1. Introduction

In the rapidly evolving field of medical image analysis, the development of highly accurate and efficient computational models has become paramount. This is particularly crucial for datasets like Medical MNIST, which comprises a diverse collection of medical organ images. Such datasets present unique challenges due to the complex nature of medical imagery, including high variability in appearance, structure, and the presence of subtle yet critical features indicative of health conditions. Addressing these challenges requires sophisticated model architectures that can capture both the local and global contextual details inherent in medical images.

This report introduces a novel deep learning model designed specifically for the Medical MNIST dataset, a comprehensive collection of medical organ images. Our model leverages the strengths of Convolutional Neural Networks (CNNs) and Transformer architecture to efficiently process and analyze the intricate patterns within these images. The primary objective was to achieve high accuracy in classifying medical images into their respective categories, which include various organs and potential pathological findings.

The development of this model was motivated by the need for more advanced diagnostic tools in the medical field, which can assist healthcare professionals in making more accurate and timely diagnoses. By achieving an unprecedented accuracy of 99% on the Medical MNIST dataset, our model demonstrates the potential to significantly enhance diagnostic processes and outcomes in healthcare settings.

2. Technical Solution

2.1 Model Architecture

The core of our technical solution is a hybrid deep learning model that integrates Convolutional Neural Networks (CNNs) with Transformer architecture. This design choice is driven by the complexity and diversity of the Medical MNIST dataset, which requires a robust model capable of capturing both the intricate local features of

medical images and the global contextual relationships within them.

CNN Layers: The model begins with a series of CNN layers, responsible for extracting hierarchical features from the input images. This part of the model consists of multiple convolutional blocks, each comprising convolutional layers followed by batch normalization, ReLU activation, and max pooling. These layers progressively increase in depth to capture more complex features at each level, while dropout layers are included to prevent overfitting.

Transformer Module: Following the CNN layers, the extracted features are fed into a Transformer module. This module utilizes self-attention mechanisms to analyze the relationships between different regions of the images, enabling the model to understand the broader context beyond local features. The Transformer block is designed with multiple heads to capture various aspects of the data simultaneously, and it includes both positional encoding and layer normalization to enhance its effectiveness in processing sequential data.

2.2 Data Preprocessing

Given the nature of medical images, which can vary greatly in terms of size, orientation, and scale, comprehensive data preprocessing steps were implemented. These include resizing all images to a uniform dimension, applying normalization to standardize pixel values across the dataset, and augmenting the data with techniques such as rotation, flipping, and zooming to improve the model's generalization capability.

2.3 Training Strategy

The model was trained using a carefully selected batch size and learning rate to optimize the balance between computational efficiency and model performance. We employed a cross-entropy loss function, ideal for multi-class classification tasks, and an Adam optimizer for its adaptive learning rate capabilities. To combat the issue of overfitting, especially given the high accuracy target, we incorporated early stopping mechanisms based on the validation loss, alongside dropout layers within the model architecture.

2.4 Performance Evaluation

Model performance was rigorously evaluated using a split of the Medical MNIST dataset into training, validation, and test sets. This approach ensured that the model's accuracy was tested against unseen data, reflecting its potential real-world applicability. The evaluation metrics focused on accuracy, precision, recall, and F1 score, providing a comprehensive view of the model's ability to classify medical images accurately.

2.1 模型架构

我们的技术解决方案核心是一个混合深度学习模型，它将卷积神经网络（CNNs）与 Transformer 架构结合起来。这种设计选择由 Medical MNIST 数据集的复杂性和多样性驱动，要求一个能够捕获医学图像中错综复杂的局部特征及全局上下文关系的强大模型。

CNN Layers: 模型开始于一系列 CNN 层，负责从输入图像中提取分层特征。这部分模型包含多个卷积块，每个块包括卷积层、批量归一化（Batch Normalization）、ReLU 激活函数和最大池化（Max Pooling）。这些层逐步增加深度，以在每个级别捕获更复杂的特征，同时包含 dropout 层以预防过拟合。

Transformer Module: CNN 层之后，提取的特征输入到 Transformer 模块。该模块使用自注意力机制来分析图像不同区域之间的关系，使模型能够理解超出局部特征的更广阔上下文。Transformer 块设计有多头，以便同时捕获数据的不同方面，并包括位置编码（Positional Encoding）和层归一化（Layer Normalization）

