Image Classification of Tomato, Cherry, and Strawberry Using Convolutional Neural Networks

Ben Shand

***Abstract*— This project investigates the application of Convolutional Neural Networks (CNNs) for the classification of tomatoes, cherries, and strawberry images. Using a dataset of 6,000 images, including extensive preprocessing and augmentation techniques, we explore model performance in handling variations in lighting, angles, and background clutter. Initial experiments with a Multi-Layer Perceptron (MLP) produced limited accuracy, while a custom CNN and a transfer learning approach using ResNet18 significantly improved classification accuracy, achieving a final accuracy of 98% with an ensemble model. This report covers data preprocessing strategies, model selection, and transfer learning on model generalisation. The findings highlight the strengths and limitations of CNNs in image classification tasks and suggest potential areas for future work, including advanced augmentation with generative models and hyperparameter optimisation.**

I. Introduction

Accurately classifying images based on visual features remains a significant challenge in computer vision, especially when dealing with similar objects. This project explores the use of deep learning, specifically Convolutional Neural Networks (CNNs), to address a multiclass classification problem where the goal is to classify images into one of three classes: tomatoes, cherries, and strawberries. CNNs have demonstrated exceptional success in various vision tasks, often surpassing human-level performance due to their ability to learn complex spatial hierarchies in images, such as patterns in colour, texture, and shape [1], [7].

For this study, we utilise a dataset of 6,000 images, with 4,500 images allocated for training and 1,500 for testing. The dataset presents a range of challenges—variations in scale, illumination, background clutter, and viewing angles. To enhance classification accuracy, we apply a series of preprocessing techniques, including image resizing, normalisation, and augmentation, to address these variations and amplify relevant features.

In addition to training a custom CNN model, we incorporate transfer learning by fine-tuning ResNet18, a pre-trained model that offers advanced feature extraction [2], [8]. Our objective is to construct a model that not only achieves high classification accuracy but also demonstrates strong generalisation on unseen data. Through this process, we provide insights into effective preprocessing, model design, and optimisation strategies for CNNs in multiclass classification.

II. Background

A. Exploratory Data Analysis (EDA)

Image Resizing: I resized all images to 128x128 pixels, which allowed for faster processing and visualisation. Given the dataset contained 4,500 images across multiple classes, I verified that the number of images matched the dataset's specifications, confirming class representation without needing further validation on distribution. This resizing did not impact the visible features of the fruits, maintaining the essential visual quality needed for analysis.

Sample Image Inspection: I displayed five random images from the dataset to gain insight into the types of images, they showed varying levels of clarity. Most images appeared slightly blurred, suggesting that sharpening or noise-reduction filters will improve feature visibility for model training. This step was important to understand the dataset's quality and to assess which pre-processing steps would be required for clearer feature extraction [3].

Fig. 1. Sample images from each class

RGB Channel Distribution: Using a box plot, I visualised the RGB distribution across classes, which revealed key colour characteristics of each fruit. Tomatoes showed the highest median red intensity, while strawberries had a broader spread of red and green due to their stems and tops. Cherries had darker hues, indicated by lower median intensities across green and blue channels. These findings suggest that colour distribution, particularly in the red channel, could serve as a significant feature for classification.

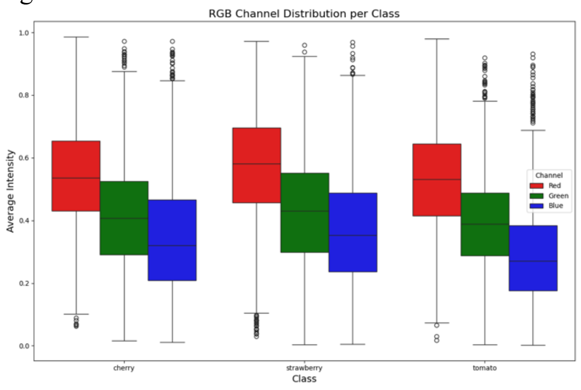

Fig. 2: RGB Distribution

Channel Histograms: To further investigate colour features, I generated histograms displaying the frequency distribution of pixel intensities across RGB channels for each class. The histograms supported observations from the box plot; for example, tomatoes had noticeable peaks in the red channel, while strawberries and cherries displayed unique green and blue distributions. This reinforces that colour intensity distributions are a distinct feature between classes and will assist in building a feature selection framework for the model.

