INTRODUCTION [4] Paragraph to briefly describe the problem and summarize approach

The problem involves classifying images of cherries, strawberries, and tomatoes. The dataset presents several challenges, including variation in image composition, resolution, and the presence of noise such as non-fruit objects. To address these issues, pre-processing techniques are applied, including uniformly resizing images, normalizing pixel values, and removing or augmenting noisy data. The model is trained using a Convoluted Neural Network, where hyperparameters such as loss and activation functions, batch sizes, optimization techniques, and learning rate are tuned using unseen data. The goal is to develop a robust image classification model capable of generalizing well across diverse images between the fruit.

PROBLEM INVESTIGATION [15] Describe EDA and justify preprocessing / model decisions

Initial Observations

1. The majority of images are square with dimensions 300 x 300. This uniformity is achieved at the cost of a handful of images appearing compressed. While this would have an effect on the trained model, as long as all images are resized in preprocessing, such consistency reduces this concern

2. There are close-up shots and distant shots. This is generally preferable for its variety at the cost of size no longer being a predictive factor as close-up shots of cherries may resemble tomatoes.

3. On the topic of variety, images contain a wide range of backgrounds, lighting conditions, shadows and viewpoints. This variability should be kept to ensure a robust model, *but only relevant variety*.

4. The data includes both ripe and unripe fruits, especially for tomatoes and strawberries, appearing green. Such images won't be removed for training as they still represent the corresponding fruit.

5. **There are some duplicate images**. While this may skew the distribution, random selection and shuffling during data augmentation should mitigate any potential bias caused by these duplicates.

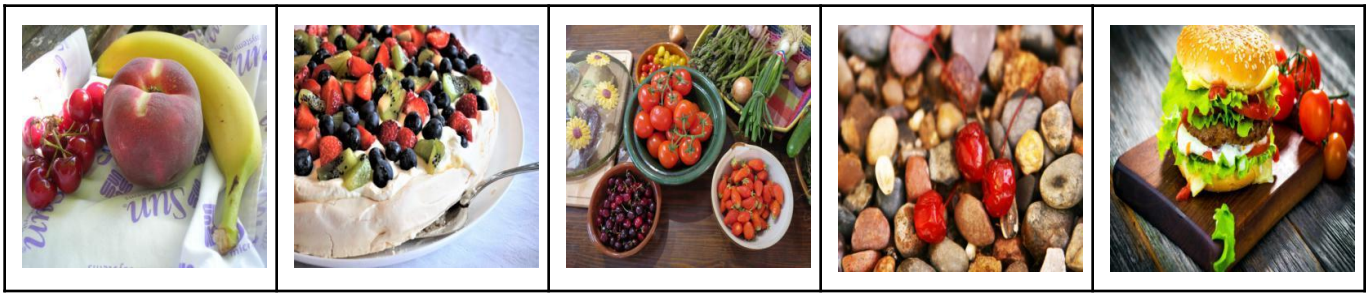
- **There exist plenty of Low-Quality images <u>that need to be removed</u> notably the following:**

1. Relevant variety was mentioned prior. Irrelevant variety are abstractions of the fruit, such as text, cartoons, food without the fruit visible etc. This introduces noise that only confuses the model.

2. A decision was made to remove all images containing people who serve as the primary subject of the image despite the fruit being visible. This is because the fruit is often so small and eclipsed.

3. Any misclassified fruit must be removed. Secondly, there exists a minority of cherries which due to color, shape, and (relative) size appear too similar to tomatoes. Controversially, these images were removed to reduce model confusion. This shows the difficulty in obtaining 95%+ accuracy.

4. Any image where the fruit was a considerable minority, e.g. placed alongside other ingredients, objects were to be removed despite the fruit being present. This is not to be confused with a plain background and the fruit appearing small. The issue lies in the abundance of edges which detract from focusing on the texture, shape, and color of the fruit to aid in adequately training the model.



