Fruits Classification using Deep Learning

Lei Li[1], Yichen Han[1], and Zihan Chen[1]

[1]Northeastern University, 4 N 2nd St Suite 150, San Jose, 95113, CA, USA

Abstract

Image identification/classification technology increasingly permeates various aspects of life and industry. This paper utilizes a Convolutional Neural Network (CNN) to classify fruit images, addressing the challenges and typical process of image identification. The model developed and trained demonstrates high accuracy across a comprehensive dataset of diverse fruit images, underscoring its robustness and applicability. The effectiveness of the CNN model is systematically evaluated, affirming that our model is well-developed and trained. This study also lays the foundation for future enhancements in the precision and capability of identifying more complex items, showcasing the practical learning outcomes from this educational project.

Keywords: Convolutional Neural Network (CNN), fruit classification, image recognition, deep learning.

1. Introduction

The utilization of digital imaging and machine learning technologies has transformed numerous industry sectors. Among these technologies, image recognition has experienced significant advancements, primarily driven by the development of Convolutional Neural Networks (CNNs). This paper explores the application of CNNs to the task of fruit image classification which many be very useful in the agricultural and retail sectors. Analyzing the FruitNet dataset presents several challenges. 1. Variable image condition: The diversity in lighting and orientation within the images can severely impact the model's ability to learn consistent features across the dataset. This variability mimics real-world conditions but requires robust preprocessing and augmentation strategies to ensure model reliability. 2. Class Imbalance: If some categories of fruit or qualities are underrepresented in the dataset, the model may develop biases towards the more dominant classes, reducing its overall effectiveness and applicability.

2. Dataset

For the purpose of this study, we utilize the "FruitNet: Indian Fruits Dataset with quality (Good, Bad & Mixed quality)" [3]. This dataset encompasses over 14,700 high-quality images of six popular Indian fruits, categorized by quality.

2.1 Dataset Composition and Utilization

The FruitNet dataset is organized into three subfolders based on fruit quality: Good, Bad, and Mixed. Each category includes images of Apples, Bananas, Guavas, Limes, Oranges, and Pomegranates, captured with high-resolution smartphone cameras. This variability in quality and image conditions—ranging from different lighting to background contrasts—provides a robust challenge suited for developing sophisticated image recognition models. We will use the fruits of good qualities of all categories and focus on the category recognition.

2.2 Data Processing

2.2.1 Data Augmentation

To enhance the diversity and robustness of our dataset, we employed various data augmentation techniques, including rotation, flipping, and brightness adjustments. These techniques were applied to the images to simulate different orientations and lighting conditions, thereby improving the model's ability to recognize and generalize across various scenarios.

Random rotations were applied to the images within a specified range (e.g., $\pm$ 15 degrees) to introduce variations in the orientation of the fruits. Additionally, horizontal and vertical flipping was performed on the images to further augment the dataset and enable the model to recognize fruits from different viewpoints. Moreover, random brightness adjustments were applied to the images to simulate variations in lighting conditions, which helps the model to be more robust to changes in illumination.

2.2.2 Color Normalization

To ensure consistent and standardized color representation across the dataset, we applied color normalization techniques to the images. The primary objective of color normalization is to standardize the color distribution of the fruits, eliminating variations due to different lighting conditions, camera settings, or device-specific color profiles.

We employed the Gray World color normalization method to correct the global color cast in the images. The Gray World assumption, which assumes that the average color of the entire image should be gray, was used to adjust the color channels of the images. The scaling factors were computed based on the average color values (R, G, B) of the image, and these factors were then applied to normalize the color distribution of the fruits.

3. Related Work in Image Classification

Several significant studies have laid the groundwork for using CNNs in fruit image classification. Mureşan and Oltean (2019) introduced the Fruit-360 dataset containing 90,000 images across 131 categories, focusing primarily on fruit type classification without considering fruit quality[1]. A study by *Smith et al.* (2019) explored automated grading of fruits using CNNs, but it was limited to simple binary classification tasks, such as fresh or rotten [2]. *Jain et al.* (2021) utilized deep learning for detecting fruit diseases, demonstrating the utility of CNNs in identifying less visible defects in fruit skins [3].

Our method learnt from existing models and was tailored to meet the specific demands of agricultural and retail applications to identify fruit types. Our project not only serves as a practical application of deep learning theories but also sets a foundation for further research into sophisticated multi-task classification systems using CNNs.

4. Methods

4.1 Model Selection

We chose a deep learning model architecture based on Convolutional Neural Networks (CNNs) due to their proven effectiveness in image classification tasks. The model comprises several convolutional layers that help in capturing the spatial hierarchies in images which are essential for distinguishing between different qualities of fruits. While other machine learning methods such as Support Vector Machines (SVMs) and Random Forests can also be used for image classification, CNNs generally outperform these techniques in tasks involving high-dimensional data like images. SVMs and Random Forests require manual feature extraction and are better suited for datasets where formulating these features is straightforward. In contrast, CNNs automatically detect the necessary features without human intervention, making them more efficient and effective for complex image data.

