FINAL PROJECT: IMAGE CLASSIFICATION

| Terrance Randolph | IST 652 |

Colab Code

INTRODUCTION

TOPIC OF INVESTIGATION

The topic I seek to investigate in this project revolves around the concept of image classification using deep neural networks. Furthermore, the primary purpose is to understand how neural networks work on the surface and to properly build and evaluate models used for image processing.

IMPORTANCE TO MY COMPANY

Since being employed for the Army National Guard as a Geospatial Analyst I have noticed the real property managers and engineers spend countless hours sifting through utility images from Armories (bases) in order to locate where a manhole or cleanout is located. Therefore, if I can train a model to accurately find an image that corresponds to their search, because the model had properly labeled the new images then perhaps it would save time and money for the state of South Carolina.

DATA ORIGIN

The data I am using comes from Kaggle and originates from the Canadian Institute for Advanced Research (CIFAR-10) dataset. It was collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton, who participated in a group at MIT and NYU that collected millions of tiny color online images. More detailed analysis from the group can be found in this whitepaper.

- **CIFAR-10**: 60,000 32x32 color images in 10 different classes.
    - Airplane, Automobile, Bird, Cat, Deer
    - Dog, Frog, Horse, Ship, Truck
    -

DATA CLEANING

CIFIR DATA

When working with imagery the pixels must be cleaned and transformed to a certain specification that can be processed by the algorithms. Therefore, the data is shaped in a 28 pixel by 28 pixel format then labels are encoded into a binary vector that creates an array with for each element, leaving every index 0 and

list index 1. Next, normalizing the pixel digits to a floating structure and ensuring it's normalized by the max number of pixels 255, creating a pixel range between 0 and 1.

GOOGLE DATA

- **GOOGLE IMAGES**: 26 color images in 10 different classes.
  - Airplane, Automobile, Bird, Cat, Deer
  - Dog, Frog, Horse, Ship, Truck
  - Photoshop convert images to 32x32
  - Replicated CIFIR Data cleaning methods in python

