AIML421 Capstone Project

Image Classification CNN

Introduction

Image Classification is the task of a machine learning model being able to recognise or interpret the focus of an image through various methods. This task is typically done using deep learning models and requires many images to learn from to correctly provide the best features for the models to learn from.

Convolutional Neural Networks, or CNNs, are good at working with high-dimensional vectors to represent instances and can learn trends at a deep level better than many other contemporary models in the field, thus they are applied to high level applications like the development of Image Classification models.

Image Classification of fruit can be useful for furthering the field of Image Detection in general [2], particularly to improve face recognition by starting with simple-to-recognise fruits and vegetables that have many features to glean, and then moving on to understanding the concepts of Image Classification and how they relate to the image itself.

Problem Investigation

EDA:

Looking through the dataset manually, it was found that a portion of the images either contained cakes or other subsidiaries that would clash with the finding of the fruit, however, this would still have the actual fruit as the focal point of the image or would blend the cakes and other subsidiaries with the background, meaning the focal point would remain as the fruit.

Other examples of tomatoes would be unripened or contain background noise from other plants that took away from the focal point of the image. This results in the image being either low quality, or hard to see. The problem with this being a lack of clarity into the shape and colour of each fruit, and rather a guessing game.

Cherries would often contain people eating them, or have the stem off screen, this reduces an important feature that defines cherries and would make it harder for the model to train to recognise them. Another problem is many colloquialised views of cherries come in pairs, however the singular cherry is more common within the dataset, meaning any caricatures of cherries or advertisements would make it harder to recognise.

Thus, the first task was to clean the dataset of the images that violate the issues found above in the investigation. The first goal was to outline a set of rules that would define the quality of an image and its viability. These rules were:

- The image must show the shape of the object in its totality or majority.
- The image must be of similar colouring to other instances.

- The image must not contain background noise from other colours or items.
- The focal point of the image must be on the object.
- Cartoon versions are not allowed.
- Subsidiary examples (eg. Strawberry Milkshake) are not allowed.
- Any images with text are removed.

Evaluating the total dataset, it's important to display an example of which images are considered a violation of the above rules as well as those who meet the criteria along with the extra features that accentuate the trends required, here is an example for the Strawberry class:

Bad, Violates Subsidiary Example	Good, Shows shape and fruit only, High contrast
	

This strict guideline of removal was then applied manually to each dataset to clean and remove unwanted instances from testing. This results in each class label going from:

- Cherry: 1,495 -> 1,418
- Strawberry: 1,495 -> 1,298
- Tomato: 1,495 -> 1,333

Preprocessing:

Images are resized to 128x128 to reduce dimensionality, then within the train set are duplicated by flipping vertically and horizontally, along with a small rotation. Each of these permutations provides a new instance to learn off due to the changes and this helps the model to learn the item itself past the image trends.

Lastly, the colour values are normalised to the RESNET 18 defined values to improve contrast and help accentuate the edges of each image [1]. This normalisation is also applied to the test set as it does not change any data within the image but merely filters it. This allows for the test data to be homogenous with the training data.

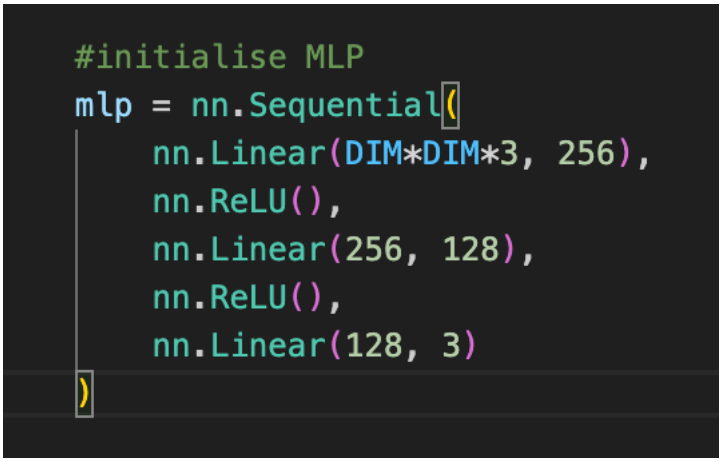
These changes to the model design as well as the preprocessing of the data will help the model to better learn each class without noise mitigating the trends present in each image. Also, the extra feature generation using different flips should help the model to recognise each class at any orientation or collection, allowing for better accuracy within the model, however this could result in overfitting in the baseline due to a repetition of class objects.

