

**School of InfoComm Technology**

**Deep Learning Assignment**

Diploma in DS / FI / IT

Oct 2022 Semester

**ASSIGNMENT 1**

(30% of DL Module)

**Submission Deadline:**

**Presentation: 18th Dec 2022 11:59PM**

**Report and Code: 18th Dec 2022 11:59PM**

|  |  |  |
| --- | --- | --- |
| **Tutorial Group** | **:** | **T01 / T02** |
| **Student Name** | **:** | Loke Kai Fong |
| **Student Number** | **:** | S10204160J |

**Penalty for late submission:**

10% of the marks will be deducted every calendar day after the deadline.

**NO** submission will be accepted after 24th Dec 2022, 11:59PM.

**Table of Contents**

[**1 Overview 3**](#_Toc122295038)

[**2 Data Pre-processing & Data Loading 4**](#_Toc122295039)

[**2.1 Data Pre-processing 4**](#_Toc122295040)

[**2.2 Data Loading 5**](#_Toc122295041)

[**3 Developing Image Classification Model (From Scratch) 5**](#_Toc122295042)

[**3.1 Model 1a (Baseline Model) 5**](#_Toc122295043)

[**3.2 Model 1b 7**](#_Toc122295044)

[**3.3 Model 1c 8**](#_Toc122295045)

[**3.4 Model 1d 9**](#_Toc122295046)

[**3.5 Model 1e 10**](#_Toc122295047)

[**3.6 Model 1f (Best Model 1) 12**](#_Toc122295048)

[**3.7 Model 1g 14**](#_Toc122295049)

[**4 Developing Image Classification Model (Pre-Trained) 15**](#_Toc122295050)

[**4.1 Model 2a 15**](#_Toc122295051)

[**4.2 Model 2b 16**](#_Toc122295052)

[**4.3 Model 2c 18**](#_Toc122295053)

[**4.4 Model 2d 19**](#_Toc122295054)

[**4.5 Model 2e 21**](#_Toc122295055)

[**4.6 Model 2f (Best Model 2) 22**](#_Toc122295056)

[**5 Evaluation on Test Images 24**](#_Toc122295057)

[**6 Prediction on Real-Life Images 25**](#_Toc122295058)

[**7 Summary & Further Improvements 28**](#_Toc122295059)

[**8 References 30**](#_Toc122295060)

# Overview

This report aims to cover the processes for building a deep learning model, as well as to evaluate the final model’s performance on real-life images.

The objective is to test and build an image classification model that is able to correctly identify and classify the images that it has been trained on. We will be utilizing an image dataset from Kaggle that consists of food images, with 1000 images per type of food. For this model, it will be trained to accurately identify and classify the 10 types of food that have been assigned to me. These 10 foods are,

* Beef Carpaccio
* Breakfast Burrito
* Churros
* Crab Cakes
* Fish and Chips
* Omelette
* Pancakes
* Panna Cotta
* Poutine
* Takoyaki

The model building process will involve building 2 deep learning models for image classification.

The first model will be a Convolutional Neural Network model trained and optimized from scratch. To fully optimize the model, various parameters must be adjusted and tested to create a model that is able to learn the data well. This will involve adjusting batch sizes, epochs, and learning rates. Methods such as configuring optimizers, adding regularization and dropout, as well as data augmentation must be done to further improve the model’s accuracy.

The second model will utilize transfer learning to build the deep learning model. Transfer learning will involve importing and loading in a pre-trained CNN model. Pre-trained models are deep learning models that have already been trained on previous data, typically on ImageNet. ImageNet is an image dataset that contains an average of 1000 images for each “synset” or “synonym set” in the WordNet database, which is a database containing English nouns, verbs, adjectives, and adverbs grouped into sets of synonyms that explain a particular concept (“synsets”) (in-text). These pre-trained models re-purposed and will be used as the starting point for the model building process. Freezing and unfreezing the layers in the pre-trained model will help to optimize the accuracy of the final model.

# Data Pre-processing & Data Loading

## Data Pre-processing

Before building the models, we have to first perform some data pre-processing steps and prepare the data.

Data Pre-processing is an important step as we have to ensure that we are using the correct data and ensure that the data we are using is suitable for modelling. This will involve extracting the necessary data from the dataset, as well as splitting the data into training, validation, and testing sets.

Splitting the data into training, validation, and testing sets is important as we do not want our model to train and learn the images in the testing or validation datasets. The training set is used to train the data, allowing it to learn the parameters of the images, to perform prediction. The testing dataset will allow us to simulate how the model would perform on new unseen data, which would let us properly evaluate the accuracy and usefulness of the model’s predictions. The validation dataset helps us to validate our model performance during training, allowing us to tune the model’s parameters and configurations.

To extract the 10 assigned food images from the dataset, the data is loaded into the “Image\_Preprocessing” notebook. The name list of assigned foods is appended from a text file to an array in the code, which will be used to extract the folders containing the images of that specific food. The images are then separated into training, validation, and testing folders. We will be utilizing 750 images of each food for training, 200 for validation, and 50 for testing.

## Data Loading

After data preprocessing, the training, validation, and testing datasets are loaded into the main jupyter notebook to start the model building process. Next, the necessary libraries and keras must be loaded into the notebook for model building. An additional line of code “print(tf.config.list\_physical\_devices('GPU'))” was used to check if tensorflow is utilizing the system’s GPU, which can affect the speed of training. Lastly, the images have to resized to 150 x 150 as neural networks require all images to be of equal size.

# Developing Image Classification Model (From Scratch)

## Model 1a (Baseline Model)

The first model we build should be a simple baseline model. A baseline model is needed as a benchmark to see how the data is fitting onto the model. The baseline model should be very simple as we want to determine the starting point of our model building. This will allow us to scale up the model until it begins to overfit, which lets us identify at which point the model is optimal. From there, we can determine the necessary changes and adjustments needed to optimize the model parameters, based on the validation curves plotted for accuracy and loss.