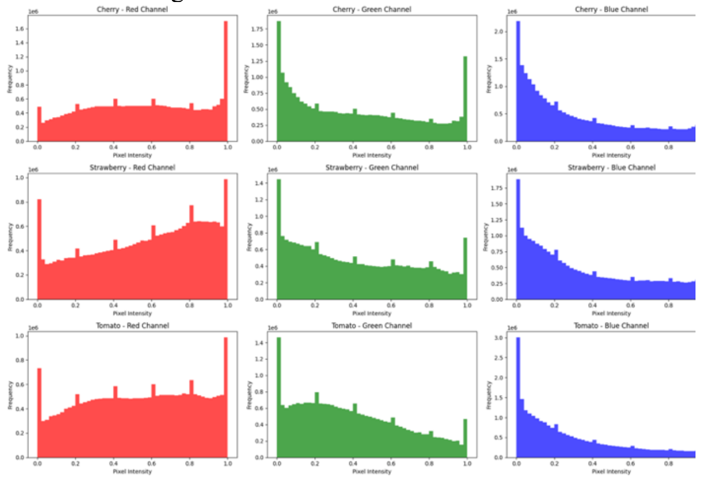

Fig. 3: RGB Channel Histograms

Thresholding Comparison: I applied various thresholding techniques—Isodata, Otsu, and Yen—to a grayscale version of a sample image. These methods provided balanced segmentation, clearly highlighting shapes and outlines, which are critical for distinguishing between classes based on fruit shapes and textures. The Isodata, Otsu, and Yen methods, in particular, maintained the distinct contours of the fruits and their branches/stalks. The use of thresholding could potentially be used, as focusing on shapes and textures rather than just colours might enhance class differentiality.

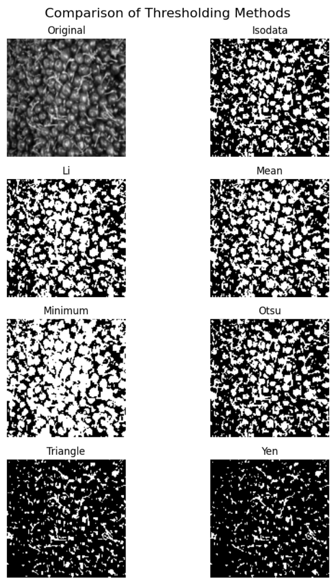

Fig. 4: Comparison of Thresholding Methods on Cherries

B. Preprocessing and Feature Selection

I created a preprocessing pipeline to address the challenges identified and improve the dataset's features. The goal was to ensure that the model could effectively generalise across varying lighting conditions, orientations, and image qualities.

Resizing to 128x128 pixels: This step was necessary to standardise the input image size, ensuring faster computation and consistency across the dataset. Resizing to 128x128 maintained the core features of the fruits, including shape and colour patterns, while reducing the memory requirements and training time.

Random Horizontal Flip and Rotation: To deal with the variation in image orientations observed during EDA, random horizontal flips with a 50% probability and rotations up to 15 degrees were applied. This augmentation step was necessary to prevent overfitting, as it encouraged the model to generalise its recognition of fruits beyond specific angles present in the dataset [3], [4].

Colour Jitter: Colour jitter, which introduces random variations in brightness, contrast, and saturation, was applied to simulate different lighting conditions. This augmentation made the model less sensitive to environmental variations, improving its stability in real-world scenarios where lighting variations could interfere with prediction accuracy [4].

Normalisation: Images were normalised using the mean and standard deviation values from the ImageNet dataset. This preprocessing step ensured consistent pixel intensity distributions, contributing to stable training and efficient convergence during model optimisation.

Red Channel Adjustment: I added a custom lambda function to amplify the red channel by a factor of 1.5. This step was specifically designed to boost the model's sensitivity to colour differences, which are important to the classification task, especially when differentiating between the classes with overlapping characteristics in the green and blue channels.

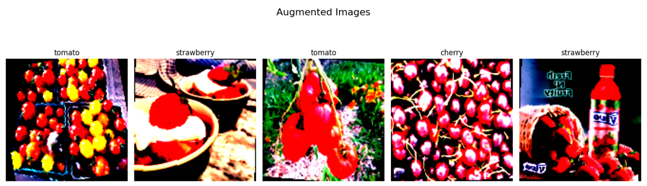

Fig. 5: Red Channel Amplification

Gaussian Blur: Gaussian blur was applied to reduce reliance on high-frequency edge details and encourage focus on broader structural features, such as shape and texture [3]. This technique mitigates sensitivity to image quality variations, allowing the model to perform reliably on both sharp and blurred images.

