Quality Based Fruit Classification

By Evaluation of Common Bangladeshi Fruit

Adam Hirshson

Introduction

My project is on the multiclassification of images, by training on a data set of 15 variations of ripe, rotten, and chemically preserved fruits of various types sourced from the FruitVision dataset (Bijoy et al.). The topic is worth exploring as classification of food can have important impacts on automated supply chain and food safety systems, allowing a computer to separate/make informed decisions based on product quality. Classification based on the FruitVision dataset is more in-depth than a standard fruit-classification project due to the similarities shared between fresh and preserved fruit, requiring specialized models to detect the subtle differences

Much of the difficulty in this project was calibrating the convolutional models to accurately and consistently distinguish fresh and preserved fruits

Related Work

Fruit classification models have seen many renditions with varying results, but many of them use data sets of only ripe and rotten fruit, and do not include formalin preserved fruits. The models trained without preserved fruit, while valuable for some tasks, will likely make false predictions when used on preserved fruit, potentially leading to harmful impacts on consumers due to formalin's toxicity.

Morshed et al. used the FruitNet data set, made up of over 19,000 images of popular fruit in India. The data was partitioned into a 60:20:20 ratio of training, validation, and test. Data was augmented at run time with different methods at each training epoch, such as rotation, flipping, shifting, and sheering. The augmentation per-epoch was so that the model could train on a wide variety of data with randomizing intensities of augmentation. The authors tested multiple pretrained CNN models by fine-tuning the top layers. The models fine-tuned were MobileNetV2, ResNet152, ResNetV2, and DenseNet201, with DenseNet201 having the highest accuracy, 99.26% (Figure 1). The authors measured model accuracy, prevision, recall, F1-scores, AUC-ROC, and support. One challenge the authors faced was data augmentation, stating that traditional methods such as under or oversampling would lead to losing potentially important data. They solved their challenge by implementing their per-epoch augmentation strategy.

Architecture	Accuracy (%)	Trainable Parameters (millions)
ResNet152	97.86	58.26
InceptionResNetV2	97.91	54.3
EfficientNetV2B0	97.95	5.88
VGG16	98.6	14.72
MobileNetV2	98.62	2.25
InceptionV3	98.8	21.81
Xception	98.98	20.84
DenseNet201	**99.26**	18.13

Figure 1: Morshed et al. page 3

In addition to fine-tuning popular models such as ResNet and MobileNet, Palakodati et al. introduced their own convolutional neural network specifically trained for fresh and rotten fruit classification (Figure 2). Their model contained three convolutional layers, spaced out with normalization, activation, and max pooling layers. These layers were used to adjust the weights and size of the images throughout processing. Lastly, the fully connected layer, made up of flattening and dense layers, was used to predict one of the many classes. Something interesting to note is that their filter amounts stayed relatively small (16 filters each), but their kernel size got larger in each layer, progressing from 3x3 to 5x5 and 7x7.

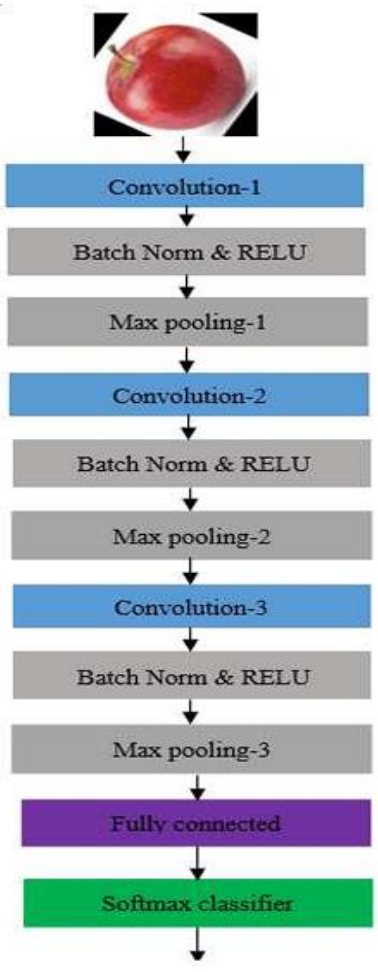


Figure 2: Palakodati et al. page 619

In the results of their paper, they tracked their optimizers, learning rates, batch-sizes, and epochs. Their findings concluded that the highest performing hyper-parameterization was with the Adam optimizer, followed by RMSprop, a learning rate of 0.0001, 225 epochs (the most they tested), and a batch-size of 16. Some interesting observations are that increased learning rates lead to a steep drop-off in accuracy after 0.001, accuracy saw a decline of roughly 20% when going from 32 to 64 batch-sizes, and the increase in accuracy due to the number of epochs was nominal going from 50 to 225 (Palakodati et al.). Their custom model and transfer learning results are shown in Table 1 below.