5. Many strawberries are depicted as desserts. While this introduces noise, it may be advantageous due to similarities in angle and presentation enabling increased predictability. As such, as long as the desserts clearly depict the fruit (such as the images below) these images of products are kept

Post Image Removal

After applying the image removal criteria, the dataset was reduced from a balanced 1500 images per fruit to 1200 Strawberry, 1300 Cherry, and 1350 Tomato. Strawberry images saw the largest reduction due to said deserts and noisy composition. Such imbalanced data is a cause for concern because it can lead to biased performance. Therefore to address this, while increasing training data, synthetic images are made

Generating Synthetic Data Methodologies.

1. Web browser Stable Diffusion was the tool used which has an ImgtoImg mode. This allows an existing image to be used as *inspiration* for the generated images alongside a text-based prompt. Automation can be achieved via batches, enabling many images to be run through this mode.

2. The images used in the batch were hand-selected, *good* real images of the fruit that emphasized the texture, shape, color, shine, designed to clearly separate the three fruit. 3 synthetic images were generated for each image in the batch. Though composed similarly, each is slightly different.

- 800 synthetic strawberries, 650 synthetic cherries, and 600 synthetic tomatoes were generated resulting in 6000 total images with a balanced 2000 images for each fruit. These are optimal numbers because generating any more would saturate the training data with artificial images.

- Too many artificial images is undesirable as the test set does not contain such images. Synthetic images are too perfect, too clean. Their purpose is to only capture the desired visual attributes.

- To aid in this purpose, the hand-picked images used in the batch were selected with variability e.g. zoom ins, zoom out, fresh harvests, photography etc while primarily focusing on the fruit only. The generated images were square by default and automatically saved enabling easy integration.

Preprocessing Methodology and Justification

1. Random cropping was applied, varying between 0.8 to 1.0 of the original size. This adds variety while simultaneously resizing all images to a consistent 300 x 300 dimension. Such processing enables the model to learn from different perspectives without introducing excessive distortion.

2. Random Horizontal flipping was incorporated to further augment the data. Vertical flipping was deliberately omitted given the contexts in which the fruits were usually presented (e.g. shadows, tables, bowls etc). Horizontal flips enhance data diversity while retaining natural orientations.

3. Color jitter is a further technique employed to vary the brightness and contrast of the images. The introduced variability aids the model in generalizing under different environmental conditions that may be encountered in the real-world. IMPORTANTLY, hue is not to be adjusted as it distorts the natural color of the fruit, leading to unrealistic representations that significantly confuse the model.

4. Random rotation was considered but ultimately not implemented due to the introduction of black borders around the images. The addition of these borders creates noise that misleads the model during training. Ultimately this highlights diminishing returns regarding the addition of variability.

5. A further preprocessing technique is feature extraction like Gaussian Blur and Sobel Filter for edge detection. The decision to avoid Gaussian Blur or Sobel Filter was based on simplicity and the need to preserve the original RGB attributes of the images as there's plenty of value there.

6. Lastly, the dataset was normalized using ImageNet mean and std making the input more uniform for the model. Standardizing input distributions improves model performance as much as it is a common preprocessing step in machine learning. This is at the expense of less human visibility.

(Images 1 depict feature extracted images –sobel filter, and the effect of hue, cherries look like tomatoes)

(Images 2 depict the similarity in the 3 generated images from one inspiration, and post-preprocessing)

METHODOLOGY [30] Describe and justify choices made when training the Convoluted Neural Net

#1 Investigating Training and Validation Splits

- Prior to removing any images, a training-validation split was conducted to benchmark the base performance using unseen data. A split of 0.8 training to 0.2 validation would suffice in estimating.

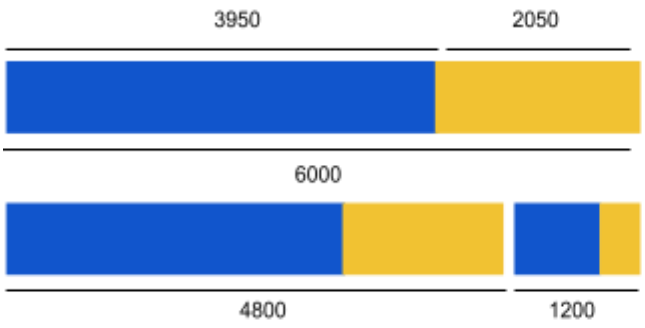
- Removing weaker images (via EDA) resulted in a 3% average increase in validation accuracy on a simple CNN with 2 convolutional layers compared to training without the removal of images.
- The issue this presented was that the fruit were now unbalanced which justified synthetic data.