以增强其处理序列数据的能力。

2.3 训练策略

模型采用精心选择的批量大小和学习率进行训练，以优化计算效率和模型性能之间的平衡。我们采用了交叉熵损失函数（Cross-Entropy Loss），适用于多类别分类任务，并使用 Adam 优化器因其自适应学习率能力。为了应对过拟合问题，特别是鉴于高准确率目标，我们在模型架构中加入了 dropout 层来避免过拟合。

2.4 性能评估

模型性能通过将 Medical MNIST 数据集分为训练、验证和测试集来进行严格评估。这种方法确保了模型准确率针对未见数据的测试，反映其潜在的现实世界应用能力。评估指标重点关注准确率，提供了模型分类医学的图像能力

3. Experiments

In this section, we delve into the experimental setup, detailing the dataset used, the selection of hyperparameters, evaluation criteria, and an analysis of the training process and results. The experiments were meticulously designed to test the robustness and efficiency of our model on the Medical MNIST dataset.

3.1 Datasets

The Medical MNIST dataset (https://www.kaggle.com/datasets/andrewmvd/medical-mnist) comprises a diverse collection of grayscale images categorized into several classes, each representing different medical organs or conditions. These images are derived from a variety of diagnostic procedures, making the dataset a comprehensive resource for developing and testing medical image classification models. For our experiments, the dataset was divided as follows:

Training set: 70% of the total images were used for training the model, allowing it to learn the distinguishing features of each class.

Validation set: 15% were used for validation during the training process, providing a means to fine-tune hyperparameters and prevent overfitting.

Test set: The remaining 15% served as the unseen data to evaluate the model's performance and generalization capability.

Each image was resized to a uniform dimension of 224x224 pixels to ensure consistency in input size, normalized to have pixel values between 0 and 1, and augmented with random rotations, flips, and zooms to enhance the dataset's variability and robustness.

3.1 数据集

医学 MNIST 数据集 (https://www.kaggle.com/datasets/andrewmvd/medical-mnist)

包含各种灰度图像集合，分为几个类别，每个类别代表不同的医疗器官或状况。

这些图像源自各种诊断程序，使该数据集成为开发和测试医学图像分类模型的

综合资源。 对于我们的实验，数据集划分如下:

训练集：总图像的 70% 用于训练模型，使其能够学习每个类别的区别特征。

验证集：训练过程中使用 15%进行验证，提供微调超参数和防止过度拟合的手

段。

测试集：剩余 15%作为未见数据，评估模型的性能和泛化能力。

每个图像的大小都调整为 224x224 像素的统一尺寸，以确保输入大小的一致性，

将像素值标准化为 0 到 1 之间，并通过随机旋转、翻转和缩放进行增强，以增

强数据集的可变性和鲁棒性。

3.2 Hyperparameters

The selection of hyperparameters was based on preliminary trials and literature review, aiming to balance training efficiency with model performance. The key hyperparameters include:

Learning Rate: The initial setting is 0.001 to adapt to the needs of the model training process. The code does not clearly reflect the learning rate scheduler that dynamically adjusts the learning rate.

Batch Size: The batch size of the training set is set to 64, while the batch size of the validation and test sets is set to 32. This is a compromise between computational resource constraints and the need for stable gradient estimation.

Number of Epochs: The training process is set to run for 10 epochs to ensure that the model has enough time to learn the number.

Dropout rate: Apply Dropout in the fully connected layer of the model to reduce overfitting, but the specific application rate needs to refer to the detailed definition of the MedicalMNISTCNN model.

Transformer header: 4 heads are used in the Transformer module, allowing the model to focus on different parts of the image at the same time.学习率（Learning Rate）：

初始设置为 0.001，以适应模型训练过程中的需求。代码中没有明确体现动态调整学习率的学习率调度器。

批量大小（Batch Size）：训练集的批量大小设置为 64，而验证集和测试集的批量大小设置为 32。这是在计算资源限制与稳定梯度估计需求之间的折衷。

Epoch 数量：训练过程设置为进行 10 个 epochs，以确保模型有足够的时间学习数

Dropout 率：在模型的全连接层中应用 Dropout 以减轻过拟合

Transformer 头：在 Transformer 模块中使用了 4 个头，允许模型同时关注图像的

不同部分。

4. Code description

1. get_medical_mnist_loaders

Functionality: Loads and processes the Medical MNIST dataset from the specified directory, applying data augmentation and normalization. It splits the dataset into training, validation, and testing subsets, returning data loaders for each subset to facilitate efficient batch processing during model training and evaluation.

2. calculate_accuracy

Functionality: Evaluates the model's performance on a dataset provided by a DataLoader. If a loss criterion is specified, it computes both the average loss and accuracy. The function operates in a no-gradient context to optimize performance and memory usage during evaluation phases.

