Computer Vision and Natural Language Processing Integrated Application for Recipe Recommendations

Dawson Damuth (CV Engineer, Backend) and Harsh Tamada (NLP Engineer, Frontend)

Abstract Natural Language Processors and Computer Vision models dominate a significant portion of commonly used applications in the current tech landscape. These models are cutting-edge in their ease of use, allowing them to be applied in a variety of use-cases. For our application we will use a pretrained BERT (Bidirectional Encoder Representations from Transformers) model and a ResNet (Residual Network) together in tandem to perform the task on images of select fruits and vegetables, classify the item, classify the state of the item (three categories, distinct per each item), then feeding these classifications into our BERT model for recipe recommendations and relevance scores. Additionally, we will create standalone applications for each of the distinct tasks as well, including a base-level classifier and an NLP which takes tags and outputs recipes with relevance scores.

Keywords Applications • Images • Classifier • Models • Machine Learning • Deep Learning • Natural language processing

1 Introduction

The objective of the application was split into four phases; data collection, CV model building, NLP model building, and application development. The first of these, data collection, was a self-directed process of collecting a total of 12,000 images for our future models. Split evenly between the four chosen fruits and vegetables, the images were then divided evenly again for each of the three states of the items, resulting in end-category batches of 1,000 each. After the data has been collected, we can begin model construction.

The structure for this task is as follows:

Images (12,000 images):

- Eggplant (3,000): Whole (1,000), Halved (1,000), Sliced and Cooked (1,000)
- Potato (3,000): Whole (1,000), Peeled (1,000), Fries (1,000)
- Cherry (3,000): With Stem (1,000), Pitted (1,000), In a Bowl (1,000)
- Orange (3,000): Whole (1,000), Peeled (1,000), Segmented (1,000)

Beginning our second phase we will be utilizing five different pretrained ResNet models, all taking the form of the ResNet50 architecture. All models are trained independently using transfer learning, once globally for the basic classifier, then in each item subfolder to classify the state of the object. The models work hierarchically, with the global classifier feeding into the latter model in the application to identify which of the state models to use. This dependency means our models must be accurate and efficient, since they are building off one another to make decisions.

The third phase involves the construction and training of our NLP model. For this task we will be using a pretrained BERT model to give us a recommendation list of the most compatible recipe list from a given database of recipes. For this we use transfer learning to fine tune our model to give us this list based on criterion such as highest rating, etc. This is achieved by filtering data into a trainable form and training it on those to fit our criteria.

Finalizing our development phase, it was time to build the framework for our application, both front and back-end. For the back end of the application, we needed to create model scripts which allowed for images to be passed through and a two-layer classification to be made, the keywords of the classification later being fed through the NLP. All the inference files also needed to be backed by the model weights, which are provided. Additionally, we needed an API (Application Programming Interface) to execute the model applications. We then add a front-end to our application that includes this paper and an interactive way to use our model which includes uploading a picture of our vegetable/fruit and adding any descriptors we can to further narrow down our search for a recipe which is outputted in a table along with the recipe name, id and a rating score from the interactions list.

2 Methodology

As mentioned, the two model types used are ResNet50 and BERT. There is a total of six models trained for the application; a general fruit/vegetable classifier, four individual state classifiers for each previously classified item, and a BERT model which outputs a list of recipes and rankings given the keyword entries.

2.1 ResNet Models

Each of the ResNet models were imported from TorchVision: PyTorch's Computer Vision library [1]. The library provides SOTA pre-trained models for ease of use in a task like ours where transfer learning is of great value, considering the limited time and computational resources.

Each model was trained using transfer learning on the existing ResNet50 architecture, with all hyperparameters constant between the models, aside from a change in learning rate to 1e-4 and batch size lowering to 16 to account for the sparsity of data in the four state-classification models, where the general model has a learning rate of 5e-4 and batch size of 64. Aside, these models share all hyperparameters and criterion for loss and optimization. The data is split into train, test, and validation sets by proportions of 70, 15, and 15 percent of their respective data sets. Additionally, both models are trained using transfer learning on the existing weights over 10 epochs. In preprocessing for both the training and final application model, the images are resized to fit

224x224 pixels, allowing uniformity in the input structure while preserving enough of the image data.

