

**CSC 180-01 Artificial Intelligence Mini-Project 3: Computer Vision using GPU and Transfer Learning**

**Due at 2 pm, Friday , October 29, 2020**

BY

Perry Gill

Rajvee Modi

Mary Ballesteros

**Problem Statement**

In this project, we practiced with image classification using Google GPU and transfer learning. Software to detect network intrusions protects a computer network from unauthorized users, including perhaps insiders. This project aims to build an AI based network Intrusion Detection System (IDS), a predictive model distinguishing between bad connections, called intrusions or attacks, and good normal connections.

In order to solve this issue, we needed to run through the dataset and filter out the null and redundant values. We treated this as a binary classification and used (CNN) Convolutional Neural Networks in order to compare the accuracy, precision score, recall score, F1-score, and support of all the models. We used a Sequential Model, Logistic Regression Model in order to compare the accuracy and plot the Confusion Matrix.

**Methodology**

We imported the cifar dataset from keras.datasets and split it into test data. We changed our class vectors to be one hot encoded. Our model without the transfer learning used the relu activation function and the output layer was softmax. The accuracy of our model was approximately 75%. In our randomly selected pictures we got ⅘ right. With transfer learning we imported the VGG16 model and added all the layers in it to our sequential model. We froze the wrights by setting trainable property to false. We added 3 dense layers and our model accuracy was 71%. From the randomly selected images we for ⅘ right.

**Experimental Results and Analysis**

We implemented a Sequential Model for our CNN without Transfer Learning and Pre-trained VGG16 Model in order to display important metrics used for comparison when implementing (CNN) Convolutional Neural Networks. We found that our activations: Sigmoid, relu, and softmax gave us the best results. We set our range to 5, and used Early Stopping to stop training once our model performance stopped. For our Sequential Model, we were able to output the three different layers: Convolutional Layer, Pooling Layer, and Fully-Connected Layer. Our hyper parameters that gave us the best result were relu and softmax for our activation. We reported FALSE in order to freeze the weights in each layer in the model.

**With Transfer Learning: Without Transfer Learning:**

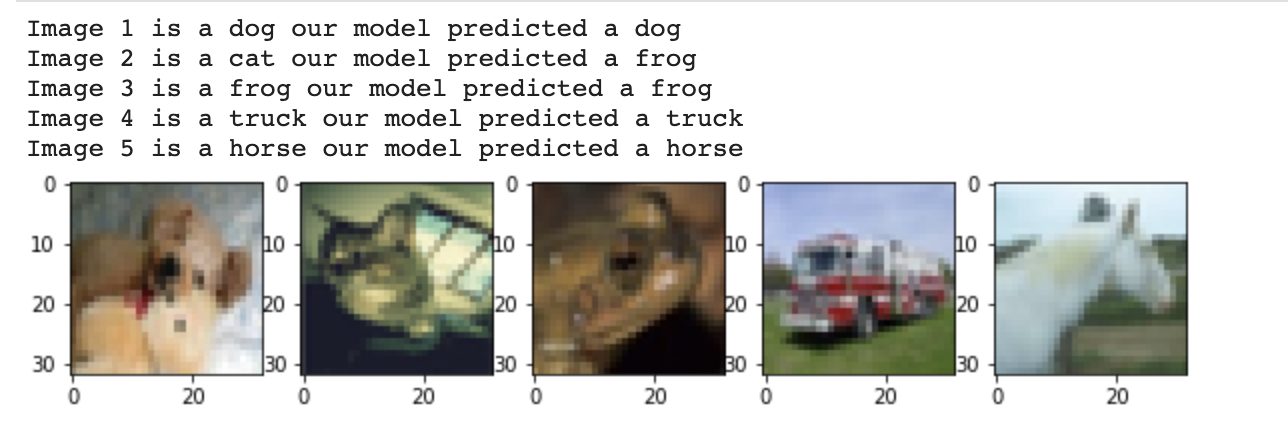
Accuracy score: 0.7104 Accuracy score: 0.7566

Precision score: 0.7179629864753917 Precision score: 0.7544874565575448

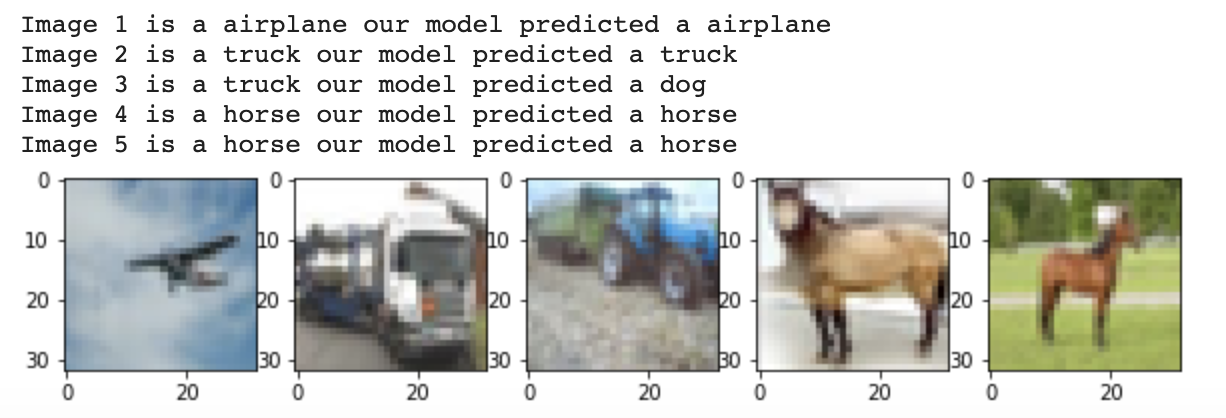
Recall score: 0.7104 Recall score: 0.7566

F1 score: 0.7118776642262932 F1 score: 0.750658291061047

**With Transfer Learning:**

****

**Without Transfer Learning:**

****

**Task Division and Project Reflection**

**Task Division:**

Perry Gill:

1. Strategize and Implementation
2. Data cleaning and preprocessing
3. Transfer Learning
4. Pre-trained model
5. Write Report

Rajvee Modi:

1. Strategize and Implementation
2. Data cleaning and preprocessing
3. Transfer Learning
4. Pre-trained model
5. Write Report

Mary Ballesteros:

1. Strategize and Implementation
2. Data cleaning and preprocessing
3. Transfer Learning
4. Pre-trained model
5. Write Report

**Project Reflection :**

From this project, we got a lot of practice with using google colaboratory. We used google colaboratory to run our code on a cloud-based Jupyter notebook environment. We used CNN without transfer learning on collab and compared it to using CNN with transfer learning. Using google colaboratory, we decided to work on this project as a unit rather than divide the work. When running through issues, we were able to figure them out together through screen sharing.

The challenge was to find the best accuracy to determine our photos. Implementing CNN with our transfer learning was a challenge because we wanted to increase our accuracy. We tried to resolve this issue by adding more layers. We were getting an accuracy of 71% with our transfer learning. While implementing different hyperparameters and layers with our model, we were able to play around with our CNN to see the different results.