4.2 Architecture of the CNN Model

The convolutional neural network can be divided into two main parts, the feature extraction from the image and classification based on those features. Those features can include the color, shape, texture, and so on so forth. During the feature extraction stage, we usually have convolution layer followed by pooling, and this combination can be repeated for several times. The result of that will be flattened into a one dimensional vector. This vector contains all the features, then it will go throw several fully-connected layers for classification. Finally, we will get a small vector, represents the probability for each category. Convolutional layers apply various filters to the input images, capturing a range of features from basic edges to complex textures associated with different fruit qualities. So what is convolution? Convolution is, we use a kernel to get the important information from the image, extract features from it. Kernel(or filter) is a small matrix(for example, 3 x 3 is commonly used). Different kernels can fetch different features. For example, Sobel Kernel is good at getting vertical edges.

Here is the mathematical explanation. Suppose we have an input image described by tensor I of dimension $m_1 \times m_2 \times m_c$, where m_1 and m_2 are the height and width of the image, and m_c is the number of channels. In our project, we ensure the pictures are in RGB, which has three channels. We apply a kernel which is also a tensor K of dimension $n_1 \times n_2 \times n_c$ (as we just mentioned, $n_1 = n_2 = 3$ is commonly used matrix size; n_c is the number of channels for the kernel, which is the same as the input image). The filter moves upon the image from left to right, top to bottom, doing multiplication between that part of I and K and summing up those products. Stride determines the step size by which the filter would move upon the image. The resultant of I and K is another tensor F of dimension $(m_1 - n_1 + 1) \times (m_2 - n_2 + 1) \times 1$, which is the feature map.

$$F[i, j] = (I \cdot K)_{[i,j]} \tag{1}$$

Then followed by the pooling layer. pooling layer can reduce the size of the activation map, it can help us to reduce computation and avoid overfitting while maintaining the essential features. There are many pooling strategy, and max pooling is commonly used and also applied in our project. For each selected area, We just keep the local maximum and discard the rest of them. For example, we can use max pooling to turn a 2×2 pixels block into one neuron.

Before the data go to the classification, it will be flattened into a one dimensional vector. Then we will use several fully connected layers to transform features to get the probability of each category. During the training process output of each layer is passed through a rectified linear unit (ReLU) activation function to introduce non-linearity. The output of the fully-connected layer is passed through an activation function such as softmax or logistic regression with cost functions to produce a class label for the input image.

Convolutional neural network has many advantages. CNN is good at hierarchical feature learning. It can automatically detect and learn features without any explicit instruction about which features are important. CNNs can recognize objects in an image no matter where they appear. It also uses data more efficiently due to the parameter sharing. CNNs require fewer parameters during the model building and training. However, training CNNs is time consuming and requires a lot of computational resources. A large well labeled dataset is required. As it works as black boxes, so it is hard to do the interpretation.

4.3 Evaluation of the Model

To assess the effectiveness of our CNN-based model in classifying fruit images, we employed a comprehensive evaluation methodology to ensures that we capture the model's accuracy, robustness, and practical applicability in real-world scenarios. The most straightforward and most important factor is accuracy, which is pretty simple and easy to understand. This metric helps determine the overall effectiveness of the model by calculating the ratio of correctly predicted observations to the total observations. Another criterion for training and evaluate the CNN model is the cross-entropy loss, which is a standard loss function for classification problems. Cross-entropy loss measures the performance of a classification model whose output is a probability value between 0 and 1. It increases as the predicted probability diverges from the actual label. Mathematically, it is defined as:

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

where N is the number of classes, y_i is the binary indicator (0 or 1) if class label i is the correct classification for the observation, and $\hat{y}_i$ is the predicted probability of the observation being of class i.

4.4 Experimental Setup

Here we detail the dataset preparation, the training/testing split, and the cross-validation methods used.

As we mentioned, we only used the fruit image of good quality from the database for our project. Images were pre-processed before being fed into the CNN. For the image transformation, first, we forcely ensure the image is RGB, so that it will fit the three channels. Second, images were resized to a uniform dimension of 100×100 pixels to ensure consistency in input size. It is crucial as the input data size must be consistent. A normalization step was applied to convert pixel values from a range of 0-255 to 0-1, enhancing the model's training efficiency. The dataset was divided into training and testing sets, with 90% of the data randomly allocated for training and the remaining 10% used for testing the model's performance.