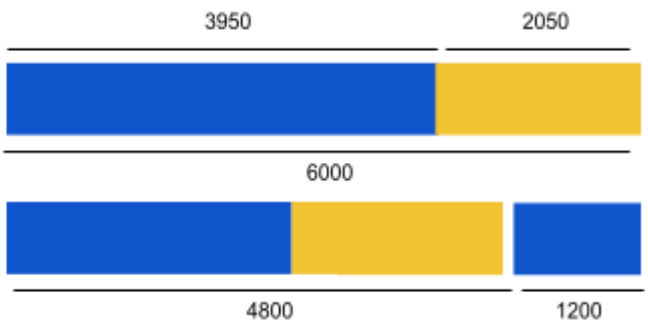
#2 Investigating Getting More Data / Synthetic Data

- Details of generating synthetic data were covered in the previous section. This section highlights the effects that adding more data had on predictive performance. On top of removing the weaker images, the addition of synthetic data, which led to 6000 total images (4800 training post split), resulted in a surprising 24% average increase in validation accuracy on our simple CNN model!

- This figure would be dismissed as the default split meant both training and validation sets contain artificial images. This is an unrealistic metric since the real test set would only contain real images

- The solution was to use a new split which calculated how much to take from the 3950 real images to keep a validation split of 0.8:0.2 ensuring the validation set only contained real unseen images.





- The result was a large departure from the 24% average increase, instead opting for a sizable 6% average increase in validation accuracy. This is a desirable gain as although it suppresses our metric lower, it is a realistic stand-in for unseen test data and we stand to gain more when tuning.

- This new split now sets the proportion of artificial images to real images in the training set at 43:57. This highlights how generating any more synthetic images would make them the majority of the training data which would not be optimal as they only stand to supplement the real images.

- As a final mention regarding data splitting. Cross validation is a suitable method in presenting a robust metric of the model's performance with all the data provided. Despite this, with CNN, cross validation takes arduously long to finish. As a result, a basic validation split as described above would be efficient when simply exploring / experimenting with the upcoming CNN settings. Cross validation would be saved strictly for final tuning / selection when there's 2-3 impressive models.

#3 Investigating Computational Resources

- Downloading Nvidia CUDA toolkit, CUDNN, and reinstalling pytorch with CUDA 12.4 were crucial steps to ensure torch could recognize and utilize my **GPU**. This allows tensors to be mapped to cuda cores resulting in noticeably faster CNN training compared to the CPU (default). The larger the task, the larger the separation between GPU and CPU completion times –Perfect for CNNs

- Such speed means more configurations can be tested in a given timeframe and a higher ceiling for CNN architecture or epochs (cross validation). The downside is whether other machines also have GPU support as a large CNN model may not be as equally speedy, particularly for testing.

#4 Investigating Optimisation Techniques

- As a baseline, **SGD** was used with a learning rate of 0.001. While effective, Stochastic Gradient Descent can be slow to converge as it uses a fixed learning rate throughout training. This became apparent when an increased learning rate of 0.01 resulted in a 4% gain in validation accuracy.

- In research, almost all CNN advice points to **Adam** as the go-to optimizer. This is because Adam combines the benefits of momentum and RMSProp, adjusting the learning rate individually for each parameter. This was justifiable as switching to Adam with learning rate 0.001 increased average validation accuracy by 8% over SGD 0.01, converging faster particularly in early epochs.

- Adam with 0.01 learning rate was tested and the validation accuracy dropped by a noteworthy 6%. This is due to overshooting caused by momentum and adaptive lr on top of the higher 0.01. Epochs are wasted on oscillations attempting to stabilize as opposed to a clean gradient descent.

#5 Investigating Regularization Strategies.