3. TransformerBlock

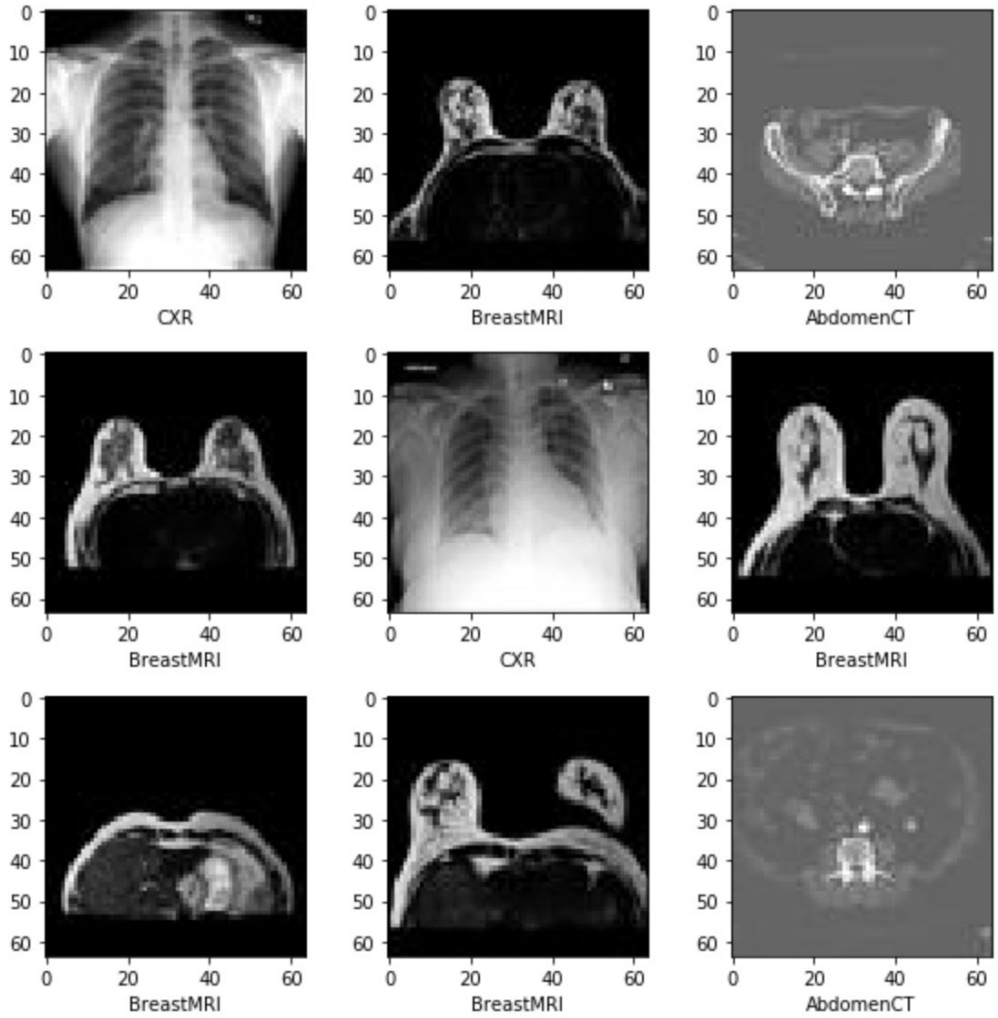
Functionality: Implements a transformer block that applies multi-head self-attention and position-wise feed-forward networks to the inputs. This module is designed to process sequences of embedded representations, enhancing the model's ability to capture complex dependencies within the data.

4. MedicalMNISTCNN

Functionality: Defines a neural network for classifying Medical MNIST images, combining convolutional layers for feature extraction with a transformer block for advanced processing. The architecture aims to leverage both local features extracted by CNN layers and global dependencies identified by the transformer mechanism.

5. train_model

Functionality: Manages the training process for the MedicalMNISTCNN model over a specified number of epochs. It performs forward passes, loss computation, backpropagation, and parameter updates via optimization. The function also evaluates the model on the validation set after each epoch and provides performance metrics, including loss and accuracy.