Methodology

To establish a baseline, a Multi-Layer Perceptron was developed to evaluate the training and validation sets along with validate the usefulness of the EDA performed along with the preprocessing pipeline. Both models were tested on an Apple M2 Mac using the GPU and CPU for the epoch evolution. The MLP pipeline has three linear activation layers to reduce dimensionality and output the class result.

Baseline Multi-Layer Perceptron:

The baseline model developed to compare against the CNN developed for this model was a Multi-Layer Perceptron developed using the PyTorch library in Python. This library can be used for the development of many different neural network applications like said perceptrons or CNNs. By outlining the stack for each model, the architecture can be directly displayed through the code itself



```
#initialise MLP
mlp = nn.Sequential(
    nn.Linear(DIM*DIM*3, 256),
    nn.ReLU(),
    nn.Linear(256, 128),
    nn.ReLU(),
    nn.Linear(128, 3)
)
```

This model runs for 15 epochs and due to the dimensionality reduction of each image, only takes 1m 19s to run completely. This baseline is quick and efficient to provide important steps to improve via the CNN architecture. The Loss function chosen is CrossEntropyLoss to help with the model's adjustment of weights and learning. The Optimiser used is Adam, chosen for efficiency in convergence along with the ability to display changes when adapting to a CNN, the learning rate is set to 0.001. This model performs well on the training set, allowing the model to find the trends learned, however performs worse when testing.

Set	Size	Accuracy
Training	3,239 Images	86.20%
Test	810 Images	49.38%

As we can see from the disparity in the accuracies despite the smaller search size, the overfitting problem present within the MLP is a problem that will likely arise within the development of the CNN model. This means steps need to be taken within the architecture of the developed CNN to mitigate this error, or within the preprocessing pipeline to provide more features that are deep enough to be interpreted by the model past recognition of similarities.

CNN Model:

The CNN model developed for this project differs from the MLP defined within the first section of the methodology by increasing the complexity of the model developed along with changing the preprocessing methods applied to the model to improve feature accuracy and overall fitness for purpose.

Architecture:

The CNN model developed contains three convolutional layers that increase the dimensionality of the data through the extraction of features from the images and each layer uses an activation function for non-linearity and a pooling method to ensure dimension size is maintained. After these convolutional layers are complete, the data is flattened to be used in the hidden layer for classification which is then passed to the output layer to give a result. Every test is done using seed 0 for random, torch random and generator seeds. The activation function used for this model is consistent throughout to match the MLP as well as the proven quality of ReLU activation and its sufficiency in most if not all cases.

```python
class CNN(nn.Module):
    def __init__(self, num_classes=3):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
        self.conv3 = nn.Conv2d(64, 128, kernel_size=3, padding=1)
        self.pool = nn.MaxPool2d(2, 2)
        self.dropout = nn.Dropout(0.25)
        self.fc1 = nn.Linear(128 * 32 * 32, 128)
        self.fc2 = nn.Linear(128, num_classes)

    def forward(self, x):
        x = self.pool(func.relu(self.conv1(x)))
        x = self.pool(func.relu(self.conv2(x)))
        x = self.pool(func.relu(self.conv3(x)))
        x = x.view(-1, 128 * 32 * 32)
        x = func.relu(self.fc1(x))
        x = self.dropout(x)
        x = self.fc2(x)
        return x
```

To replicate the different features of the Baseline MLP, the same optimiser and loss functions were used, Adam and CrossEntropyLoss respectively. This allows us to directly compare the results of the CNN model against the MLP on the same metrics, and then perform improvements to this model to maximise its performance.