- An added benefit of Adam is the support for **weight decay** directly which works by constraining model weights, forcing it to learn more generalized patterns rather than memorizing the training data. By setting a weight decay of 1e-4, the CNN model improved validation accuracy by 3%. Interestingly, a weight decay of 1e-3 and 1e-5 both drop validation accuracy compared to 1e-4. This is because 1e-3 is strongly regularized i.e underfitting, while 1e-5 shows more overfitting.

- **Batch Normalization** was a strategy that was applied to the CNN architecture directly. After countless tests, the conclusion is that batch normalization simply results in reduced performance, often losing 9% validation accuracy. This may be a case of underfitting as batch normalization is a notably effective regularization strategy which may not bode well for our problem's complexity.

- Unlike Batch Normalization, implementing **Dropout** to the CNN architecture resulted in a notable 8% increase in validation accuracy on top of weight decay. Dropout is utilized in the dense layers, randomly deactivating neurons which forces the model to learn redundant, more general patterns. The expense of higher unseen accuracy is minor inconsistency +/- 2% due to 50% random drops.

- 0.001 lr Adam,1e-4 weight decay, 0.5 dropout, and 0 batch normalization are now staple features of the Convoluted Neural Network, consistently resulting in 70% or greater validation accuracy.

#6 Investigating Hyperparameter settings / CNN architecture

- **Kernel / Filter Width**: Initial 5. Smaller kernels 3x3 focus on fine details, detecting small-scale patterns like edges or textures and get more generalized the larger the kernel. In testing, only 5 and 9 gave rounded CNN outputs. 5 was found to perform better as intricacies weren't being lost.

- **Stride Length**: Initial 1. Similarly, testing a stride any larger than 1 reduced validation accuracy. Larger strides skip image details, leading to an incomplete understanding of texture, edges etc.

- **Pooling layer Dimensions**: Initial 2.Pooling layers reduce the spatial dimensions of the input, helping to simplify computations. Larger pooling (4) excessively simplified the feature map, losing info necessary for higher accuracies. Therefore 2 was kept as it balances with computational load

- **Activation Function**. Initial ReLU. The Relu activation function introduces non-linearity, enabling the CNN to reach optimal validation accuracies by learning more complex representations of the data. Other activation functions, such as sigmoid or tanh, simply didn't yield competitive results.

- **No. Convolutional Layers**: Initial 2. Adding a third and a fourth convolutional layer improved the CNN's ability to convert fine details into high-level patterns, increasing validation accuracy by 7% on the 4th layer. Interestingly, adding a fifth convolutional layer yielded diminishing returns while increasing computational cost / completion time potentially due to overfitting on the training data.

- **No. Dense Layers**: Initial 2. Similarly, introducing a third dense layer did not yield any significant change except increasing completion time regardless of the number of neurons in the new layer.

- **Dense Layer Outputs.** Initially Flat -> 120 -> 84 -> 3 taken from cifar10_tutorial. Following many tests, Flat -> 1028 -> 512 -> 3 yielded an acceptable increase in accuracy. 1028 -> 1028 -> 3 or outputs any higher than 1028 saw little to no gain except an undesirable increase in computation.

- **Convolutional Layer Outputs**: Of all hyperparameters, increasing these outputs always resulted in higher validation accuracy at the cost of a definite increase in training completion time. With 4 convolutional layers, outputs 3 -> 18 -> 90 -> 360 -> 1080 (3 x 6 x 5 x 4 x 3) was an acceptable set, enough to average 77% validation accuracy in 10 epochs with everything above considered. These could be increased further, but without a GPU in testing, it may be inconvenient waiting.

#7 Investigating Batch Sizes

- As of all prior CNN testing, the batch size was an overlooked factor which remained at 96 since it greatly outperformed a batch size of 4 during very early MLP testing. Testing a batch size of 256 significantly reduced CNN validation accuracy which led to a deeper dive into this subject matter.

- A batch size of 4 was retested but with the CNN. This likewise saw reduced performance vs 96. However, a batch size of half, 48, improved validation accuracy by 2%, and half again, a batch size of 24 added another 2% on top of that. Any less resulted in weaker CNN performance.