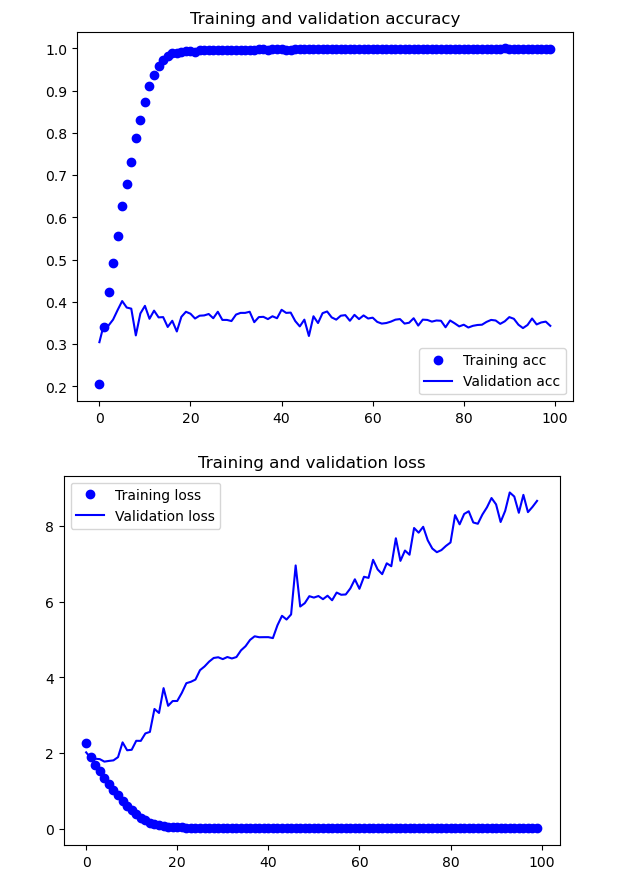
The first model uses 2 conv2D layers with ‘ReLU’ activation function as hidden layers. The ReLU or Rectified Linear Unit function is preferred as the activation function for hidden layers as it has a faster computational and convergence speed compared to other activation functions. Furthermore, it eliminates the vanishing gradient problem, which can be detrimental for our model’s performance.

The output layer uses a ‘softmax’ function with 10 outputs as we are building a multi-classification model to classify 10 foods. As opposed to the sigmoid function which is used for binary classification problems. Max pooling, Flatten, and Dense layers were also used in the model.

Max pooling is used to reduce the dimensions of the feature maps during the training process. This effectively reduces the number of parameters and computations performed in the network, thus reducing overfitting, and preventing the GPU memory from running out while training. The Flatten layer is needed to convert the image array with multiple dimensions into a single array to be compiled in the dense layer. The dense layer is a fully connected layer needed for image classification models, as it allows the flattened feature map to be pass through the neural network into the output layer to obtain the predicted classes.

For model 1a, the model begins to overfit around the 6-8 epoch and the validation accuracy hovers around 30-35%. Which is a poor accuracy value for a deep learning model.

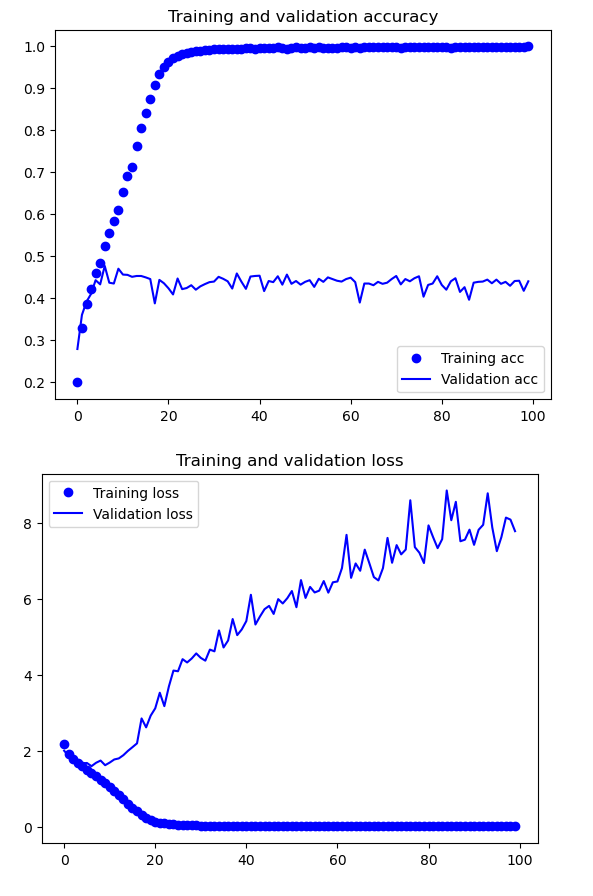
This is where we start to scale the model up. Accuracy may be further improved by adding more layers, implementing regularization techniques, trying different activation functions, making use of data augmentation, testing the model with different learning rates, and experimenting with different batch sizes and epochs. Also, using lesser epochs may reduce unnecessary learning.



## Model 1b

Model 1b aims to experiment with increasing the model’s complexity to improve the model’s training and allow it to understand the parameters of the data better. This will allow it to make more accurate predictions on the dataset.

Based on the validation curves and the model results, it can be observed that model 1b begins to overfit later than model 1a, around the 15-17 epoch. Moreover, Model 1b has a slight increase in validation accuracy at 44-46%. However, the increase in accuracy is insufficient. Model accuracy can be further improved by adjusting batch sizes and epochs in the next iteration.



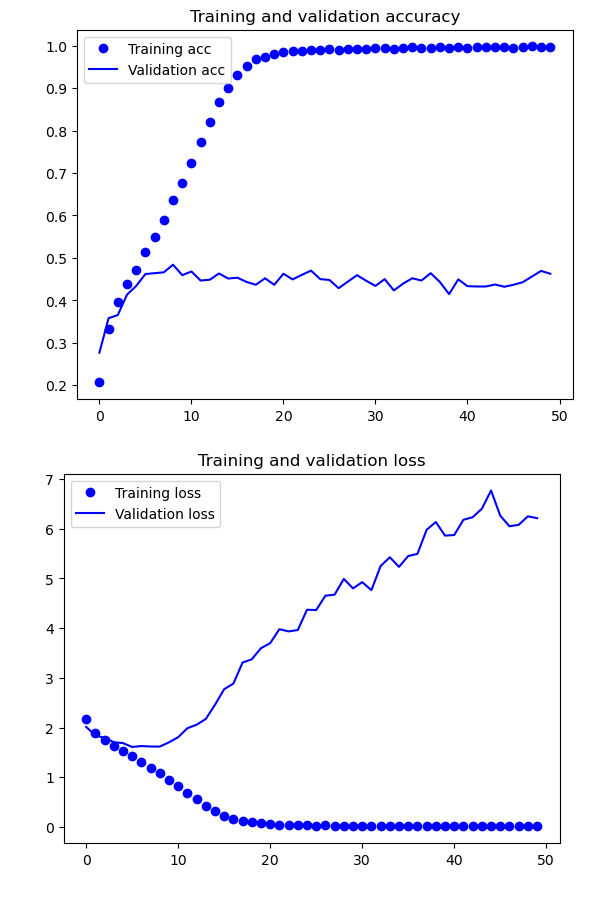
## Model 1c

This model focuses on fine-tuning the accuracy of the model by reducing the batch size in an attempt to improve accuracy and performance.

