Cardiac Image Classification Using Deep Learning

University of Jordan

May 17, 2025

Prepared by:

- Mohammad Hajjaj — ID: 2212886

- Ghayth Bani Yaseen — ID: 2212977

- Mohammad Mustafa — ID: 2213409

- Anas Nahas — ID: 2213438

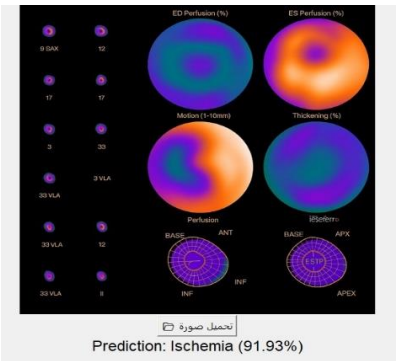
Abstract

This project presents a deep learning model for classifying myocardial perfusion imaging (MPI) scans into three categories: *Infarction*, *Ischemia*, and *Normal*. The core of the system is a fine-tuned MobileNet model pretrained on ImageNet, with its early layers frozen and higher layers retrained for transfer learning.

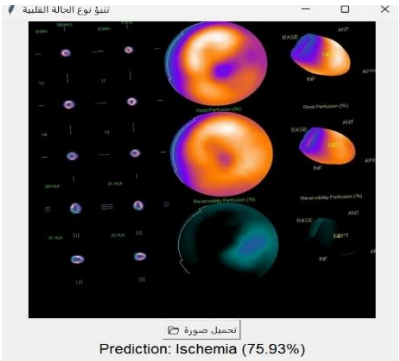
To improve generalization and mitigate overfitting, data augmentation techniques—such as rotation, shifting, and brightness adjustment—were applied only to the training set. Additionally, dropout layers and early stopping were used during training, and class weights were computed to address dataset imbalance.

The model was trained using TensorFlow and achieved a test accuracy of 91%, with strong performance in both precision and recall across all categories. Evaluation included a confusion matrix and performance visualization using Matplotlib and Seaborn.

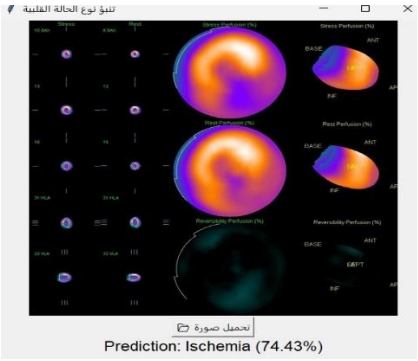
For real-world usability, a GUI interface was built with Tkinter, allowing users to upload MPI images and receive live predictions with confidence scores. The GUI resizes and displays the image, loads the model, and outputs the classification result in a user-friendly format.



Prediction on generated ischemia pic from GAN ChatGPT



Prediction on generated *ischemia pic* via *data augmentation*



Prediction on real ischemia picture from the test set

Contents

Introduction

Cardiovascular diseases are considered one of the main causes of death around the world. Because of that, early detection and diagnosis are very important to help patients and improve their chances of survival. One of the most common methods used in hospitals for this purpose is Myocardial Perfusion Imaging (MPI), which shows how blood flows through the heart.

However, analyzing MPI images manually can be time-consuming and prone to human error. This challenge motivated us to explore how deep learning can be used to support the diagnosis process, especially with the rise of powerful models like Convolutional Neural Networks (CNNs).

In our project, we developed a deep learning model to classify MPI images into three categories: Normal, Ischemia, and Infarction. We chose to fine-tune a MobileNet model pretrained on ImageNet due to its efficiency and good performance with limited data. We also applied data augmentation and used regularization techniques such as dropout and early stopping to improve the model's generalization.

Our goal is to assist medical professionals by providing a fast and reliable classification system. To make the model user-friendly, we also built a graphical user interface (GUI) using Tkinter, which allows users to upload MPI images and receive predictions along with confidence scores.

Dataset Description

The dataset used in this project consists of **myocardial perfusion imaging** (MPI) data collected from **88 patients**, after excluding incomplete cases from an original set of 97. Each patient had a complete set of standard MPI views: **QPS AC**, **Stress QGS AC**, and **Rest QGS AC**.