Model	Train Accuracy	Test Accuracy
MLP	86.20%	49.38%
CNN	95.34%	64.81%

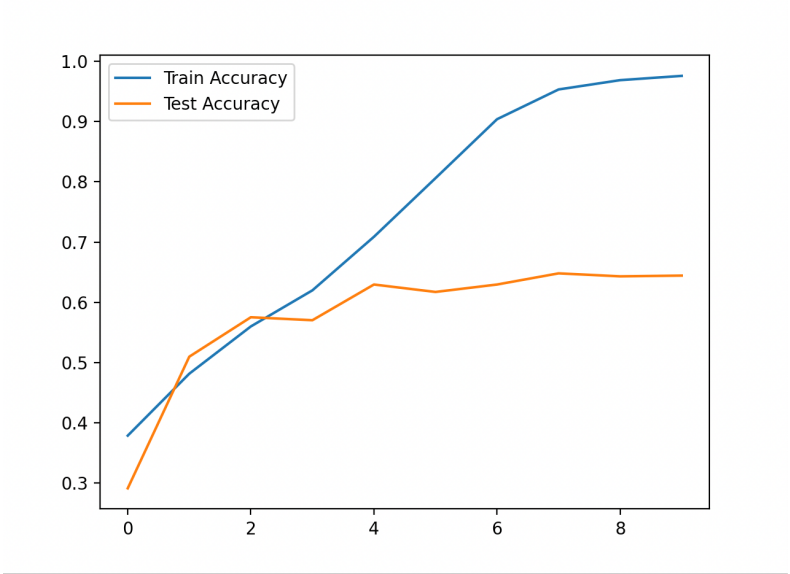
Looking at the results replicating the same optimiser and loss function, we can see that the CNN performs better in all aspects but still fits heavily to the training data, this can be mitigated by making more strict stopping criteria. Overall, improvements need to be made to the model through testing and iterative design to improve the results further, this project aims to find five or more interesting ways to adapt this model for its purpose, by implementing different improvements such as the stopping criteria, the image augmentation, the optimiser used, the regularisation strategy as well as using transfer learning to adapt the model further.

1. Stopping Criteria

The stopping criteria for the model is either:

- 10 epochs have passed.
- The test accuracy has been below the best result twice in a row.

By not giving the model the chance to bounce back from a decrease in test accuracy, this could reduce overfitting and improve the model. However, to get the best test accuracy, this stopping criterion is required. When looking at the base CNN and its trends for accuracy, we can see that this does help the model, but results in the model overfitting to the training data to get the best test accuracy.



Even with this change in stopping criteria, the model overfits to the data to get the best performance out, however, this is to maximise the test accuracy over time. Mitigating this would result in a harsher condition for stopping and would overall reduce the best accuracy produced during training. Further implementation of the second stopping criteria will become important further in the model development where the model can improve past the epoch limit and is tested to do so.

2. Data Loading and Split Methodology:

The data is loaded from the train data folder which houses each class within the respective folder. This data is loaded into a single loader with no transformations and then is split with the generator using the seed 0. This data is split using an 80/20 split in which the training set is applied different transformations, a horizontal, vertical and rotation to reduce recognition in the model by randomly transmutating images to remove trends but still maintain the

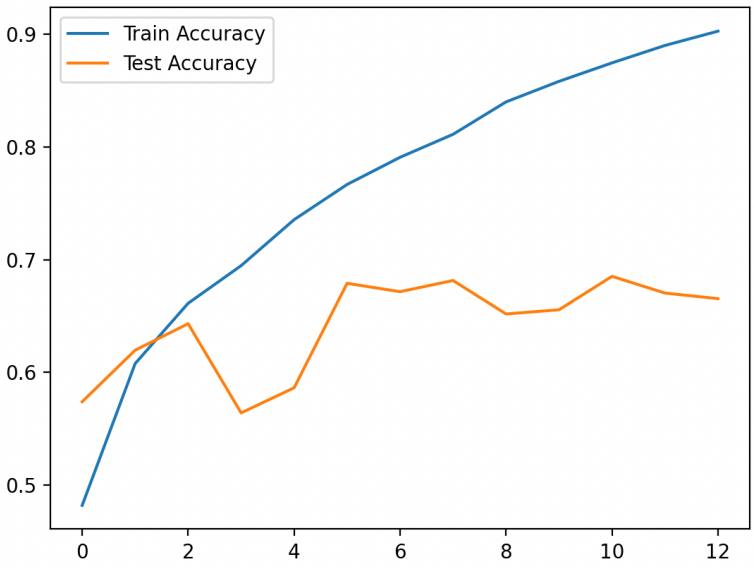
sample quality. The test set is not applied this transformation set, but both are resized to 256x256, and colour filter normalisation is set to 0.5 for all values to unify the colour values for every image.

3. Optimiser Choice

The optimiser is an important choice when developing a Convolutional Neural Network, different optimisers target different gradients for how to update weights on each computation, meaning that a change in optimiser could improve or reduce the accuracy of the model. To ensure that the CNN model is optimal for use, evaluating different optimisers during training is essential. The base model uses the Adam optimiser, which is a combination of AdaGrad and RMSProp advantages along with adaptive learning rates.
Other optimisers tested are SGD, AdaGrad and RMSProp. These are different optimisers that differ from Adam in terms of aims and weight calculations.
All use a learning rate of 0.001, the same as the base model, to provide consistent results. Specifically, SGD uses a momentum of 0.9, this is a typical value for SGD with momentum and is evaluated to be an optimal value in most instances [3].