Batch sizes refers to the number of samples processed before the model is updated or in a single iteration. By reducing the batch size from 30 to 20, the model should be able to learn the parameters quicker, which reduces the computation time needed for training.

After running the model, Model 1c begins to overfit earlier than previous models. From the validation curves, overfitting seems to begin at around the 7-8 epoch. Moreover, the validation accuracy of the model does not seem to show any signs of improvement as it has stagnated at 45%-46%.

Adjusting the base model parameters does not seem to have a large impact on the model performance. However, we can still adjust other parameters that can improve the model. The next model will be tested with a different optimizer, from 'RMSprop' to 'Adam’.

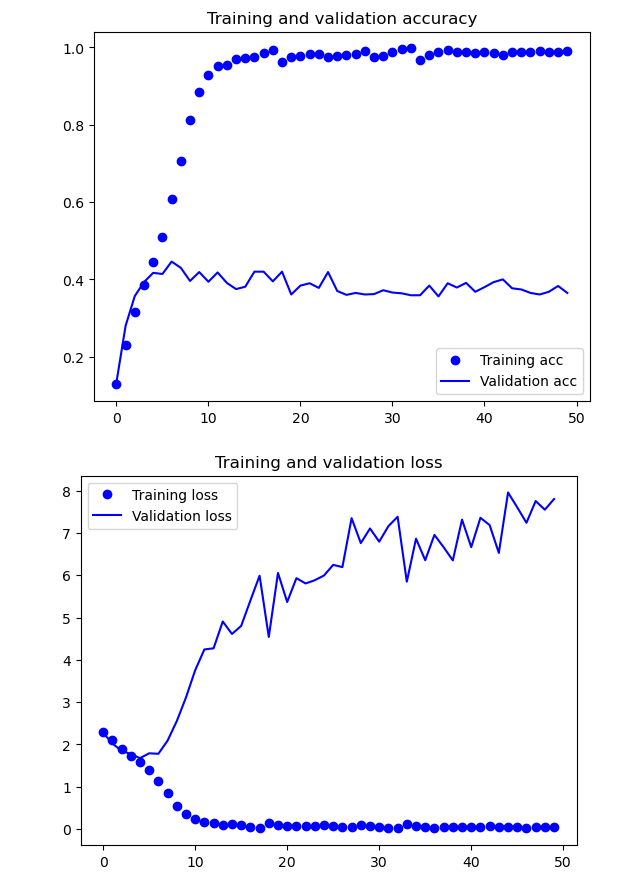


## Model 1d

This model utilizes the Adam optimizer instead of RMSProp. This is to test and evaluate the different optimizers on the model to see if they can improve the performance and accuracy of the model. The Adam optimizer is a optimization algorithm for deep learning models that combines the advantages of the Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). The Adam optimizer is said to have a faster computation time, require less parameters to tune and it can handle scattered gradients on noisy models.

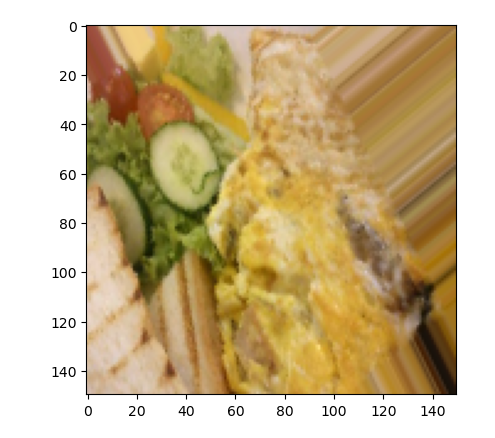
Validation accuracy of the model (1d) has decreased to 36%-38%, which indicates a poorer performance. However, this optimizer has only been tested on one instance of the model and the decrease in accuracy may not hold true when testing with models that have implemented data augmentation and regularization.

Again, it seems that adjusting the base model parameters does not seem to have a large impact on the model performance. To see further improvements in the accuracy of the model, Data Augmentation and Regularization techniques may have to be implemented. Data Augmentation in particular has to be tested as it effectively adds more data to the model, allowing it to perform better as it will have more data to learn and train on.



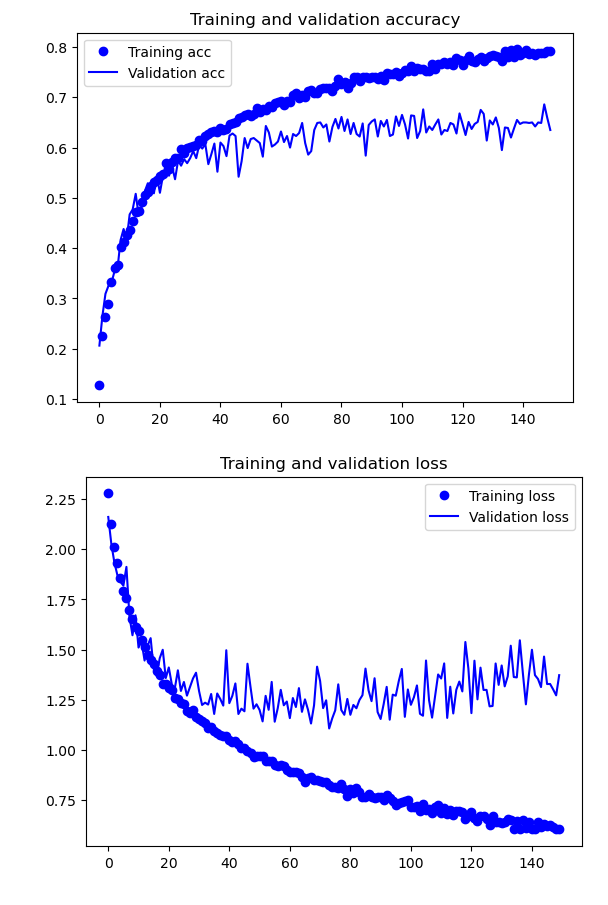
## Model 1e

This model will employ the use of data augmentation to artificially increase the amount of data. This is done by rescaling and performing minor transformations on the images such as, flipping, stretching, and rotating the images. These augmented images are seen as new images in the model, thus effectively increasing the amount of training data. Which will increase performance and accuracy of the model.



By applying implementing data augmentation to model, we are able to increase the accuracy of the model by a considerable amount, from 36%- 38% to 62%. This can be attributed to the higher amount of data used to train the model in model 1e obtained from data augmentation.