The images were categorized into three clinical classes based on diagnosis: **Normal**, **Ischemia**, and **Infarction**, with the following distribution:

- **Normal**: 189 images
- **Ischemia**: 60 images
- **Infarction**: 12 images

Each image contains both the visual cardiac data related to the patient's heart condition and an embedded section showing basic patient-related details, such as age or ID, as they appear in the original scan format.

Data Preprocessing

Before training the model, several preprocessing steps were performed to prepare the dataset for classification.

Initially, the dataset contained MPI images from **97 patients**. However, some patients were missing one or more of the required imaging views (**QPS AC**, **Stress QGS AC**, and **Rest QGS AC**). These incomplete cases were **manually excluded**, resulting in **88 complete cases**.

Each of these 88 cases was then **manually categorized** into one of three folders—**Normal**, **Ischemia**, or **Infarction**—based on the patient's medical diagnosis. All available MPI views per patient were placed in the appropriate class folder.

To preserve patient privacy and focus the model on relevant cardiac data, a **cropping script** was implemented to automatically remove identifying information (e.g., name, age, ID) from each image.

After organizing the data, the dataset was split using a **custom script** into three subsets with the following proportions:

- **Training:** 55%
- **Validation:** 20% (with slight adjustments)
- **Testing:** ~25%

The dataset was **highly imbalanced**, particularly between the **Normal** and **Ischemia** classes. To mitigate this, we generated additional synthetic **Ischemia** samples using a **Generative Adversarial Network (GAN)** via ChatGPT's image generation feature. These images were **manually verified** before being added to the **training set** and, to a lesser extent, the **validation set**, to improve representation and reduce class confusion.

During training, **data augmentation** was applied **only to the training set**, including slight **rotations**, **brightness changes**, and **spatial shifts**. The **validation** and **test sets** were used **without augmentation**, except for **rescaling pixel values to [0, 1].**

Model Architecture

In this project, we developed a **Convolutional Neural Network (CNN)** based on the **MobileNet (V1)** architecture to classify **Myocardial Perfusion Imaging (MPI)** scans into three categories: **Normal**, **Ischemia**, and **Infarction**.
MobileNet was selected for its **computational efficiency** and proven performance on relatively small medical imaging datasets.

We initialized the model using **pretrained weights from ImageNet**, with the **top classification layers excluded** (include_top=False). A **custom classification head** was then appended, consisting of the following:

- A **Global Average Pooling** layer to reduce dimensionality
- A **Dense layer** with 128 units and **ReLU** activation
- A **Dropout layer** (rate = **0.3**)
- A second **Dense layer** with 256 units and **ReLU** activation
- Another **Dropout layer** (rate = **0.4**)
- A final **Dense output layer** with 3 units and **softmax** activation for multiclass classification

Initially, **all layers of the base MobileNet model were frozen** to retain the generic features learned from ImageNet. Subsequently, we performed **partial fine-tuning** by unfreezing the model from **layer index 60 onward**, enabling the deeper layers to better adapt to the MPI domain.

The model was compiled using the **Adam optimizer** with a **learning rate of 0.0001**, and trained using the **categorical cross-entropy** loss function. To address **class imbalance**, **class weights** were calculated from the training distribution and applied during training.

To further improve generalization and prevent overfitting, we incorporated **early stopping** with a patience of **6 epochs**, restoring the best weights based on validation loss.

Training Details & Hyperparameters

The training process was conducted using the TensorFlow framework with Keras API. The MobileNet-based model was trained on augmented MPI images with categorical labels (Normal, Ischemia, Infarction). Training and validation were monitored across several hyperparameters and optimization settings as outlined below:

- **Batch Size:** A batch size of 16 was used for training, 10 for validation, and 1 for testing. A relatively smaller batch was selected for validation to allow more granular evaluation per batch while keeping GPU memory usage efficient, given the limited size of the validation set.

