IN3063: Mathematics and Programming for AI 2025-26 Coursework

Report 2

Lung and Colon Cancer Histopathological Images Classification

Group 2

https://github.com/IvanUkrRada/P-MforAI/tree/IVAN/TASK2

Ivan Radavskyi, Mohamed Roshid, Sujit Bhatta, Wael Khafagi, Yaseen Mneimneih

Introduction

In this report we will closely investigate, analyse and make use of the Lung and Colon Cancer Histopathological Images dataset. It contains 25,000 color histopathology images grouped into 5 balanced classes (5,000 images per class).

Classes:

1. Lung benign tissue
2. Lung adenocarcinoma
3. Lung squamous cell carcinoma
4. Colon adenocarcinoma
5. Colon benign tissue

Image properties:

- Format: JPEG
- Resolution: 768 × 768 pixels
- Total images: 25,000

How the dataset was created:

- The 25,000 images were produced by augmenting a smaller set of original tissue images using the Augmentor package
- The original sample size reported is 750 lung images (250 per lung category) and 500 colon images (250 per colon category), then augmented to reach the final dataset size. (Borkowski et al., 2019)
-

Topic of study
Goal: Build and evaluate deep learning models using PyTorch for multi-class classification of histopathological images into the five tissue categories above.
Why this matters:
Histopathology is a gold-standard for cancer diagnosis, but reviewing slides is time-consuming and requires expert interpretation. Moreover, automation of the process will reduce misinterpretation made by bias. A reliable image classifier can support computer-aided diagnosis by helping distinguish benign tissue from cancerous tissue, and also separating cancer subtypes. Still it is crucial to apply both human and AI analysis since theobjective is human health. At the same time both sided analysis will strongly reduce the workload on doctors. Moreover it will provide a reliable tool that can be applied in areas where qualitative human analysis is not possible(time, price, etc).

Key considerations/assumptions

- Balanced classes make accuracy and macro-averaged metrics easier to interpret.
- Because many images are augmented versions of fewer originals, there is a risk that very similar images could appear across splits if splitting is not done carefully. This can inflate performance if not addressed in the experimental design.

This dataset setup supports a clear supervised learning problem: predict one of five pathology labels from an input histology image, then compare baseline CNN(firstbaseline.ipynb) vs improved architectures and tuned hyperparameters in later sections.

Model description (Baseline Convolutional Neural Network)

First model
To address the core requirement of classifying histopathology images into 5 classes, the first model was implemented as a small convolutional neural network (CNN). This initial attempt focuses on the essential building blocks needed for image classification and provides a working foundation that can be extended later (e.g., deeper feature extraction, stronger regularization, better preprocessing).(firstbaseline.ipynb)
Input and preprocessing

All images are converted to RGB and resized to 128×128, then converted to tensors. Resizing ensures a consistent input shape for batching and for the fully connected layer.

Architecture:
1) Convolutional layers (feature extraction)
- **Conv1:** Conv2d(3 → 8, kernel=3×3, padding=1)
- **Conv2:** Conv2d(8 → 16, kernel=3×3, padding=1)

Convolution layers are included because they are the standard mechanism for learning spatial patterns in images. The 3×3 kernels capture local textures and edges (important in tissue images/detalisation), while padding=1 preserves spatial resolution before pooling, so important boundary information is not lost too early(since it captures edges more precisely). The second convolution increases channel depth (8→16), allowing the model to represent more complex visual patterns than a single layer.
2) ReLU activation (non-linearity)
After each convolution, **ReLU** is applied. This is necessary because it introduces non-linearity, allowing the network to learn complex decision boundaries rather than behaving like a purely linear model.
3) Max pooling (downsampling and robustness)
A **2×2 max pooling** layer follows each ReLU output. Pooling is included to:
- reduce spatial size (**128→64→32**), lowering compute/memory,
- keep the strongest activations, helping the model focus on prominent features,
- add some robustness to small shifts/variations in tissue patterns.

After the second pooling step, the feature map size becomes:
- **16 channels × 32 × 32**

4) Flatten + Fully Connected layer (classification)
The feature maps are flattened into a vector of size:
16×32×32=1638416 \times 32 \times 32 = 1638416×32×32=16384
Then a single linear layer maps these features to 5 outputs:
- Linear(16384 → 5)

This layer is included to transform the learned visual features into class scores for the five categories. The model outputs logits because training uses CrossEntropyLoss, which internally applies softmax during optimization.

Training choices:

- **CrossEntropyLoss:** appropriate for multi-class classification (5 classes).
- **SGD (lr=0.01):** a standard, reliable optimizer for a first implementation; it verifies that the training pipeline (forward/backward/update) functions correctly.
- **80/20 split (800 train, 200 test):** provides a simple held-out test set to check generalization.
- **3 epochs:** enough to confirm learning behavior and get a first performance signal.