- This suggests that larger batch sizes may not provide enough gradient updates, resulting in slower convergence and poorer generalization. Whereas extremely small batch sizes, 4, 8, likely demonstrated gradient overfitting while also taking a longer time to finish training. (450s to 520s).

#8 Investigating Any Extra Technical Advancements

- **Learning Rate Scheduler.** Although Adam provided an adjustable learning rate, out of curiosity, OneCycleLR learning rate scheduler was tested with 0.01 specified as the max_lr. Surprisingly, after 10 epochs, the validation accuracy sky-rocketed to 84% and the training set accuracies were hitting 80% in just 3 epochs. This highlights the strength of OneCycleLR which boosts the learning rate up to the max 0.01 in early epochs to overcome local minima / plateaus and slowly lowers the learning rate in later epochs to ensure finer adjustments and prevent overshooting. Lowering the max_lr to 0.001 almost negates OneCycleLR as early epochs get stuck around 70%

- **Epochs.** This is the final adjustment that determines whether the CNN overfits or generalizes well with unseen data. All the above testing was conducted with 10 epochs, the key being consistency in order to accurately evaluate the changes being made. With the CNN architecture completed, to properly assess epochs, validation accuracy will now be tested for each epoch for 20 epochs. Results/graphs of this are provided in the Summary section after describing the final structure.

SUMMARY [7] Describe & compare the structure and settings of base MLP and best CNN

MLP structure and settings

6000 Total Images, cleaned and synthetic added
RandomResizedCrop 0.8-1.0, 50% HorizontalFlip
Brightness +/- 0.2 Contrast +/- 0.2. 0 hue change
ImageNet mean/std normalization. 24 Batch Size
4800 training, 1200 fully real unseen validation set

SGD Optimization 0.001 LR
Dense Layer 1: 270000 (flat) -> 1028
Dense Layer 2: 1028 -> 512
Output Layer: 512 -> 3

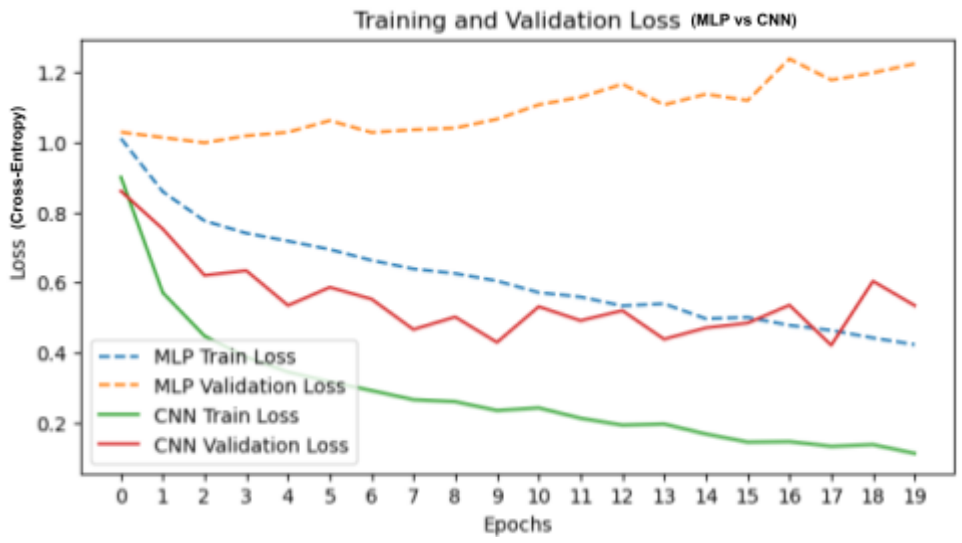
(identical dense layer sizes for a good comparison on the effectiveness of CNN's additional settings)