Optimiser	Train Accuracy	Test Accuracy
Adam (Base Model)	95.34%	64.81%
SGD	55.36%	56.30%
AdaGrad	81.11%	68.15%
RMSProp	59.65%	54.69%

The best optimiser is AdaGrad for accuracy on the same testing as the others, this optimiser uses historical gradients to alter weights, which allows for better adaptation to infrequent features. This optimiser will be used going forward into further evaluation of parameters. After applying Adagrad to the model and increasing the epochs to check if the model reaches the second stopping criteria, the convergence graph results in a slightly better performance past the base 10 epochs.



On epoch 11, the best accuracy is found, and right after the second stopping criteria is reached, with a training accuracy of 87.43% and test accuracy of 68.52%. Using 15 epochs will be the standard going forward.

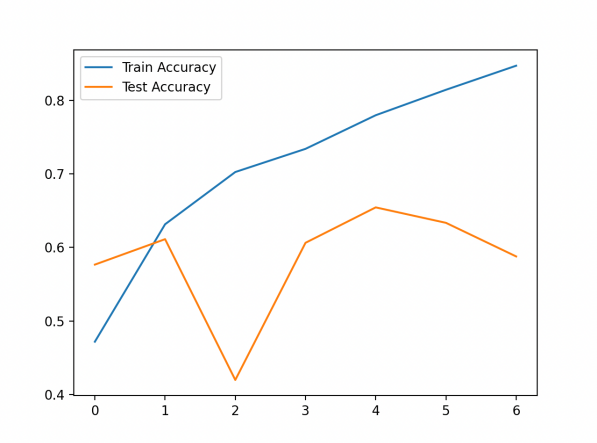
4. Regularisation Strategy

Important strategies are required in CNNs to reduce the gradients present within the model to allow for regularisation of data. The most common regularisation strategies used in CNNs are Dropout and Batch Normalisation, Dropout is used for removal of random connections to reduce complexity of the model. Batch Normalisation is used to shift scales and adapt connections to introduce variance, these both ensure a reduction in overfitting through different methods and should be evaluated differently to ensure the model is performing optimally with these outlined examples.

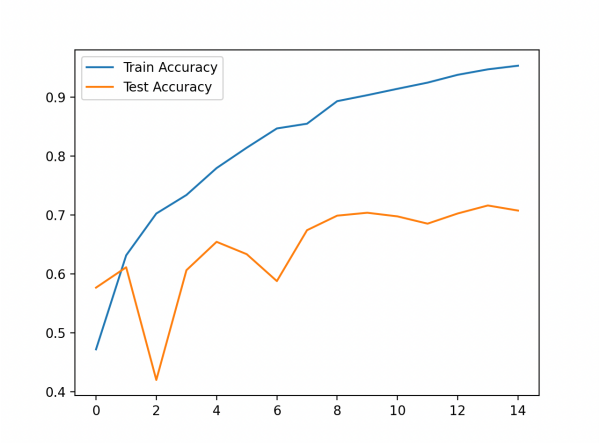
Implementing both a dropout layer, then swapping it for implementation of batch normalisation, this results in these metrics:

Regularisation Technique	Train Accuracy	Test Accuracy
Dropout of 0.25	87.43%	68.52%
Batch Normalisation	77.96%	65.43%

Using batch normalisation resulted in lower overfitting and converged quicker, however resulted in more variance between the outputs on each epoch, this results in the model being inconsistent with the output and gradually getting better over time, this could be improved further by making the stopping criteria less strict due to the architecture change, but also reduces computation time. The convergence ends at epoch 7 and the graph is smaller than the drop out method, indicating a better overall performance, by increasing the stopping criteria to 3 in a row, this could result in better performance over the 15 allotted epochs.