Model	Accuracy
VGG-16	89.42
VGG19	76.18
MobileNet	68.72
Xception	78.68
Proposed Model	**97.82**

Table 1: Palakodati et al. model accuracies

Data

Acquisition

I used the unmodified FruitVision dataset created by Bijoy et al., comprised of 10,154 images of fruit, with class frequencies in Table 2.

	Apple	Bananas	Grapes	Mangos	Orange
Fresh	765	749	770	763	753
Rotten	630	632	630	630	656
Formalin	643	660	610	616	647

Table 2: FruitVision Class Frequencies

The fresh and rotten fruit were from vendors local to the authors in Bangladesh. The authors took all the images with the same camera. The formalin preserved fruit data was created with a 1:9 formalin to water. Constant concentration of formalin prevented formalin variation from skewing the data. Labels were assigned by the data collectors, samples of which are depicted in Figure 3.


(a) Formalin Preserved


(b) Fresh


(c) Rotten

Figure 3: FruitVision Apple Class Samples

Model Implementation

Libraries

Models were created using Python Tensorflow and its Keras library, the Matplotlib library's pyplot function was used to graph the models' metrics, and scikit-learn metrics was used for the confusion matrix. Tensorflow does not work with Python versions after 3.12 (3.12 subversions are acceptable), and can be installed with, "pip install tensorflow". I developed with version 2.19, which can be downloaded with, "pip install tensorflow=2.19". "pip install matplotlib" and "pip install scikit-learn" can be used to download the other libraries.

Pipeline

First, I moved the dataset into my local project directory and loaded it into a dataset variable via the Keras "image_dataset_from_directory" function. Then split it into training, validation, and test sets; I applied preprocessing only to the training and validation sets.

I ran the training and validation sets through many different convolutional neural network model designs. Some design patterns tested are the following:

1. CNN blocks made of two CNN layers, pooling, and dropout, with increasing filter amounts ranging anywhere from 16-256
2. single CNN layer blocks, with pooling and no dropout, with constant filter amounts, but increasing filter kernel sizes
3. single CNN layer blocks with increasing filter numbers (ranging from 16-256) and constant filter sizes, pooling, and dropout.

All models had a flattened fully connected layer at the end. Hyper-parameters such as optimizer learning rate and batch-size fluctuated from 0.0001-0.005 and 16-64 respectively,

however only the Adam optimizer was used. Models with increasing filter numbers per layer did not necessarily always go from 16-256, some went 32-256, others went 16-64. A callback function with patience values ranging from 5-8 was used to prevent training that was likely to be unproductive. All custom CNN designs used the Relu activation function, but placement varied from being built into the CNN layer and being its own layer after batch normalization.

Fine-tuning was done with the MobileNetV2 due to its relatively low number of parameters compared to higher performing models, such as DenseNet201. The model was frozen, given a new top layer to fit the classification task and fitted on the FruitVision dataset. The top 50 layers were unfrozen for fine-tuning. The same callback and optimizer configurations as the custom CNNs were used for the MobileNetV2 transfer learning.

All custom models and transfer learning models were evaluated on the performance metrics described in the Performance Measurements section.

Preprocessing

Once I downloaded the unmodified dataset from the Data Mendeley page for the FruitVision paper (Bijoy et al.) and removed one of the directory layers so that the images could more easily be accessed. I split the dataset using a 60:20:20 partitioning for training, validation, and testing. The images were then augmented via rotation, horizontal and vertical flipping, zoom, rescaling, and translation. Images were resized to a variation of dimensions, such as 28x28, 32x32, 64x64, 128x128, 254x254, and 512x512 throughout testing.

Performance Measurements

I chose three metrics to measure my models: accuracy, loss, and confusion. Accuracy to get an idea of how well the model is doing overall, and a confusion matrix to analyze what the model struggles with. Accuracy and loss were tracked with the Keras.Model “fit” method and visualized with matplotlib’s pyplot. The confusion matrix was calculated when evaluating on the test dataset and was displayed with the scikit-learn library. Model training times ranged from 90-450 seconds per epoch, with a full training run being 30-50 epochs, unless stopped by an early stopping callback function.