- **Epochs:** The model was trained for up to 50 epochs with early stopping enabled to halt training when validation performance plateaued.
- **Optimizer: Optimizer:** The Adam optimizer was used with a learning rate of **0.0001**. This value was chosen after experimenting with multiple learning rates (including 0.01, 0.001, and 0.00005). The selected rate provided the most stable convergence with minimal validation loss and reduced overfitting. Adam was preferred for its adaptive learning capabilities and consistent performance in medical image classification tasks.
- **Loss Function:** Categorical cross-entropy was selected as the loss function since the classification task involves more than two classes, and the labels were one-hot encoded. This loss is particularly suitable when the model's final activation function is **softmax**, as it measures the distance between the predicted probability distribution and the actual distribution of class labels.
- **Dropout Rates:** Dropout regularization was applied with a rate of 0.3 after the first dense layer and 0.4 after the second dense layer. These values were determined experimentally by observing the model's behavior during training. Lower dropout rates led to signs of overfitting, while higher values caused underfitting. The selected rates offered the best trade-off, helping the model generalize better without sacrificing learning capacity.
- **Fine-Tuning Strategy:** Initially, all layers of the pretrained **MobileNet** base were frozen to preserve the generic image features learned from ImageNet. After completing the initial training phase, fine-tuning was enabled by unfreezing layers starting from index 60 onward (approximately the last 26 layers). This configuration was selected based on empirical experiments, where several unfreezing points were tested. Starting from layer 60 resulted in noticeable improvements in validation accuracy without degrading performance, making it the most effective setup among the trials.
- **Class Weights:** Class weights were computed based on the training set distribution to compensate for the imbalance between classes, especially the underrepresented "Infarction" class.
- **Data Augmentation:** Augmentation was applied only to the training set, using medically safe transformations such as slight rotations (±15°), minor width and height shifts (0.01), and limited brightness variation (range: 0.8–1.2). These specific augmentations were selected based on domain research and expert guidance, as excessive modifications in medical imaging—such as flipping, heavy zooming, or noise injection—can distort clinically relevant features. The chosen transformations preserved diagnostic integrity while helping the model generalize better across natural variations in image capture
- **Early Stopping:** EarlyStopping callback was used with a patience of 6 epochs to prevent overfitting, with the model automatically restoring the best-performing weights based on validation loss.

The training and validation processes were visually monitored using plots of loss and accuracy over epochs. These visualizations helped confirm convergence and ensure that no significant overfitting occurred during training.

Evaluation and Results

The trained model was evaluated on a separate test set consisting of 68 images. The overall test accuracy reached **91%**, indicating strong generalization capability despite the limited dataset size and class imbalance.

Classification Metrics

A detailed classification report was generated using scikit-learn's classification report function. The report includes precision, recall, and F1-score for each class showed in **Figure 1.**

```
Classification Report:
                precision    recall  f1-score   support

InfarctionCrop       1.00      1.00      1.00         3
  IschemiaCrop       0.93      0.72      0.81        18
    NormalCrop       0.90      0.98      0.94        47

      accuracy                           0.91        68
     macro avg       0.94      0.90      0.92        68
  weighted avg       0.91      0.91      0.91        68
```

Figure 1

These results reflect excellent performance on **Normal** and **Infarction** cases, with nearly perfect classification. The **Ischemia** class showed slightly lower recall, likely due to its smaller representation in the dataset and its visual similarity to both Normal and Infarction images.

Confusion Matrix

To better understand class-wise prediction behavior, a confusion matrix was generated for the test set, as shown in **Figure 2.**

