Project Final Report

Max L. Chen PID: 730841674
Yan Zhu PID: 730615250
Paeder Wall PID: 730720363
Dinara Aliyeva PID: 730793930

Abstract

This project aims to apply machine learning techniques to the problem of classifying fresh and rotten fruits and vegetables. It will do so with a convolutional neural network along with various pre-processing methods for optimization. The data processing pipeline will include both unsupervised and supervised training methods including principal component analysis, autoencoders, and various stochastic optimization methods. This report describes a preliminary test of the convolution neural network for classifying fresh and rotten produce with a single small dataset and default hyperparameters that resulted in a test accuracy of 0.9688.

1 Introduction

Ensuring the freshness of fruits and vegetables is a critical issue existing in all levels of the food supply chain, raised by farmers, retailers and consumers. An accurate identification method of rotten produce will not only reduce wasted food and financial loss, but also contribute to minimizing health risks caused by consuming spoiled food. With the increasing demand for efficient food quality control solutions in contexts ranging from supermarkets, warehouses to households, there is a need for automated and efficient approaches to detect the freshness of groceries.

Driven by this need, we address the problem of distinguishing between "fresh" and "rotten" produce images by using binary classification. One of the important tasks waiting to be solved is domain adaptation. The images of fruits and vegetables can vary greatly depending on environmental conditions like lighting conditions, camera quality, and background clutter. These variations often lead to performance drops in models trained on ideal or uniform data. Thus, our model should not only be accurate but also adaptable across image domains, as well as focusing on training efficiency and strategies like regularization and optimization to improve generalization.

To achieve the goal, our project approaches the problem by using convolutional neural networks (CNNs), which are resilient against translations of objects within an image and reduce features needed to train through pooling. However, despite their advantages, high-performance CNNs often require significant compute resources, large training datasets, and careful model tuning to avoid overfitting. To address these concerns, we aim to minimize computational and data burdens through fast prototyping and model refinement, careful selection of datasets and use of methods for optimization such as Stochastic Gradient Descent, which updates the weights per sample, or Adam optimizer, which updates per mini-batch with added adjustment of learning rate. In addition, we apply several data preprocessing techniques to improve model performance and accelerate training speed. We resize the images and apply principal component analysis (PCA) to optimize the training and testing process. We also use different training approaches, such as training from scratch, feature extraction, and transfer learning, to find the most efficient and scalable solution lightweight yet accurate classifier that can be used in diverse settings, from personal mobile apps to large-scale retail quality control systems.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

2 Related Work

2.1 Convolutional Neural Networks

The most common approach to perform image classification is Convolutional Neural Network, the use of which was highly popularized due to the paper Krizhevsky et al. [2012], which showed groundbreaking results in pure supervised learning on ImageNet dataset. The AlexNet network, introduced in this paper, has the architecture that consists of eight learnable layers, including five convolutional layers and three fully connected layers. This design extracts features in a hierarchical way, starting with simple details like edges and textures in the early convolutional layers and then moving to more complex and abstract features, which improves classification accuracy. It uses Rectified Linear Unit (ReLU) activation function instead of traditional sigmoid or tanh functions. This is one of the key ideas of this project because ReLU uses the non-saturating function f(x) = max(0, x) to accelerate the training process. This paper also uses overlapping pooling techniques. This method not only helps to reduce the error rate by preserving more spatial information from the images but also plays a key role in preventing overfitting.

2.2 The Effects of Optimizer Choice

In our project we plan to test performance of the model with different optimizers, however at this stage, the most frequently used Adam optimizer (Kingma and Ba [2015])is utilized. While other optimizers, such as SGD, take more time to converge and may generalize better in some deep learning tasks, Adam converges faster and performs better in early training, especially for image classification, according to Wilson et al. [2017]. Zhang et al. [2020] also suggests Adam often finds better minima than SGD in practice, especially in noisy environments, or unclear images, such as the case of fresh/rotten produce, where it is not always possible to directly see the difference between produce images. For the specific task of image classification, the comparative study of different optimizers in Dogo et al. [2022] also points out the superiority of Adam or its different variations, such as NAdam (RMSProp + Nesterov momentum mentioned in Dozat [2016]).

