Metastatic Tissue Detection (Cancer Detection)

Ildar Mamin *Department of Information Technology Course: INFO-6147 Deep Learning with PyTorch December 9, 2025*

Abstract

Pathologists today face a big challenge: they must examine thousands of tissue slides manually to find cancer. This process is very slow, and because doctors get tired, they can make mistakes. This project proposes a solution using Deep Learning to help automate the detection of metastatic tissue. I created a Convolutional Neural Network (CNN) and trained it on the PatchCamelyon (PCAM) dataset. Because of hardware limits, I used a subset of 5,000 images. After training for 20 epochs, the model achieved a validation accuracy of 85.50%. I also built a web application using Streamlit to show how this tool can work in real life. This report explains my methods, the model architecture, and the results.

***Keywords—Deep Learning, CNN, Cancer Detection, PyTorch, Medical Imaging*.**

I. Introduction

Cancer diagnosis is a critical task in medicine. Currently, the main way to find metastasis is by looking at tissue slides under a microscope. This is done by pathologists. However, this method has serious problems. It is very time-consuming and subjective. When a pathologist has to check hundreds of slides in one day, fatigue sets in, and this can lead to errors in diagnosis.

To solve this, we can use automated tools. Deep Learning, and specifically Computer Vision, allows computers to analyze images much faster than humans. The goal of this project is to build an AI model that can look at a scan of lymph node tissue and decide if it is healthy or if it contains a tumor.

This project serves as a "second opinion" tool. It triages the images and flags the suspicious ones for the doctor. This way, the doctor can focus only on the important areas. The objectives of this work were to implement a CNN using PyTorch, train it on histopathology data, and deploy it as a web application for real-time inference.

II. Methodology

A. Dataset Description

I chose the **PatchCamelyon (PCAM)** dataset for this project. This is a standard dataset used for testing machine learning models in pathology. It contains images of H&E stained lymph node sections.

- **Image Type:** Color images (RGB).
- **Size:** 96x96 pixels.
- **Classes:** Binary classification. Label 0 is Healthy, and Label 1 is Tumor.

The original dataset is very large. To make the project manageable and to test how well a model can learn from limited data, I selected a balanced subset of **5,000 images**.

B. Data Preprocessing

Before training, I prepared the data using several steps:

1. **Resizing:** I resized all images to 96x96 pixels to make sure they are consistent.
2. **Normalization:** I normalized the pixel values. I used a mean of 0.5 and a standard deviation of 0.5 for all three color channels. This helps the neural network train better.
3. **Splitting:** I divided the dataset into two parts. 80% of the data was used for Training, and 20% was reserved for Validation to check the accuracy.

C. Model Architecture

I designed a custom Convolutional Neural Network (CNN). I called it MyCNN. I decided to use a custom architecture instead of a pre-trained one to better understand the feature extraction process. The model has 3 main convolutional layers:

1. **First Layer:** It takes the 3 input channels (RGB) and outputs 32 feature maps. I used a kernel size of 3 and padding of 1. This is followed by Batch Normalization and Max Pooling (2x2) .
2. **Second Layer:** It increases the depth from 32 to 64 feature maps. Again, it uses Batch Normalization and Max Pooling .
3. **Third Layer:** It goes from 64 to 128 feature maps .

After the convolutional layers, I flattened the data. I added a fully connected layer with 512 neurons. Importantly, I added a **Dropout layer with a rate of 0.5**. This randomly turns off neurons during training, which prevents the model from memorizing the specific images (overfitting).

The final output is a single neuron with a **Sigmoid activation function**. This gives a probability score between 0 and 1.

III. Implementation

I wrote the code in Python using the **PyTorch** library and **Jupyter Notebook** .

A. Training Settings

For the training process, I used the following hyperparameters:

- **Optimizer:** Adam. It is a very popular optimizer that adjusts the learning rate automatically. I set the learning rate to 0.001.
- **Loss Function:** Binary Cross Entropy Loss (BCELoss). This is the standard loss function for binary classification tasks.
- **Batch Size:** 64.
- **Epochs:** 20.