III. Methodology

A. Image Usage and Dataset Splitting

The dataset used in this project consists of images from three categories: cherry, tomato, and strawberry. To prepare the dataset, the images were split into training and validation sets using an 80/20 split. This ensures that 80% of the dataset is used for model training, while 20% is kept for validation. The dataset splitting was performed using random_split() to guarantee consistency across multiple training runs. This approach, using a manually set random seed, helps in evaluating the models' generalisation performance by exposing them to unseen validation data after each training epoch. For the validation set, no augmentations were applied to maintain consistency in evaluating the model's performance on clean, unaltered images. The validation set underwent only resizing, normalisation, and the same red channel adjustment to ensure that the model's predictions were based on real-world data without augmentation artefacts.

B. Loss Function

The CrossEntropyLoss function was chosen as the loss function for the CNN model. This function is particularly suitable for multi-class classification tasks, as it calculates the difference between the predicted probabilities for each class and the actual class labels [5].

During training, CrossEntropyLoss provided feedback by penalising incorrect classifications—guiding the optimisation process. It was observed that using CrossEntropyLoss helped the final model converge efficiently, as reflected in the steady decrease in loss over epochs (Figure 5).

C. Optimisation Method

The models were optimised using the Adam optimiser rather than Stochastic Gradient Descent (SGD). Adam was selected due to its adaptive learning rates, which improve the efficiency of training, especially with complex datasets and deeper architectures. For the MLP and BaselineCNN models, a learning rate of 0.001 was found to be optimal, while a lower learning rate of 0.0001 was used for the EnsembleCNN, allowing for gradual and stable fine-tuning when combining BaselineCNN with ResNet18.

D. Regularisation Strategy

To reduce overfitting, I used L2 regularisation. This regularisation technique adds a penalty term to the loss function, proportional to the square of the magnitude of the model parameters. By penalising large weights, L2 encourages the model to maintain smaller weights, which contributes to a more generalisable model [6].

Data augmentation served as an implicit form of regularisation as well. By artificially expanding the training dataset, data augmentation allowed the model to learn from a larger, more diverse set of images [3], [4].

E. Activation Function

The ReLU (Rectified Linear Unit) activation function was used in all hidden layers of the CNN, as well as in the final layers before the output of the CNN's fully connected layers. ReLU was chosen for its efficiency in deep networks, as it avoids the vanishing gradient problem by zeroing out negative values [5], making it suitable for training architectures like the CNN and ResNet18. The final output layers used a linear transformation without activation, as the logits produced by these layers were directly passed to the CrossEntropyLoss function, which computes the softmax internally.

F. Hyperparameters

The hyperparameters selected for the models were chosen to balance convergence speed, computational efficiency, and generalisation.

Learning Rates: 0.001 for MLP and BaselineCNN, and 0.0001 for EnsembleCNN.
Batch Size: A value of 16 was chosen to balance computational efficiency and the reliability of gradient updates.
Epochs: Training was set to 10 epochs for all models. Early stopping was applied based on validation loss, ensuring training halted if performance plateaued, preventing overfitting.

H. Use of Transfer Learning

BaselineCNN was combined with ResNet18, a model pre-trained on ImageNet. This was done to enhance classification performance through a parallel feature extraction approach. In this setup, both BaselineCNN and ResNet18 are trained simultaneously, each learning distinct but complementary feature representations.

To adapt ResNet18 to the task, its original fully connected output layer was replaced with a layer that matches the three classes of the dataset (cherry, tomato, and strawberry). During forward propagation, both models process the input images independently, generating separate sets of class logits. The outputs from BaselineCNN and ResNet18 are then concatenated to form a combined feature vector, which is passed through another fully connected layer to combine the complementary information and produce the final classification.

This parallel training and feature combination approach improves the model's ability to capture both specialised features relevant to this specific dataset (via BaselineCNN) and more generalisable high-level features (via ResNet18).

IV. Discussion

A. MLP Architecture

The baseline Multi-Layer Perceptron (MLP) model was designed as a simple benchmark for image classification. It treats the input images as a 1D vector of features, ignoring the spatial structure of the images. Each 128×128 RGB image is flattened into a 1D tensor with 128×128×3=49,15212 features, representing each pixel's RGB values.