2.3 General CNN architecture, Activation Function and Hyperparameters

The recent interesting works which focus on tackling the image classification for the specific case of fresh/rotten produce incorporate ReLU and sigmoid activation functions, attain high accuracy of over 98% (Chouhan et al. [2023]). Additionally, the authors use dropout to capture intricate produce characteristics.
While most of the related work focused on the specific fresh/rotten produce classification uses a wide variety of architectures, including both well-known models (e.g., the paper on Deep Learning Methods for Rotten Fruits Identification (Chakraborty et al. [2021]) uses MobileNetV2, and the paper on classification of fresh/rotten fruits (Göksu et al. [2023]) - ResNet50) and custom designs, most of the works still incorporate pure CNN architectures (Sia and Baco [2023], Chouhan et al. [2023]).

2.4 Unsupervised Pre-Training Preliminary Research

Although we can achieve considerably high accuracy with solely using supervised learning (CNN in our case), the data dimensionality reduction approaches have been used, such as PCA (Jolliffe [2002]), and most recently, Autoencoders (Hinton and Salakhutdinov [2006]). Autoencoders essentially add a layer of training on the unlabeled data (the unsupervised learning part), as input to the following classification task tackling Neural Network (supervised learning part). By encoding the original data to lower dimensions, it provides opportunity for network to only focus on the essential features during classification, discarding irrelevant information. Non-linear methods such as autoencoders also achieve lower reconstruction errors and capture more complex data structures than linear methods, such as PCA. Therefore, the PCA and Autoencoders dimensionality reduction techniques for semi-supervised learning of the classification task for fresh/rotten produce can also be considered as the next step in our project.
As proposed by Rasmus et al. [2015], combining supervised learning with unsupervised helps greatly with denoising, so that it achieves higher performance on CIFAR-10 and MNIST benchmarks in terms of lower error rate, comparing to the approaches which used only supervised learning. Specifically 0.57 % error rate on all labeled data of MNIST was achieved, becoming the state-of-the-art. One

of the approaches that used solely supervised learning on MNIST image classification was Maxout Networks paper (Goodfellow et al. [2013]), and the other was the original Dropout paper (Srivastava et al. [2014]), both of which were outperfomed by the semi-supervised learning technique of Ladder Networks (Rasmus et al. [2015]).

3 Methods

3.1 Main CNN Model

We trained a convolutional neural network with an Adam optimizer and ReLU activation functions at each layer. Convolutional neural networks (CNN) are commonly used in image recognition, which is applied to the problem of classifying fresh and rotten produce in this instance. The performance of the neural network trained on the Adam optimizer will be compared against other neural networks trained on different activation functions and optimizer settings. Before resizing, the 3 color channels were normalized to a scale from 0 to 1. The CNN pipeline consisted of the following: a 32 channel Conv2D layer with 3x3 patch and ReLU activation, a MaxPooling2D layer of size 2x2, a 64 channel Conv2D layer with 3x3 patch and ReLU activation, a MaxPooling2D layer of size 2x2, a 64 channel Conv2D layer with 3x3 patch and ReLU activation, a flattening layer, 32 channel dense layer, and the final layer with 2 channels matching our binary classification problem. We use k-folds cross validation, with k = 5, split our dataset into 5 separate subsets, which we then use each as a test set and the remaining 4 as training sets.

3.2 Image Pre-processing

As CNN requires inputs of images with the same pixel dimensions, pre-processing is required across all training and testing images. Pre-processing will consist of resizing input images and performing principal component analysis (PCA) on inputs. Two methods of image resizing for pre-processing have been considered.
The first method is a direct scaling to a size of 224 by 224, where images are directly compressed to a square aspect ratio. For the preliminary test, a smaller size of 32 by 32 was chosen due to memory constraints. The advantage here is that all color pixels are preserved and no extra pixels are added. However, the scaling may cause distortions in the shape of the objects within the images, which may impact model training at the convolution stage. The second method involves padding images with white pixels on the bottom and right margins to match the image with the widest aspect ratio. The shape of objects within the images is not distorted, but the introduction of extra white pixels may impact the performance models trained on padded images.
In addition to resizing the images, principal component analysis will be done on input images to further reduce the size of training data. Other methods such as unsupervised learning techniques and encoders will also be added to the image pre-processing pipeline. The performance of the convolutional neural networks with varying settings will be compared between the different pre-processing methods.

