Medical Image Classification Using Deep Learning

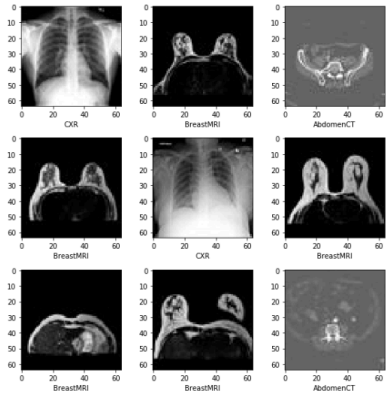
Salvador S. Sandoval
California State University Long Beach
CECS 456
Github

I. Introduction

Medical image classification is vital for healthcare applications. Accurate diagnosis and well-timed interventions can save countless lives with technology. Using the "Medical MNIST" dataset from Kaggle, this project looks into the application of a convolutional neural network (CNN) to analyze and classify medical images into its six categories. The objective was to develop a rich and intelligent model capable of achieving high accuracy and fast modeling while minimizing computational resource costs. The report outlines the dataset characteristics, related works, the methodology adopted, the experimental setup, and the results obtained.

II. Dataset and Related Work

The dataset used in this project is the Medical MNIST dataset, available on Kaggle. It comprises 59,000 grayscale images divided into six classes: AbdomenCT, BreastMRI, ChestCT, CXR, Hand, and HeadCT. Each class represents a different type of medical imaging, making the dataset a varied benchmark for testing image classification models.



Sample Images from the Kaggle Dataset

Due to their ability to learn hierarchical feature representations, previous works have shown the utility of CNNs for medical image classification. VGGNet and ResNet, for instance, have been successfully applied to classify medical imagery with high accuracy. This project, however, implements a simpler CNN architecture to balance accuracy and computational efficiency, suitable for smaller datasets.

III. Methodology

A CNN model was implemented to classify the images into their respective categories. The architecture consists of:

- Three convolutional layers with ReLU activation and max-pooling for feature extraction.
- A flattening layer to prepare features for fully connected layers.
- Three dense layers with 128, 64, and 6 neurons each for further feature

learning, followed by classification through a softmax layer.

```
# define model using Sequential API
model = Sequential([
  ## first convolutional layer, image shape is 64x64 with grayscale (1)
  Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 1)),
  MaxPooling2D((2, 2)),
  Dropout(0.3), #randomly disabiling neurons during training to prevent overfitting
  # apply 32 filters of size 3x3 to the input
  Conv2D(64, (3, 3), activation='relu'),
  MaxPooling2D((2, 2)),
  # second convolutional layer
  Conv2D(128, (3, 3), activation='relu'),
  # second convolutional layer, further reduces spatial dimensions
  MaxPooling2D((2, 2)),
  Flatten(), # flatten from 3D feature maps to a 1D vector
  # Dense layer with 128 neurons, relu introduces non-linearity
  Dense(128, activation='relu', kernel_regularizer=l2(0.01)),
  Dropout(0.5),
  Dense(64, activation = "relu"),
  Dropout(0.5),
  Dense(6, activation='softmax') # Output layer with 6 classes
])
```

Model structure in code

Data augmentation techniques such as zooming, rotation, and horizontal/vertical flipping were applied during training to ensure the model generalizes well. The model was optimized using the Adam optimizer and trained with categorical cross-entropy loss.