However, when observing the validation curves, it seems that the training accuracy did not reach 95% and above and has stagnated at around 62%. The model does not seem to be overfitting as well. This maybe a sign that the model isn’t complex enough. Thus, the next iteration of the model will focus on increasing the complexity by adding another conv2d layer or testing regularization and dropout methods.



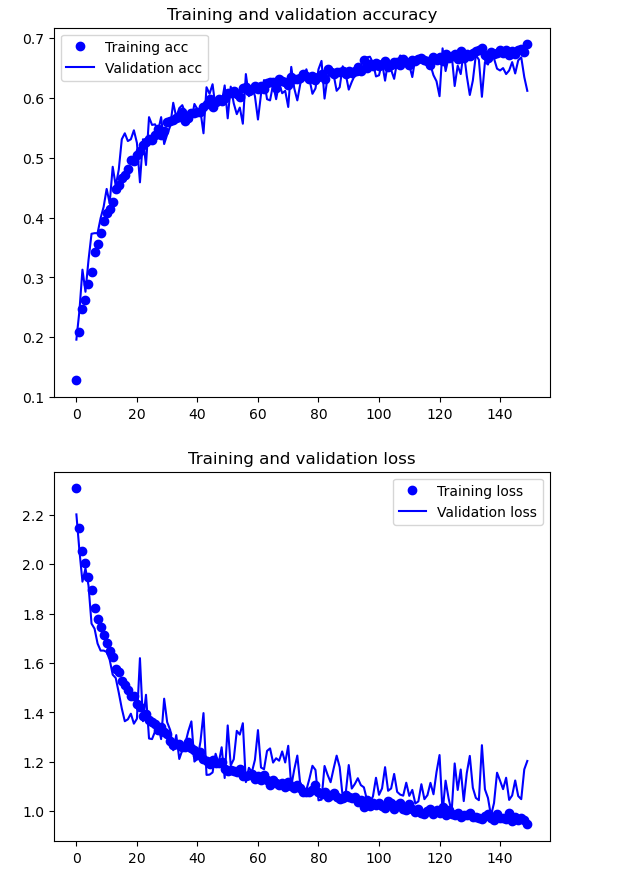
## Model 1f (Best Model 1)

This model aims to implement regularization techniques such as l2 regularization and adding dropout layers to reduce the complexity of the model. This will minimize the adjusted loss function and prevent the model from overfitting or underfitting.

By adding dropout layers and regularization, the overfitting present in the previous model (1e) has been eliminated from the model. Using regularization and dropout has reduced the complexity of the model, thus reducing overfitting. However, the accuracy of the model seems to be unchanged from the previous model (1e), as the accuracy seems to increase and stagnate around 65%. However, model 1f appears to have the highest accuracy with minimal overfitting among the models in model 1.

Another Conv2D layer was not added as when testing the model with an additional layer, the validation accuracy of model did not break past 15% even with 100 epochs.

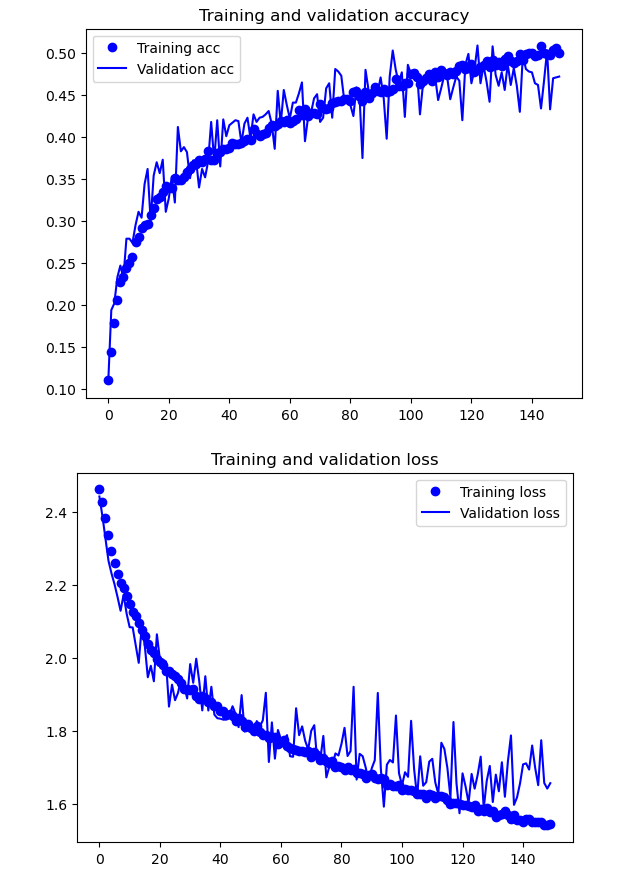
The next iteration of the model will test the same model with a different optimizer to determine which optimizer will be used for subsequent models.



## Model 1g

When building and testing the model with a different optimizer (RMSProp), the overfitting present in the validation curves is identical to the previous iteration (1f). This proves that using regularization and dropout will reduce overfitting in the model.

However, the validation accuracy of the model has decreased significantly. The accuracy has dropped from around 65% to around 49%. This is a significant drop in accuracy and the cause can be attributed to the optimizer used as it was the only change made from the previous iteration. This shows that the most suitable optimizer for this model is the “Adam” optimizer.



# Developing Image Classification Model (Pre-Trained)

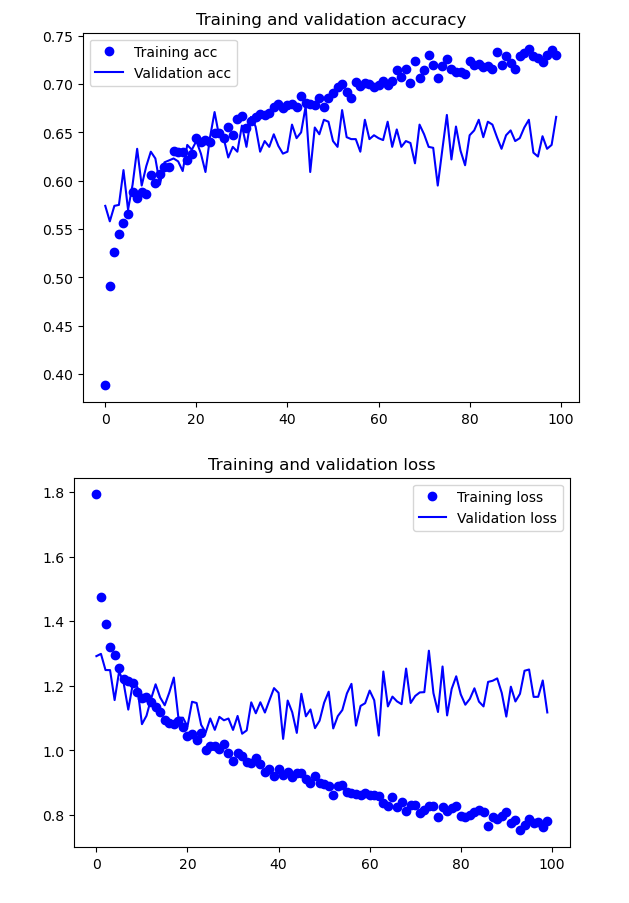
## Model 2a

The model 2a was tested with the pre-trained CNN VGG16, which stands for Visual Geometry Group. There are 2 versions of VGG, one with 16 convolutional layers and another with 19 convolutional layers. These versions are denoted by the number at the end of VGG. The VGG16 is trained on the ImageNet dataset and managed to achieve an accuracy of 92.7%, meaning it can classify numerous objects accurately.

