Project 3

Ezekiel Sung: 219507107
Hung Truong: 301035748
CSC 180-01
Due:4/2/21

Problem Statement

Our goal for this project is to have hands-on experience with image classification using google GPU and transfer learning. The dataset we will be working with is the CIFAR-10 dataset which is preinstalled with Tensorflow.

Methodology

_____We will be using Google Colab and other libraries to train a model to classify an image out of 10 different classes. As usual, we will first import all the necessary libraries and help functions. The first model we will attempt is an image classification model without transfer learning and without earlystopping. We first load the CIFAR-10 dataset and split it between train and test sets. Then we will convert the y datasets from 2D to 1D, one-hot encoded them, converted it from int to float and normalized it. Then we began to build a CNN model. We will add a combination of all these layers: Conv2D, activation, MaxPooling2D, flatten, dropout and dense layers. Within that we will test out different parameters for activations, optimizers, neurons, kernel size and stride. After finding the best model for this basic model, we will then user earlystopping to see how much our model can improve. From there we will show 5 images in the test set as well as their true labels and their predicted label to see how well our model did.

Next we will use transfer learning where we will use a pre-trained model as the starting point. The usage of transfer learning allows rapid progress and improved performance. Again we will load up the data and split it between train and test sets. First thing we have to do is upsample our images because the pre-trained model we are using supports down to 48x48 images as input. For that reason we will resize our x dataset and change our images from 32x32 to be 64x64. Due to the fact that the function resize() also normalizes the data, we will not do any additional normalizing for our dataset. Then just like before we will convert our y dataset to 1D and convert to one hot format. Now for our model, we will load the pre-trained model VGG16 except for the top layer and add each layer to our model. We will then set the layer trainable to false to freeze the weights in each layer in the new model. Add a few extra dense layers and

output layers to our model. Then finally we will add compile, fit and earlystopping to our model and proceed with fitting the model. With our final model we will show 5 images in the test set again and as well as their true labels and their predicted label to see how well our final model did.

Experimental Results and Analysis

Basic CNN Model

We first started with a basic CNN model and after tuning the different parameters and comparing the results of each model, we found that the results of all the different settings were fairly close to each other. We did notice that with adam optimizer, we did get slightly better results regardless of the other parameters settings. Starting with sgd optimizer, we tried many different settings, but here are the main ones we tried.

1. With 2 Conv2D layers of 32 and 64 neurons, a dense layer with 64 neurons and relu we got a final accuracy score of .58.
2. With 2 Conv2d Layers of 256 and 124 neurons, a dense layer with 124 neurons and tanh we got a final accuracy score of .62.
3. With 2 Conv2d Layers of 32 and 64 neurons, a dense layer with 64 neurons and sigmoid we got a final accuracy score of .1.

With adam optimizer, we were able to get slightly better results:

1. With 2 Conv2d Layers of 32 and 64 neurons, a dense layer with 64 neurons and relu we got a final accuracy score of .67.
2. With 2 Conv2d Layers of 256 and 124 neurons, a dense layer with 64 neurons and relu we got a final accuracy score of .65.