Results

Findings

Dozens of layer designs and hyper-parameterizations were evaluated; results of only a few models were saved in Table 3. The same optimizer, Adam with learning rate of 0.0001, was used for all recorded models, chosen based on the findings of Palakodati et al.. Table 4 displays the metrics for the recorded models. Table 5 provides the index-to-label key used in the confusion matrices shown in Figure 4.

A MobileNetV2 model was given a new top layer to fit this classification task, and its results were recorded both prior and post fine-tuning, rows T1 and T2 in Table 4. It trained on 256x256 image sizes, with a fully connected layer of global average pooling and dense layers with 128 and 15 neurons, using the Relu and Softmax activation functions within the layers respectively. The top 50 layers were unfrozen for fine-tuning.

Model	CNN Blocks	Layers per Block	Increase Filters	Increase Filter Sizes	Batch Norm.	Activation with CNN	Dropout	Image Dimensions
C1	4	2	Yes	No	No	Yes	Yes	512x512
C2	3	1	No	Yes	Yes	Yes	No	512x512
C3	3	1	No	Yes	Yes	No	No	254x254
C4	3	2	Yes	No	Yes	No	No	128x128

Table 3: Custom CNN Designs

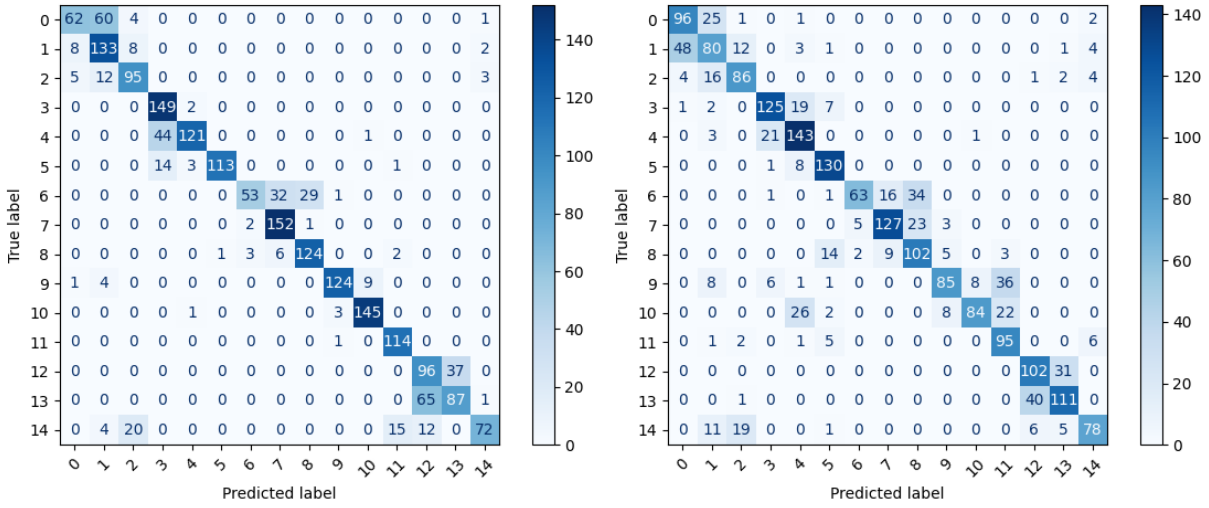
Model	Final Training Accuracy	Final Validation Accuracy	Evaluation Accuracy	Evaluation Loss	Confusion Matrix
C1	79%	78%	79%	0.52	Figure 4.a
C2	79%	61%	72%	0.72	Figure 4.b
C3	80%	81%	82%	0.44	Figure 4.c
C4	79%	78%	78%	0.55	Figure 4.d
T1	87%	79%	81%	0.46	Figure 4.e
T2	95%	90%	90%	0.30	Figure 4.f

Table 4: CNN Performance

Apple Formalin-Mixed = 0	Apple Fresh = 1	Apple Rotten = 2
Banana Formalin-Mixed = 3	Banana Fresh = 4	Banana Rotten = 5
Grape Formalin-Mixed = 6	Grape Fresh = 7	Grape Rotten = 8
Mango Formalin-Mixed = 9	Mango Fresh = 10	Mango Rotten = 11
Orange Formalin-Mixed = 12	Orange Fresh = 13	Orange Rotten = 14

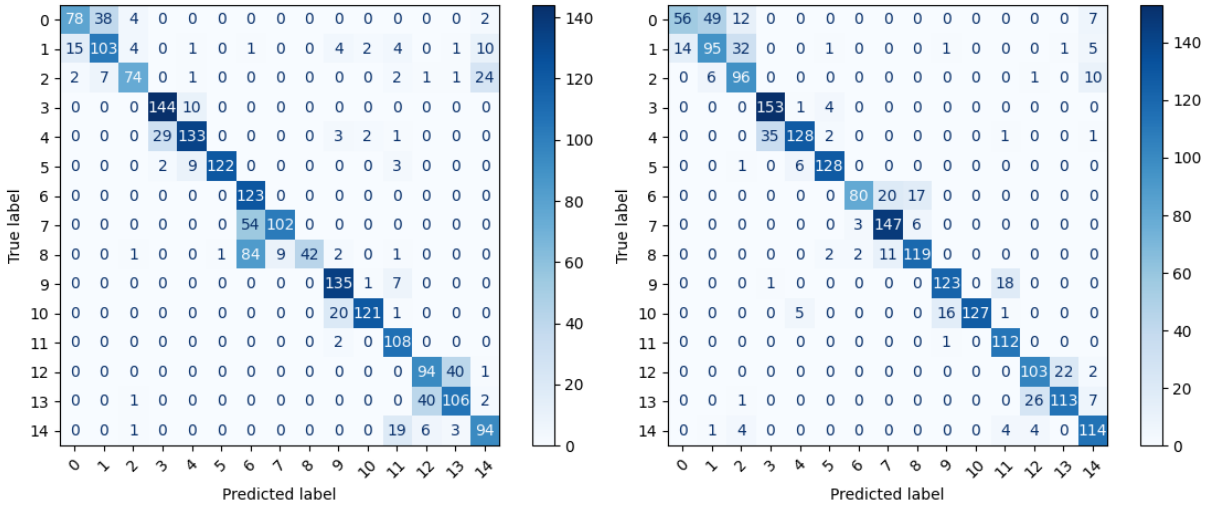
Table 5: Confusion Matrix Label Key

Figure 4: Confusion Matrices



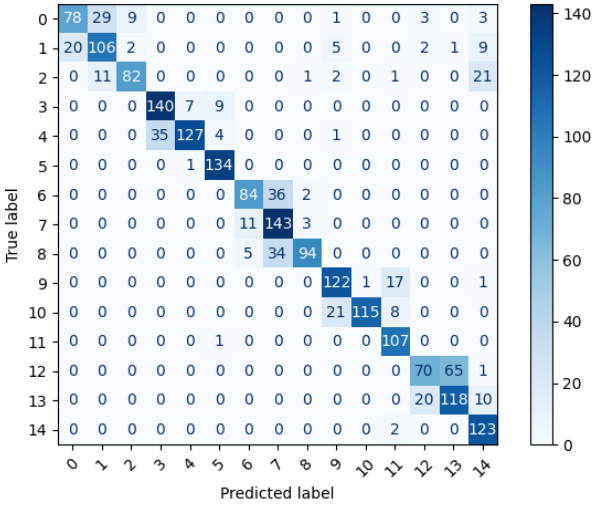
a

b

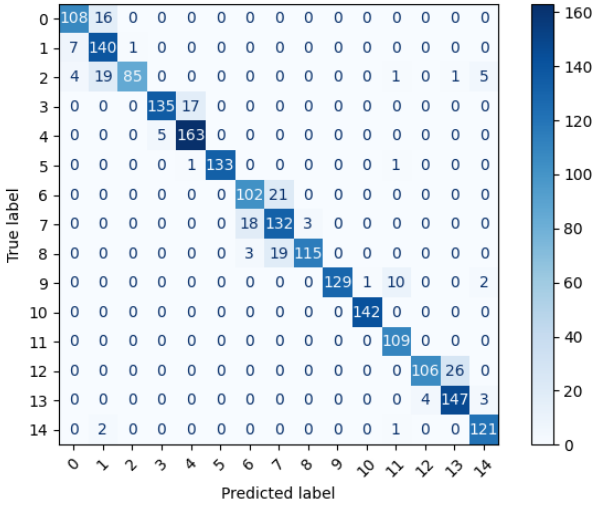


c

d



e



f

Interpretation

The highest performing custom model was C3 at 82% evaluation accuracy, which was surprising as my hypothesis going into the project was that the more complex architectures (those increasing in filter numbers as layers increased) would be the most efficient at this task. However, C3 was the simplest architecture, with 16 filters per CNN layer. C3 was a near copy of the CNN described in the Fresh and Rotten Fruits paper (Palakodati et al.), with only some of the weight initialization and regularization missing and trained for less epochs. The second most accurate model was C1, which differed strongly from the C3 design. C1 had multi-CNN-layered blocks with no normalization and increasing filter amounts. Despite having nearly equal accuracy, it was much less efficient as it was more complex and less accurate. Each model did very well at distinguishing the type of fruit (not classifying an apple as an orange for example), however the models had issues at times with separating the fresh and formalin-mixed fruit, see square permutations of (0,1), (6,7), and (12,13) in Figure 4 for examples.

In many of the unrecorded designs, mass overfitting was observed, such as training accuracies in the 90-95% range, and validation accuracies in the 60-65% range. The steps taken to achieve convergence were implementing batch normalization, increasing image size (original models used dimensions 32x32 and 64x64), and extracting activation functions to their own layer. Training and validation measurements for all the recorded custom designs either achieved or approached convergence.

The transfer learning model with MobileNetV2 saw validation accuracies greater than and equal to that of the custom models, with the fine-tuned model reaching 90% accuracy. Both

the base MobileNetV2 with the custom fully connected layer and the fine-tuned model showed overfitting, contrary to some of the weaker performing custom models, but had overall higher accuracy. The MobileNetV2 model did not suffer from confusion between fresh and formalin mixed fruits nearly as much as the custom models.

Final Discussion

Future Work

Possible improvements to this project would be to reattempt it with stronger hardware, as the high training times restricted the experimentation. Some potential CNN designs to test would be to rebuild C3 with multi-layered blocks, and C1 with changing filter sizes and batch normalization. Another interesting experiment would be to test the current customizations on datasets without formalin preserved fruits, to see how the models would perform. Based on the confusion matrices, the likelihood of achieving the original goal of 90% accuracy outlined in the interim report is high.

The preprocessing process was rudimentary, with only the image size and position being adjusted. Due to some images containing shadows, such as Figure 3.b, it would be worth exploring to see how model performance changes if the dataset was preprocessed such that the shadows were removed or minimized. The authors of the FruitVision dataset also created an augmented version of the dataset containing nearly 8x the number of images, it would be interesting to see how their data augmentation compares to my augmentation in terms of model accuracy.

Given how well the MobileNetV2 model performed in comparison to the custom models, it would be worth testing different differently constructed fully connected layers, as well as training with more than the 30 epochs used in this instance. Use of the DenseNet201 model is also worth exploring, as its performance was shown in Morshed et al. to be the highest for fruit classification out of the popular pre-trained models tested. However, it was not used in this project due to its high complexity leading to training times that were unfeasible, such as epoch times being an hour and up.

Finally, the lower than anticipated accuracies (using the 97% and 98% accuracy values achieved by Palakodati et al. and Morshed et al. as goals) of the models outline the importance of specialized models for detecting formalin preserved fruits in safety and inspection systems, due to how similar they appear to the naked eye.

References

Hasan, M., Syeda Zarin Tasnim, Syed Ali Awsaf, & Hasan, M. Z. (2025). FruitVision: A Benchmark Dataset for Fresh, Rotten, and Formalin-mixed Fruit Detection. *Data in Brief*, *61*, 111752–111752. https://doi.org/10.1016/j.dib.2025.111752

Md. Samin Morshed, Ahmed, S., Ahmed, T., Muhammad Usama Islam, & A.B.M. Ashikur Rahman. (2022). Fruit Quality Assessment with Densely Connected Convolutional Neural Network. *ArXiv (Cornell University)*. https://doi.org/10.1109/icece57408.2022.10088873

Palakodati, S. S. S., Chirra, V. R., Dasari, Y., & Bulla, S. (2020). Fresh and Rotten Fruits Classification Using CNN and Transfer Learning. *Revue d'Intelligence Artificielle*, *34*(5), 617–622. https://doi.org/10.18280/ria.340512

Hasan, M. (2025). FruitVision: A Benchmark Dataset for Fresh, Rotten, and Formalin-mixed Fruit Detection. *Mendeley Data*, *2*. https://doi.org/10.17632/xkbjx8959c.