LINEAR CLASSIFICATION

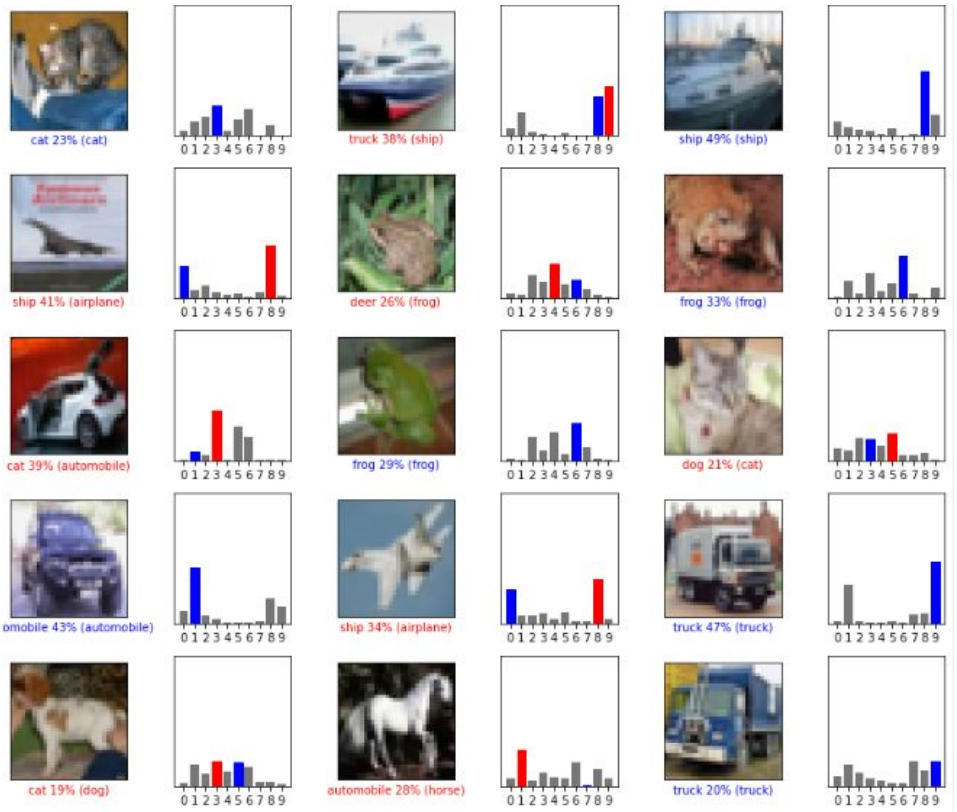
MODEL PARAMETERS

The linear classifier used stochastic gradient descent (SGD) learning. Meaning, linear combination of x and y would be any expression of the form ax + by, where a and b are constants. The gradient estimates the loss of each sample at a time and the model is updated along the way with a decreasing learning rate with the weight defaulted at 1. Furthermore, this model was built with max_interations of 1000 (epochs) meaning the training set of 60,000 images was processed with the linear classification specification n-epochs.

PREDICTION ACCURACY & METRICS

Accuracy: 37% Processing Time: 304sec

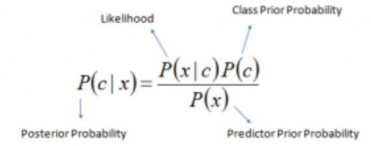
PREDICTION VISUALIZATIONS



NAIVE BAYES

MODEL PARAMETERS

Naive Bayes is a bayesian classifier that assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Moreover, it determines unique characteristics of a class (image labels) based on pixel value combinations and decides what the overall category is based on probability of a culmination of unique categories. NB



$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

- P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).

- P(c) is the prior probability of class.
- P(x|c) is the likelihood which is the probability of a predictor given class.
- P(x) is the prior probability of the predictor.

PREDICTION ACCURACY & METRICS

Accuracy: 28% Processing Time: 19 sec

PREDICTION VISUALIZATIONS

	precision	recall	f1-score	support
	WORST			
airplane	0.26	0.42	0.32	57
automobile	0.35	0.20	0.25	41
bird	0.19	0.10	0.13	51
cat	0.06	0.02	0.03	49
deer	0.18	0.33	0.23	40
dog	0.26	0.21	0.23	48
frog	0.32	0.52	0.40	54
horse	0.27	0.06	0.10	47
ship	0.38	0.49	0.43	57
truck	0.37	0.39	0.38	56
	BEST (NOT GOOD)			
accuracy			0.28	500
macro avg	0.26	0.27	0.25	500
weighted avg	0.27	0.28	0.26	500