So for the basic CNN model, our best model was given to us from the first settings of the adam optimizer with an accuracy score of .68 and here is the model summary as shown.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 2304)              0
_________________________________________________________________
dense (Dense)                (None, 64)                147520
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 167,562
Trainable params: 167,562
Non-trainable params: 0
_________________________________________________________________
```

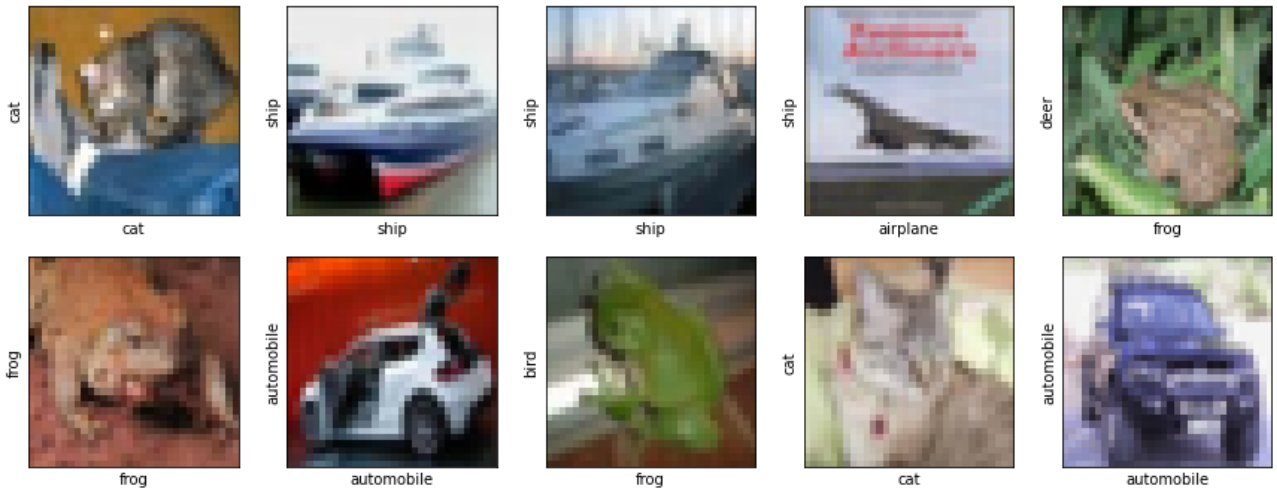
CNN Model with Earlystopping

We improved our previous CNN model by adding a checkpointer and a monitor to avoid local minimum and to avoid overfitting. We fit our model with a checkpointer that saved the best weight in a for loop. The results with earlystopping were slightly better than the previous models. In this case, we found that relu was the best activation function for the model and gave us the best results whereas sigmoid gave us the worst results. Our best model was given to us by two Conv2d layers of 32 and 64 neurons. A dense layer of 64 neurons and Adam optimizer. This model gave us an accuracy score of .71 which is about .04 better than without earlystopping. The results were not as significant as we thought, but this might also be because we have not found the superior parameter settings that can showcase the difference. As shown below are the classification reports and the predicted images.

	precision	recall	f1-score	support
0	0.78	0.70	0.74	1000
1	0.87	0.78	0.82	1000
2	0.59	0.62	0.60	1000
3	0.56	0.49	0.53	1000
4	0.62	0.72	0.66	1000
5	0.61	0.66	0.63	1000
6	0.79	0.78	0.79	1000
7	0.77	0.80	0.78	1000
8	0.82	0.82	0.82	1000
9	0.79	0.81	0.80	1000
accuracy			0.72	10000
macro avg	0.72	0.72	0.72	10000
weighted avg	0.72	0.72	0.72	10000

Final accuracy: 0.7174

```
#x label is actual class
#y label is predicted class
```



	Actual Class	Predicted Class
0	cat	cat
1	ship	ship
2	ship	ship
3	airplane	ship
4	frog	deer
5	frog	frog
6	automobile	automobile
7	frog	bird
8	cat	cat
9	automobile	automobile

CNN Model with Transfer Learning

After we applied transfer learning and used the VGG16 model, we ran into a lot of RAM issues. We could not fit our model for more than 1 epoch without the program crashing due to hitting maximum RAM. So without any additional dense layers, we fit our model with an epoch of 1 and got an accuracy score of .64 which is already really good considering that we weren't able to add onto it.

Task division

_____Hung was in charge of building the basic CNN model and the CNN model with earlystopping, which includes loading the data, reshaping data, normalizing data, training the model and testing different parameters for both the different models. He was also in charge of creating the video and creating it.

Ezekiel was in charge of the CNN model with transfer learning which also includes loading the data, reshaping data, resizing data, load pre-trained model, adding each layer to our model except the top, freezing the weights, training the model and also testing different parameters and comparing the results. He was also in charge of writing the report and compiling all the data.

Reflection

_____Although we had some issues with the transfer learning like how to exactly resize our x data set and how to use the pre-trained model, we both think that this project really helped us understand transfer learning. Our first issue was how to resize the images correctly. We had to do some research and after we were successfully able to resize, we kept running into ram issues while trying to fit our model. We were not able to find a solution around it except for lowering the epoch.We could not add any additional dense layers, increase the epoch or use a for loop without crashing due to RAM. But it made us realize how simple it is to use an existing pre-trained model to tackle a task that is similar to it. It was also really helpful that this time we were provided a skeleton of what needed to be and it really helped us understand the proper process for transfer learning and how to utilize it. We also learned to appreciate google colab because it is a great resource for multiple people to work on a project without having to constantly send each other an updated file. In addition, although we consistently ran out of disk and RAM on google colab, it was great to be able to use google's gpu instead of our own.