Sun Nov 3, 03:00:54	4m 13s
FinalProject_model5\validation	0.917	0.917	29	Sun Nov 3, 08:22:59	3m 32s
FinalProject_model6\validation	0.925	0.925	29	Sun Nov 3, 03:08:00	3m 11s

Figure 4. Model Optimization: Validation Scores

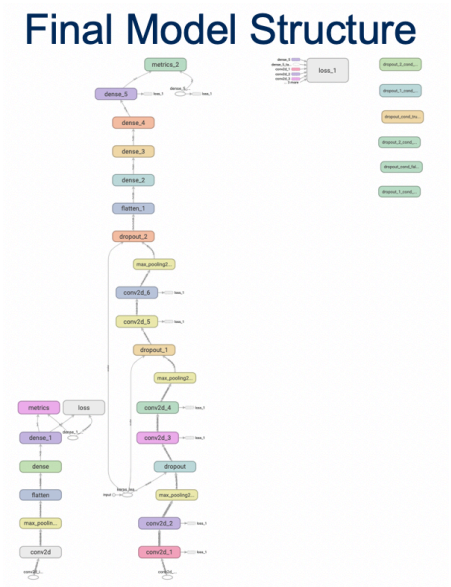


Figure 5. Model Optimization: Final Model Structure

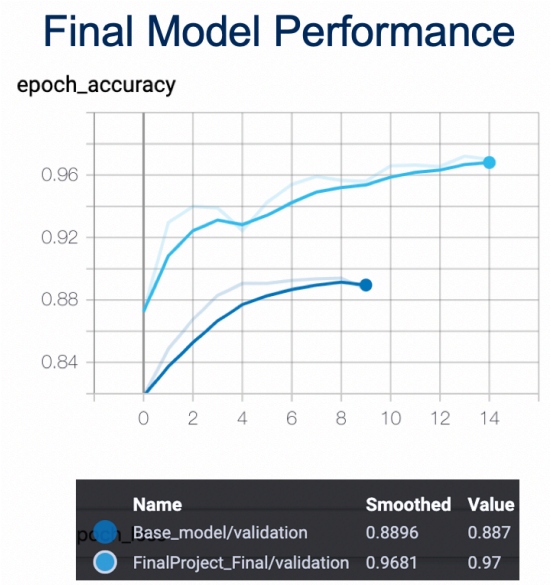


Figure 6. Model Optimization: Final Model Performance

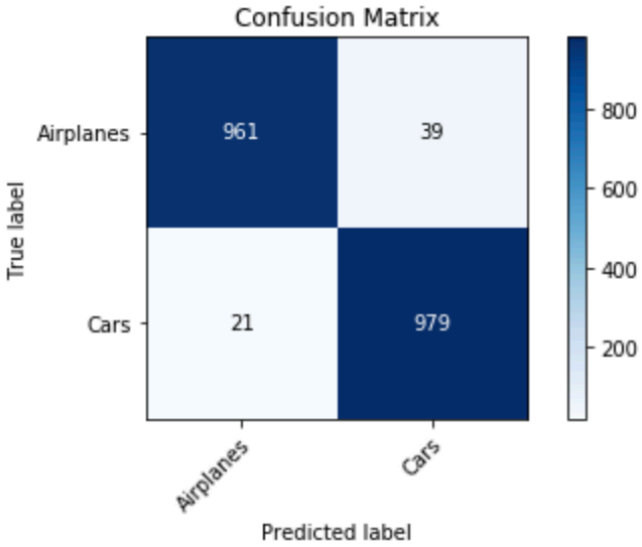


Figure 7. Model Optimization: Final Model Confusion Matrix

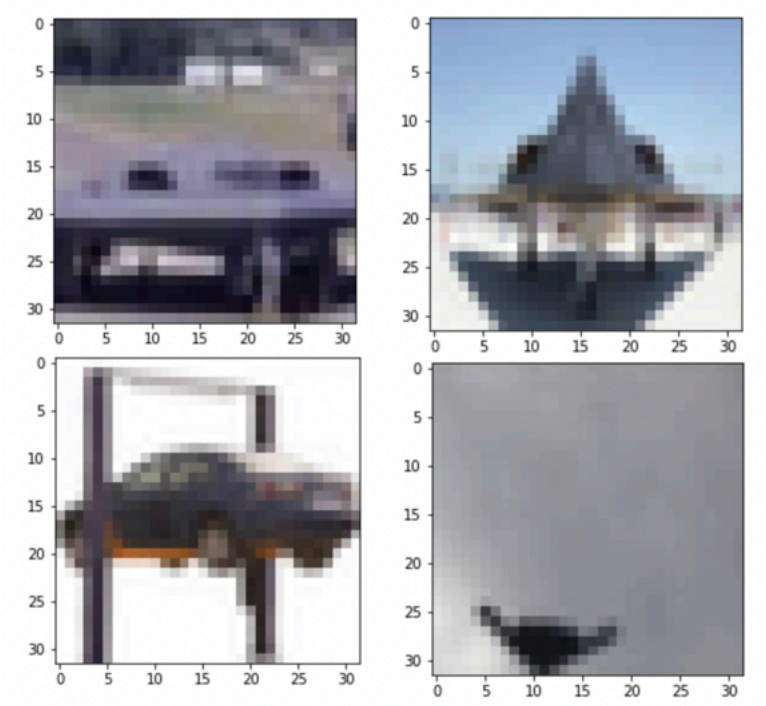


Figure 8. Model Optimization: Final Model Example Misclassified Images

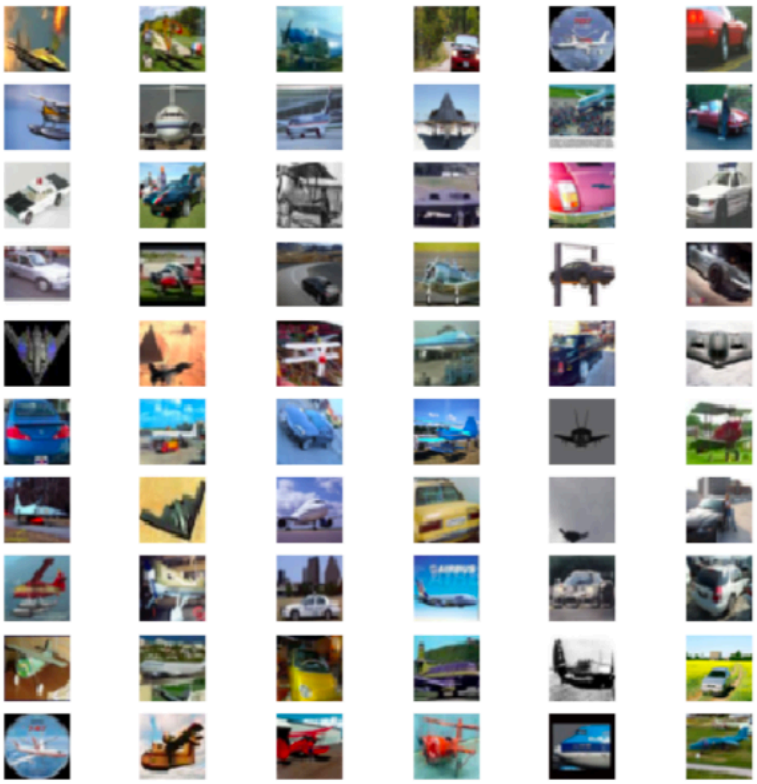


Figure 9. Model Optimization: Final Model All Misclassified Images