B. Training Loop

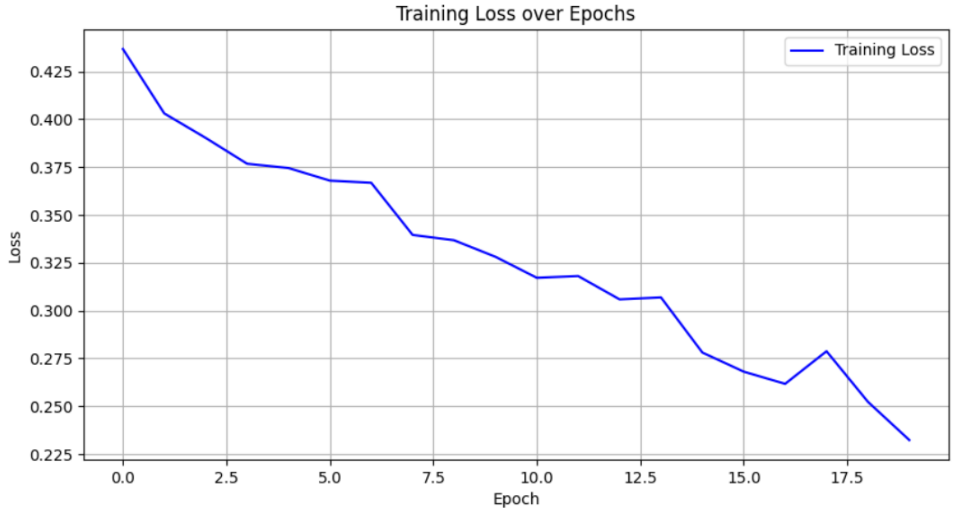
The training loop was standard. For each batch of images, the model calculated the prediction and the loss. Then, it updated the weights using backpropagation. I monitored the training loss and the validation accuracy at the end of every epoch .

IV. Results

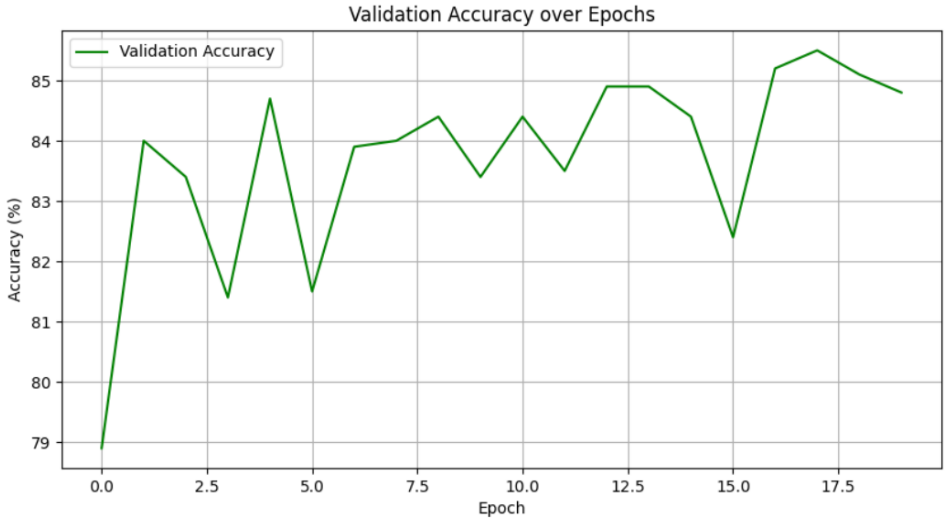
A. Model Performance

The training results were quite good.

- **Loss:** The training loss started high at around 0.43. Over the 20 epochs, it went down steadily to **0.23**. This shows the model was learning well.



- **Accuracy:** The validation accuracy improved quickly. By epoch 18, it reached a maximum of **85.50%**. In the final epochs, it stayed stable around 84-85%.



Considering I only used 5,000 images, an accuracy of 85.5% is a strong result. It proves that the 3-layer CNN is capable of extracting the right features to identify cancer cells.

B. Web Application

To make this project useful for end-users, I deployed the model using **Streamlit**. The app has a simple interface where a user can drag and drop an image file.

- When I tested it with a "Healthy" image, the model correctly said "Healthy Tissue" with **77.11% confidence**.
- When I tested it with a "Tumor" image, it detected the tumor with **98.58% confidence**.

V. Conclusion

In this project, I successfully built a deep learning system to detect metastatic tissue. By training a custom CNN on a subset of the PCAM dataset, I achieved an accuracy of **85.50%**. The project shows that AI can effectively assist pathologists by filtering out healthy images and flagging suspicious ones.

However, there are limitations. The main one is the dataset size. I only trained on a small subset (5,000 images). If I used the full dataset, the accuracy would likely be higher.