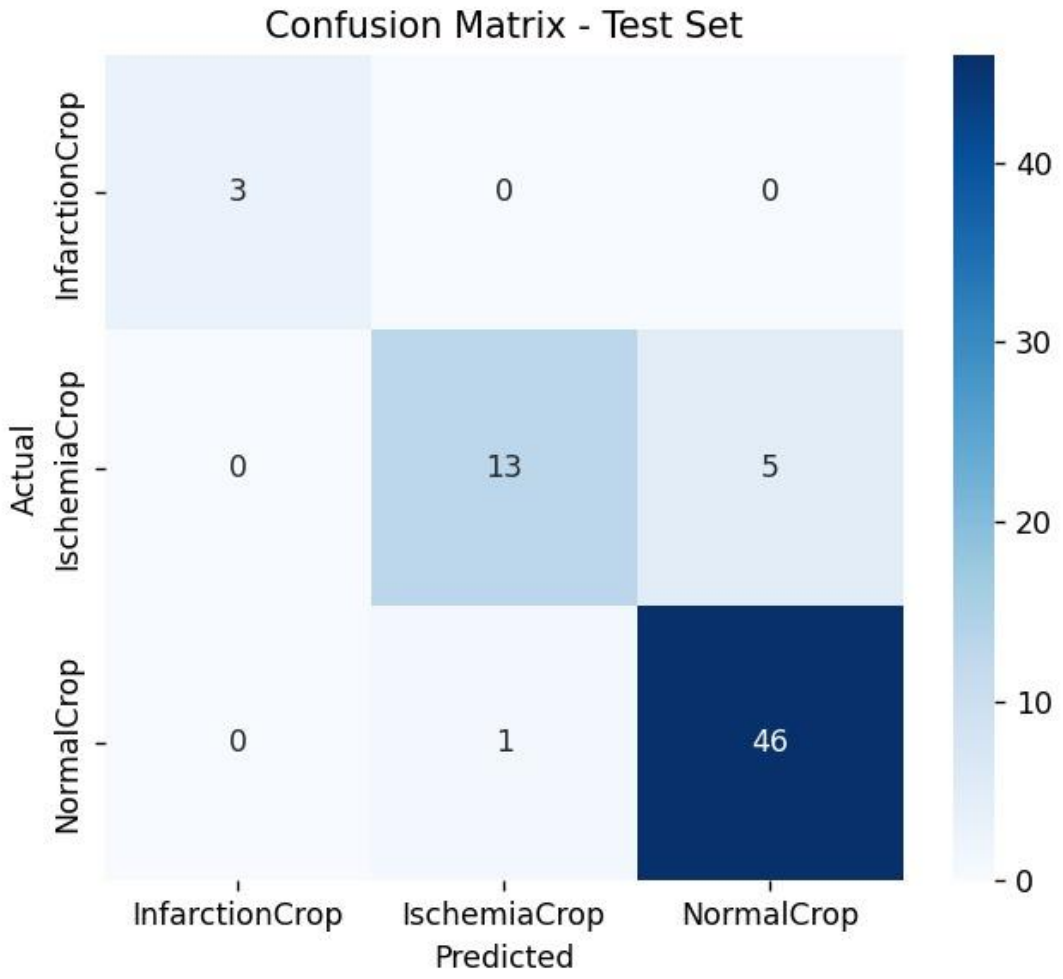


Figure 2

As seen in the matrix:

- The model correctly classified **all Infarction cases (3/3)**.
- For the **Ischemia class**, 13 out of 18 were correctly classified, while 5 were misclassified as Normal.

- For **Normal cases**, the model achieved 46 correct predictions out of 47, with just one misclassified as Ischemia.

This confusion between **Ischemia and Normal** is expected and clinically relevant. Even experienced cardiologists often find it challenging to distinguish **mild ischemic cases** from normal perfusion, especially when defects are regional or borderline.

Training and Validation Curves

Figures 3 and 4 show the model's loss and accuracy progression over 25 training epochs. These plots were used to monitor training behavior and assess convergence quality.