Experiment setup

- 2 convolutional layers (8, 16 channels)
- 1 fully connected layer
- No batch normalization
- No dropout
- SGD optimizer (no momentum)
- 3 epochs training
- 1000 total samples
- Dataset split: 800 train / 200 test
- Parameters: 83,317

Training performance

- Epoch 1: Train Acc **29.5%**, Test Acc **23.5%**, Train Loss **1.5532**
- Epoch 2: Train Acc **34.4%**, Test Acc **39.5%**, Train Loss **1.4331**
- Epoch 3: Train Acc **40.6%**, Test Acc **61.5%**, Train Loss **1.2683**

Final summary

- Best training accuracy: **40.6%**
- Best test accuracy: **61.5%**
- Final test accuracy: **61.5%**
- Final training loss: **1.2683**
- Total execution time: **68.5 s**

Improvement 1 - More feature extraction capacity (3 conv layers instead of 2, wider channels)

Change:

- First attempt: Conv(3→8) → Conv(8→16)
- Developed model: Conv(3→16) → Conv(16→32) → Conv(32→64) (three stages)

Motivation:

Histopathology images contain patterns at different levels (texture, cell clusters, tissue organization). Adding a third convolution layer and increasing channels gives the network enough capacity to learn hierarchical features, where deeper layers can represent more complex structures than shallow layers.

Improvement 2 - Stronger classifier head

Change:

- First attempt: one linear layer from flattened features directly to 5 classes
- Developed model: fc1 (16384→128) → fc2 (128→5)

Motivation: The extra dense layer (fc1) acts as a learned “feature mixing” stage that combines the spatial features extracted by the CNN into a compact representation before final classification. This typically improves separability between classes that look similar.
Improvement 3 - Input normalization

Change:
- First attempt: only Resize + ToTensor
- Developed model: adds
 Normalize(mean=[0.5,0.5,0.5], std=[0.5,0.5,0.5])
-

Motivation:
Normalization keeps input values in a consistent range, improving numerical stability and helping the optimizer converge faster and more reliably (especially important with Adam).
Improvement 4 - Better training setup (Adam + more data + train/val/test split)

Change:
- Optimizer: from SGD(lr=0.01) → Adam(lr=0.001)
- Data: from 1000 samples → 25,000 samples
- Split: from 80/20 train/test → 70/15/15 train/val/test

Motivation:
- Adam adapts learning rates per parameter, often speeding up learning early on.
- Using the full dataset improves generalization and reduces sampling bias.
- A validation set is essential for hyperparameter tuning (so decisions aren’t made using the test set).

Results of model:
Experimental setup
- Dataset: 25,000 images total (5 classes × 5000)
- Splits: 17,500 train / 3,750 val / 3,750 test
- Batch size: 32
- Optimizer: Adam, lr=0.001
- Epochs: 5
- Input: 128×128 RGB, normalized
- Parameters: 2,121,509

Training + validation performance (per epoch)
- **Epoch 1:** Train Acc **82.30%**, Val Acc **89.57%**, Val Loss **0.2361**
- **Epoch 2:** Train Acc **91.50%**, Val Acc **91.52%**, Val Loss **0.2161**
- **Epoch 3:** Train Acc **95.09%**, Val Acc **94.61%**, Val Loss **0.1479**
- **Epoch 4:** Train Acc **97.15%**, Val Acc **93.65%**, Val Loss **0.1694**
- **Epoch 5:** Train Acc **98.19%**, Val Acc **95.57%**, Val Loss **0.1226**

Final:
- Final training accuracy: **98.19%**

- Final validation accuracy: **95.57%**
- Total training time: **2436.7 s**

Interpretation:
Training accuracy steadily increases; validation peaks at 95.57%, suggesting the model generalizes well, with a small sign of overfitting around epoch 4 before recovering.
To optimize performance, we carried out a small grid search over key training hyperparameters that strongly affect convergence and generalization: learning rate (lr), batch size (bs), and weight decay (wd).
Each configuration was trained using the same fixed 70/15/15 split (train/validation/test) to ensure a fair comparison, and the validation set was used to select the best hyperparameters.

Hyperparameter search range
We evaluated the following combinations:

- **Learning rate (Adam):** 0.001, 0.0003, 0.0001
- **Batch size:** 32, 64
- **Weight decay:** 0.0, 0.0001

For each run, we recorded:

- Best validation accuracy
- Validation loss
- Time(Used GPU for all trainings/different GPU can produce different runtime, so all models have to be runned to evaluate models not GPU)