3.3 PCA Analysis on Images

Principal component analysis was chosen to only be carried out on the images directly scaled to a square of 224 by 224. Some images in the dataset were rotated along a dark background, resulting in dark corners of varying lengths along the margins. The dark corners present in the training dataset would exist in random locations in the padded images as the padding algorithm adds white pixels onto the right and bottom of each image to match the dimensions of the largest image. Principal component analysis would then take into account these dark corners and create basis images with these dark corners and use them to reconstruct images of fruits. The PCA algorithm would then be unable to distinguish between the dark corners and any blemishes on the fruits. This resulted in extremely blurry images even with a large number of principal components. Only the scaled images kept the dark corners in relatively similar locations across all images in the training dataset, where they would not interfere with the fruits during reconstruction, resulting in clearer images.
After performing PCA on sets images labelled as fresh or rotten, it was found that the principal component images gradually became less differentiated from one another, making classification extremely unreliable. Using these principal component images to train the convolutional neural network would result in inaccurate predictions since no discernable features could be identified.

3.4 Autoencoder Unsupervised Pre-training Analysis

Motivated by enhancing classification with encoder weights from Autoencoder unsupervised training, described in Section 2.4, we trained an autoencoder on train dataset only from the train/test split of CNN-based classifier of the main CNN model, described in Section 3.1.
The autoencoder structure is a common one used in image-modality based tasks, such as Convolutional Autoencoder, adapted from the tensorflow.org website. Autoencoder is comprised of Conv2D layers in the encoder, and Conv2DTranspose layers in the decoder. We use 3 Conv2D and Conv2DTranspose layers in each encoder and decoder, respectively. As for hyperparameters, the optimal 32 codes, 128 hidden layers, 2 strides, ReLU activation function. For the last layer of encoder, we remove activation function and It maps from a 3-channel image to code with codesize filters (32). No activation (linear output) is used to avoid distorting the latent space with a nonlinearity.
We first trained on 1 dataset utilised in preliminary results to see the patterns in training/validation loss. The dataset was scaled to full 224 by 224 images, and additionally padded as described previously in Section 3.3. We assume the benefits and limitations of padding also had extensive effect on the autoencoder quality.
According to Figure 1, we observe an initial drop in loss at epochs 1–2, followed by a rapid decrease in both training and validation loss. This suggests that the model is learning meaningful reconstructions. The sudden increase in both training and validation loss at epoch 3 is likely due to a batch anomaly, especially when using large datasets or high-resolution images (224×224). The recovery after the spike shows that the loss sharply drops again and stabilizes. Small fluctuations are normal, particularly with a limited number of epochs. Overall, the loss drops to near-zero levels (≈ 0.001–0.002), which indicates good reconstruction performance.

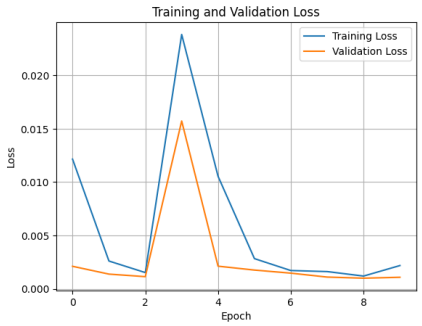

Figure 1: Autoencoder Validation Loss

3.5 Encoder based Classifier

We save the encoder weights from autoencoder training and utilize them as a top layer to the binary classifier head. The classifier head consists of the following. GlobalAveragePooling2D compresses the encoder's output feature maps by computing the average of each channel, effectively reducing the spatial dimensions while retaining important feature activations. This is supposed to reduce overfitting and makes the model more translation-invariant. Then, Dense layer with 128 units and ReLU activation are added, which adds non-linearity and learns intermediate representations from the pooled encoder features. A Dropout layer (rate=0.3) follows to randomly deactivate 30% of the neurons during training, helping to prevent overfitting. Another Dense layer with 64 units and ReLU activation introduces a second level of abstraction. A second Dropout layer (rate=0.3) continues regularization to improve generalization. Finally, a Dense layer with 2 output units produces the logits for binary classification (used with SparseCategoricalCrossentropy(from_logits=True) during training).
This classifier is lightweight but deep enough to transform encoder outputs into class probabilities. Using sigmoid on a single neuron aligns with binary classification and avoids one-hot labels.

