Amir Aynede
Department of Pharmacy and Biotechnology (FaBiT)
Alma Mater Studiorum – University of Bologna

LC25000 Cancer Classification Project Report

Introduction

Cancer diagnosis through histopathological examination is a gold-standard method in clinical practice. However, traditional manual analysis of tissue slides is both time-intensive and subjective, often leading to inter-observer variability. Leveraging deep learning models, particularly convolutional neural networks (CNNs), presents a promising solution for automating and standardizing diagnostic workflows. This project aimed to design, implement, and evaluate a complete machine learning pipeline for classifying cancer subtypes from histopathology images using the publicly available LC25000 dataset. The pipeline includes data preprocessing, model training, evaluation, visualization of results, and interpretability through Grad-CAM analysis.

Project Objectives

Develop an end-to-end deep learning system for classifying five histopathological classes: lung adenocarcinoma, lung squamous cell carcinoma, benign lung tissue, colon adenocarcinoma, and benign colon tissue. Employ a ResNet18 architecture as the base model for fine-tuning on the dataset. Utilize comprehensive evaluation metrics such as accuracy, precision, recall, F1-score, and confusion matrices.

Ensure model interpretability using Grad-CAM Heatmaps

Follow a modular structure with clean code organization and reproducibility.
While the original plan served as a guiding framework, certain adaptations were made during implementation to improve robustness and usability. These adjustments are reflected in the final structure documented in the project's README.md.

Data Handling

The dataset was downloaded from Kaggle and automatically extracted into a dedicated data directory. To facilitate experimentation and ensure a consistent workflow, the dataset was split into training, validation, and test sets using a custom script. This separation was necessary to support reliable evaluation and prevent data leakage across stages.
Unlike the initial plan that suggested loading the full dataset and then splitting randomly, the updated implementation created explicit folders (train/, val/, test/) in the filesystem. This allowed the use of torchvision.datasets.ImageFolder, which leverages folder names as class labels, streamlines data loading and class indexing.

Model Architecture

A pretrained ResNet18 model was used as the core of the classifier. The final fully connected layer was replaced with a new layer to match the number of target classes (five). During training, earlier layers were optionally frozen to leverage the pre-trained features while reducing the risk of overfitting. A critical enhancement involved dynamically adjusting the model's output layer based on the dataset class count. This ensured the model architecture matched the data split regardless of future dataset changes, flexibility not explicitly mentioned in the proposal but implemented during development for long-term maintainability.

Training

The training process was orchestrated with detailed logging of performance metrics across epochs. The model was optimized using the Adam optimizer and cross-entropy loss function. The script saved the best-performing model based on validation accuracy, preserving checkpoints for future evaluation. To aid interpretability, training curves for loss and accuracy were generated and saved as static plots and animations. These visualizations helped monitor convergence trends and detect signs of underfitting or overfitting.

Evaluation

Evaluation was performed on the test dataset using standard classification metrics. A dedicated script outputted a full classification report, a confusion matrix, and per-image predictions saved in CSV format. These results were used for further visualization and analysis. One deviation from the proposal was the addition of utilities to visualize correct and incorrect predictions. These scripts generated image grids for random predictions and misclassifications, offering intuitive insights into model behavior.

Grad-CAM Visualization

Grad-CAM was employed to interpret the model's decision-making process. By generating Heatmaps overlaid on original images, the system highlighted areas most influential in prediction. A random image from the test set was selected during each run to automate and diversify the interpretation workflow. This interpretability tool added practical value beyond what was originally scoped, making the model outputs more transparent to users and potentially aiding diagnostic trust.

Pipeline Automation

The entire workflow was codified into a GitHub Actions CI pipeline, enabling reproducible runs upon each push to the main branch. Steps included:
Installing dependencies
Downloading and preparing data

Training the model
Evaluating performance
Generating visual outputs, including Grad-CAM
This automated pipeline ensures end-to-end reproducibility and reflects modern machine learning development practices.

Jupyter Notebook & Usability Enhancements

To accommodate interactive use (e.g., on Google Colab), a Jupyter notebook version of the pipeline was created. It includes all essential steps with markdown documentation for clarity. The user is guided through environment setup, dataset preparation (including placement of the kaggle.json token), training, and visualization. This adaptation was not part of the initial plan but proved crucial for ease of use, especially for users unfamiliar with running CLI-based scripts.

Conclusion

The LC25000 Cancer Classification project successfully delivered a comprehensive and reproducible pipeline for histopathology image classification using deep learning. While rooted in the original proposal, the project evolved to include valuable practical enhancements such as automated data splits, dynamic class handling, CI integration, and Grad-CAM grids. By adhering to modularity and transparency, the system offers a strong foundation for further experimentation or deployment. This work underscores the promise of AI-assisted diagnostics in pathology and offers a framework that can be extended to larger or more complex medical imaging datasets in the future.

Future Directions and Related Projects

The current project serves as a robust baseline for automated cancer image classification. However, several potential directions exist for expanding its scope and impact:
1. Multi-class and Multi-label Expansion:
Future datasets could include more tissue types or allow multiple labels per image (e.g., tumor type + grade). This would simulate real-world diagnostic complexity and require more sophisticated models.
2. Whole-Slide Image (WSI) Analysis:
Instead of using cropped patches, working with WSIs and applying tiling, attention-based aggregation, or transformers, could significantly elevate realism and clinical relevance.
3. Semi-supervised and Self-supervised Learning:
Many medical datasets have limited annotations. Exploring models that learn from unlabeled data or use contrastive/self-supervised pretraining can reduce reliance on expert labeling.
4. Model Calibration and Uncertainty Estimation:
For deployment in clinical workflows, models must be reliable and trustworthy. Techniques

like Monte Carlo dropout or deep ensembles can help quantify model uncertainty in predictions.

5. Integration with Clinical Metadata:
Combining image data with patient-level features (e.g., age, smoking history, genetic markers) could enable multi-modal models and yield more personalized diagnostic insights.

6. Deployment as a Web App or API:
The trained model could be deployed via Flask/FastAPI and integrated with frontends to support clinician-facing tools. Real-time Grad-CAM visualizations could boost interpretability.

7. Federated Learning in Multi-institutional Settings:
To preserve data privacy across hospitals, future projects could use federated learning, training models on distributed data without centralizing it.

8. Domain Adaptation and Robustness:
Models trained on a single dataset may fail on new data distributions. Future work could explore domain generalization or style transfer techniques to improve robustness to new hospitals or scanners.

9. Active Learning for Efficient Annotation:
Building annotation tools that incorporate model suggestions (e.g., most uncertain or influential samples) could streamline the process of curating new datasets.

10. Collaboration with Pathologists:
Co-designing tools with medical experts, incorporating feedback loops into training, and validating predictions on expert-verified datasets could make the system clinically deployable.

These directions would not only refine the technical pipeline but also bring the work closer to real-world diagnostic integration, scaling both its impact and complexity.