2.2 BERT Model

The BERT models used for the NLP model here is an SBERT (or Sentence Transformer) particularly the “all-MiniLM-L6-v2". This falls into the BERT class of models which was chosen for its strong semantics embedding. This fits the prompt of finding the most relevant recipes based on an input that consists of ingredients and/or descriptors. It is also a compact and fast model

The model was trained using transfer learning, using training pairs that embed positive and negative queries. A query in our case is ingredients and/or descriptors in the given list of recipes (RAW_recipes.csv, RAW_interaction.csv). These were further filtered down into a training dataset that consisted of recipe info (id, name, ingredients and descriptors) and a mean rating related to each recipe. To train, we first make training pairs for each query that include positive (itself) and negative query (randomly selected query that has a rating lesser than 3.0). We then train it for 3 epochs with batch sizes of 16 and a cosine similarity loss function (this is used for the purpose of semantic similarity to recommend recipes).

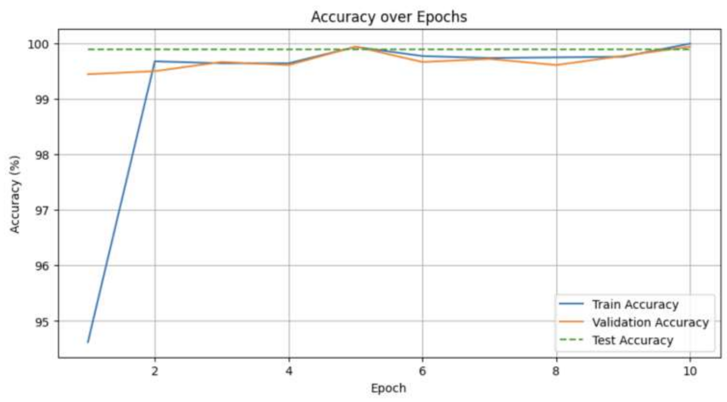
3 Training and Results

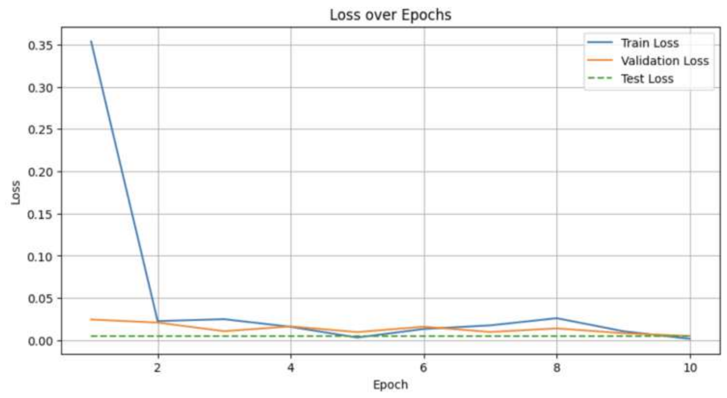
All models in training achieved exemplary results, including the state-classifiers which were trained on higher data sparsity. We can confidently claim that the models are fit to be implemented in the latter application and will work well both independently and in-tandem.

3.1 ResNet Results

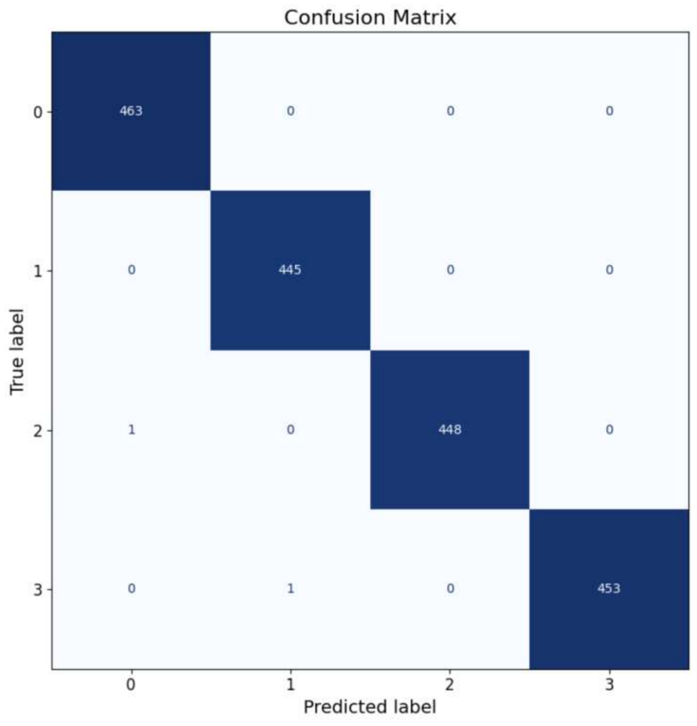
For our ResNet50 models, we saw all models achieved above 99% accuracy within all data sets, including validation and testing, ensuring this wasn’t a product of overfitting.

