Food Cuisine Classification with Deep Learning

Qing Snyder
University of Missouri-St. Louis

1 Abstract

The goal of this project is to build a convolutional neural networks to classify 5 food cuisines: American Food, Chinese Food, Indian Food, Italian Food and Japanese Food. There were several steps to complete the entire project. First, I collected the images from the internet, for each cuisine, there were 1 to 2 types of classic dishes were picked. The reason that I only chose 1 to 2 dishes for each cuisine is trying to make the images between different cuisines are as different as possible since so many dishes are similar to each other across the cuisines, and we only have about 1000 images to train the model. Then I built an overfitting model to reach the best performance for the training data, which gave us 100 percent accuracy on training data, however, this "perfect" model is overfitting and only explained the training data well, for new images, we can not guarantee the model predict the class of cuisine perfectly. Therefore, I split the images into training data, validation data and test data to train the model on training data and optimize parameters on validation data, left test data out for the final evaluation. On the top of this step, data augmentation and regularization both were applied to improve the model performance. We can see there is an increase if using data augmentation. Also, to prevent the overfitting issue, regularization techniques such as L2, dropout and batch normalization can help to solve the overfitting issue. Finally, pre-trained model VGG-16 and ResNet architecture were used. VGG-16 performed well on the cuisine classification images.

2 Data Preparation

2.1 Collect and Organize the Data Set

1267 food cuisine images were collected from internet to build the convolutional neural network. For keeping the variety of the images, most duplicates images were removed and only unique images were kept in the data set. For the purpose of training a better model, the variety of dishes within a specific food cuisine is low, which means only one or two types of dish for each cuisine have been collected to create the images data. All of the images were placed into 5 folders corresponding to their labels: 'AmericanFood', 'ChineseFood', 'IndianFood', 'ItalianFood', 'JapaneseFood'. At this phase, I have not separated images to train, test and validation sets yet. In the future I will create random train, test and validation data sets and place the images into sub-folders by food cuisine type.

A subset group of images for visualization:

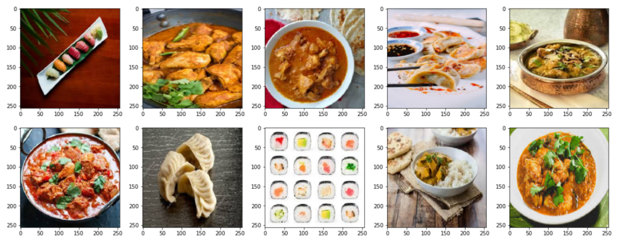


Figure 1: Food Images Sample

2.2 Distribution Of Output

- There are 299 images in Japanese Food folder.
- There are 210 images in Italian Food folder.
- There are 191 images in Chinese Food folder.
- There are 249 images in Indian Food folder.
- There are 319 images in American Food folder

Bar plot to show the distribution of output labels:

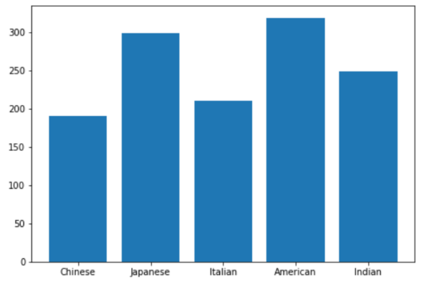


Figure 2: Distribution Bar Plot

2.3 Normalize Input Images

Re-scaling

The most common method to do the image normalization is re-scaling by dividing by maximum. The goal of normalizing is to make the images pixels to be in the range 0 to 1.

In python, 2 ways were processed for re-scaling.

1. Use ImageDataGenerator in Keras to set rescale=1./255.
2. Use asarray in numpy to check the range of pixels, then divide all pixels by the maximum, here is 255.

3 Build an Overfitting Model

3.1 Overfitting Model

5 CNN models were built to fit for comparing the the filter and layer number changes accuracy.

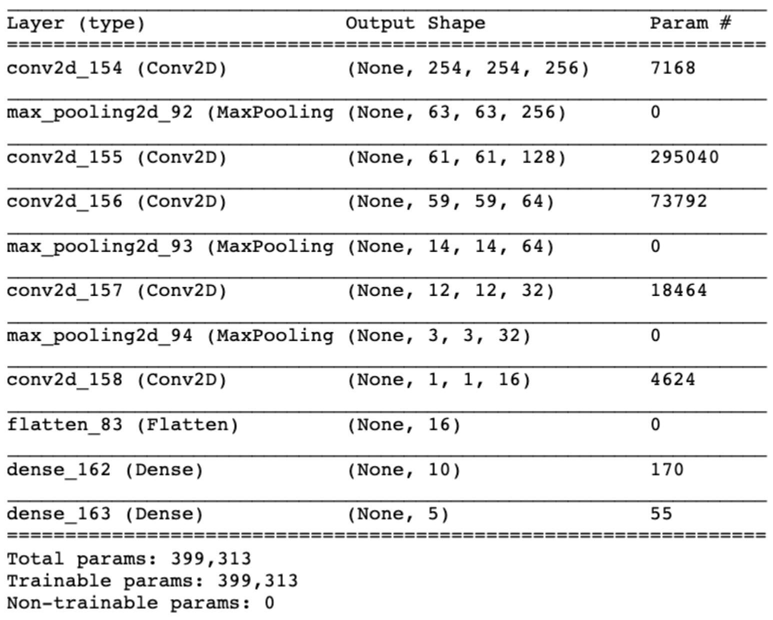
Model 1: 1 convolution layer with 16 filters

Model 2: 2 convolution layers with 16 and 8 filters for 2 layers separately

Model 3: 2 convolution layers with 32 and 16 filters for 2 layers separately

Model 4: 3 convolution layers with 64, 32 and 16 filters for 3 layers separately

Model 5: 5 convolution layers with 256, 128, 64, 32 and 16 filters for 5 layers separately



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_154 (Conv2D)          (None, 254, 254, 256)     7168
_________________________________________________________________
max_pooling2d_92 (MaxPooling (None, 63, 63, 256)       0
_________________________________________________________________
conv2d_155 (Conv2D)          (None, 61, 61, 128)       295040
_________________________________________________________________
conv2d_156 (Conv2D)          (None, 59, 59, 64)        73792
_________________________________________________________________
max_pooling2d_93 (MaxPooling (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_157 (Conv2D)          (None, 12, 12, 32)        18464
_________________________________________________________________
max_pooling2d_94 (MaxPooling (None, 3, 3, 32)          0
_________________________________________________________________
conv2d_158 (Conv2D)          (None, 1, 1, 16)          4624
_________________________________________________________________
flatten_83 (Flatten)         (None, 16)                0
_________________________________________________________________
dense_162 (Dense)            (None, 10)                170
_________________________________________________________________
dense_163 (Dense)            (None, 5)                 55
=================================================================
Total params: 399,313
Trainable params: 399,313
Non-trainable params: 0
```

Figure 3: Model 5: 5 convolution layers with 256, 128, 64, 32 and 16 filters

How the Accuracy Changes for Different Filter and Layer Number:

From below learning history and learning curve plots, there are some observations we can learn:

- Model 1 did not learn anything from the images, so there is no increase for accuracy or decrease for loss.
- When layer number changes from 1 to 2, the filter number are 16 (same as Model 1) and 8, Model 2 achieved 0.94 accuracy at 7th epochs.
- Model 3 stays the same layer number to Model 2, however, the filter numbers increased from 16 and 8 to 32 and 16. Model 3 achieved 0.93 accuracy at 8th epochs.

- Model 4 increased layer number to 3 with filters are 64, 32 and 16. Model 4 achieved over 0.9 accuracy at 21st epoch.
- Model 5 increased layer number to 5 with filters are 256, 128, 64, 32 and 16 achieved 0.92 accuracy at 24th epoch.
- Both Model 2 and Model 4 achieved 100 percent accuracy within 50 epochs, Model 1 and Model 5 both have accuracy very close to 1 within 50 epochs.
- In summary, Model 1 is not an effective model. Model 2 and Model 3 learned pretty fast, which is around 8 epochs both achieved to over 0.9 accuracy. Model 4 and 5 learned slower than Model 2 and Model 3 because they have more layers and more filter numbers, which means they are more complicated models. 4 Models all performed pretty well on accuracy: achieved or very close to 1 within 50 epochs. However, considering the learning speed, Model 2 is sufficient to be considered as an overfitting model.

Models Learning Curve Plots

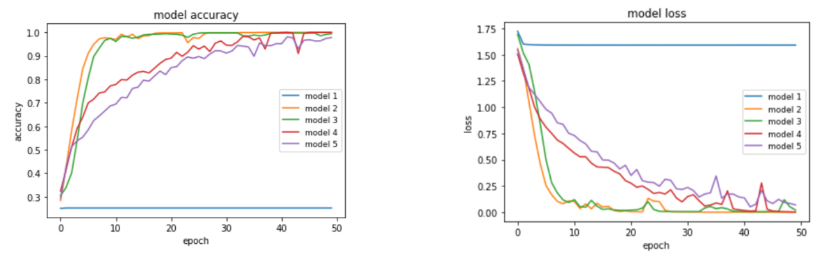


Figure 4: Learning Curve

3.2 Build a Model Use Output as an Input Channel

Summary:

- We are curious if put the output as an input, what is the simplest model we can build to achieve 100 percent accuracy.
- The true output from the images, which are the classes every image belongs to is considered the 4th channel on the top of the 3 colors. The 4th channel is labeled as 0,1,2,3,4, then divided by 10 for scaling purpose.
- As seen from the model structure table, only one convolution layer and 4 filters with size 3x3 are used to build the model, this is sufficient to achieve 100 percent accuracy, model achieved over 90 percent accuracy in 6 epochs and achieved 100 percent accuracy in 34 epochs.

Model Output as Input Learning Curve - Accuracy and Loss:

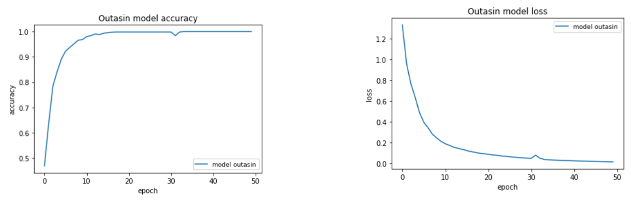


Figure 5: Learning Curve

4 Split Data and Evaluate on Test Set

4.1 Training Models

At this phase, there were 4 models trained on the training set. The Early-stop method was used on the validation set, also the model's performances were evaluated on the test set. The patience of early stop is set to 15.

Model 1 Structure: 2 Convolution Layer with 8 filters for each layer

Model 1 Learning History for Train and Validation Sets:

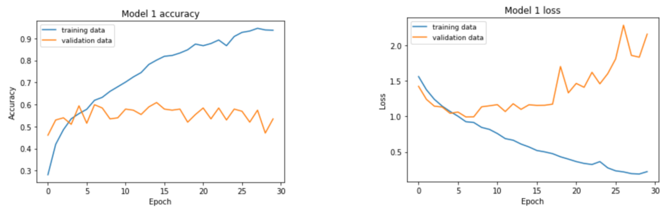


Figure 6: Model 1 Learning History

Model 2 Structure: 2 Convolution Layers with 16 filters for each layer

Model 2 Learning History for Train and Validation Sets

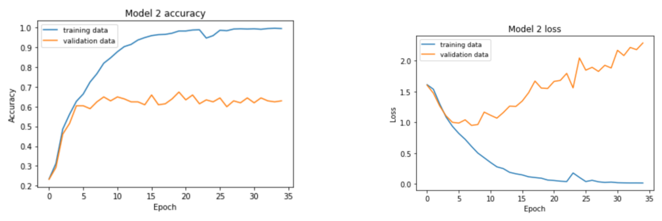


Figure 7: Model 2 Learning History

Model 3 Structure: 3 Convolution Layers with 32 filters for each layer

Model 3 Learning History for Train and Validation Sets

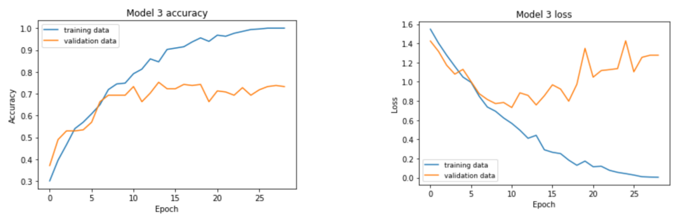


Figure 8: Model 3 Learning History

Model 4 Structure: 4 Convolution Layers with 64 filters for each layer

Model 4 Learning History for Train and Validation Sets

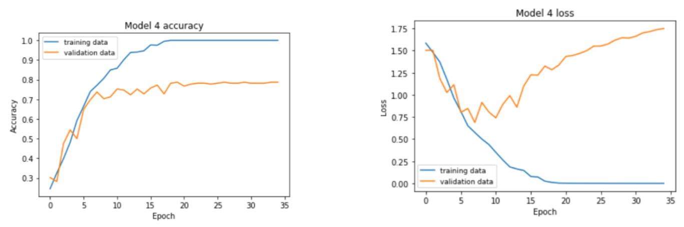


Figure 9: Model 4 Learning History

4.2 Test Accuracy Changes for Different Filter and Layer Number:

Model	Filter #	Layer	Test Accuracy
1	8	2	59.45%
2	16	2	66.54%
3	32	3	75.98%
4	64	4	76.38%

Summary for Splitting data to Train the Model

- Model 1 includes 2 layers and each layer has 8 filters. Test accuracy achieved 59.45 Percent. As we can see from the learning history, validation accuracy achieved to the highest point (about 60 percent) within 16 epochs, after 15 epochs more, there is no increase therefore, model stopped training.

- Model 2 includes 2 layers and each layer has 16 filters.Test accuracy achieved 66.54 Percent. As we can see from the learning history, validation accuracy achieved to the highest point (about 67 percent) within 20 epochs, after 15 epochs more, there is no increase therefore, model stopped training.

- Model 3 includes 3 layers and each layer has 32 filters.Test accuracy achieved 75.98 Percent. As we can see from the learning history, validation accuracy achieved to the highest point (about 75 percent) within 15 epochs, after 15 epochs more, there is no increase therefore, model stopped training.

- Model 4 includes 4 layers and each layer has 64 filters.Test accuracy achieved 76.38 Percent. As we can see from the learning history, validation accuracy achieved to the highest point (about 77 percent) within 21 epochs, after 15 epochs more, there is no increase therefore, model stopped training. This the model gives us the best performance.
- Other models with higher layers and filters were tried too, the results either gave us crazy overfitting or performed similar with Model 4.
- Testing images were randomly left out from the training and validation images, we can say that test accuracy 76.38 percent is reasonably robust, since if we just predict the image classes without building any CNN model, the baseline accuracy is only 20 percent.

5 Train CNN Model with Data Augmentation

At this phase, I used the best model from previous step - Model 4 to train the images applying data augmentation techniques on training data, validation data and test data. Data augmentation is created by the data generator. There are 3 different augmentation techniques applied for comparison purposes.

Data Augmentation Techniques

Tech	Rotation	W Shift	H Shift	Shear	Zoom	Flip	Fill	Test Acc
1	10	0.1	0.1	0.1	0.1	FALSE	Nearest	81.89%
2	20	0.2	0.2	0.2	0.2	FALSE	Nearest	85.83%
3	30	0.3	0.3	0.3	0.3	TRUE	Nearest	85.83%

Model 4 Learning History Applying Augmentation Technique 1

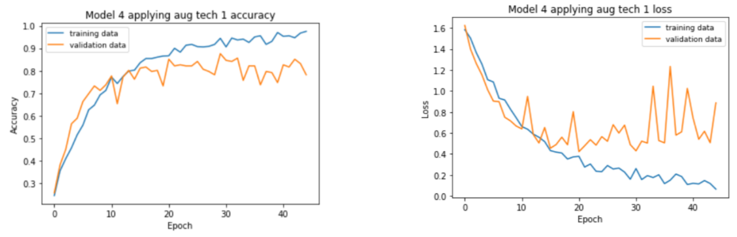


Figure 10: Model 4 Learning History Applying Augmentation Technique 1

Model 4 Learning History Applying Augmentation Technique 2

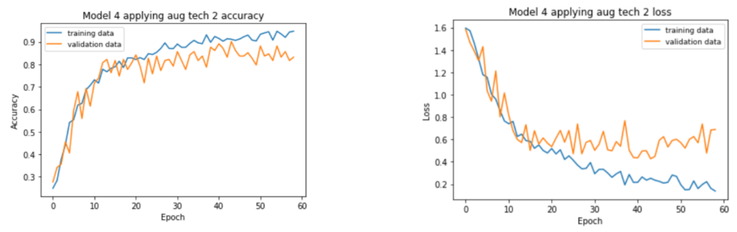


Figure 11: Model 4 Learning History Applying Augmentation Technique 2

Model 4 Learning History Applying Augmentation Technique 3

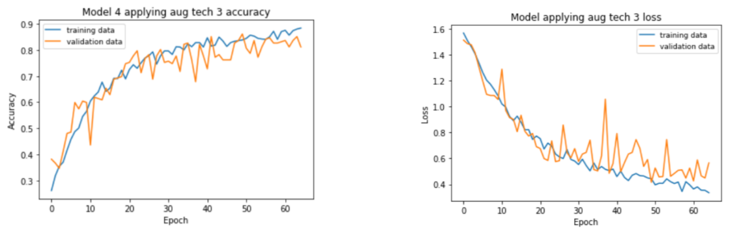


Figure 12: Model 4 Learning History Applying Augmentation Technique 3

Summary of Using Data Augmentation

- Model 4 with data augmentation technique 1 achieved 81.89 percent test accuracy.
- Model 4 with data augmentation technique 2 achieved 85.83 percent test accuracy.
- Model 4 with data augmentation technique 3 achieved 85.83 percent test accuracy.
- We can see from the result, if we do some augmentation techniques, model performance will be improved compared with the original Model 4, Model 4 achieved 76.38 percent test accuracy while after applying augmentation technique 1, the test accuracy increased over 5 percent. If we increase the rotations, make the shifts range wider and allow generator to flip image with a nearest filling, Model 4 with augmentation technique 2 and 3 can both finally achieved to 85.83 percent which is over 7 percent better than original Model 4.

6 Effects of Regularization

At this phase, the best model from last section was used for studying the effects of regularization techniques. There are 3 regularization techniques applied: L2, dropout and batch normalization. Let us see how the model performs after adding in different regularization in the model for preventing overfitting.

Regularization Techniques

Regularization Technique	Test Accuracy
Weight Regularization (L2)	85.43%
Dropout	82.28%
Batch Normalization	79.92%

Model 4 Learning History using L2 regularization

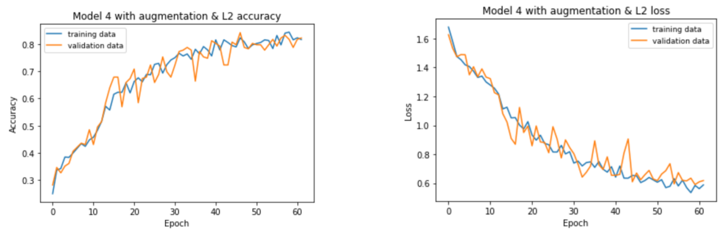


Figure 13: Model 4 Learning History using L2 regularization

Model 4 Learning History using Dropout regularization

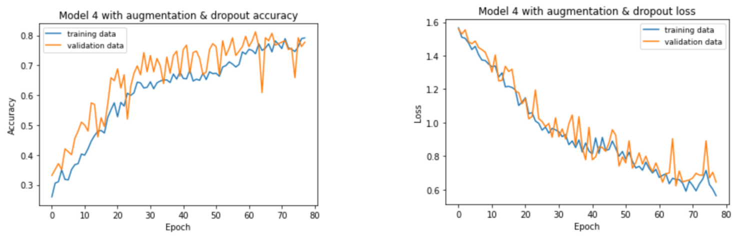


Figure 14: Model 4 Learning History using Dropout regularization

Model 4 Learning History using Batch Normalization regularization

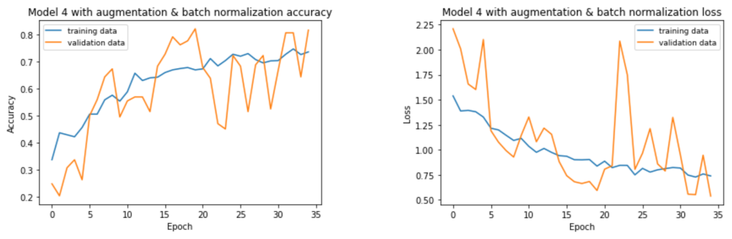


Figure 15: Model 4 Learning History using batch normalization regularization

Summary of Regularization Effects

- Model 4 with data augmentation technique 3 applying L2 regularization achieved 85.43 percent test accuracy.
- Model 4 with data augmentation technique 3 applying Dropout regularization achieved 82.28 percent test accuracy.
- Model 4 with data augmentation technique 3 applying Batch Normalization regularization achieved 79.92 percent test accuracy.
- We can see from the result, if we apply L2 regularization technique, model performs similar to Model 4 with augmentation technique 3. The model achieved 85.43 percent test accuracy. If we applied dropout regularization on the same layers, the model performs slightly worse than using L2, which is 82.38 percent test accuracy. If we used batch normalization, the model performance is worse than both using dropout and L2 techniques.

7 Use Pre-trained Models

7.1 Use Pre-trained Model VGG-16

At this phase, the feature extraction technique was used to take the convolutional base of pre-trained model VGG-16, then ran the new data through it and trained a new classifier on top of the output.
Below is the learning curve for training data and validation data. The test accuracy achieved about **92** percent, compared with previous best model about **85** percent accuracy in proved **7** percentage, which is a really good increase.

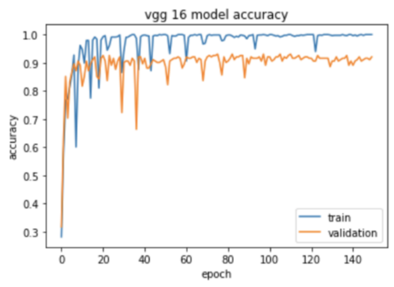


Figure 16: Model VGG-16 Learning Curve

7.2 Use ResNet Architecture

At this phase, the Deep Learning architecture ResNet was also used to train the model on food cuisine image data. ResNet Model 1 only has 1 loop of the ResNet architecture while ResNet Model 2 has 2 loops of the ResNet architecture.

Below is the ResNet Model 1 architecture:

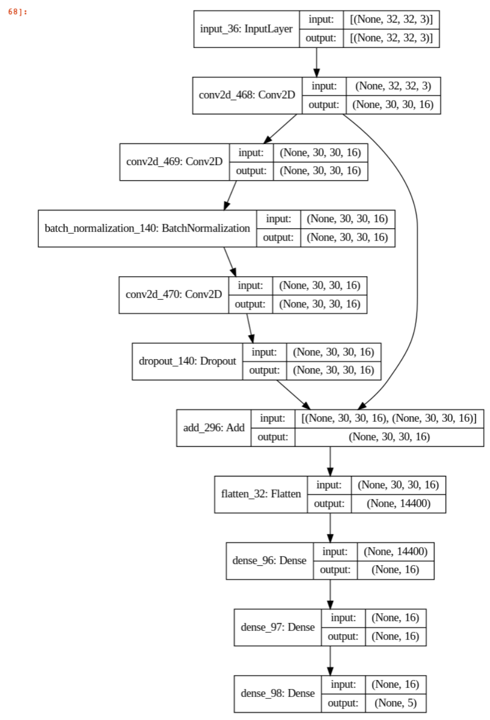

68]:
input_36: InputLayer | input: [(None, 32, 32, 3)] | output: [(None, 32, 32, 3)]
conv2d_468: Conv2D | input: (None, 32, 32, 3) | output: (None, 30, 30, 16)
conv2d_469: Conv2D | input: (None, 30, 30, 16) | output: (None, 30, 30, 16)
batch_normalization_140: BatchNormalization | input: (None, 30, 30, 16) | output: (None, 30, 30, 16)
conv2d_470: Conv2D | input: (None, 30, 30, 16) | output: (None, 30, 30, 16)
dropout_140: Dropout | input: (None, 30, 30, 16) | output: (None, 30, 30, 16)
add_296: Add | input: [(None, 30, 30, 16), (None, 30, 30, 16)] | output: (None, 30, 30, 16)
flatten_32: Flatten | input: (None, 30, 30, 16) | output: (None, 14400)
dense_96: Dense | input: (None, 14400) | output: (None, 16)
dense_97: Dense | input: (None, 16) | output: (None, 16)
dense_98: Dense | input: (None, 16) | output: (None, 5)

Figure 17: Model ResNet Structure

Below is the ResNet Model 1 and ResNet Model 2 learning curves for training data and validation data. The test accuracy for ResNet Model 1 achieved about **63** percent and for ResNet Model 2 achieved about **47**, compared with previous best model about **85** percent accuracy and VGG-16 pre-trained model's accuracy **92**, we can see ResNet model 1 performed worse than VGG-16 and previous best model.

ResNet Model 1 Learning Curve

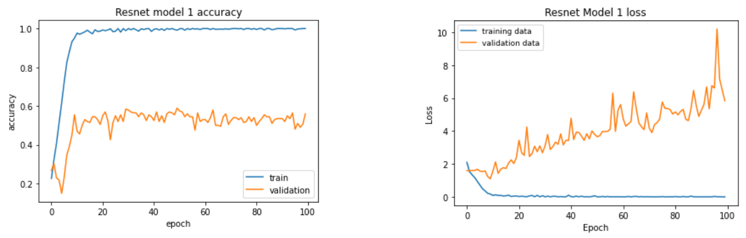


Figure 18: ResNet Model 1 Learning Curve

ResNet Model 2 Learning Curve

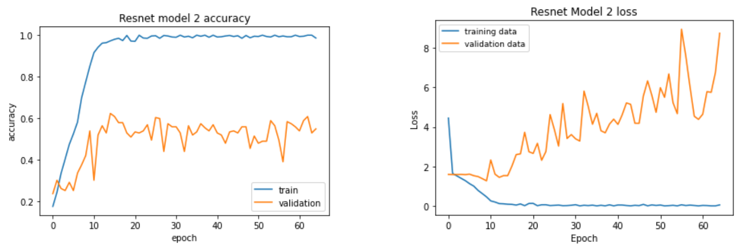


Figure 19: ResNet Model 2 Learning Curve

8 Conclusion

The goal of this project was the construction of a convolution neural networks and to demonstrate how a CNN model was trained and predict the food classification. An initial model was developed and capable of overfitting the data without splitting the data, which means evaluate the model on the images. The hyperparameters were adjusted to yield the minimal loss on all images data. Therefore, the model must be overfitting, after adjustments to the hyperparameters. After splitting the data to training, validation and testing date sets and tuned the hyperparameters based on the validation loss, the model's performance(accuracy) decreased on the testing data. The effects of data augmentation and the regularization on preventing overfitting were both observed since the performance on testing data significantly increased compared with the model without using these 2 techniques. From around 76 percentage to 85 percentage. For regularization, the L2 helped to increase the model's performance most compared with other 2 regularization techniques: dropout and batch normalization. The use of a pretrained model, VGG-16 stood out on the testing performance indicates that there is more space for us to adjust the model and improve the performance and the data is suitable for a CNN model. ResNet architecture was used to build a deep CNN model to train, however, the performance was actually worse than original models that were trained from previous steps. Overall, the CNN models performed well, at around 85 percentage accuracy achieved on testing data with preventing overfitting techniques. With pretraining VGG-16, the performance on the testing data achieved 92 percent. Compared to 20 percent accuracy, which is the baseline for 5 classes prediction problem, the CNN model in this paper performed significantly better, which indicates our CNN model was trained quiet well.