K-NEAREST NEIGHBOR

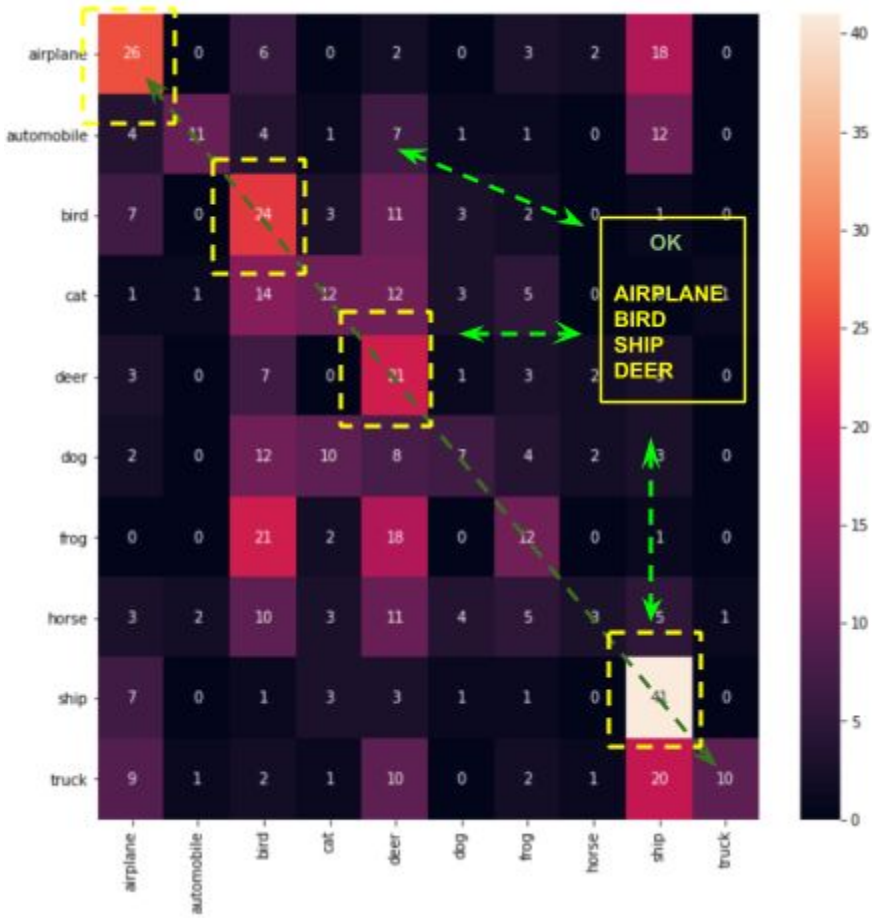
MODEL PARAMETERS

KNN moves through the image array to measure the pixel and its surrounding pixels based on a distance function (this case ‘minkowski’). However, I could have improved the image with perhaps using euclidean as the distance and increasing the neighbors will broaden the spatial aspect/boundary causing more images to be grouped together. However, too high of a ‘K’ can cause high errors by miscalculating images in a category.