Table 1: Datasets

Name	Size
Fresh And Rotten Dataset	∼30,400
Fruit and Vegetable Disease Dataset	∼29,300
Fruits and Vegetables Dataset	∼12,000

4 Results

4.1 Preliminary Results

Our original plans were to train models on local hardware in addition to cloud based platforms; unfortunately, software incompatibilities prevented a local model training trial for now. For our first attempt to train the CNN model, we used the 32 by 32 scaled-down images on the Google Colab machine learning platform, using Google Drive as the data storage source. It was necessary to copy the entire dataset from Google Drive to the local Colab drive storage, as training on the images directly from Google Drive resulted in a severe bottleneck. Our initial model training used 10 epochs, and resulted in an accuracy of 0.9688 and a loss of 0.1152, with a train time of 118 seconds (refer to Figure 2).

From these preliminary results, we gained an understanding of the importance of preprocessing of

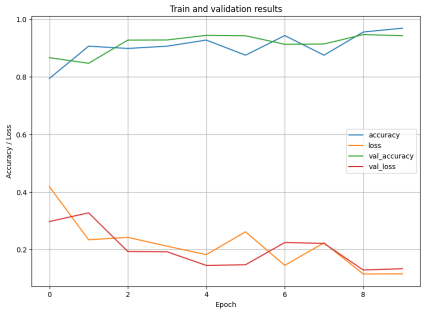

Figure 2: Accuracy and Loss Graph

the dataset, and the significant effects it can have on the training time of our model. Also of note are the potential bottlenecks encountered for the machine learning platform of choice, in our case the lengthy dataset uploads to Google Drive that slowed down our training process. The result's fast train time also gives us confidence in our future training on larger datasets and CNNs of increased complexity.

4.2 Results on 1 dataset only

Our initial training of the CNN on 1 dataset resulted in a high validation accuracy of 0.949, and a validation loss of 0.1605, trained over 10 epochs. These promising results may be due to the relatively small size of the singular dataset used, and the quality of the provided train/test split.

We evaluated the encoder-based classifier with one dataset (as described in Section 3.5) by monitoring both accuracy and loss over 10 training epochs. The result in Figure 2 shows both the trends of validation accuracy and loss. The accuracy of this model increased across epochs steadily, hovering around 0.86. The validation accuracy also remained consistently high, stabilizing around 0.89. This trend suggests that the model was able to generalize well to unseen data and there is no major overfitting. The convergence of training and validation accuracy further indicates that the encoder successfully captured informative features during unsupervised pretraining.

However, there are some significant fluctuations for the validation loss. The graph shows that the loss spikes at epochs round 6, even though accuracy remains high.a sharp spike around epoch 6, even though the validation accuracy remained relatively stable. This trend might be caused by a few misclassified examples with high confidence, which has softmax output near 0 or 1. These instances might inflate the cross-entropy loss. In addition, since k-fold cross-validation was not applied in this

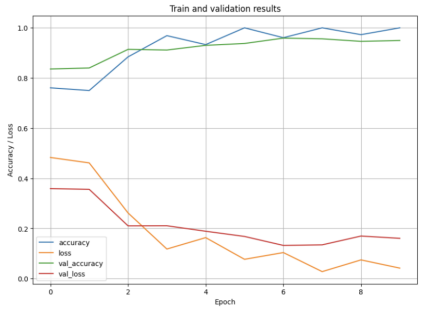

Figure 3: Accuracy of CNN-based classifier [main model described in Section 3.1]

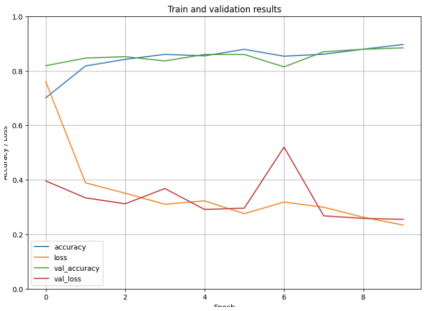

Figure 4: Accuracy of Encoder based classifier [described in Section 3.5]

phase, the static validation set may not have provided a representative error distribution, making the model more sensitive to individual outlier samples. Regardless of these variances, both training and validation loss decreased overall and present with the final values around 0.2.
In general, these results confirm that the encoder-based classifier is effective in leveraging unsupervised representations for binary classification and achieving high accuracy with low reconstruction and classification loss.