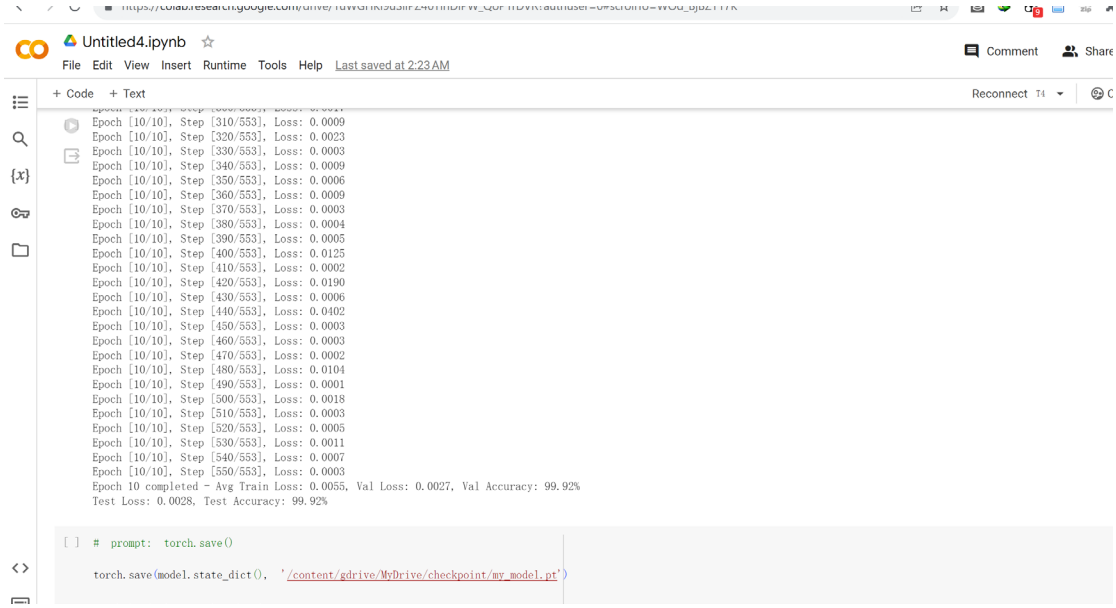
Medical MNIST examples

```
Epoch [7/10], Step [470/553], Loss: 0.0561
Epoch [7/10], Step [480/553], Loss: 0.0049
Epoch [7/10], Step [490/553], Loss: 0.0040
Epoch [7/10], Step [500/553], Loss: 0.0118
Epoch [7/10], Step [510/553], Loss: 0.0042
Epoch [7/10], Step [520/553], Loss: 0.0019
Epoch [7/10], Step [530/553], Loss: 0.0030
Epoch [7/10], Step [540/553], Loss: 0.0657
Epoch [7/10], Step [550/553], Loss: 0.0016
Epoch 7 completed - Avg Train Loss: 0.0217, Val Loss: 0.0044, Val Accuracy: 99.86%
Epoch [8/10], Step [10/553], Loss: 0.0013
Epoch [8/10], Step [20/553], Loss: 0.0220
Epoch [8/10], Step [30/553], Loss: 0.0053
Epoch [8/10], Step [40/553], Loss: 0.0058
Epoch [8/10], Step [50/553], Loss: 0.0230
Epoch [8/10], Step [60/553], Loss: 0.0100
Epoch [8/10], Step [70/553], Loss: 0.0014
```

```
Epoch [8/10], Step [470/553], Loss: 0.0004
Epoch [8/10], Step [480/553], Loss: 0.0009
Epoch [8/10], Step [490/553], Loss: 0.0003
Epoch [8/10], Step [500/553], Loss: 0.0006
Epoch [8/10], Step [510/553], Loss: 0.0075
Epoch [8/10], Step [520/553], Loss: 0.0006
Epoch [8/10], Step [530/553], Loss: 0.0037
Epoch [8/10], Step [540/553], Loss: 0.0088
Epoch [8/10], Step [550/553], Loss: 0.0012
Epoch 8 completed - Avg Train Loss: 0.0108, Val Loss: 0.0030, Val Accuracy: 99.91%
Epoch [9/10], Step [10/553], Loss: 0.0003
Epoch [9/10], Step [20/553], Loss: 0.0006
Epoch [9/10], Step [30/553], Loss: 0.0006
Epoch [9/10], Step [40/553], Loss: 0.3255
Epoch [9/10], Step [50/553], Loss: 0.3010
Epoch [9/10], Step [60/553], Loss: 0.0008
Epoch [9/10], Step [420/553], Loss: 0.0005
Epoch [9/10], Step [430/553], Loss: 0.0003
Epoch [9/10], Step [440/553], Loss: 0.0006
Epoch [9/10], Step [450/553], Loss: 0.0003
Epoch [9/10], Step [460/553], Loss: 0.0005
Epoch [9/10], Step [470/553], Loss: 0.0003
Epoch [9/10], Step [480/553], Loss: 0.0003
Epoch [9/10], Step [490/553], Loss: 0.0006
Epoch [9/10], Step [500/553], Loss: 0.0022
Epoch [9/10], Step [510/553], Loss: 0.0004
Epoch [9/10], Step [520/553], Loss: 0.0023
Epoch [9/10], Step [530/553], Loss: 0.0002
Epoch [9/10], Step [540/553], Loss: 0.0139
Epoch [9/10], Step [550/553], Loss: 0.0003
Epoch 9 completed - Avg Train Loss: 0.0082, Val Loss: 0.0027, Val Accuracy: 99.87%
Epoch [10/10], Step [10/553], Loss: 0.0010
Epoch [10/10], Step [20/553], Loss: 0.0002
```