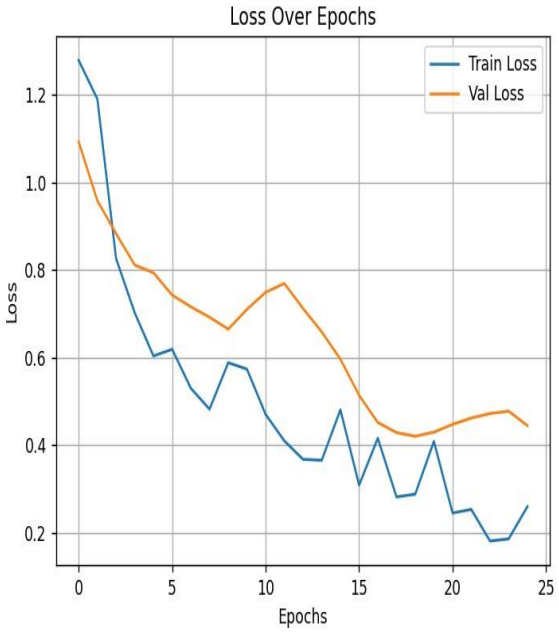


Figure 3

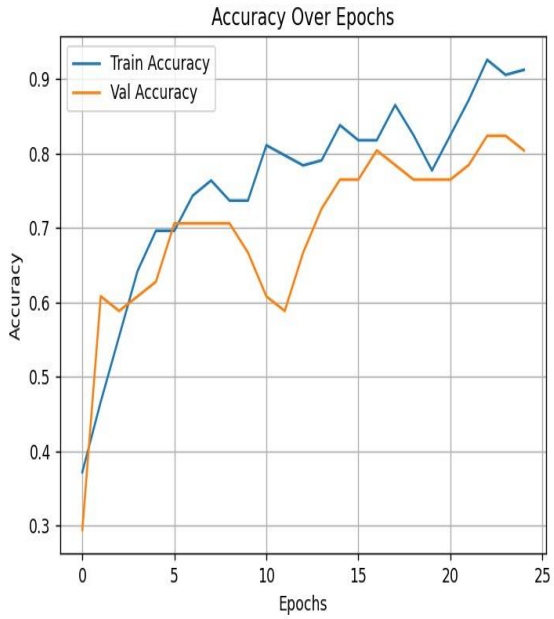


Figure 4

As observed:

- **Training loss** steadily decreased, while **validation loss** followed a similar trend before plateauing and slightly increasing near the end, a sign that **early stopping** was triggered at an appropriate time.
- **Training accuracy** improved consistently and reached over **92%**, while **validation accuracy** stabilized around **81%**.
- The visible gap between training and validation performance indicates a mild overfitting trend, which was mitigated through dropout regularization and proper augmentation.

These plots demonstrate that the model was able to learn meaningful features without overfitting severely, which is crucial when working with limited medical data

GUI Application

Although the primary goal of this project was to build a robust classification model, a simple **Graphical User Interface (GUI)** was also developed using **Tkinter** to assist with rapid testing of the trained model on new unseen images.

This interface was designed mainly for the developer's use, allowing for quick validation of the saved model on individual MPI images after training, without rerunning the full pipeline or writing additional evaluation code.

Key Functionalities:

- **Image Upload:** Users can upload any MPI image directly from disk.
- **Model Integration:** The interface loads the saved Keras model (MobileNet-based) and preprocesses the uploaded image (resizing to 224×224 and normalization).
- **Prediction Output:** The predicted class is shown on the GUI along with the confidence score.
- **Image Display:** The uploaded image is also displayed in the interface for visual confirmation.

This tool proved valuable during post-training validation, enabling fast experimentation and checking how well the model generalizes to newly acquired or augmented MPI images as shown in figure 5,6.