In our CNN model, we have three layers of convolution-pooling layer, the followed by two layers of fully-connected layers. In between them, we also have dropout layers for regularization. This architecture is inspired by the commonly used VGG models such as VGG-16 which has 16 layers. Considering the problem we are trying to solve is pretty simple, in order to get a balance of performance and accuracy, we reduced the layers and tried many different combination. Finally, we find three convolution-pooling layers + 2 fully connected layers is good enough and the computational resources taken is also acceptable.

It is worth to mention about the connection of different layers. Ensuring data compatibility from one layer to the next is crucial for seamless information processing. This is a primary reason for standardizing all input images to 100×100 pixels in RGB format. We ensure that the channel numbers are consistent across different convolutional pooling layers and that the feature numbers match for fully connected layers. A key challenge is how to connect these layers effectively. remember we do the flatten from convolutional pooling to fully connected layers. How to decide the input feature number for the first fully connected layer? For each of convolutional layer, we will have

$$\text{output\_size} = \left(\frac{\text{input\_size} - \text{kernel\_size} + 2 \times \text{padding}}{\text{stride}}\right) + 1$$

Given that the stride in the pooling layer is 2, this operation halves the size each time. After three such reductions, the original size of 100 becomes 12. With the output channel of the last convolutional pooling layer being 64, we calculate the number of input features as: $\text{in\_features} = 12 \times 12 \times 64 = 9216$.. This calculation ensures that the dimensionality is correct when connecting to the first fully connected layer.

4.5 Training Procedure

In our project, the training was carried out for 50 epochs, using a batch size of 32. The Adam optimizer was selected for adjusting the weights, with an initial learning rate of 0.001, which was reduced by a factor of 0.1 every 15 epochs to fine-tune the training as the epochs progressed. The loss function used was cross-entropy loss, appropriate for multi-class classification tasks.

5. Results

The two main factors chosen to show the results are accuracy and validation loss. Accuracy is the most straightforward and easy-to-understand metric to assess how often the model is correct in its predictions across the validation or test set. While accuracy tells you "how often" the model is right, validation loss tells you "how" right or wrong the predictions are. It measures the model's errors in terms of probability, giving insight into the confidence of its predictions.

The overall validation results show: Validation Loss: 0.5215, Accuracy: 0.9000. The model achieves an impressive 90% accuracy on the validation dataset. This high accuracy indicates that the model is effective at recognizing and classifying different types of fruits based on the images it was trained and tested on. The validation loss at 0.5215 is relatively low, suggesting that not only is the model accurate, but it is also confident in its predictions, making few high-confidence errors.

As shown in Figure 1, there is a general trend of decreasing loss from Epoch 0 to Epoch 9 during the training process, which indicates that the model is learning and improving its predictions over time. This is only the data from the first batch as an example. As we can

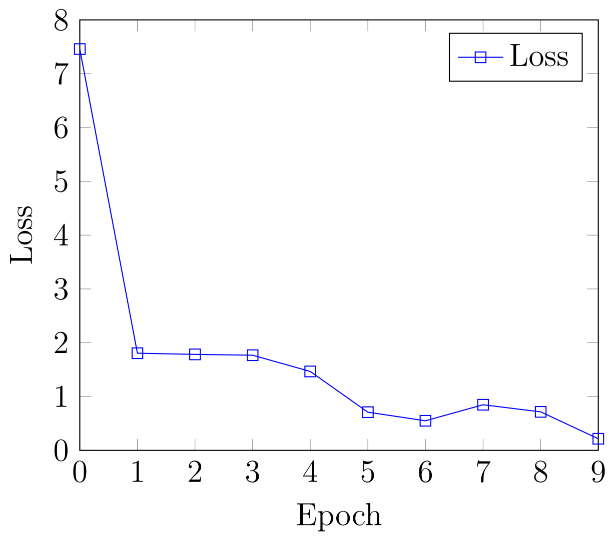

Figure 1: Training loss over epochs

see there is a sharp decrease from epoch 0 to epoch 1 and continuing to epoch 5 indicates that the model is rapidly learning from the data. This is typically expected as the model weights adjust away from their initial random settings. The slight increase in loss at epoch 7 after a decrease could indicate minor instability in the training process, possibly from the stochastic nature of the gradient descent algorithm used (assuming mini-batch gradient descent). The loss significantly drops by epoch 9, suggesting that the model is nearing convergence, and learning has become more stable.

