# Histopathology Image Classification with LoRA Fine-Tuning

Corentin Chabanol - Paul Tabbara

April 7, 2025

## 1 Introduction

This report documents the development of a deep learning model for histopathology image classification. The goal was to classify images into two categories using a pre-trained ResNet50 model with LoRA (Low-Rank Adaptation) fine-tuning. The project involved data preprocessing, model training, and evaluation, with a focus on improving performance through advanced fine-tuning techniques.

## 2 Dataset and Preprocessing

The dataset consists of histopathology images stored in HDF5 files, split into training, validation, and test sets. The following preprocessing steps were applied:

- **Normalization**: Images were normalized using mean $[0.485, 0.456, 0.406]$ and standard deviation $[0.229, 0.224, 0.225]$.
- **Data Augmentation**: For training, random resized crops, horizontal and vertical flips, and color jitter were applied.
- **Validation/Test Transformations**: Images were resized to 256x256 and center-cropped to 224x224.

## 3 Model Architecture

The base model is a ResNet50 pre-trained on ImageNet. Two fine-tuning approaches were explored:

### 3.1 Baseline Fine-Tuning

- The final fully connected layer was replaced with a dropout layer (p=0.5) followed by a linear layer with a single output.

- Only the new layers were trained, while the rest of the model was frozen.

### 3.2 LoRA Fine-Tuning

- LoRA layers were added to all convolutional layers in the model.
- The rank of the LoRA matrices was set to 8.
- The original convolutional layers were frozen, and only the LoRA parameters and the final classifier were trained.

## 4 Training

The models were trained with the following hyperparameters:

- Batch size: 32
- Learning rate: $1 \times 10^{-4}$
- Weight decay: $1 \times 10^{-4}$
- Loss function: BCEWithLogitsLoss with class balancing
- Optimizer: Adam
- Scheduler: ReduceLROnPlateau with patience=5
- Early stopping: Patience=10

## 5 Results

The performance of the models on the validation set is summarized below:

Table 1: Model Performance

| **Model** | **Validation Accuracy** | **Validation Loss** |
|---|---|---|
| Baseline | 0.8285 | 0.3830 |
| LoRA Fine-Tuning | 0.9391 | 0.2252 |

The LoRA fine-tuned model achieved significantly better performance, demonstrating the effectiveness of the approach.

## 6 Conclusion

The LoRA fine-tuning approach outperformed the baseline method, achieving a validation accuracy of 0.9391. This highlights the potential of LoRA for adapting large pre-trained models to specific tasks with limited computational resources. Future work could explore different LoRA ranks and layers to further optimize performance.

# Appendix: Code

The complete code for this project is available in the Jupyter notebook `report.ipynb`. Key components include:

- Custom dataset class (`HistoDataset`) for efficient data loading.
- LoRA layer implementation (`LoRAConv2d`).
- Training loops with progress tracking and model saving.