The general model achieved an accuracy rating of 100% in training, 99.94% in validation, and 99.89% in testing. For additional comparison, we can also analyze the f1-score of the model, the average f1-score being 99.89% as well. This is generally expected to be the case in models which have balanced classes, such is the case in our dataset. Below, we can see a visual display of our accuracy and loss over epochs for validation and training, with a constant for the test values.

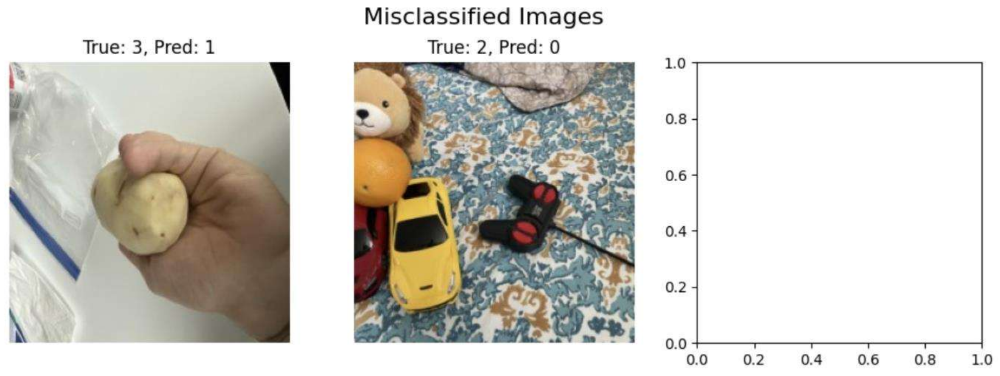




From this, we can see the convergence in training and validation happens almost immediately, offering very minimal improvement in the latter epochs. We can also look at the confusion matrix and misclassed images for the general model during testing:



```
Total misclassified images: 2
```



(Class Labels: [0:3]: [‘Cherry’, ‘Eggplant’, ‘Orange’, ‘Potato’])

The first image is a potato misclassed as an eggplant, the second is an orange misclassed as a cherry. Looking at the images, the obstructions and noise in the image allow us to understand how these misclassifications happened, as the first image looks vaguely similar to a halved eggplant, and the second has two round red objects in the focus of the image which are easily mistakable for a pitted cherry.

Moving to the state classification models, we have several similar evaluation metrics to discuss. We will discuss the models and their performance in an item-by-item format. For this section, the accuracy and f1-score will be solely on the test set, with the accuracy for the other sets displayed in the accuracy plots. Note that all plots for this section will be in the appendix. [i:viii]