The MLP architecture is made up of three fully connected layers. The first layer applies a linear transformation with 256 neurons. The second layer has 128 neurons, allowing the model to capture higher-order interactions between the features. Finally, the output layer contains three neurons corresponding to the three classes—cherry, tomato, and strawberry—and outputs raw logits. The model relies solely on the linear transformations and CrossEntropyLoss function to convert the logits into probabilities.

Although the MLP provided a quick and simple model to benchmark performance, it was limited by its lack of spatial awareness. The model treated each pixel independently and did not take advantage of the spatial structure present in the images, which is important image classification tasks.

B. BaselineCNN Architecture

In contrast the BaselineCNN model was specifically designed to use the spatial relationships in the image data, which is crucial for image classification. This CNN architecture includes three convolutional layers, each followed by a max-pooling layer to progressively reduce spatial dimensions while increasing the feature depth.

The first convolutional layer uses a 3×3 kernel with padding to retain the spatial dimensions, generating 16 feature maps and applying ReLU activation to introduce non-linearity. The following layers expand the feature depth (32 and then 64 feature maps) while reducing spatial dimensions via max-pooling, enabling the model to capture complex hierarchical features. Finally, the output from the convolutional layers is flattened into a 1D tensor, passed through a dense layer with 128 neurons, and then outputted as class probabilities after applying CrossEntropyLoss.

B. EnsembleArchitecture

The EnsembleCNN was designed to maximise performance by combining the baseline CNN with a pre-trained ResNet18 model. In this ensemble configuration, both the baseline CNN and ResNet18 were fine-tuned and their respective outputs concatenated, followed by a final classification layer that combines the learned features from both models.

This model achieved the highest performance, reaching a test accuracy of 0.98. This significant improvement is attributed to the combined strengths of the baseline CNN's custom features and the ResNet18's pre-trained generalisable features.

Although training the EnsembleCNN required a substantially longer time (approximately 1,500 seconds), its accuracy gains justify the extended training duration. The smoother loss curves in the EnsembleCNN's training and validation plots indicate accurate learning with minimal overfitting, suggesting effective generalisation to various image conditions, such as varying lighting and backgrounds.

C. Comparison of MLP and EnsembleCNN

The performance comparison between the MLP and CNN models clearly highlights the benefits of using a convolutional approach. The MLP achieved a test accuracy of 0.68, while the CNN significantly outperformed it, achieving a test accuracy of 0.89. The EnsembleCNN further improved the accuracy to 0.98, as demonstrated in the accuracy plots in Figure 5.

In terms of training time, the MLP model converged the fastest, completing training in approximately 55 seconds (Figure 5, top-left). However, its quick training did not yield high performance, as reflected by the erratic loss curves and low test accuracy. The CNN, on the other hand, required more time to converge, completing training in about 275 seconds (Figure 5, middle-left). This increase in training time was expected due to the complexity of the convolutional layers, which required additional operations to extract spatial features. Despite this longer training duration, the CNN's superior performance justifies the additional time. The CNN showed steady improvements in both training and validation loss, with consistent increases in validation accuracy, reflecting better generalisation to unseen data compared to the MLP.

The EnsembleCNN model took even longer to train, with a total training time of approximately 1,500 seconds (Figure 5, bottom-left). However, this longer training time was compensated by the model's outstanding performance, achieving the highest test accuracy of 0.98 (Figure 7). This improvement can be credited to the transfer learning approach, where the pre-trained ResNet18 layers provided rich feature representations from the ImageNet dataset.

The MLP's poor performance is further reflected in the fluctuating training and validation loss curves, indicating overfitting and difficulty in generalising to the validation and test data (Figure 5, top-left). In contrast, the CNN and ResNet18 models showed steady declines in loss, with the ensemble model demonstrating the smoothest curves and least overfitting, suggesting robust learning and generalisation (Figure 5, bottom-left).

The precision, recall, and F1-score plots (Figure 6) further highlight the CNN and ensemble models' superior performance, particularly in handling difficult-to-classify images. The ensemble model demonstrated the most balanced performance across all metrics, ensuring high classification accuracy for each of the fruit classes.

D. Differences and Observations

The most obvious difference between the MLP and CNN lies in their ability to handle spatial information. The MLP, by treating images as flattened vectors, lost all spatial structure, making it difficult for the model to identify patterns such as edges, textures, or shapes. As a result, the MLP struggled to differentiate between the fruit classes. On the other hand, the CNN's ability to preserve spatial information through its convolutional layers allowed it to capture more complex patterns, leading to significantly higher accuracy.