4.3 Results on 3 combined datasets

Moving to training the models on the combined dataset of three different sources, we evaluated the results of the accuracy of the CNN-based classifier, the autoencoder loss and and the accuracy of the encoder-based classifier. For each model, we conducted 5-fold cross-validation, training for 10 epochs per fold. Thus, we can learn both classification performance and generalization ability across datasets from the results.
The CNN-based classifier was trained using traditional supervised learning. The results show that the training accuracy starts very high, which is close to 100% in most folds, indicating that the model is easily fitting the training data. However, the training accuracy for all the folds drop significantly after reaching the second epoch. One possible reason for the low accuracy for both training and validation is overfitting, as it memorizes training data but fails to generalize due to the heterogeneity of the combined dataset. Meanwhile, the validation accuracy remains relatively low across all folds, which is around 50%. This result might suggest that our model is struggling with generalization and might fail to address domain shifts successfully.

For the encoder-based classifier, the training accuracy remains consistently high across folds. The accuracy of around mostly 90–98% indicates that the classifier head successfully fits the training data. Validation accuracy, however, fluctuates significantly. While some folds, like Fold 1 and Fold 3, show relatively stable performance, the others drop sharply after a few epochs. Fold 5 is particularly problematic, with validation accuracy floating around 47%, which is equivalent to random guessing and no learning is happening. The high training accuracy with low and fluctuating validation accuracy suggests overfitting, despite having the unsupervised pretraining. The encoder also may not be extracting generalizable features across folds or datasets. The drop in validation

Figure 5: Accuracy of CNN-based classifier [described in Section 3.1]

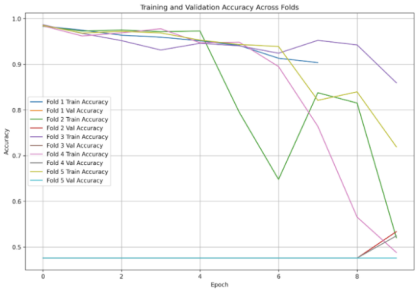

Figure 6: Accuracy of Encoder based classifier [described in Section 3.5]

accuracy for certain folds implies dataset heterogeneity or distribution shift across folds.
As what is shown in Figure 7, the training and validation loss for the autoencoder converge smoothly across all folds. Both training and validation losses drop rapidly within the first few epochs and plateau around very low values. This indicates that the autoencoder is effectively minimizing reconstruction error and is learning consistent latent representations of the image data. However, strong reconstruction performance does not guarantee the utility for classification tasks. While the autoencoder reconstructs images well, the encoder may be capturing features unrelated to produce freshness as features, such as background texture or lighting. It might limit its effectiveness for classification. It is also notable that the validation loss does not exhibit sharp increases, which suggests that the issue is not typical overfitting, but rather a latent space misalignment with the decision boundaries needed for accurate classification.

5 Discussion and Future Works

In general, the results we got from the models are not very satisfactory. Our failures might be caused by several problems and concerns. First of all, there might be some limitations during the unsupervised pretraining. As the autoencoders optimize for reconstruction but not classification, the encoder may emphasize reconstructing lighting or shape, rather than discriminative features related to freshness or rottenness. Second, there is likely to be a variance across the datasets. Since the data is from 3 different datasets, feature distributions may vary drastically. Thus, PCA and encoder can both struggle with inconsistent padding, lighting, or orientation, which make our model fail to handle these variations. In addition, there might be limited data augmentation. Without strong augmentation like color jitter, cropping, horizontal flip, rotation, the model may not learn invariance to lighting, orientation, or occlusion, which make our model unadaptable to different domains. Finally, although the classifier head includes multiple dense layers and regularization, it remains insufficient to overcome poor input representations from a suboptimal latent space. A moderately deep classifier cannot compensate when the encoder fails to extract task-relevant features.
As our current approaches highlight several limitations, particularly in terms of domain generalization and representation learning, our future work will address these challenges through the following directions. First, we plan to fine-tune the encoder jointly with the classifier head, rather than freezing