Before Applied Changes



After Applied Changes

When extending the stopping criteria to allow the model to slowly converge past the previous result due to being allowed to better learn the data over time and thus better learn the trends when working with unfounded data. The model runs 8 epochs longer and

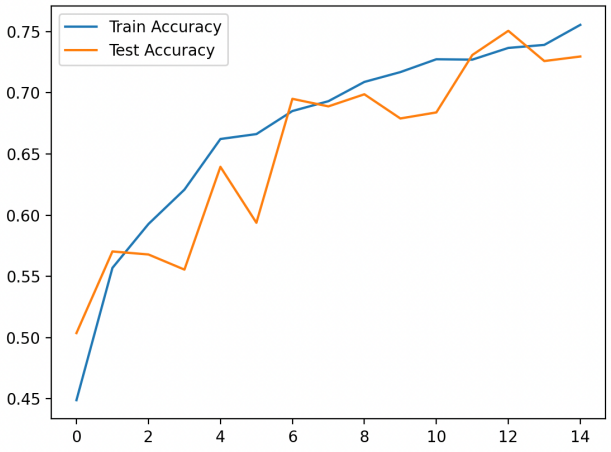
improves slower past this stage, but performs better past this, getting a training accuracy of 94.72% and test accuracy of 71.60% at its best epoch, which is saved at training end.

5. Resnet Normalisation and Transfer Learning

The Resnet pre-trained models use specific normalisation for hex values and shading to normalise the colour representation within the collection of images, this allows for the model to recognise expected trends within colour connections and learn them. This normalisation can be specified as mean= [0.485, 0.456, 0.406], std= [0.229, 0.224, 0.225]. This brings the normalisation process back in line with the MLP design outlined as the baseline, as this can improve trend declaration and is a more recognisable output. Also, a manual split was performed on the data to split the dataset 80/20, and uses a different seed from the previous, meaning the results of using the ResNet normalisation before the addition of transfer learning provides insight into the consistency of the model on different splits.

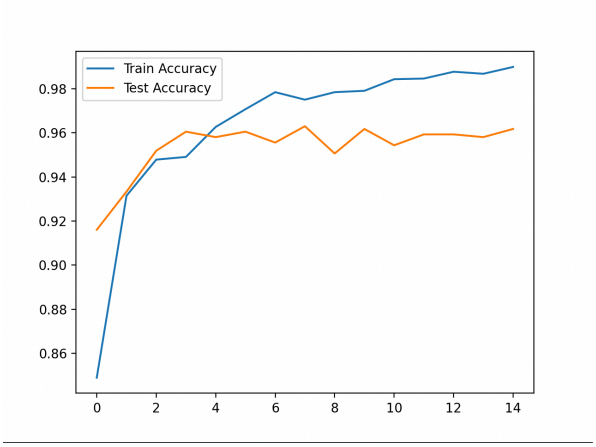
Normalisation Type	Train Accuracy	Test Accuracy
0.5 – Standard	73.66%	75.06%
ResNet Normalisation	73.51%	73.83%

With these findings, we can attribute test accuracy to be the important metric in deciding which normalisation to use, by diversifying from the baseline MLP and using 0.5 as the baseline, this slightly improves the test accuracy. Using 0.5 normalisation results in this training and validation accuracy trend:

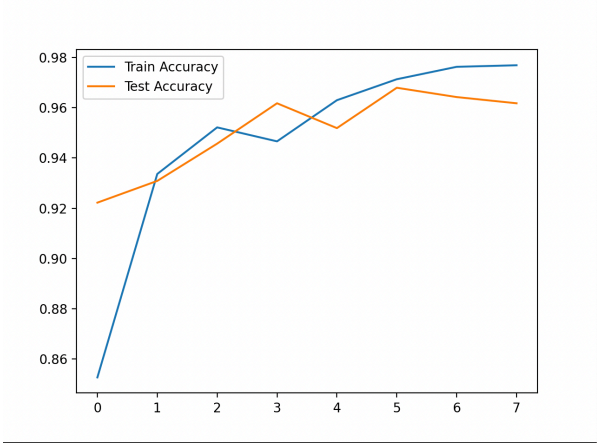


By using transfer learning to further improve this model using a pre-trained model, this model could use pre-defined architecture and weights to define a model fit for use without having to train it. By using the Resnet 18 model developed by Microsoft Research for a base to transfer weights from, this model can perform better than our trained one through transfer learning. This model is trained using images that are normalised using the resnet normalisation tested previously, thus the transfer learning process will compare the difference of these as well.