PREDICTION ACCURACY & METRICS

Accuracy: 33% Processing Time: 311 sec

PREDICTION VISUALIZATIONS



RANDOM FOREST

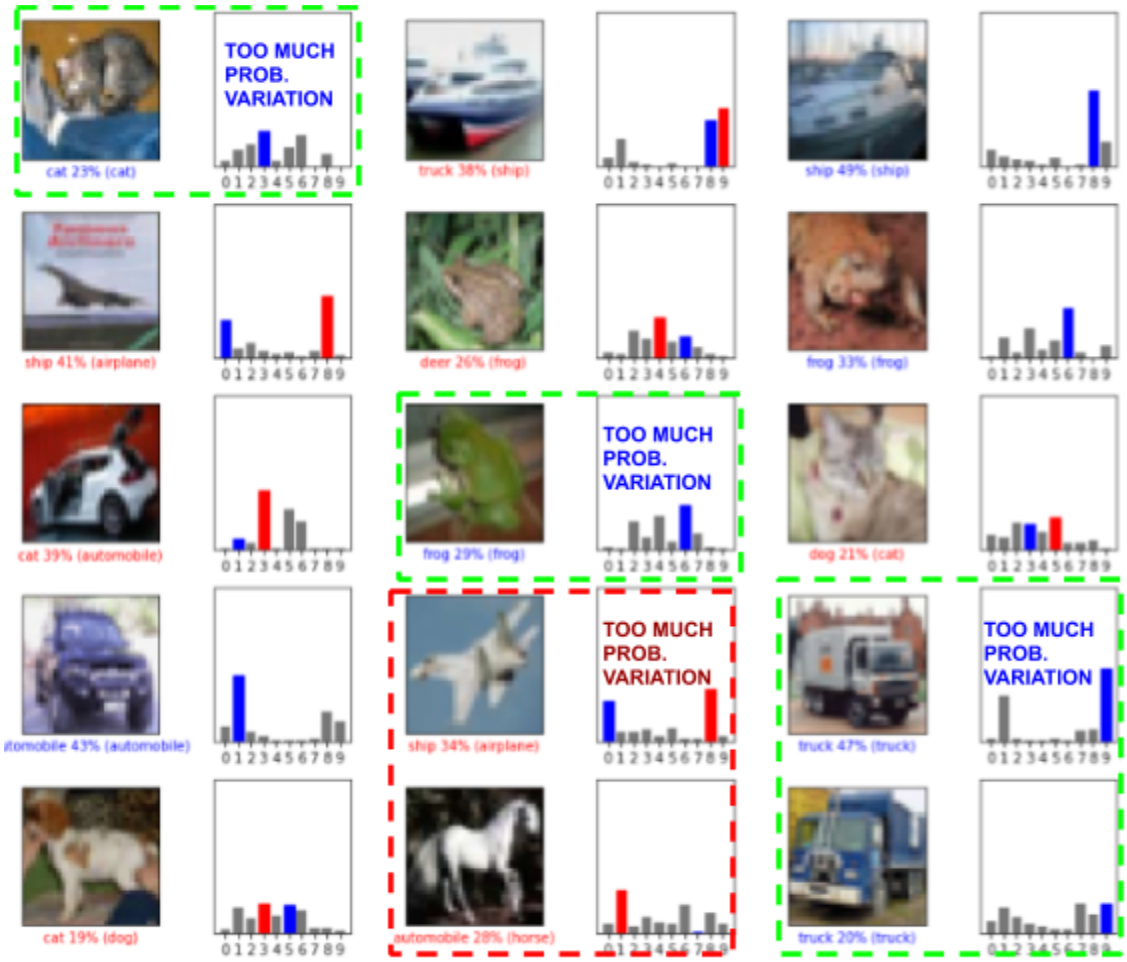
MODEL PARAMETERS

Random Forest is a classifier that uses the mean of multiple decision trees to determine the accuracy and proper classification of a class. Moreover, the decision trees determine boolean results based on probability for each image categorical classification based on grouped pixels over 100 epochs.

PREDICTION ACCURACY & METRICS

Accuracy: 44% Processing Time: 400 sec

PREDICTION VISUALIZATIONS



MULTI-LAYER PERCEPTRON

MODEL PARAMETERS (675 Epoch)

MLP is a neural network that classifies the images based on multiple based on multiple activation levels of perceptrons. Therefore, in this particular case Rectified Linear Units are used but with Stochastic Gradient Descent (SGD) as the optimizer. Whereby, the loss is minimized quickly resulting in a decent accuracy. Although, 100 nodes with an equal amount of hidden layers and Iteration of 630 loss 0.01006517 were introduced the accuracy for this particular dataset resulted in 39%.

PREDICTION ACCURACY & METRICS

Accuracy: 39% Processing Time: 2,922 sec

PREDICTION VISUALIZATIONS

	precision	recall	f1-score	support
airplane	0.27	0.67	0.38	57
automobile	0.55	0.63	0.59	41
bird	0.33	0.35	0.34	51
cat	0.42	0.41	0.41	49
deer	0.24	0.17	0.20	40
dog	0.21	0.15	0.17	48
frog	0.48	0.26	0.34	54
horse	0.50	0.36	0.42	47
ship	0.55	0.46	0.50	57
truck	0.60	0.38	0.46	56
accuracy			0.39	500
macro avg	0.42	0.38	0.38	500
weighted avg	0.42	0.39	0.38	500