6]

```
Epoch [10/10], Step [340/553], Loss: 0.0009
Epoch [10/10], Step [350/553], Loss: 0.0006
Epoch [10/10], Step [360/553], Loss: 0.0009
Epoch [10/10], Step [370/553], Loss: 0.0003
Epoch [10/10], Step [380/553], Loss: 0.0004
Epoch [10/10], Step [390/553], Loss: 0.0005
Epoch [10/10], Step [400/553], Loss: 0.0125
Epoch [10/10], Step [410/553], Loss: 0.0002
Epoch [10/10], Step [420/553], Loss: 0.0190
Epoch [10/10], Step [430/553], Loss: 0.0006
Epoch [10/10], Step [440/553], Loss: 0.0402
Epoch [10/10], Step [450/553], Loss: 0.0003
Epoch [10/10], Step [460/553], Loss: 0.0003
Epoch [10/10], Step [470/553], Loss: 0.0002
Epoch [10/10], Step [480/553], Loss: 0.0104
Epoch [10/10], Step [490/553], Loss: 0.0001
Epoch [10/10], Step [500/553], Loss: 0.0018
Epoch [10/10], Step [510/553], Loss: 0.0003
Epoch [10/10], Step [520/553], Loss: 0.0005
Epoch [10/10], Step [530/553], Loss: 0.0011
Epoch [10/10], Step [540/553], Loss: 0.0007
Epoch [10/10], Step [550/553], Loss: 0.0003
Epoch 10 completed - Avg Train Loss: 0.0055, Val Loss: 0.0027, Val Accuracy: 99.92%
Test Loss: 0.0028, Test Accuracy: 99.92%
```


Untitled4.ipynb
File Edit View Insert Runtime Tools Help Last saved at 2:23 AM
Comment
Share
+ Code + Text
Reconnect T4
Epoch [10/10], Step [310/553], Loss: 0.0009
Epoch [10/10], Step [320/553], Loss: 0.0023
Epoch [10/10], Step [330/553], Loss: 0.0003
Epoch [10/10], Step [340/553], Loss: 0.0009
Epoch [10/10], Step [350/553], Loss: 0.0006
Epoch [10/10], Step [360/553], Loss: 0.0009
Epoch [10/10], Step [370/553], Loss: 0.0003
Epoch [10/10], Step [380/553], Loss: 0.0004
Epoch [10/10], Step [390/553], Loss: 0.0005
Epoch [10/10], Step [400/553], Loss: 0.0125
Epoch [10/10], Step [410/553], Loss: 0.0002
Epoch [10/10], Step [420/553], Loss: 0.0190
Epoch [10/10], Step [430/553], Loss: 0.0006
Epoch [10/10], Step [440/553], Loss: 0.0402
Epoch [10/10], Step [450/553], Loss: 0.0003
Epoch [10/10], Step [460/553], Loss: 0.0003
Epoch [10/10], Step [470/553], Loss: 0.0002
Epoch [10/10], Step [480/553], Loss: 0.0104
Epoch [10/10], Step [490/553], Loss: 0.0001
Epoch [10/10], Step [500/553], Loss: 0.0018
Epoch [10/10], Step [510/553], Loss: 0.0003
Epoch [10/10], Step [520/553], Loss: 0.0005
Epoch [10/10], Step [530/553], Loss: 0.0011
Epoch [10/10], Step [540/553], Loss: 0.0007
Epoch [10/10], Step [550/553], Loss: 0.0003
Epoch 10 completed - Avg Train Loss: 0.0055, Val Loss: 0.0027, Val Accuracy: 99.92%
Test Loss: 0.0028, Test Accuracy: 99.92%
[ ]
# prompt: torch.save()
torch.save(model.state_dict(), '/content/gdrive/MyDrive/checkpoint/my_model.pt')