The model was tested with similar parameters as before, with a simple flatten and dense layer of 256 neurons using the softmax function as the final layer and “Adam” as the optimizer. The VGG16 layer was frozen to prevent any changes to the weights of the model. Data augmentation was also kept, as increasing the amount of data with data augmentation has proved to help increase the validation accuracy greatly based on previous testing on model 1.

One change that was made was to reduce the epochs as it had been observed that overfitting typically does not begin later than 80 to 90 epochs. Therefore, in the interest of time and efficiency, the number of epochs was reduced from 150 to 100 and the steps per epochs was reduced to 375 to reduce computation time.

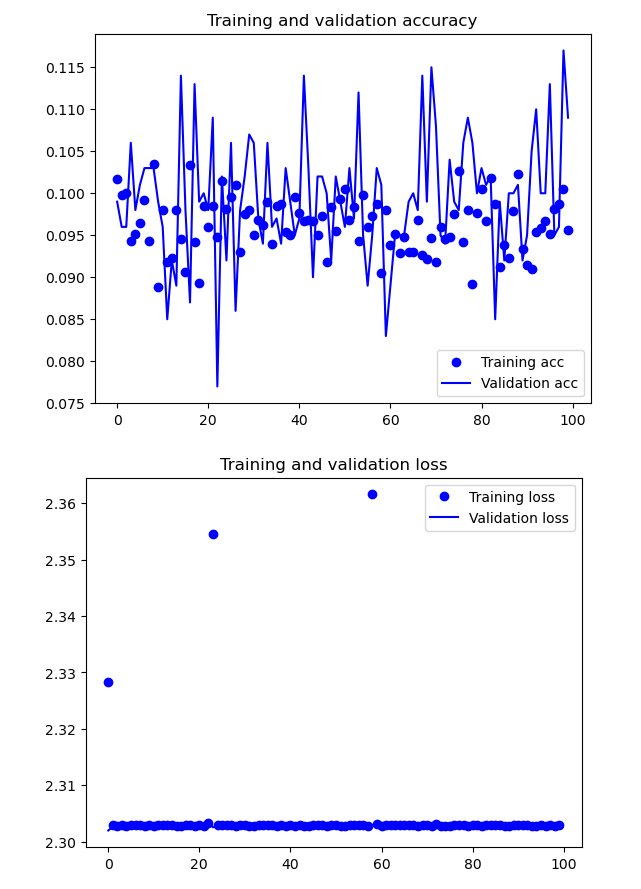
From the validation curves of the model, it appears the model begins to overfit on the data around the 60th epoch. The accuracy obtained from the model was similar to our final iteration for model 1 (f), staying around the 65% mark. To improve the accuracy further, unfreezing the layers may allow the model to learn some of the weights for our data, which may allow it to classify the images better.



## Model 2b

Unfreezing certain layers in a pre-trained model will allow the weight of the layer to be updated. Allowing it to be more fine-tuned on the dataset used.

When allowing the VGG16 model to learn the weights of the data, the model 2b became unable to learn anything meaningful from the images. The validation curves show the model was randomly classifying the images as the model is unable to learn the representations from the image data. The accuracy did not reach past 12% indicating extremely poor performance. This means that unfreezing the conv layer and allowing the weights of the VGG16 model to be updated based on the image data is ill-advised for this problem.