CONVOLUTIONAL NEURAL NETWORK

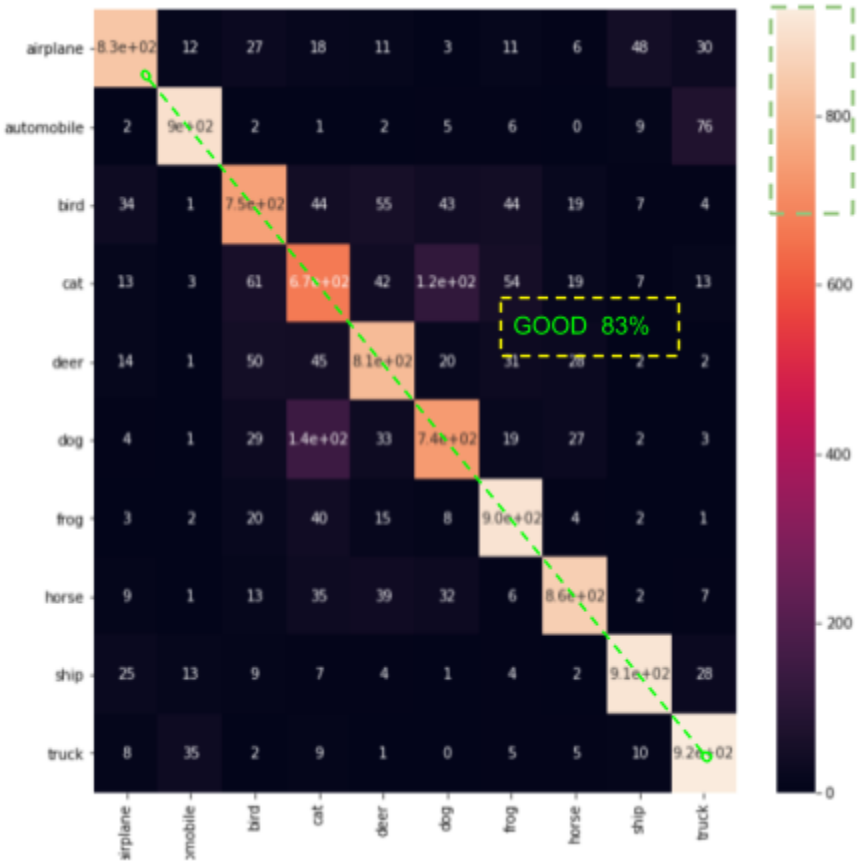
MODEL PARAMETERS

CNN uses filters for each pixel that is passed through each node (100 Nodes) that captures the value from 0 - 1 and all other values as 0 (rectified Linear Units). Therefore, quickly eliminating unnecessary values, then pooling the curved or angled filtered pixel groups to the last layer (10 Nodes) which captures the probability that the curve is there based on a softmax algorithm.

