Project 2
FYS-STK4155

Krithika Gunasegaran, Oskar Idland, Erik Røset & Arangan Subramaniam
University of Oslo, Department of Physics

(Dated: November 20, 2024)

Add abstract here

https://github.com/Oskar-Idland/FYS-STK4155-Projects

I. INTRODUCTION

II. THEORY

III. METHODS & IMPLEMENTATION

A. Data Preprocessing and Model Architecture

The dataset consisted of breast cancer histopathological images classified into three categories: normal, malignant and benign. The latter two also came with an image mask file for the region of interest for each image. These were not included in our project. All images were preprocessed to a standardized size of 224 × 224 pixels. The dataset was split into training (64%), validation (16%) and test (20%) sets, with stratification maintained across splits to preserve class distribution. Data augmentation techniques, including random horizontal flips (probability 0.9), rotation (±15 degrees) and color jittering, were applied to address class imbalance particularly for the minority classes malignant and normal.

Two deep learning architectures were implemented and compared: a custom CNN and a pre-trained ResNet101 model[1]. The custom CNN comprised two convolutional layers ($3 \rightarrow 16 \rightarrow 32$ channels) with 3×3 kernels, each followed by ReLU activation and max pooling operations. Three fully connected layers ($325656 \rightarrow 512 \rightarrow 128 \rightarrow 3$) with dropout (p=0.25) completed the architecture. The ResNet101 model was pre-trained on ImageNet and fine-tuned for our specific task, with its final layer modified to accommodate the three-class classification problem.

B. Training and Evaluation

Both models were trained using the Adam optimizer with a learning rate of 0.001 for the custom CNN and 0.00005 for ResNet101. A step learning rate scheduler was implemented with a step size of 7 and γ of 0.1. Training utilized cross-entropy loss and incorporated early stopping with a patience of 2 epochs to prevent overfitting. Models were evaluated using accuracy, precision, recall, and F1-score metrics. Confusion matrices were generated to assess class-wise performance.

C. Implementation Details

The implementation used PyTorch framework and was executed on GPU hardware. Training was conducted with a batch size of 8, and model convergence was monitored through training and validation losses. The final evaluation was performed on the held-out test set to ensure unbiased performance assessment. All experiments were conducted with a fixed random seed for reproducibility.

IV. RESULTS & DISCUSSION

V. CONCLUSION

REFERENCES

[1] The PyTorch Foundation, "ResNet101," https://pytorch.org/vision/main/models/generated/torchvision.models.resnet101.html (Accessed: November 2024).

Appendix A: Code

Link to our GitHub repository: https://github.com/Oskar-Idland/FYS-STK4155-Projects