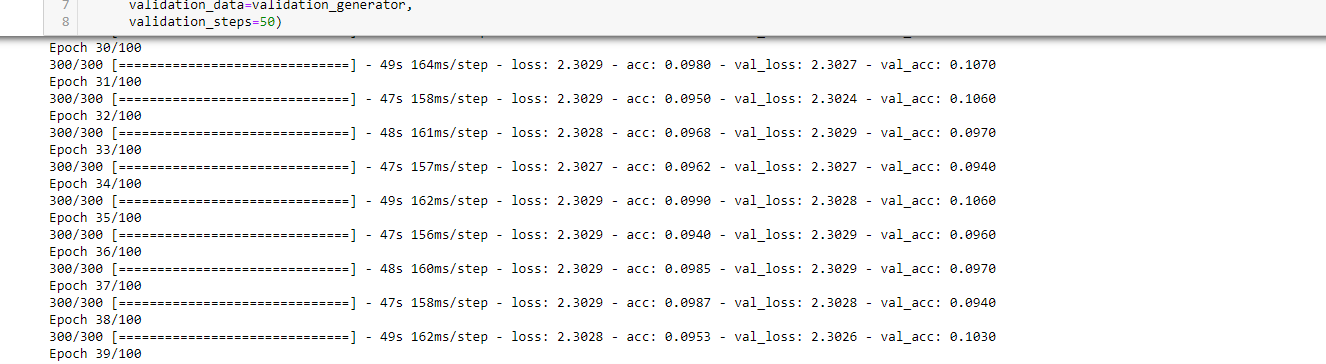


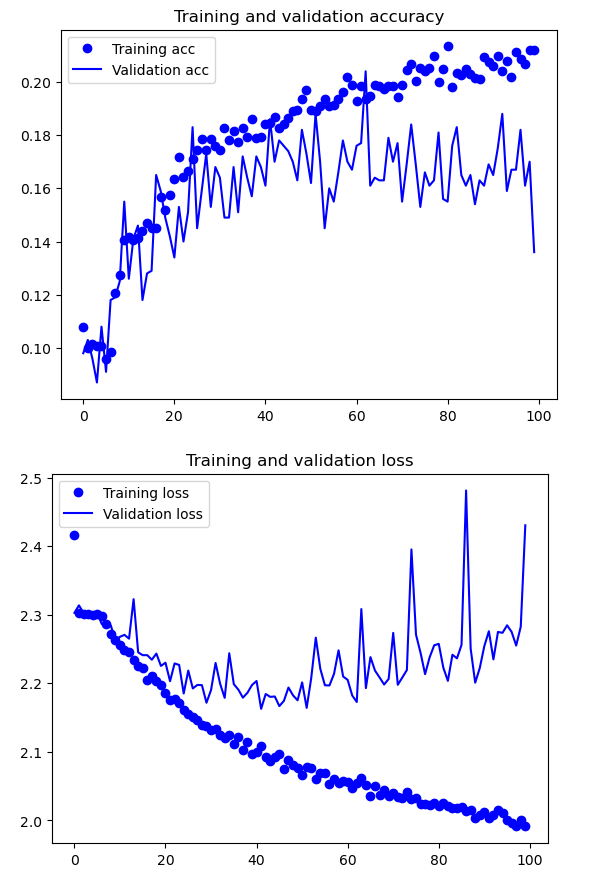
## Model 2c

The next pre-trained model that was tested is ResNet, which stands for Residual Neural Network. Unlike other neural network models, ResNet uses feedforward neural network with hundreds of layers, making it much deeper than the other neural networks used in the report. A feedforward neural network does not cycle or loop, meaning that there is no backward propagation occurring in the network. ResNet utilizes skip connections to effectively “skip” over layers. This is done to circumvent the problem with vanishing gradients and speed up training time.

The model 2c using the ResNet model was tested with the same parameters as the previous iterations of model 2. However, the number of steps per epoch was increased to its original value of 375. This was done as it was observed that the model would train better with more images within each epoch. Apart from this change, the model parameters were kept the same with data augmentation. The ResNet layer was also frozen to compare the changes after unfreezing.

However, when the model was trained with augmented data, the validation accuracy was extremely poor, around 9-11% at the 40-50th epoch. This led me to believe that the ResNet model may not work well with the augmented data of these images. Terminating the model and testing it without data augmentation did not improve the validation accuracy significantly as it only managed to peak at 19%, with overfitting beginning around the 50th epoch.



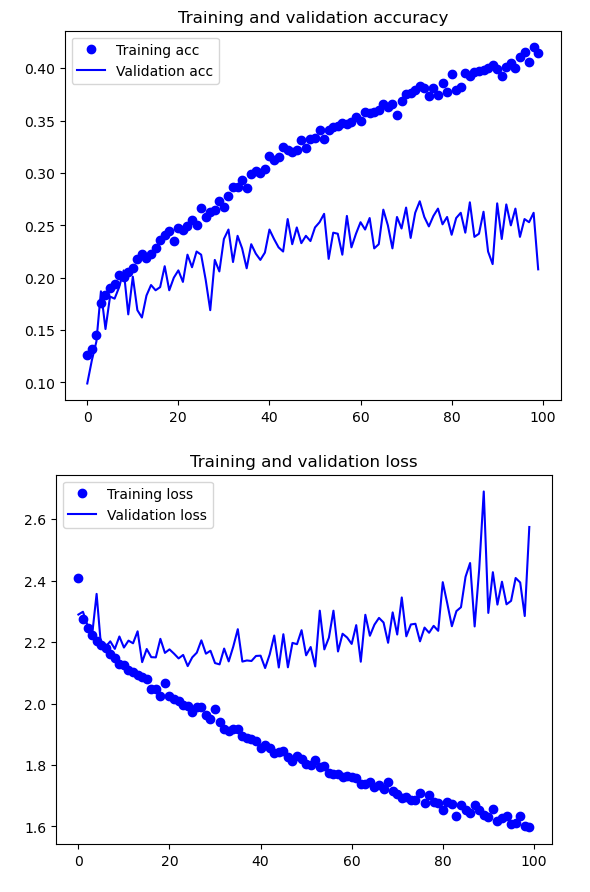


## Model 2d

This iteration of the model will be testing the same ResNet model with the ResNet layer unfrozen. This is to see if the model will be able to classify better if we allow it to learn the images and update the weights.

When testing the model 2d with the unfrozen ReNet layer, the model was able to learn and classify the images slightly better than the previous iteration where the ResNet layer was frozen. The model was able to achieve a validation accuracy of 25%, with overfitting starting around the 50th epoch. The validation accuracy of the model is still quite poor and leaves much to be desired. However, this indicates that the ResNet model performs better when the weights are updates in this problem.

Instead of trying to further optimize this model, it would be more efficient to test another pre-trained model that may fit the data better.

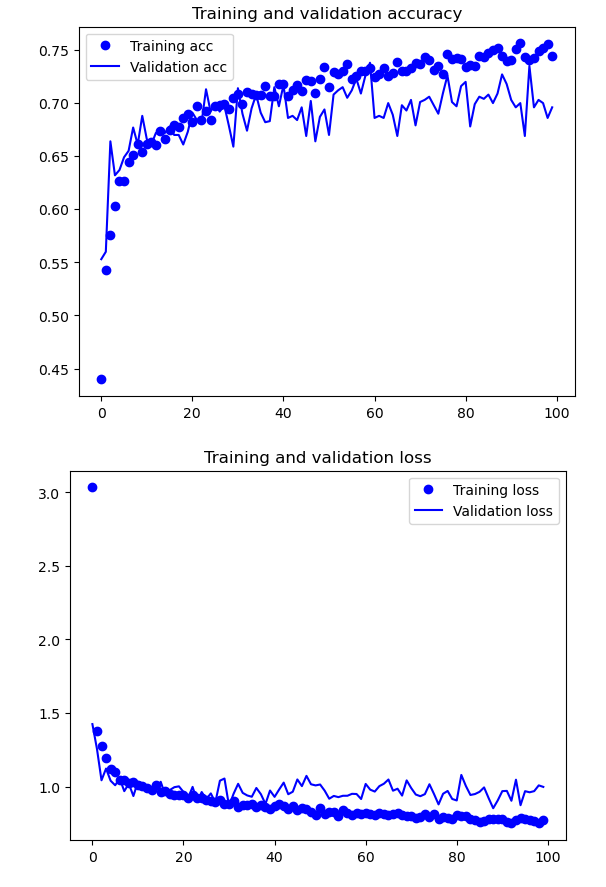


## Model 2e

The next and final pre-trained model that was tested is InceptionV3. InceptionV3 is an image recognition model that has been shown to achieve an accuracy greater than 78.1% on the ImageNet dataset.

The model is 42 layers deep and has been optimized extensively from its predecessors. The improvements made to the InceptionV3 architecture include factorization into smaller convolutions, efficient grid size reductions, and the use of auxiliary classifiers to improve the convergence of the neural network. The model computes loss with the softmax function and extensively makes use of batch normalization and applies it the activation inputs of the model. This makes it suitable for image classification problems.