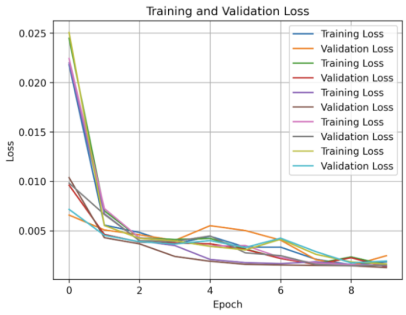

Figure 7: Autoencoder Loss [described in Section 3.4]

its weights after unsupervised pretraining. Allowing gradient updates throughout the entire model can help align latent features with classification objectives and improve the performance of the model potentially. Second, we want to explore contrastive learning or triplet loss. These methods make encoder map similar instances closer together and dissimilar instances further apart, which helps produce representations that are more effective for classification and enhance the discriminative power of the latent space. Additionally, since the datasets exhibit domain differences, we will explore domain adaptation further and try to implement it successfully. Techniques such as domain adversarial training, feature normalization, or domain-specific batch statistics may help mitigate the distribution shift and improve generalization across varied datasets. Lastly, we aim to implement stronger data augmentation pipelines that can expose the model to more visual patterns, reducing overfitting to dataset-specific artifacts and encouraging robustness to real-world variability.

6 Conclusion

In this project, we explored and created models that can classify fresh and rotten produce using convolutional neural networks, with a particular focus on the use of unsupervised pretraining through autoencoders and domain adaptation. With the goal of developing a lightweight yet accurate binary classifier capable of generalizing across images drawn from different sources with varying domains, we evaluated three main approaches: a CNN trained from scratch, PCA-based dimensionality reduction, and an encoder-based classifier pretrained with an autoencoder.

Even with strong performance demonstrated on individual datasets, the generalization to combined datasets of our model was limited. The CNN-based classifier and encoder-based classifier both achieved high training accuracy, but their validation accuracy dropped significantly across several folds. Although the autoencoder achieved excellent reconstruction performance, the latent space it learned was not sufficiently aligned with the decision boundaries required for effective classification. This highlights a fundamental limitation of unsupervised pretraining when used in isolation. Our findings suggest that domain adaptation is a challenging but essential component in real-world classification problems involving heterogeneous data sources. In addition, issues such as freezing pretrained encoders and variant datasets and a lack of robust data augmentation all contributed to suboptimal model performance.

Our work offers insights into how pretrained features generalize across data distributions. This mirrors real challenges in machine learning deployment, where training and test data are often drawn from different domains. The analysis exposes the fragility of encoders trained on combined datasets without domain adaptation, offering empirical support for the need for more robust feature extractors in cross-domain applications. Apart from this, our case study provides ideas of the future of the agricultural automation, such as quality control of fresh produce.

Looking forward, we plan to address these limitations by fine-tuning the encoder during classifier training, adopting contrastive learning to shape the latent space, applying explicit domain adaptation techniques and gaining stronger data augmentation. Despite the setbacks, this project has provided important insights into the complexity of building domain-robust image classifiers and sets the foundation for more adaptive and scalable solutions in future work.

7 Appendix

7.1 PCA Analysis on Images

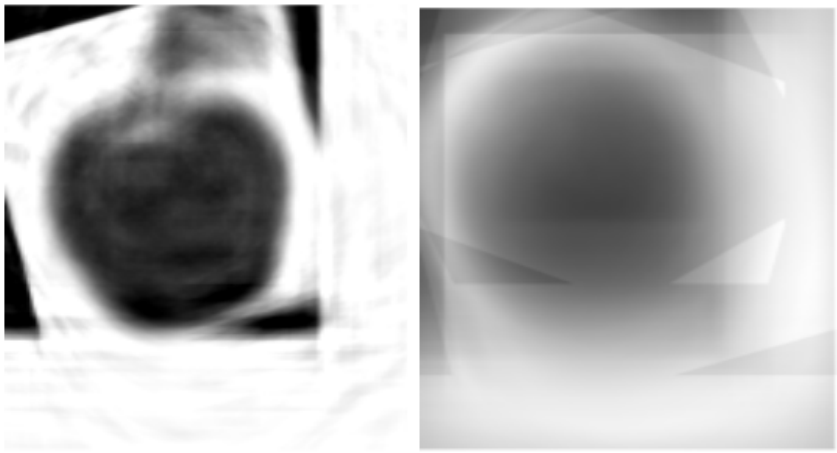

Figure 8: PCA Reconstruction from padded image (250 PCs and 4 PCs). Note the interference of the dark corners in the image