The EnsembleCNN model further demonstrated the benefits of transfer learning. By building upon the pre-trained ResNet18 architecture, the ensemble model was able to leverage more generalisable features, such as textures and shapes, which improved its ability to handle variations in the dataset, such as lighting, background clutter, and angles [2], [8]. The EnsembleCNN's final accuracy of 0.98 reflects its superior ability to classify the fruit images, particularly in challenging cases where the visual distinctions between the classes were subtle.

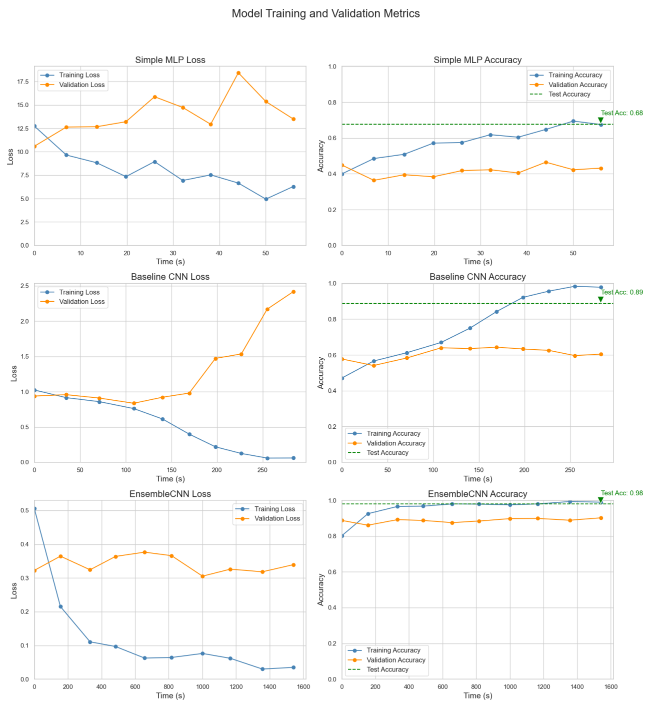

Fig. 5: Training and Validation Metrics for Model Performance

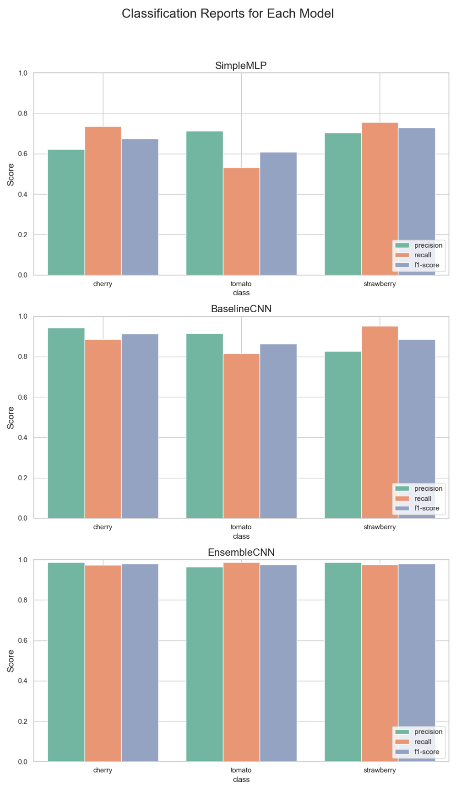

Fig. 6: Precision, Recall, and F1-Score Across All Models

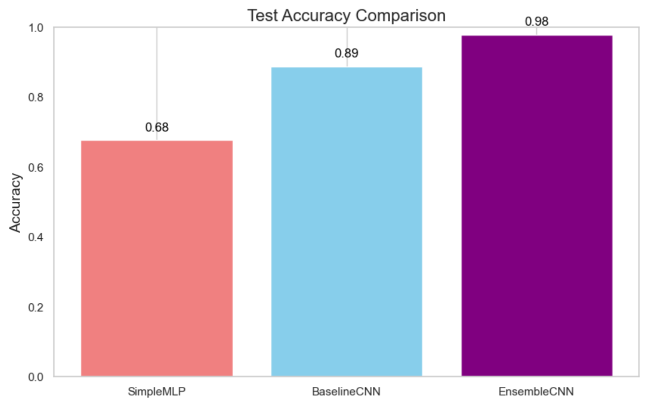

Fig. 7: Test Accuracy Comparison Between SimpleMLP, BaselineCNN, and Final Ensemble Model

VI. Conclusion & Future work

The project effectively demonstrated the benefits of CNNs over MLPs for classifying images of tomatoes, cherries, and strawberries, achieving an accuracy of 98% through transfer learning. CNNs excelled by capturing spatial relationships, which were essential for accurate classification, while MLPs, though quicker to train, lacked the spatial depth needed for this task.