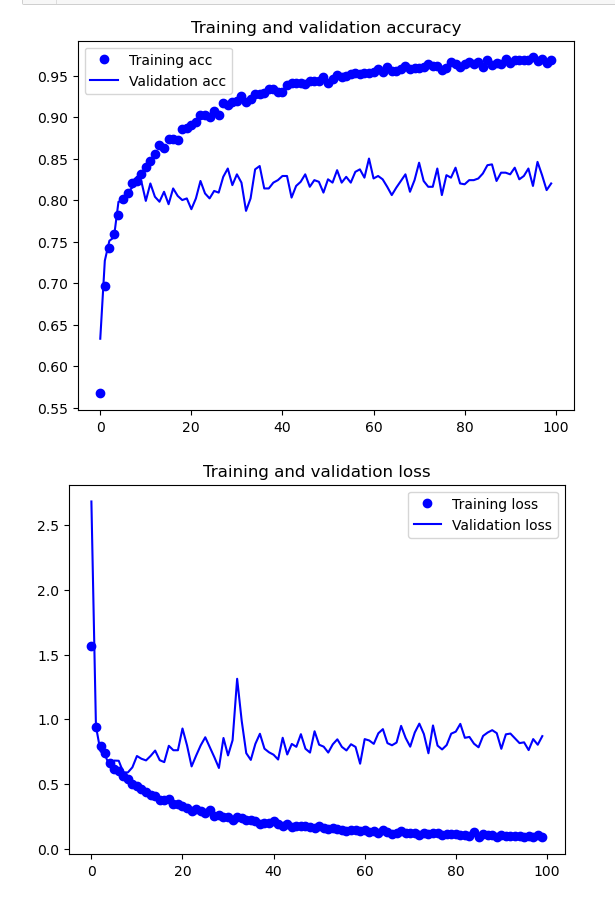
The model was tested with the same parameters as the ResNet model, with the inclusion of data augmentation to further improve the accuracy by allowing the model to train on a larger range of data. After running and evaluating the model on its validation curves, the validation accuracy hit around 71% on the dataset and the loss seems to show minimal overfitting. This means that model 2e is able to perform prediction on the image data well.



## Model 2f (Best Model 2)

This iteration of the model will unfreeze the pre-trained InceptionV3 layer, to allow it to better adjust the weights of the model, to further learn the data and optimize the accuracy of the model.

After unfreezing the layers from “mixed 6” and running the model on the data, the model was able to adjust its learning weights to better learn and classify the images. The model was able to achieve a validation accuracy of 82% on the data. Examining the validation curves, it is observed that the model has minimal overfitting, but the validation accuracy stagnates around 80%. Overall, this is the most accurate model among the iterations tested and will be useful for prediction as it achieved an accuracy of 80%.



# Evaluation on Test Images

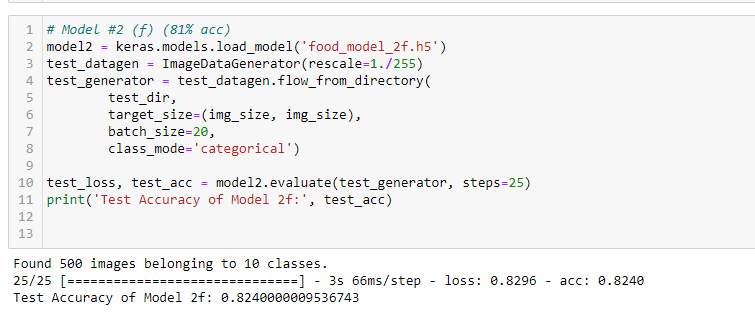
The CNN model evaluated is model 1f. It was the model with the highest accuracy and minimal overfitting among the CNN models built for model 1 that was tested.

When evaluated on test data, it is observed that the accuracy of the model has dropped considerably, from 65% to 47%. This may indicate that the model has not fully grasped and learnt the data during training and validation. Moreover, accuracy under 50% is trivial and would not be useful for making predictions. Additional improvements and optimizations should be made to further improve the model’s performance. Text

Description automatically generated

For model 2, model 2f was used as the InceptionV3 pre-trained model with unfrozen layers was able to provide the best accuracy and performance among the models tested.

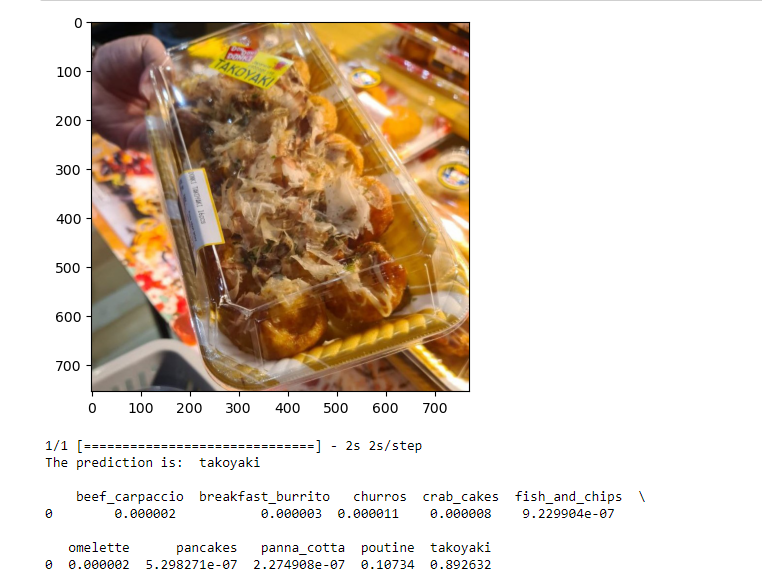
When evaluating on unseen test data, it is clear that the model has successfully been trained. The test accuracy of the model is 82%. This is means that the model is more than 80% accurate when predicting on unseen data, thus making model 2f the better model for prediction when comparing the two.



