

CNN Cancer Detection

Introduction to Deep Learning - Project Report

Date - 19/06/2025

Report By - Wai Lin Kyaw

University of Colorado Boulder

Outline

1. Problem Statement
2. Dataset
3. Exploratory Data Analysis (EDA)
4. Experiment 1 - Simple CNN
5. Experiment 2 - ResNet18 Transfer Learning
6. Conclusion

Complementary Information

1. [Github Repository](#)
2. [Jupyter Notebook](#) (with details analysis, training and validation)
3. [Kaggle Competition](#)

Problem Statement

We would like to create an algorithm to identify metastatic cancer in small image patches taken from larger digital pathology scans.

This solution and our approach will follow the framework of [Kaggle Competition](#).

Dataset

We will use the dataset outlined by the kaggle competition, PatchCamelyon. Here is the original dataset - [PCam Dataset](#).

However, we will use the variant published by kaggle. The difference is that the original images are in 92x92 for resolution while the kaggle competition dataset only focuses on 32x32.

The images are taken from digital pathology scans.

Exploratory Data Analysis (EDA)

For EDA, we analysed the dataset with a few steps.

1. Understand what files are included (training labels, image files, test files, sample submission, etc.,)
2. Deeper look into the structure of each of the above files
3. EDA on training labels (checking imbalance, duplicates, the volume of the dataset)
4. Finally, we try to inspect both malignant & non-malignant images.

As I am not an expert in this area and neither a pathologist, I don't have any idea what the scan of the tissues mean. However, I was able to verify that the image files are indeed tissue scans.

The data provided by kaggle is well structured and clean so that we didn't need to do further data cleaning.

Please refer to the EDA section of [Jupyter Notebook](#) for the code & details.

Image Augmentation

For our model to be less prone to slight variation in how the images are fed in, we also applied image augmentation techniques, such as rotation, horizontal / vertical flips, resizing & cropping.

We do this exhaustively on the training data, while keeping minimal transformation on the validation dataset.

Experiment 1

We will start by experimenting with *Simple Convolutional Neural Network (CNN)*. We want to have three convolution layer, following the below thought process:

Layer	Channels	Reasoning
Conv 1	32	To extract low level features (e.g., edges, textures)
Conv 2	64	To learn abstract features (e.g., shapes, blobs)
Conv 3	128	Deeper layer to learn high level overview (e.g., tumor boundaries, structure)

Each layer gets applied max pooling to improve computational efficiency. This convolution layer is followed by a fully connected layer to produce a single value - as we are targeting binary class classification.

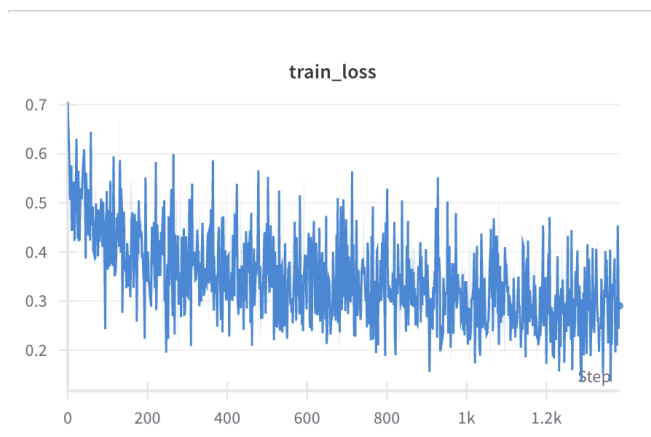
We also added error loss tracking using [Weights & Biases](#).

Please refer to Experiment 1 section of the [Jupyter Notebook](#) for the training script.

Training & Loss



The model is trained on Google Colab with 1 x T4 GPU with 12 GB of RAM. Below is our training config.

```
{  
  "num_epochs": 5,  
  "batch_size": 64,  
  "learning_rate": 1e-4  
}
```



We can see that the training loss decreases gradually. Instead of logging by epoch, I chose to log every 10 steps. Due to limited time & compute, I only trained 5 epochs. As a result, I achieved AUC of 0.9554 with this setting.

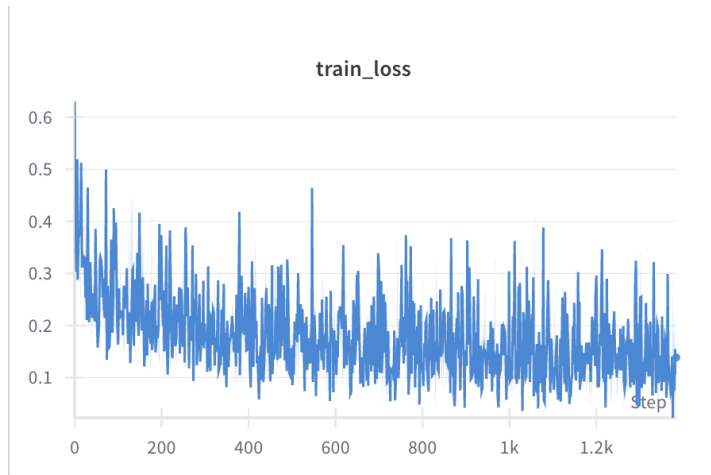
When I submitted the results to Kaggle, I got a score of **0.9239**.

Submission and Description	Private Score ⓘ	Public Score ⓘ	Selected
 resnet18-submission.csv Complete (after deadline) · 2h ago · Transfer learning with ResNet18. Trained on T4 GPU with ...	0.9384	0.9648	<input type="checkbox"/>
 simple-cnn-submission.csv Complete (after deadline) · 3h ago · The model is trained with Simple Convolutional Neural Ne...	0.8523	0.9239	<input type="checkbox"/>

You can also check the training metrics recorded here - [W&B Metrics](#).



Experiment 2

I also tried out transfer learning by fine tuning on resnet18 - which is a residual neural network released from microsoft. I used the same config setting, environment, and training parameters.



Since the earlier steps, the training loss is under 0.5 which is smaller compared to our Simple CNN earlier. The training loss decreases gradually, which is a good indicator. We used Adam optimizer with a learning rate of $1e-4$.

This produces even better AUC. When submitted to Kaggle with the predictions on the test dataset, I got a score of **0.9648**.

Submission and Description	Private Score ⓘ	Public Score ⓘ	Selected
 resnet18-submission.csv Complete (after deadline) · 3h ago · Transfer learning with ResNet18. Trained on T4 GPU with ...	0.9384	0.9648	<input type="checkbox"/>
 simple-cnn-submission.csv Complete (after deadline) · 4h ago · The model is trained with Simple Convolutional Neural Ne...	0.8523	0.9239	<input type="checkbox"/>

Based on our two experiments, we can see that even our Simple CNN produced an impressive score for 5 epochs.

I also saved the model weights so that we can reuse them later. These can be found on the Github repository.

Conclusion

If time permits, and more compute is available, I would like to try out training with more epochs – probably up to 20 epochs.

In summary, we can say that both Simple CNN & ResNet18 transfer learning yield good results. However, in the real world setting, we should test with more data, check on false negatives which may be critical for use cases like cancer detection.