GRID RESULTS
```
Run  1 | val_acc= 95.87% | lr=0.001 bs=32 wd=0.0 | val_loss=0.1086 | 1085.1s
Run  2 | val_acc= 94.21% | lr=0.001 bs=32 wd=0.0001 | val_loss=0.1495 | 1040.0s
Run  5 | val_acc= 93.52% | lr=0.0003 bs=32 wd=0.0 | val_loss=0.1769 | 966.6s
Run  4 | val_acc= 92.85% | lr=0.001 bs=64 wd=0.0001 | val_loss=0.1702 | 952.8s
Run  3 | val_acc= 92.51% | lr=0.001 bs=64 wd=0.0 | val_loss=0.1854 | 988.5s
Run  6 | val_acc= 92.05% | lr=0.0003 bs=32 wd=0.0001 | val_loss=0.1967 | 968.6s
Run  8 | val_acc= 90.72% | lr=0.0003 bs=64 wd=0.0001 | val_loss=0.2324 | 957.0s
Run  7 | val_acc= 90.35% | lr=0.0003 bs=64 wd=0.0 | val_loss=0.2521 | 957.2s
Run  9 | val_acc= 88.69% | lr=0.0001 bs=32 wd=0.0 | val_loss=0.2740 | 962.8s
Run 12 | val_acc= 87.07% | lr=0.0001 bs=64 wd=0.0001 | val_loss=0.3154 | 995.2s
Run 10 | val_acc= 86.88% | lr=0.0001 bs=32 wd=0.0001 | val_loss=0.3058 | 965.4s
Run 11 | val_acc= 86.35% | lr=0.0001 bs=64 wd=0.0 | val_loss=0.3443 | 1044.8s
```

Best hyperparameters (selected by validation)
The best-performing configuration on the validation set was:

- lr = 0.001
- batch size = 32
- weight decay = 0.0
- Best validation accuracy: 95.87% (val loss 0.1086)

Observations from the grid search

- The learning rate had the biggest effect: 0.001 consistently outperformed 0.0003 and 0.0001.
- Batch size 32 performed better than 64 for most settings (likely due to noisier gradients improving generalization).
- Adding weight decay did not help in this setup; wd=0.0 beat wd=1e-4 for the best lr and batch size.

Final evaluation on the test set

After selecting hyperparameters using validation, we evaluated the chosen model **once** on the held-out test set:

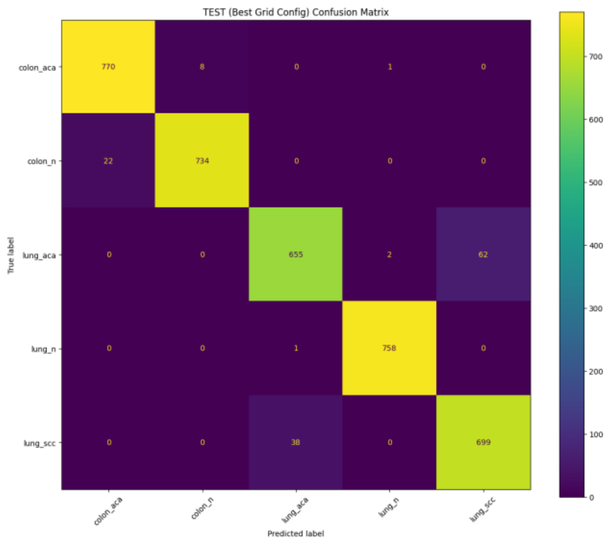
- Test Accuracy: 96.43%
- Macro Precision: 96.40%
- Macro Recall: 96.35%
- Macro F1-score: 96.36%

Per-class results

- **colon_aca:** Precision 97.22%, Recall 98.84%, F1 98.03% (n=779)
- **colon_n:** Precision 98.92%, Recall 97.09%, F1 98.00% (n=756)
- **lung_aca:** Precision 94.38%, Recall 91.10%, F1 92.71% (n=719)
- **lung_n:** Precision 99.61%, Recall 99.87%, F1 99.74% (n=759)
- **lung_scc:** Precision 91.85%, Recall 94.84%, F1 93.32% (n=737)

Interpretation

Overall performance is strong, with most classes above ~97% F1. The weakest class is **lung_aca**, which shows the lowest of recall 91.10% with 655 predicted labels (Figure 1), suggesting it is the hardest to predict class (consistent with similar visual morphology across lung subtypes).



(Confusion Matrix BR, Figure 1)

Why dropout was not included (finale.ipynb):

We tested dropout as an improvement (Dropout2d + Dropout) to reduce overfitting. However:

- Validation accuracy decreased compared with the non-dropout model.
    - Dropout run: best val ≈ 91.52%
    - Final tuned non-dropout: best val 95.87%

- This suggests dropout caused underfitting in this setting: too much information was being dropped, reducing the model's effective capacity.
- Additionally, the tuned non-dropout model already generalizes strongly (test 96.43%), so regularization from dropout was not necessary and hurt performance.

REFERENCE

Borkowski, A.A., Bui, M.M., Thomas, L.B., Wilson, C.P., DeLand, L.A. and Mastorides, S.M. (2019). Lung and Colon Cancer Histopathological Image Dataset (LC25000). *arXiv:1912.12142 [cs, eess, q-bio]*. [online] Available at: https://arxiv.org/abs/1912.12142v1.

Larxel (2019). *Lung and Colon Cancer Histopathological Images*. [online] Kaggle.com. Available at: https://www.kaggle.com/datasets/andrewmvd/lung-and-colon-cancer-histopathological-images/data.