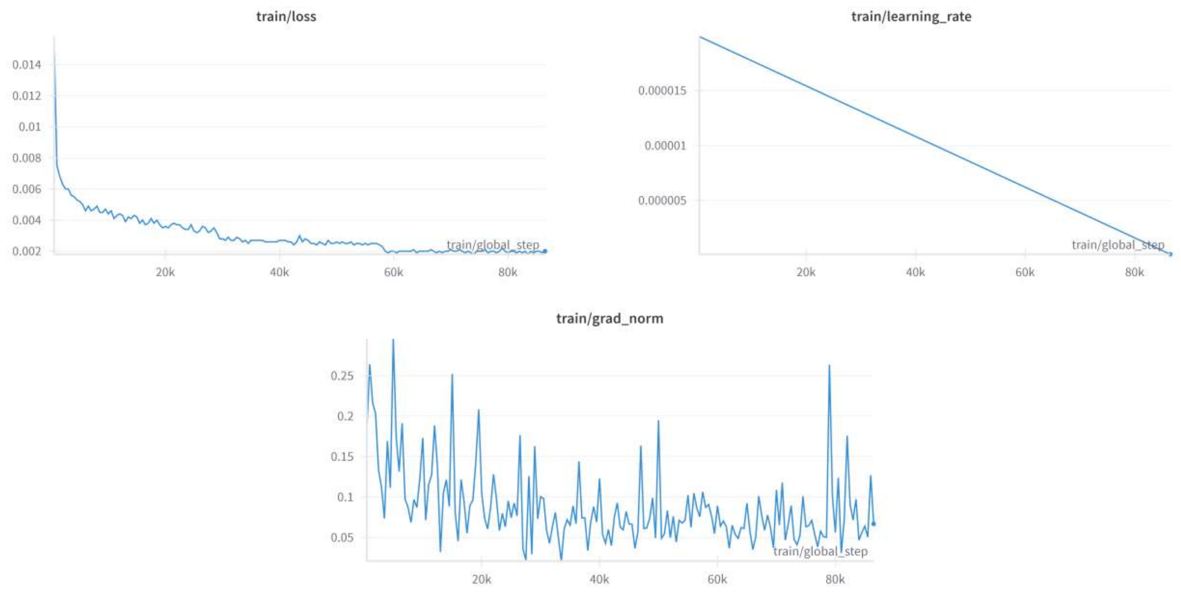
Model:	Cherry	Eggplant	Orange	Potato
Accuracy (Test)	1.0000	0.9978	1.0000	1.0000
F1-Score	1.0000	0.9978	1.0000	1.0000

Table 1: Model statistics for state-classification models

From the table above, we observe that all state-classification models achieved near-perfect test accuracies and F1-scores, indicating highly effective performance. While such results in computer vision tasks might typically raise concerns about overfitting or poor generalizability, several factors support the credibility of these outcomes: the models were evaluated on an unseen test split, and both accuracy and F1-score consistently support the findings. Although true generalization can only be confirmed with external validation, these results strongly suggest that the models are genuinely robust.

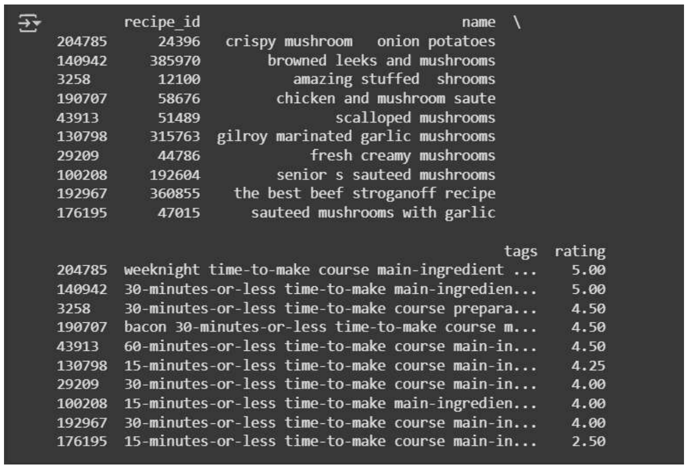
3.2 BERT Results

The training phase of the SBERT model was captured on wandb.ai, through which we see statistics of the model training such as the loss per epoch, learning rate and gradient norm as seen below.



We see that the learning rate, loss and gradient is seen as ideal. The loss is almost minute by the last epoch of training. The gradient norm seems more unpredictable, but the gradual decrease is seen as ideal for our training.

For testing, we embed a query through the model and use the cosine similarity function from the torch library with our model's tensor. We first test with a random query we generate and see if the list of recommendations is acceptable and has a good rating. For example, using a query "potato mushroom stew dinner easy-to-make" passed to the recommender function we get the following result.



```
        recipe_id                             name  \
204785      24396   crispy mushroom   onion potatoes
140942     385970        browned leeks and mushrooms
3258        12100           amazing stuffed  shrooms
190707      58676         chicken and mushroom saute
43913       51489                scalloped mushrooms
130798     315763  gilroy marinated garlic mushrooms
29209       44786             fresh creamy mushrooms
100208     192604         senior s sauteed mushrooms
192967     360855      the best beef stroganoff recipe
176195      47015        sauteed mushrooms with garlic

                                                    tags  rating
204785  weeknight time-to-make course main-ingredient ...    5.00
140942  30-minutes-or-less time-to-make main-ingredien...    5.00
3258    30-minutes-or-less time-to-make course prepara...    4.50
190707  bacon 30-minutes-or-less time-to-make course m...    4.50
43913   60-minutes-or-less time-to-make course main-in...    4.50
130798  15-minutes-or-less time-to-make course main-in...    4.25
29209   30-minutes-or-less time-to-make course main-in...    4.00
100208  15-minutes-or-less time-to-make main-ingredien...    4.00
192967  30-minutes-or-less time-to-make course main-in...    4.00
176195  15-minutes-or-less time-to-make course main-in...    2.50
```