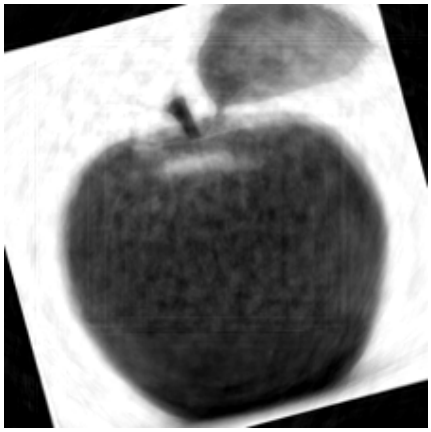

Figure 9: PCA Reconstruction from direct scaled image (250 PCs)

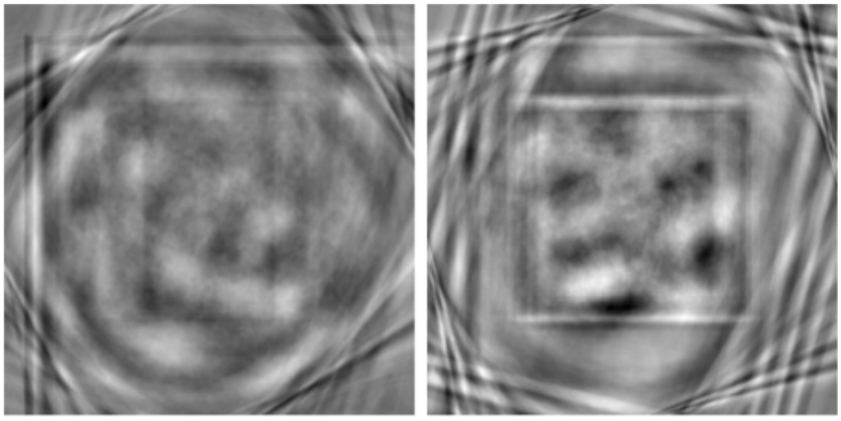

Figure 10: Principal component 100 for rotten and fresh fruits (from left to right), note the similarities in the dark spots where the "fruit" would be located

References

S. Chakraborty, F. J. M. Shamrat, M. M. Billah, M. A. Jubair, M. Alauddin, and R. Ranjan. Implementation of deep learning methods to identify rotten fruits. In *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 1207–1212, 2021. doi: 10.1109/ICOEI51242.2021.9453004.

M. Chouhan, P. S. Banerjee, A. Kumar, and J. Kushwaha. Classification of rotten fruits vs fresh fruits using sequential model with convolutional neural network. In *2023 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pages 869–874, 2023.

E. M. Dogo, O. J. Afolabi, and B. Twala. On the relative impact of optimizers on convolutional neural networks with varying depth and width for image classification. *Applied Sciences*, 12(23): 11976, 2022. doi: 10.3390/app122311976.

T. Dozat. Incorporating nesterov momentum into adam. In *Proceedings of the 4th International Conference on Learning Representations, Workshop Track*, pages 1–4, 2016.

I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 1319–1327, 2013.

T. Göksu, Z. Kaya, and S. Sahmoud. Classification of fruit images as fresh and rotten using convolutional neural networks. In *2023 3rd International Conference on Computing and Information Technology (ICCIT)*, pages 297–301, 2023. doi: 10.1109/ICCIT58132.2023.10273897.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. doi: 10.1126/science.1127647. URL `https://www.cs.toronto.edu/~hinton/absps/science.pdf`.

I. T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, 2nd edition, 2002. doi: 10.1007/b98835.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*, 2015.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105. Curran Associates, Inc., 2012.

A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

F. Sia and N. S. Baco. Hyperparameter tuning of convolutional neural network for fresh and rotten fruit recognition. In *2023 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*, pages 112–116, 2023. doi: 10.1109/IICAIET59451.2023.10291915.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting, 2014.

A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, volume 30, pages 4149–4159. Curran Associates, Inc., 2017.

J. Zhang, S. P. Karimireddy, A. Veit, S. Kim, S. J. Reddi, S. Kumar, and S. Sra. Why are adaptive methods good for attention models? In *Advances in Neural Information Processing Systems*, volume 33, pages 15383–15393, 2020.