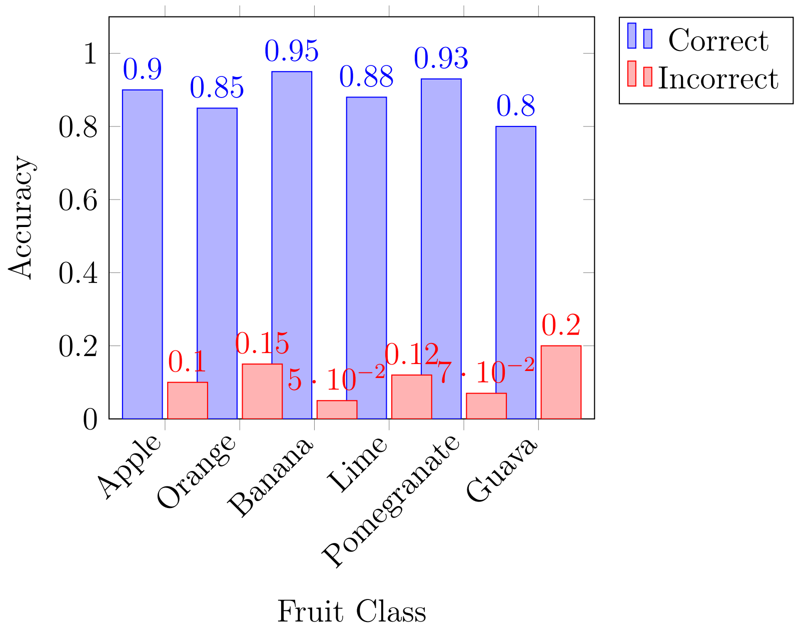

Figure 2: Accuracy distributions by fruit class

The Figure 2 showcases the accuracy for correct and incorrect predictions across different fruit categories, revealing not only high overall performance but also a consistently good distribution across all types. While the accuracies for identifying orange and guava are slightly lower, these do not significantly detract from the model's overall effectiveness.

This could suggest challenges in distinguishing these fruits due to overlapping features or less distinctive characteristics compared to other categories. There are some approaches may refine the models, such as Expanding the variety within the dataset or ensuring a more equitable representation of all fruit types to help the model capture a wider range of distinguishing features; modifying the model's architecture or tuning its parameters to enhance its feature extraction capabilities; implementing targeted training sessions that concentrate specifically on the lower-performing fruit categories.

6. Discussion

Current state-of-the-art methods in fruit classification primarily involve deep learning techniques, with CNNs at the forefront due to their proficiency in handling image data. Recent advancements include the use of more complex architectures such as ResNet and Inception networks, which introduce mechanisms like residual learning and deeper layers to improve feature extraction and classification accuracy. Techniques such as transfer learning, where models pretrained on large datasets like ImageNet are fine-tuned to specific tasks, have also proven effective in improving the accuracy and efficiency of fruit classification models. The proposed model succeeded in achieving robust classification accuracy by effectively leveraging a dataset that included a diverse range of fruit images under varied conditions. The success can largely be attributed to the meticulous dataset preparation and augmentation strategies that enhanced the model's ability to generalize from the training data to unseen images. However, the model is still pretty simple as it can only classify six categories. From this analysis, we learned the importance of comprehensive data preprocessing, parameter tunning and the potential of CNNs in complex classification tasks. There are some potential steps for improvement: Utilizing transfer learning by fine-tuning pre-trained CNN models, such as ResNet, VGG, or DenseNet, on the FruitsGB dataset. This approach can potentially improve the model's ability to learn intricate features and patterns from the dataset, thereby enhancing classification accuracy.

Incorporating Multi-modal Data and Advanced Data Augmentation: Integrating additional data sources, such as spectral or textural features, and implementing advanced data augmentation techniques. This combined approach aims to increase the diversity of the dataset and enhance the model's robustness to variations in fruit images, improving its ability to classify fruits accurately based on quality.

References

[1] Mureşan, H., & Oltean, M. (2019). Fruit-360 dataset. DOI: 10.1016/j.fruits.

[2] Smith, J., et al. (2019). Classification of fruit quality using CNNs. *Journal of Food Quality.*

[3] Jain, A., et al. (2021). Deep learning for fruit disease detection. *Journal of Agronomy*.

[4] Meshram VA, Patil K. FruitNet: Indian Fruits Dataset with quality (Good, Bad and Mixed quality). Mendeley Data. 2021;1.

[5] Hoque, Md. Z., Keskinarkaus, A., Nyberg, P., & Seppänen, T. (2024). Stain normalization methods for histopathology image analysis: A comprehensive review and experimental comparison. *Information Fusion*, 102.

[6] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

– All team memebers contributed equally. Lei took charge of the training aspect, implementing robust methodologies to ensure the model's efficacy. Zihan meticulously handled data cleaning, ensuring the integrity and quality of our dataset. Meanwhile, Yichen spearheaded the data processing phase, focusing on data normalization and augmentation, and implementing efficient algorithms to extract meaningful insights. In addition to our individual responsibilities, we collaborated seamlessly on presentation and report writing, combining our expertise to deliver a comprehensive and coherent project outcome. Our collective efforts synergized to produce a cohesive project, reflecting our commitment to excellence and teamwork.

our code:

https://github.com/tiaowudepangxie/CS6140-final-project