It is seen that the recommendations are in line with query and match our criteria for a close enough recipe. On testing with queries directly from the training data, we get the exact recipe that we wanted. Even with scrambling of the tags, we tend to get the recipe in the top few recommended recipes.

4 Conclusion

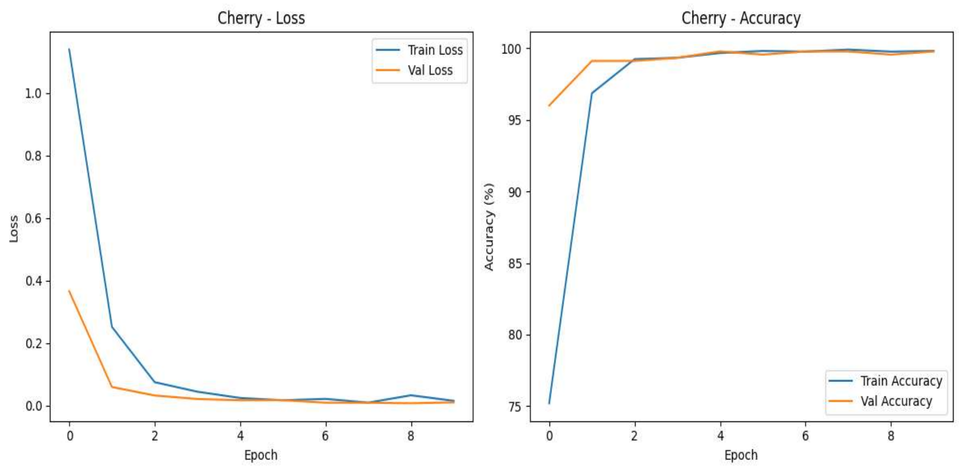
In our paper, we investigated the effectiveness of transfer learning in the integration of SOTA deep learning models in a common-day application. This experiment showed that not only can these types of models be applied in a variety of use-cases, but they're also incredibly flexible in their usage. From the point of model evaluation, integrating these models in the intended application was an exciting task which took intricate planning, but ultimately taught us a way to put our domain knowledge into use. Although the application is at a small scale currently, it would be quite easy to begin expanding the model to include a plethora of modern-day ingredients to make a much more fleshed out end-user application.

References

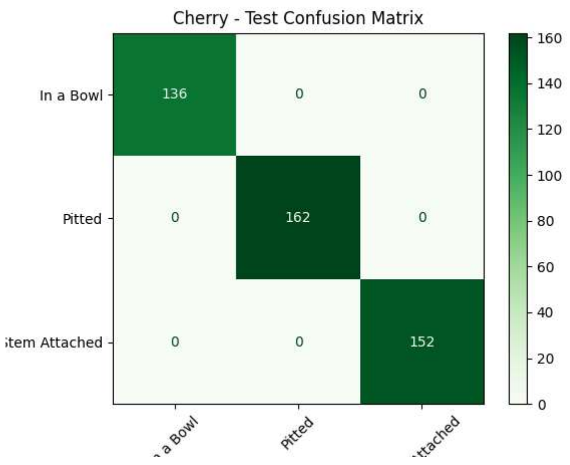
1. maintainers, TorchVision, and contributors. "TorchVision: PyTorch's Computer Vision Library." *GitHub*, 1 Nov. 2016, github.com/pytorch/vision.
2. He, Kaiming, et al. *Deep Residual Learning for Image Recognition*. 10 Dec. 2015.
3. Devlin, Jacob, et al. *BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding*. 24 May 2019.
4. Reimers, N., & Gurevych, I. (11 2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Retrieved from https://arxiv.org/abs/1908.10084

Appendix

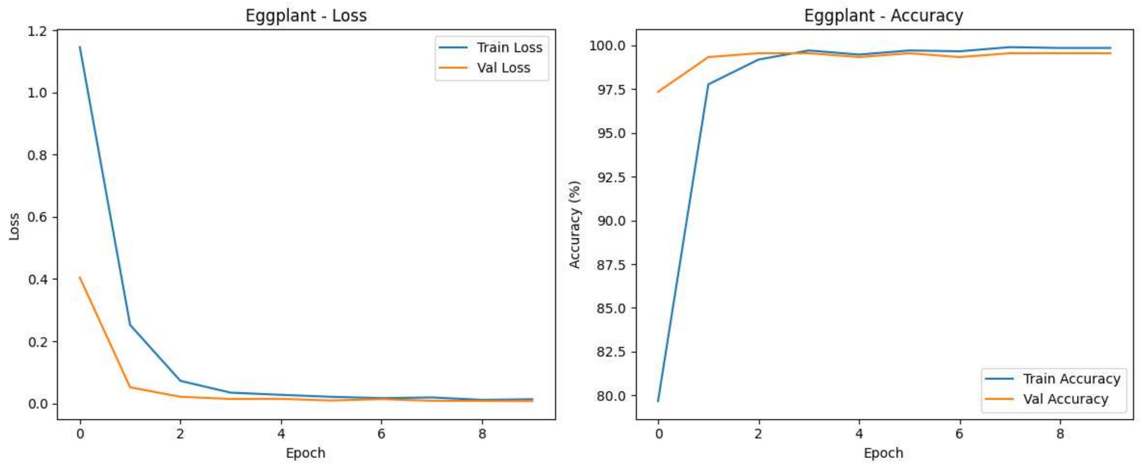
i. Cherry - Loss/Accuracy Plots:



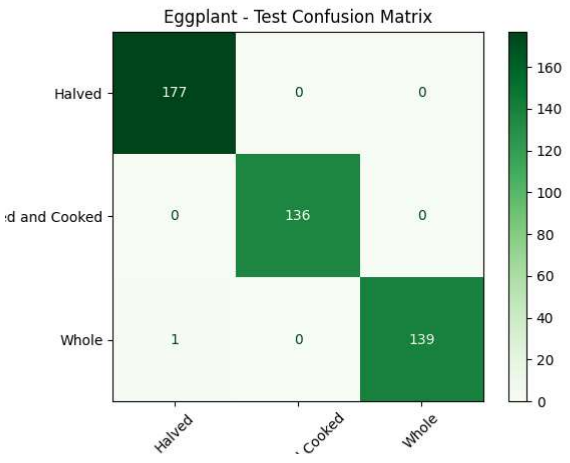
ii. Cherry - Confusion Matrix:



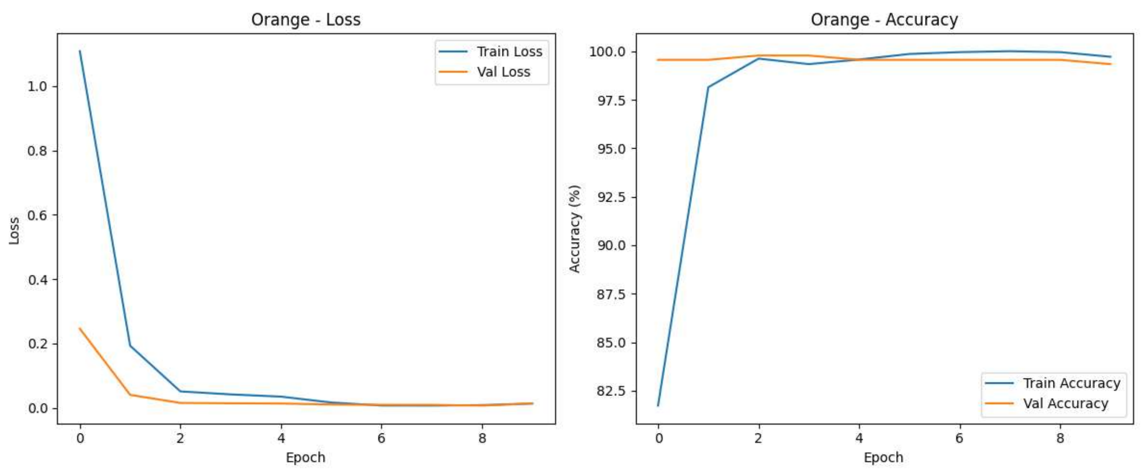
iii. Eggplant - Loss/Accuracy Plots:



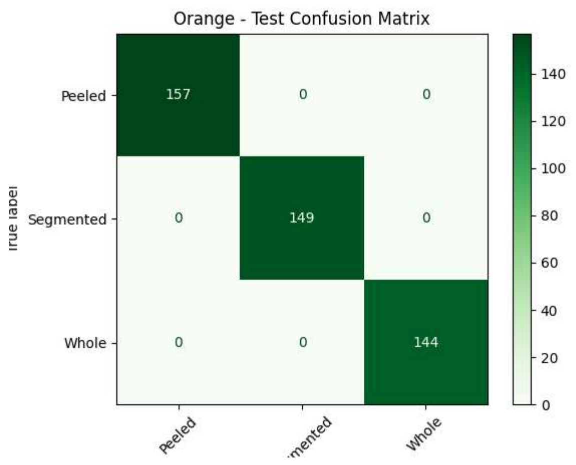
iv. Eggplant - Confusion Matrix:



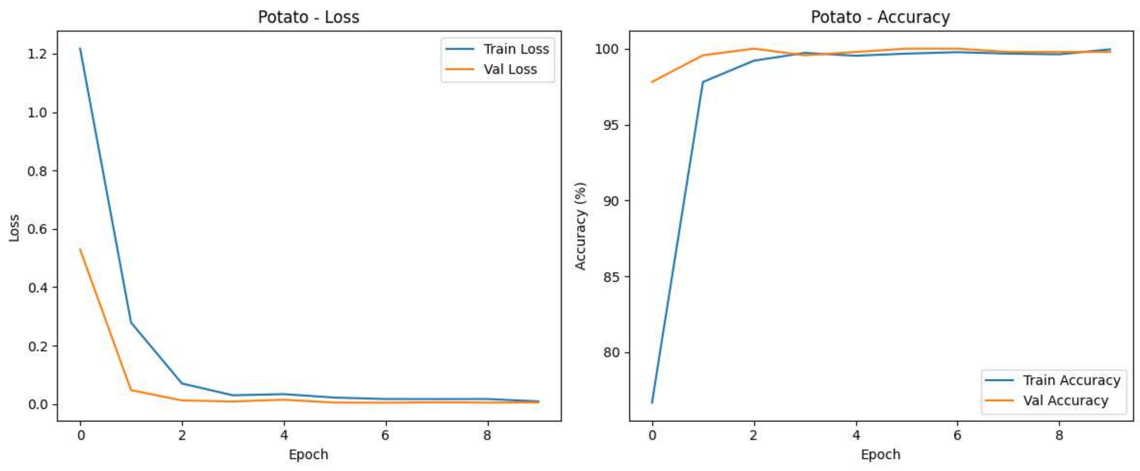
v. Orange - Loss/Accuracy Plots:



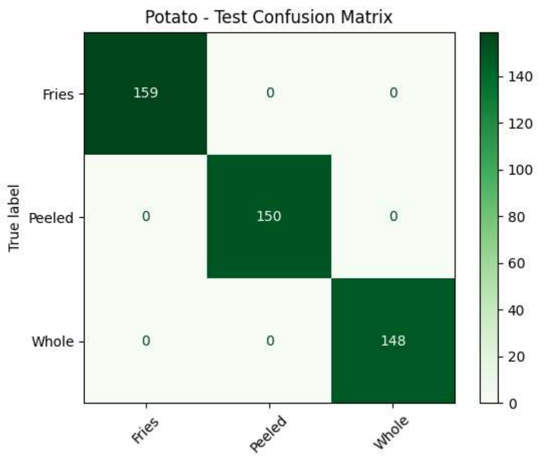
vi. Orange - Confusion Matrix:



vii. Potato - Loss/Accuracy Plots:



viii. Potato - Confusion Matrix:


Potato - Test Confusion Matrix
True label
Fries
Peeled
Whole