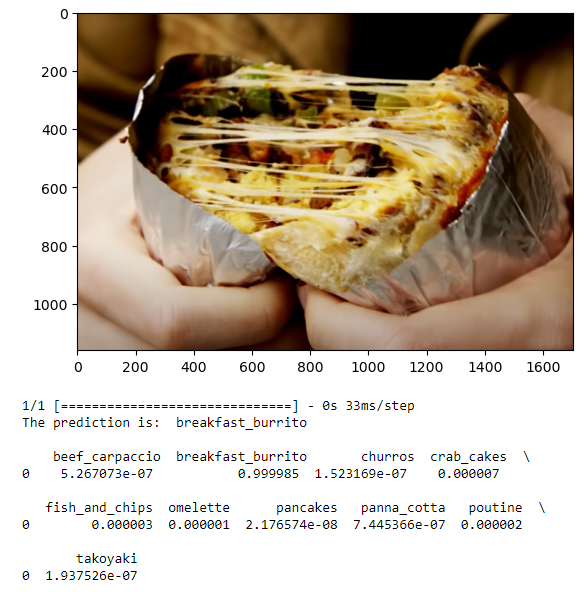
# Prediction on Real-Life Images

The final step to evaluate the model is to test the model’s accuracy on real-life images. The images will be loaded into the notebook and resized to 150 x 150, which is the sized used for training the model. The names of the foods are also instantiated in alphabetical order as an array and will be used to print out the names of the predicted results. I have taken 3 images based on the 10 food images from other sources to test on the final model. These 3 images are new unseen data that will test the model’s ability to recognize the 10 food images.

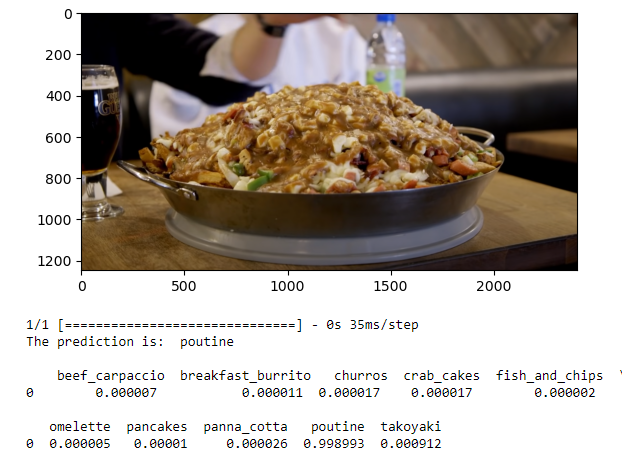
The first image is an image of Takoyaki taken personally by me. When the image is applied onto the model, the model was able to successfully identify the food in the image. The model returned a value of 89.2% for Takoyaki, which indicates the probability of the prediction. This means that it was almost 90% sure that the image is classified as a Takoyaki.



The next image is an image of breakfast burrito taken from the internet. When loading it into the model for prediction, the model returned a 99.9% for the label breakfast burrito. This shows that the model is able to classify the breakfast burrito images with a high degree of accuracy.



The final image that shows a poutine was also taken from the internet. Again, when running the image through the model, the model returned a probability of 99.8% for poutine. This means that the model has correctly classified the image as a poutine with a high degree of confidence.



Based on the evaluation, the model appears to be accurate and useful for predictions on the 10 foods that the model was trained on. However, only a small sample of images was tested, and it might not be able to classify images that do not clearly show the food or look similar to other foods in the training data.

# Summary & Further Improvements

This report has covered the processes involved in creating a Convolutional Neural Network (CNN) model from scratch and with transfer learning, to accurately predict and classify 10 types of food images. The model’s accuracy and effectiveness was evaluated by testing the model with new unseen food images taken from the internet and other sources.

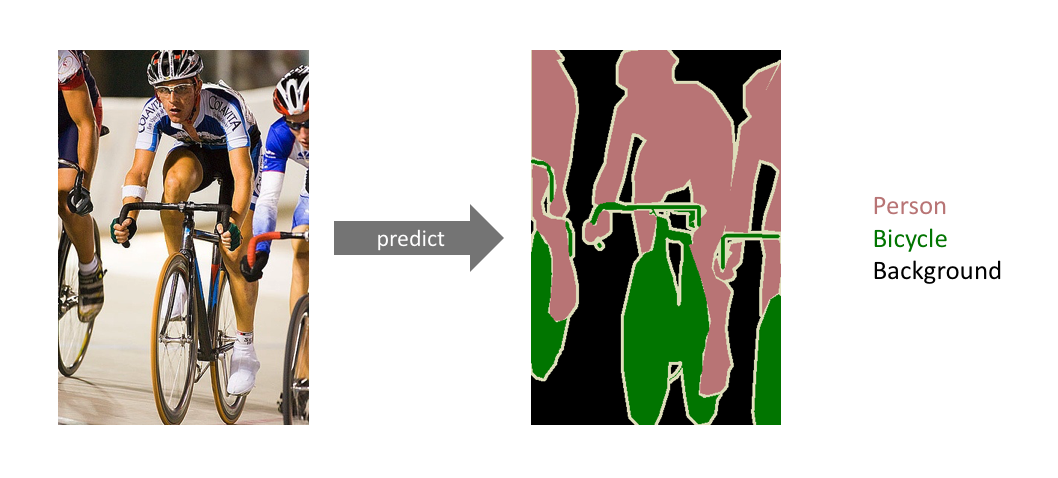
Overall, the final model’s prediction accuracy is sufficient at 82% test accuracy. Furthermore, it was able to predict 3 out of 3 of the real-life test images which shows that the model has been properly trained to classify the assigned food images.

Some basic improvements can be made to further enhance the accuracy and usefulness of the final model. Firstly, the model could be trained on a higher amount of data which would improve the learning of the model. Next, further tuning and optimizing of the model parameters can also be done to improve performance. Testing other pre-trained models could also be considered as the model used may not necessary be best suited for the dataset. Other basic improvements include, redesigning the network design, starting with a less complex model, and having a better understanding of how the parameters affect the model’s performance.

For additional improvements to model accuracy, we can look into using image segmentation for the problem. More specifically, Semantic Segmentation, as Instance Segmentation is more applicable for counting the number of the specified “objects” in the image. While Semantic Segmentation is used more to identify and label the “object” in the image.

Semantic Segmentation is used to label specific regions of an image according to the object, which makes image analysis and recognition easier. It does this by labeling each pixel of an image with the corresponding class. This may allow the model to more accurately identify the food shown in the image as, for some images, the food can be difficult to distinguish from the background. Moreover, Semantic Image Segmentation already has many real-life applications in medical science to identify tumors or in self-driving vehicles.to identify the road or other vehicles.

Example:



Source: <https://www.jeremyjordan.me/semantic-segmentation/> & <http://host.robots.ox.ac.uk/pascal/VOC/voc2012/#devkit>

# References

ImageNet. (n.d.). *About ImageNet*. <https://www.image-net.org/about.php>

Jordan, J. (2018, May 21). *An overview of semantic image segmentation*. Jeremy Jordan. <https://www.jeremyjordan.me/semantic-segmentation/>

Princeton University. (n.d.). *WordNet*. <https://wordnet.princeton.edu/>

TensorFlow. (2022, December 15). *Image Segmentation.* <https://www.tensorflow.org/tutorials/images/segmentation#what_is_image_segmentation>