IV. Experimental Setup

The experiments were conducted on a machine with the following specifications:

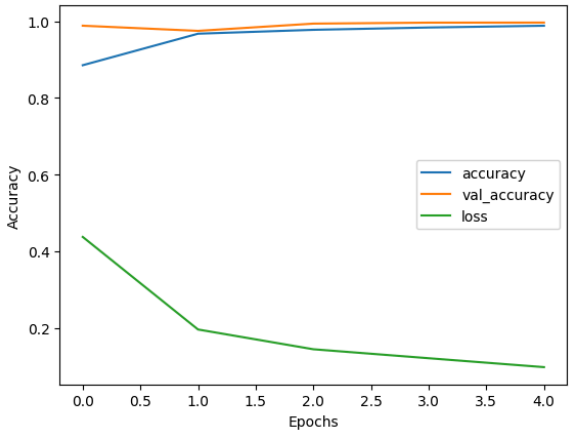
- Processor: AMD Ryzen 7 5800X
- GPU: AMD Radeon RX 6600
- Framework: TensorFlow/Keras
- Operating System: Windows 10

The dataset images were divided into a 20:80 ratio for the subsets of validation and training. To improve convergence, dataset images were resized to pixels of 64x64 and normalized to the range [0, 1]. The training was conducted over five epochs with a batch size of 32. Three dropout functions in the model and the kernel regularizer were implemented to prevent overfitting.

V. Measurement

Model performance was measured using the following metrics:

- Accuracy: Percentage of correctly classified images.
- Loss: Categorical cross-entropy value indicating how well the model fits the data.
- Confusion Matrix: For detailed class-wise performance analysis.



A graph containing the accuracy, value accuracy, and loss performance

VI. Result Analysis, Intuitions, and Comparison

The model achieved an accuracy of 98% on the validation set, with a loss of 0.13. This indicates strong performance on the given dataset. Below are key observations:

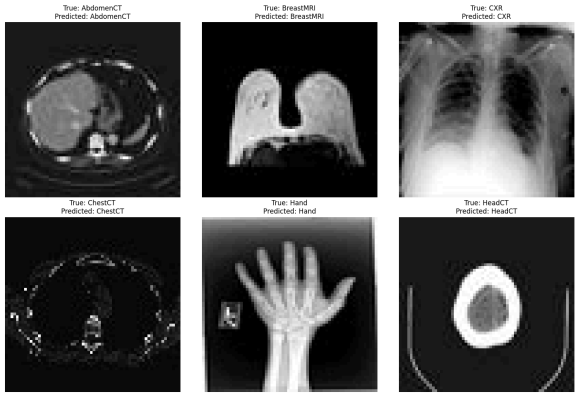
Class-wise Performance: The confusion matrix revealed that most misclassifications occurred between structurally similar categories but different imaging methods, such as ChestCT and CXR.

Data Augmentation: Augmentation helped improve generalization, which is evident from reduced overfitting during training.

Computational Efficiency: The lightweight CNN architecture allowed faster training compared to deeper models like ResNet,

making it suitable for limited computational resources.

Compared to related works, this model's simplicity comes at the cost of potentially lower generalizability to larger or noisier datasets. However, it performs well on the Medical MNIST dataset, demonstrating the viability of compact architectures for small-scale problems.



Sample of a random image from each category and the model prediction

VII. Conclusion

This project successfully implemented a CNN-based classifier for the Medical MNIST dataset, achieving high accuracy and low loss. The combination of data preprocessing, augmentation, and architectural simplicity contributed to its strong performance. Future work could involve extending the model to handle larger, more diverse datasets and experimenting with transfer learning to further enhance generalization. This study highlights the potential of deep learning and CNN's in medical image classification and their implications for improving diagnostic tools in healthcare and medicine.

VIII. References

- https://github.com/SebastianSSandoval/CECS-456/tree/main/aiFinalProject
- Dataset: https://www.kaggle.com/datasets/andrewmvd/medical-mnist
- "VGG-Net Architecture Explained." GeeksforGeeks, GeeksforGeeks, 7 June 2024, www.geeksforgeeks.org/vgg-net-architecture-explained/.
- "Residual Networks (Resnet) - Deep Learning." GeeksforGeeks, 10 Jan. 2023, www.geeksforgeeks.org/residual-networks-resnet-deep-learning/.
- "Guide : Tensorflow Core." TensorFlow, www.tensorflow.org/guide. Accessed 15 Dec. 2024.