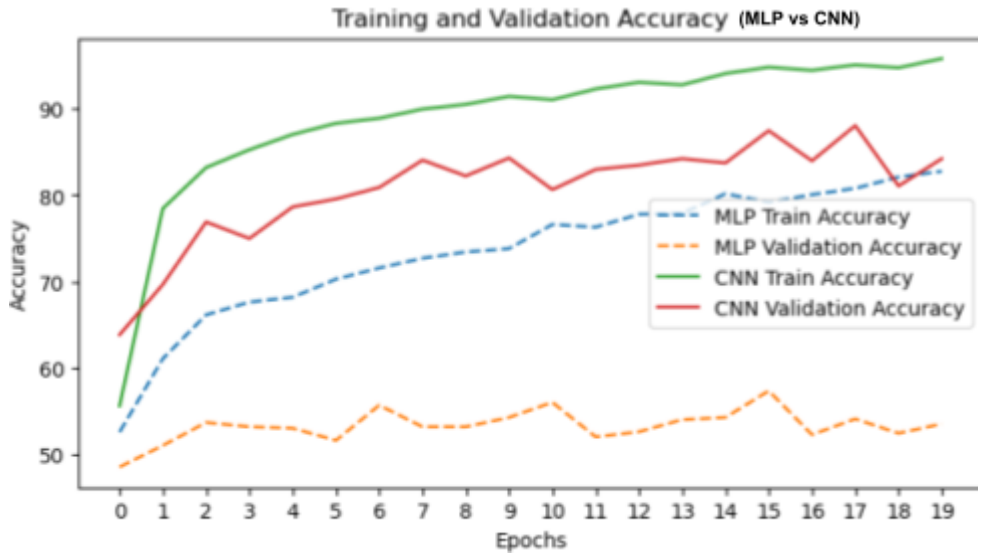
CNN structure and settings

6000 Total Images, cleaned and synthetic added
RandomResizedCrop 0.8-1.0, 50% HorizontalFlip
Brightness +/- 0.2 Contrast +/- 0.2. 0 hue change
ImageNet mean/std normalization. 24 Batch Size
4800 training, 1200 fully real unseen validation set

5x5 Kernel, 1 Stride, 2x2 Pooling, ReLU activation
4 CNN Layers 3 -> 18 -> 90 -> 360 -> 1080 output
Adam Optimization 0.001 LR, 1e-4 weight decay
OneCycleLR scheduler w/ 0.01 max learning rate.

Dense Layer 1: 81000 (flat) -> 1028 0.5 dropout
Dense Layer 2: 1028 -> 512 0.5 dropout
Output Layer: 512 -> 3





Immediately, CNN shows lower Cross Entropy Loss than MLP for both Train & Validation sets, followed by a steeper descent in loss in the early epochs.

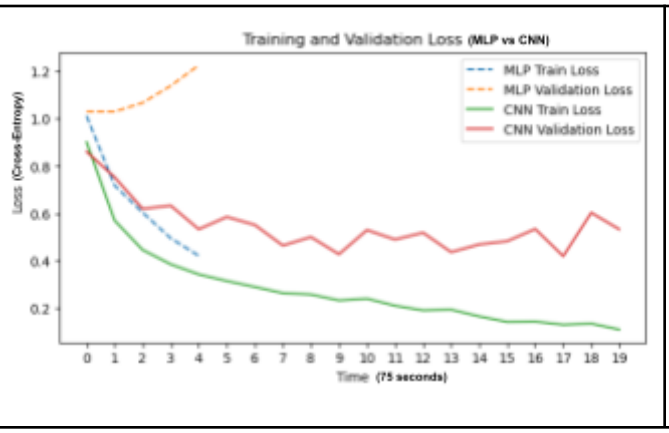
After 2 epochs, MLP loss for validation increases whereas training loss decreases. This highlights very early MLP overfitting.