PREDICTION ACCURACY & METRICS

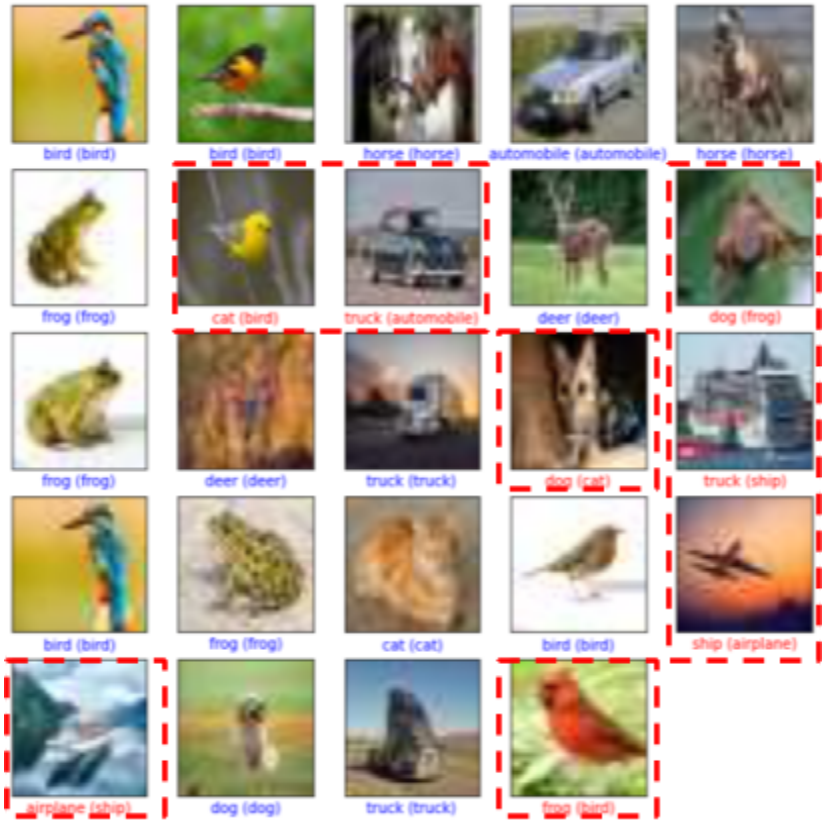
Accuracy: 83% Processing Time: 967 sec

PREDICTION VISUALIZATIONS



SELF COLLECTED DATA

Accuracy: 67% Processing Time: 2 sec



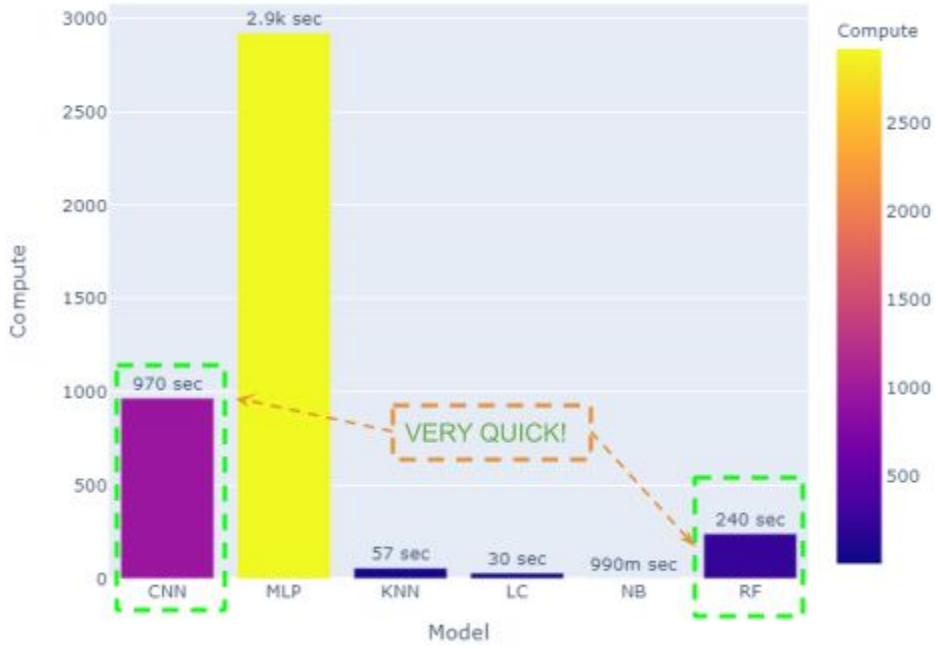
CONCLUSION

Classifying images can be a daunting task when the wrong model is chosen. However, after implementing and reviewing the models I believe implementing CNN is the best option, but capturing the mean from high numbers of cross validation and adding more layers would dramatically improve the 83% accuracy. However, certain parameter alterations to Random Forest could prove to be an optimal choice due to its fractional computation time. Therefore, when seeking to build a viable image classification model the analyst must understand and test model parameters and performance to determine which algorithm to implement for the best results.

MODEL ACCURACY & METRICS

Best method for prediction based on accuracy and compute time is random forest and CNN.


83%
GOOD!
BETTER WITH
PARAM. CHANGE
48%
39%
33%
22%
28%
Accuracy
80
70
60
50
40
30
0
10
20
30
40
50
60
70
80
CNN
MLP
KNN
LC
NB
RF
Model


2.9k sec
970 sec
VERY QUICK!
240 sec
57 sec
30 sec
990m sec
Compute
3000
2500
2000
1500
1000
500
0
CNN
MLP
KNN
LC
NB
RF
Model

REFERENCE

Brownlee, J. (2019, October 3). How to Develop a Deep CNN for Fashion-MNIST Clothing Classification. Retrieved from https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-fashion-mnist-clothing-classification/

CIFAR-10 - Object Recognition in Images | Kaggle. (2014). Retrieved from https://www.kaggle.com/c/cifar-10/data

Scikit-Learn. (2011). SGDClassifier: Journal of Machine Learning Research. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html

Scikit-Learn. (2011). GaussianNB: Journal of Machine Learning Research. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

Scikit-Learn. (2011). KNeighborsClassifier: Journal of Machine Learning Research. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Scikit-Learn. (2011). RandomForestClassifier: Journal of Machine Learning Research. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier.predict_proba

Scikit-Learn. (2011). MLPClassifier: Journal of Machine Learning Research. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

Weights & Biases. (2019). Retrieved from https://www.wandb.com/tutorial/multilayer-perceptrons

Krizhevsky, A. (2009, August 4). Learning Multiple Layers of Features from Tiny Images. Retrieved from http://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf