Homework 3

Kelsey Hawkins

Introduction

Food is a staple in everyones diet, and its important to know which food you are consuming, as some are hard to tell apart. This model takes images of 36 different types of foods, processes them so they are suitable for input to a deep learning convolutaional model, and outputs predictions. With these predictions, sorting through a basket of food will be easy and identifying specific foods will be accessable to all. This model utilizes an image classificaion solution, starting with a self-made convolutional neural network, then moving to a transfer learning approach.

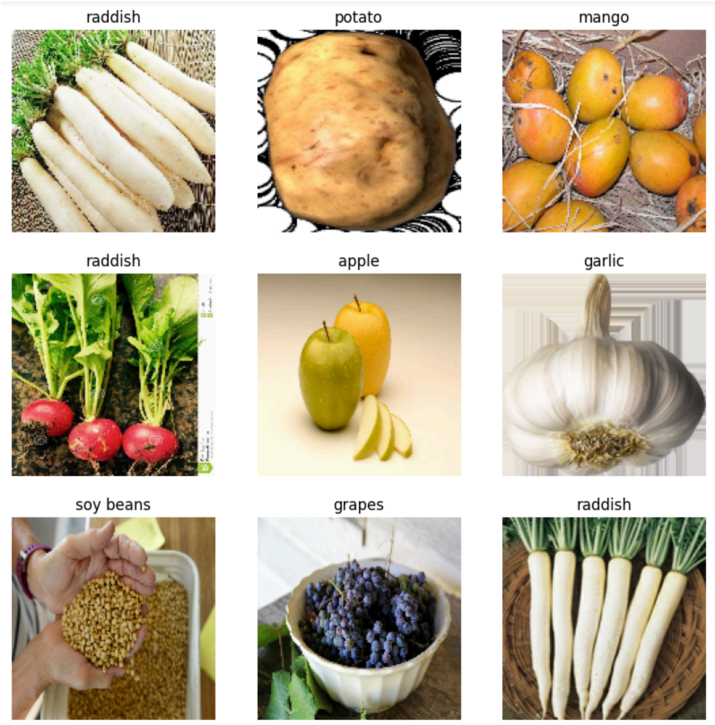

Figure 1: Visualizing Images from Dataset

Analysis

The first exploratory item I did was visualizing a few images from the dataset. Not only did I want to ensure the data accuracy, but I wanted to see what resolution I was working with. The original images were huge (542x542x3), so I needed to make them smaller for the model as a 256x256x3 standardized size. It was hard to do any summary statistics or heatmaps, etc. due to the data being flattened images so no EDA would have been useful in this case, other than examining the classes for imbalances.

The dataset from Kaggle came in folders of the type of food, so I was able to pull all 36 food item labels that I was working with. The foods range from Apples to turnips, all images were resized to a 256x256x3 shape for inputting to the model. Each class had equal numbers of images(10), so there were no imbalances. The images were then one-hot-encoded for input to the model.

Methods

Beginning this assignment, I took a simple model and built them up into deeper models while changing items to maximize the model's feature extractions and metrics. By changing one hyperparameter at a time, I am able to grasp which ones are benefitial to the model, and will solve the image classification problem the best with the least overfitting possible. I concentrated on changing the layer and general architecture, regularizations, activation functions, dropout, optimizer function, loss functions, and callbacks.

The first model I created is a Deep Feed-Forward Convolutaional Neural Network. I started with a simple model, then built my way up from there. The final model has 15 layers, not including data augmentation. I chose to include data augmentation, specifically flips, zooms, and rotations, to enhance my training data for the model to train on and reduce overfitting to the ten images that originally exist in the dataset. This will help the model learn the feature better and not suffer from translational invariance nor bias towards the training data. The body of the model has five convolutaional layers, five max pooling layers, and three dense layers. The last dense layer has a softmax activation function for predicted probabilities per class, and all other convolutional layers have relu activation functions. I utilized a loss function of Categorical Crossentropy and Adam as my optimizer, while validating on categorical accuracy. Callbacks including early stopping, and model checkpoint. This model was evaluated by printing out the training and validation scores over the epochs, gradcam, as well as accuracy and F1 scores for quantifiable analysis.

The second model i created is a transfer learning approach, utilizing the VGG-16 model from Keras. I kept the top 15 layers frozen so I could leave the top convolutional block trainable for fine tuning. I then created a feed forward network for the head as I wanted to tune them myself. This includes seven layers, consisting of one flatten and three dense layers. The last

dense layer has a softmax activation function for predicted probabilities per class, and all others have relu. I utilized the loss function of Categorical Crossentopy and Adam as my optimizer, while validating on categorical accuracy. Callbacks including early stopping, model checkpoint. This model was also evaluated by printing out training and validation scores over the epochs, gradCam, as well as accuracy and F1 scores for quantifiable analysis to compare to the other model.

In choosing those hyperparameters, I tested each against other parameters that are commonly used for image classification problems. With a trial and error approach, I was able to find the best hyper parameters to suit this problem and maximize my models. By creating and analyzing two different image classification models, I am able to prove if transfer learning is needed or if homemade models will work just as well.

Results

Given these two models, they performed quite differently. It is clear that transfer learning may theorhetically be benefitial in this image classification task, however, it suffered from overfitting even when steps were taken to prevent it. The homemade model had better scores and output plots created, which will be discussed.

Due to the VGG-16 model being pre-trained on the imagenet dataset for the weights, it is very good in classifying food, in theory. In my attempts to change the hyperparameters listed before, I was able to find a middleground and optimize my model as best as I could. My model outputted multiple metrics included Accuracy, F1, and visualizations of training epochs, with some predictions to visualize them as well.

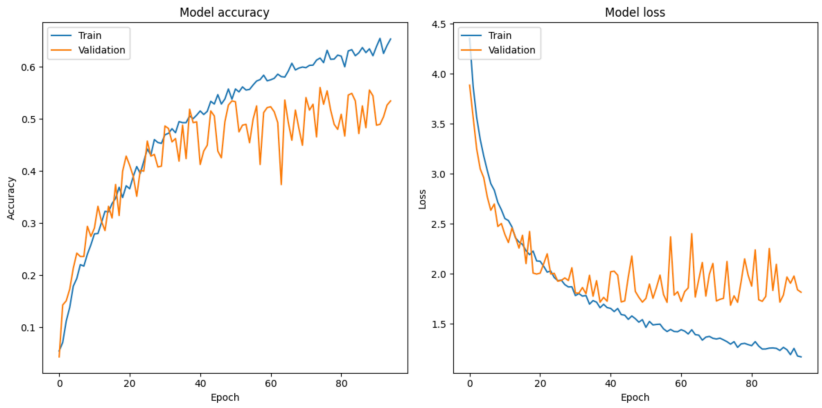

Figure 2: Deep Learning Plot - Train & Validation over Epochs

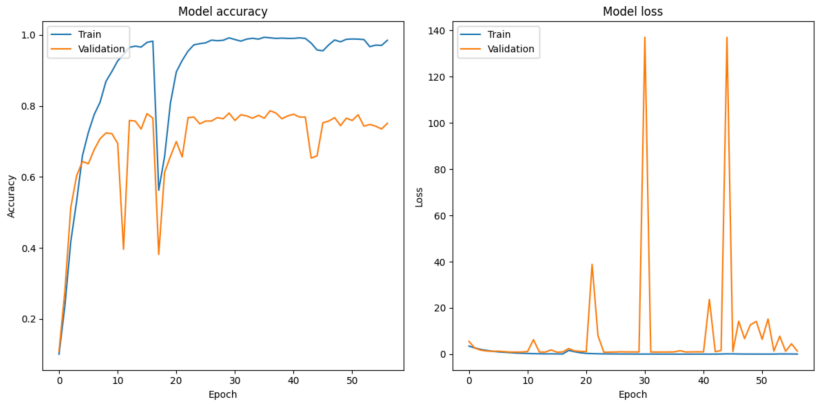

Figure 3: Transfer Learning Plot - Train & Validation over Epochs

From these two plots, its clear that the transfer learning model suffered from overfitting, but the homemade model did not. Starting with the homemade model, the early stopping ended around epoch 100, and the train accuracy showed a constant upwards trend, but the validation accuracy had huge variations after epoch 20, which means the model may have had trouble generalizing to the data further into training. Since the two metrics go from convergence to being about 20% apart, this means the model may have trained for too long due to learning rate being too big, or the model was finding patterns in the training data. For the loss plot, the model is following the same trend, which is clear towards the end of the epochs where the two lines are far apart. The training loss had a perfect downward slope, while the valiation loss had unpredictable movements with huge changes. This is the same issue from the accuracy plot, the model suffered towards the end of training. The final metric outputs converged at a acceptable level which shows the model is training to the data well, and able to predict images into the correct class with confidence.

In the transfer learning plot, it is much worse than the homemade model. There is clear overfitting and issues with the validation metrics for both accuracy and loss. It is concerning that the loss graph has values bouncing around all the way up to 140% around epoch 20. This could mean the model is suffering from not generalizing to the data as well as making huge steps around finding the best metrics calculating the negative gradients. Due to the dropout layers, the model may have lost a important node and suffered grearly. From the workings of Adam as well as the learning rate, there may be issues with both of these when it comes to the transfer learning model. There may have been huge jumps in metrics due to the exploding gradient problem, and the fixed learning rate not helping the fine-tuning nature of transfer learning. The model shows a concerning trend of the training loss remaining unchanged while the validation loss changes drastically. Unfortunately, many steps were taken to improve the model but it did not show success.

In the table below, we can see the training and testing accuracy and F1 scores. Its clear there is overfitting present in the transfer model as the train metrics are larger than the test metrics

for both F1 and accuracy. Both train and test scores are about 10% away from eachother, and tying it back to the previous graph, it makes sense why they are lower as well as far apart from eachother. This may have been caused by the learning rate being constant through training, or the model finding patterns in the training dataset and suffering later in the training epochs through training.

In conclusion, the transfer learning model was overfit and shows that the model struggled to generalize to the training data, and failed to predict unseen data correctly. As seen in Figure 4, we can see a few images taken from the test dataset and each morel predicted upon the images.

Table 1: Model Output Metrics

Model	Train F1	Test F1	Train Accuracy	Test Accuracy
Homemade	83.66%	94.44%	84.61%	91.66%
Transfer	81.84%	80.55%	83.33%	75.00%

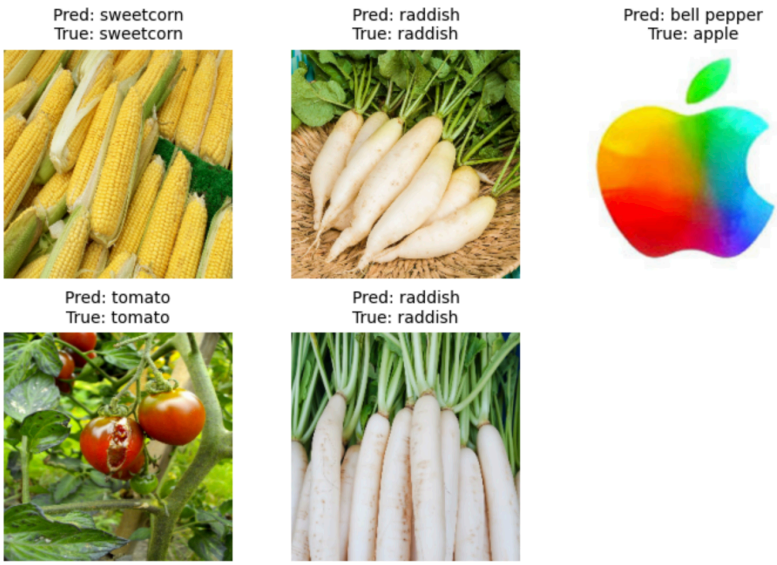

Figure 4: Homemade Model Predictions on Images

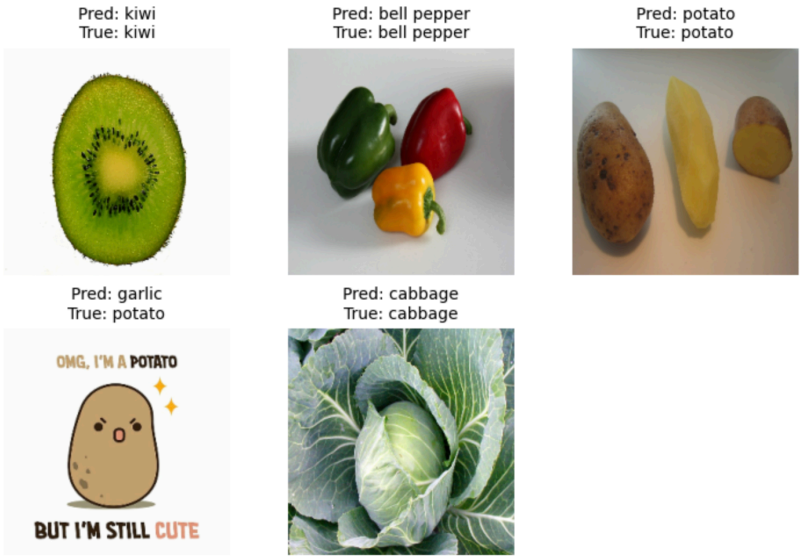

Figure 5: Transfer Learning Model Predictions on Images

Reflection

In this assignment, I did not come across many struggles or problems with the code or analysis. It was very interesting to look into the architecture and math behind the models to prove why one model did better than the other. Since I am also in a higher level class that teaches us computer vision, its very interesting to do the same concepts in a different class, but knowing the same materials.