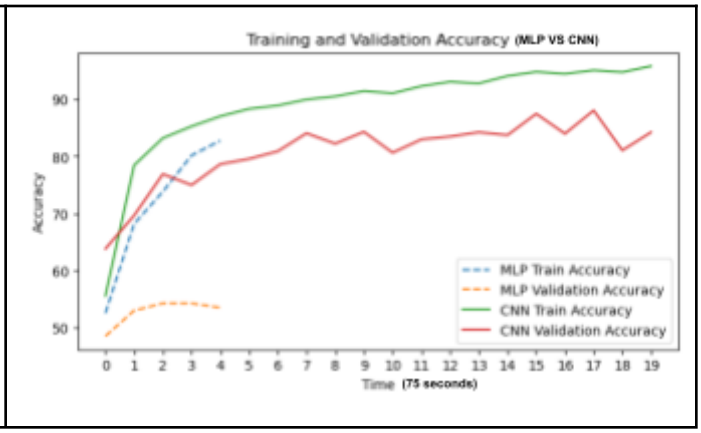
CNN training loss similarly decreases so does loss in validation which plateaus after 9 epochs instead of 2 again showing overfitting in the later half of epochs.

For accuracy, CNN again begins higher than MLP.

MLP validation accuracy hovers around the mid 50s whereas CNN climbs from mid 60s to around 85% in epoch 9 where it begins to show diminishing returns.

Random Dropout / weight decay may play a role in CNN's earlier oscillations particularly for validation.





Adjusting for time, where each increment on the x axis represents 75 seconds on the two graphs above, MLP completes 20 epochs in the time it takes CNN to complete 5 epochs. In that same amount of time, CNN still greatly outperforms MLP. This is because CNN's convolutional layers are specialized for image pattern recognition and further enhanced by Adam, Regularization, & OneCycleLR to push convergence.

CONCLUSION [4] Describe the overall conclusion, pros/cons, and insightful future next steps

The project successfully explored various strategies that enhance a Neural Network's ability to classify images of strawberries, cherries, and tomatoes. The addition of convolutional layers, regularization, and considerations into architecture and optimization strategies formed a model that on average, attains 30% higher unseen validation accuracy over a base Multi-layer Perceptron. Following a careful selection of real and synthetic images, preprocessing that attempts to reduce bias in a batch size of 24 images, and a justified 9 epochs of training, the final model, on average, classifies unseen images with 85% accuracy.

Pros

- Restricting the validation set to only contain real images unseen during training made for a robust metric to evaluate the model. Initially, this significantly lowered the validation accuracy to 52%. Therefore, finding strategies that increased this metric to average 85% inspires confidence.

Cons

- The selective removal of images during cleaning and the criteria used for the addition of synthetic images are subject to personal biases on what images are good / bad. While justifications were presented, the complexity of CNN's may support images previously considered to be weaker.
- The reliance on cuda to boost performance by enabling more complex CNN architecture to be run in reasonable time poses limitations toward testing on systems without access to such hardware. Mitigations would include reducing convolutional layer outputs at the expense of reduced results.

Future work

- Transfer learning was not pursued in this project in order to gain experience developing a CNN from scratch. This enabled a better understanding and control over the pieces that contributed to better classification performance. With this goal complete, future work may see the utilization of transfer learning to offer an efficient path to higher performance. This is because Pre-trained models using large datasets like ImageNet have already learned generalizable image features.

- Ensemble learning was not implemented due to the complexity of managing multiple models, which like transfer learning, contradicts the project's primary focus which is learning to develop ONE strong CNN model. Regardless, the added robustness and generalization that ensemble methods can provide would be highly beneficial in future work. Ensemble techniques such as bagging, voting, or stacking, could address slight inconsistencies in classification, increasing predictive performance while allowing the flexibility of different yet still optimal model settings.

- A deeper dive into validation metrics would be a worthwhile investigation for future work. This project strictly saw the use of accuracy to gauge model performance primarily because the data used was balanced and the metric was easily interpretable. Considerations into precision, recall, f1 and roc_auc especially may highlight better, robust CNN models that accuracy cannot capture.

- Reflecting on the methodology used in this project, hyperparameter tuning was performed entirely manually. While it provided valuable hands-on control, it was quite inefficient and increased time. Therefore, in future, automated optimization such as custom grid search functions or Bayesian optimization would enable a systematic approach to finding optimal learning rates, weight decay, batch sizes, and other parameters. This means certain combinations aren't missed due to human memory / error and better-performing, more precise configurations may be found in reduced time.