Using a pre-trained ResNet18 model greatly increased classification accuracy, benefiting from feature representations learned from the ImageNet dataset. This highlights the advantage of transfer learning, especially with limited data. Data augmentation methods—including random rotations, flips, and colour adjustments—further improved the model's strength, simulating different environmental conditions to support accurate classification on unseen data.

However, the model's reliance on the red channel intensity for distinguishing fruit classes could limit adaptability with less colour-diverse datasets or variable lighting. Additionally, limited computational power constrained the project's ability to perform hyperparameter tuning or train larger, more effective models.

Future work can enhance this model by exploring lightweight architectures like MobileNet or SqueezeNet that could reduce computational time and costs [11], [12]. Hyperparameter optimisation, like Bayesian methods, could further refine the model's performance [13]. Also, generative adversarial networks's (GANs) for data augmentation could broaden the dataset, creating synthetic images that increase model adaptability across different angles, scales, and conditions [9], [10]. Access to more computational power—either through the cloud or better hardware—would allow for deeper experimentation and model improvement.

By addressing these areas, future iterations of this CNN image classification model can achieve higher accuracy, efficiency, and adaptability.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Advances in Neural Information Processing Systems, vol. 25, pp. 1097–1105, 2012, [Online]. Available: https://dl.acm.org/doi/10.5555/2999134.2999257

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778, [Online]. Available: https://doi.org/10.1109/CVPR.2016.90

[3] C. Shorten and T. M. Khoshgoftaar, "A Survey on Image Data Augmentation for Deep Learning," Journal of Big Data, vol. 6, no. 1, pp. 1–48, 2019, [Online]. Available: https://doi.org/10.1186/s40537-019-0197-0

[4] L. Perez and J. Wang, "The Effectiveness of Data Augmentation in Image Classification using Deep Learning." 2017, [Online]. Available: https://arxiv.org/abs/1712.04621

[5] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," Journal of Machine Learning Research, vol. 15, pp. 1929–1958, 2014, [Online]. Available: https://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

[6] A. Krogh and J. A. Hertz, "A Simple Weight Decay Can Improve Generalization," in Advances in Neural Information Processing Systems, 1992, pp. 950–957, [Online]. Available: https://papers.nips.cc/paper/1991/file/8eefcfdf5990aeb10f0666b448fd16f9-Paper.pdf

[7] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition." 2014, [Online]. Available: https://arxiv.org/abs/1409.1556

[8] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How Transferable are Features in Deep Neural Networks?," in Advances in Neural Information Processing Systems, 2014, pp. 3320–3328, [Online]. Available: https://proceedings.neurips.cc/paper/2014/file/375c71349b295fbe2dcdca9206f20a06-Paper.pdf

[9] I. Goodfellow et al., "Generative Adversarial Nets," in Advances in Neural Information Processing Systems, 2014, pp. 2672–2680, [Online]. Available: https://papers.nips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

[10] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "GAN-based Synthetic Medical Image Augmentation for Improved Liver Lesion Classification," IEEE Transactions on Medical Imaging, vol. 38, no. 3, pp. 839–853, 2018, [Online]. Available: https://doi.org/10.1109/TMI.2018.2865683

[11] A. G. Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." 2017, [Online]. Available: https://arxiv.org/abs/1704.04861

[12] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level Accuracy with 50x Fewer Parameters and <0.5MB Model Size." 2016, [Online]. Available: https://arxiv.org/abs/1602.07360

[13] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," in Advances in Neural Information Processing Systems, 2012, pp. 2951–2959, [Online]. Available: https://dl.acm.org/doi/10.5555/2999325.2999464

[14] L. Li and A. Talwalkar, "Random Search and Reproducibility for Hyper-Parameter Optimization," in Proceedings of Machine Learning Research, 2020, pp. 565–575, [Online]. Available: http://proceedings.mlr.press/v97/li19f.html