Normalisation Type	Train Accuracy	Test Accuracy
0.5 – Standard	98.98%	96.17%
ResNet Normalisation	97.13%	96.79%



0.5 Normalisation



ResNet Normalisation

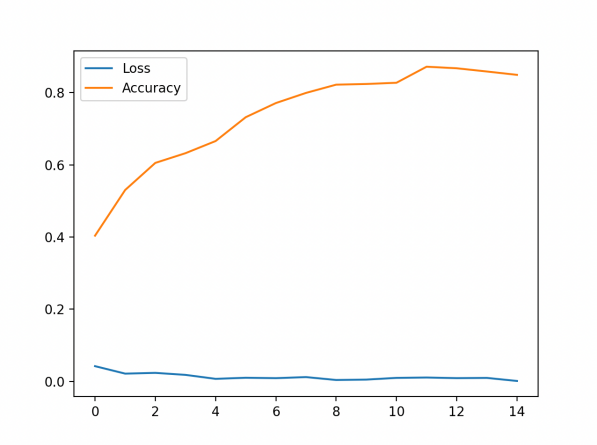
Overall, the model performs slightly better when implementing the normalisation of values that the resnet expects, but not by much, this is due to the ResNet pretrained architecture being high-dimensional and having a good ability to recognise images regardless of colour normalisation.

Summary and Discussion

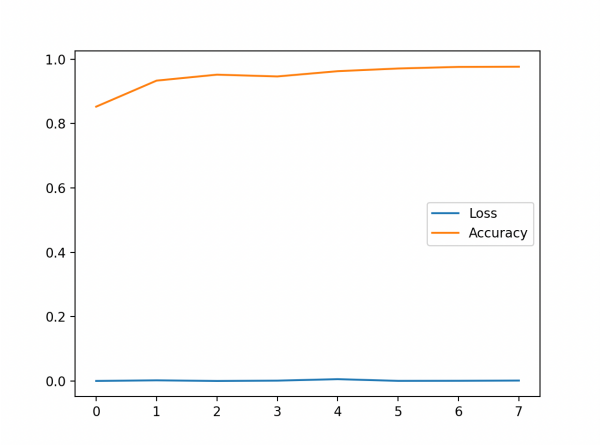
Compared to the MLP baseline implemented, the complexity of this model evolved over time to use different methods to adapt the data as well as ensure that the optimisers and features were optimal for the task, through adaptation of optimiser and normalisation or regularisation strategies. Looking at the results from the MLP, the results clearly improved over the development of the model, resulting in the gradient depicted in the ResNet Normalisation figure.

Model	Train Accuracy	Test Accuracy
MLP	86.20%	49.38%
CNN	97.13%	96.79%

However, this necessitated the use of a transfer learning process, as the computational requirement to increase accuracy increases, the use of higher fidelity models is required. This is due to the smaller dataset applied to this problem with the Flickr dataset, as well as the lack of computational resources and time at the disposal of the AIML421 corpus, it simply does not compare to Microsoft or other companies that have their own image classifiers. Regardless, the accuracy of the CNN model vastly outperforms the MLP designed as the baseline through the various methods outlined in this report and meets the criteria.



MLP Loss and Train Accuracy



CNN Loss and Train Accuracy

Conclusions and Future Work

Further work could be applied to different parameters that the CNN use, like the learning rates for optimisers or the batch sizes. In this model, the optimiser learning rate was controlled by a scheduler that had little effect due to a lack of plateauing from variable convergence. The batch sizes were limited by the hardware used for the model development due to a requirement to minimise memory usage during development, this could have changed parts of the model development, however still proved sufficient for the task at hand. Lastly, many experiments were conducted one after another, a more comprehensive overview of the quality of the model would have cross validated all of these components to ensure the highest purity of output, however due to computational limitations this was not tested, there could be missed combinations in which these values work together better than on their own, and thus there is more room for validation of this model present in this paper.

References

[1] A. Venkata, “Resnet18 Model With Sequential Layer For Computing Accuracy On Image Classification Dataset,” vol. 10, no. 5, pp. 2320–2882, Jul. 2022, Accessed: Oct. 21, 2024. [Online]. Available: https://www.researchgate.net/publication/364345322_Resnet18_Model_With_Sequential_Layer_For_Computing_Accuracy_On_Image_Classification_Dataset

[2] G. Latif, N. Mohammad, and Jaafar Alghazo, “DeepFruit: A dataset of fruit images for fruit classification and calories calculation,” *Data in Brief*, vol. 50, pp. 109524–109524, Oct. 2023, doi: https://doi.org/10.1016/j.dib.2023.109524.

[3] M. Wang, “Why to Optimize with Momentum,” *Analytics Vidhya*, Oct. 23, 2020. https://medium.com/analytics-vidhya/why-use-the-momentum-optimizer-with-minimal-code-example-8f5d93c33a53

[4] “(PDF) A Study on CNN Transfer Learning for Image Classification,” *ResearchGate*. https://www.researchgate.net/publication/325803364_A_Study_on_CNN_Transfer_Learning_for_Image_Classification