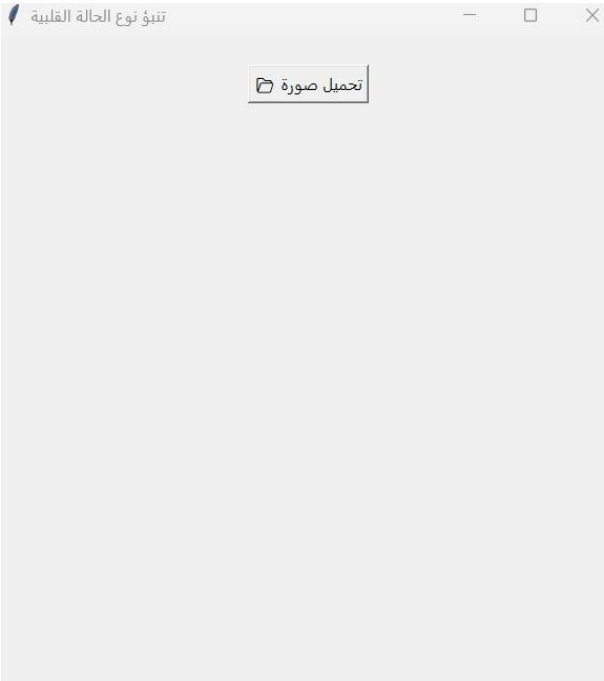


Figure 5

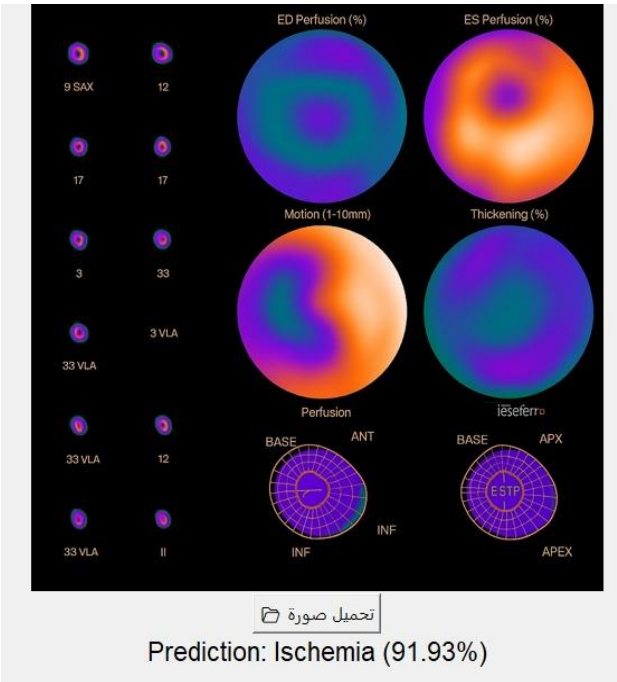


Figure 6

Limitations and Challenges

While the model achieved high performance on the test set, several limitations and challenges were encountered during the development of this model:

1. **Limited Dataset Size :** The total number of patients included was 88 after excluding incomplete cases. In particular, the Infarction class was underrepresented with only 12 images, which can negatively affect the model's ability to generalize and detect rare conditions reliably.
2. **Class Imbalance :** There was a significant imbalance between the three classes, especially between Normal (189 images) and Ischemia (60 images). Although class weights and synthetic augmentation (GAN-based) were used to partially address this issue, the model still showed some confusion between Ischemia and Normal.
3. **High Visual Similarity :** Visual differences between Normal and mild Ischemia cases are often subtle. This makes classification difficult not just for AI models, but even for experienced cardiologists. This explains why most of the misclassifications occurred between these two classes.
4. **Dependence on Preprocessing :** The performance of the model relied heavily on the consistency of preprocessing steps such as cropping and rescaling. Any deviation or noise in preprocessing may affect prediction accuracy

---

Conclusion

This project successfully demonstrated the potential of deep learning in classifying myocardial perfusion imaging (MPI) scans into clinically relevant categories: Normal, Ischemia, and Infarction.

By fine-tuning a pretrained MobileNet model and applying targeted data augmentation, regularization techniques, and class balancing strategies, the model achieved a test accuracy of 91%, with strong precision and recall across all classes. Despite challenges such as limited dataset size and class imbalance, the model generalized well and showed consistent performance during evaluation.

A simple GUI application was also developed to facilitate post-training testing, allowing for quick experimentation and real-time predictions on new MPI images.

Overall, the results indicate that deep learning models—when properly optimized—can provide valuable support tools in the cardiac imaging workflow, offering fast and accurate second opinions for diagnostic decision-making.

References

1. Howard, A. G., et al. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. arXiv preprint arXiv:1704.04861.
2. Henzler, P., et al. (2020). *Myocardial perfusion imaging: A review of techniques and applications*. Journal of Nuclear Cardiology, 27(6), 2283–2295.
3. Chollet, F. (2017). *Deep Learning with Python*. Manning Publications.
4. Goodfellow, I., et al. (2014). *Generative adversarial nets*. Advances in Neural Information Processing Systems, 27, 2672–2680.
5. Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization*. arXiv preprint arXiv:1412.6980.
6. Shorten, C., & Khoshgoftaar, T. M. (2019). *A survey on image data augmentation for deep learning*. Journal of Big Data, 6(1), 60.
7. Keras Documentation. (n.d.). *Working with Imbalanced Datasets*. Retrieved from https://keras.io