CS230

Identifying Metastatic Cancer through Deep Learning

Jonathan Tan
jontan27@stanford.edu

Lucky Adike
luckya@stanford.edu

Swadhin Ajay Thakkar
swadhin@stanford.edu

Abstract

This project aims to use a deep Convolutional Neural Network (ConvNet) to identify the presence of metastatic cancer in lymph nodes. We trained a four layer ConvNet from scratch against a subset of the PatchCamelyon (PCam) dataset, taken from a 2019 Kaggle Competition*. The latest model achieved 97.2% accuracy against the test set.

1 Introduction

With over 2 million new cases in 2018 alone, breast cancer is the second most commonly occurring cancer in the world [1]. Metastatic involvement of lymph nodes is a key prognostic characteristic of breast cancer. Prognosis is poorer when cancer has spread to the lymph nodes, so pathologists take exceptional care when examining and diagnosing lymph nodes. Even for the most experienced pathologists, the diagnostic procedure is tedious and time-consuming. According to Cancer Research UK, almost half of the people who get cancer are diagnosed late [2]. Luckily, with the rise of digital pathology, high resolution digital images of histological scans of lymph node sections are becoming increasingly available, presenting us the opportunity to develop a deep learning model to aid in identifying metastatic cancer cells.

The problem we are tackling is a binary classification one. The input to our algorithm are 96x96 pixel color images of hematoxylin and eosin (H&E) stained slides of sentinel lymph node sections from the PatchCamelyon (PCam) challenge dataset. We use a Convolutional Neural Network to output a predicted value specifying whether or not the input image contain signs of metastatic cancer.

2 Related work

2.1 Previous Learning Methods / Algorithms

Among those surveyed, some studies directly used pre-trained deep learning model as the feature extractor to extract high-level features to train a classifier and some used fine-tuned the pre-trained models by changing the last layer and trained models on new medical datasets. In addition, complex CNN models were also used for transfer learning in Esteva et al [3] eg. a pre-trained inception v3 in transfer learning, which included 98 convolutional layers and 14 pooling layers. One another study investigated and compared two CNN models as feature extractor, one was pre-trained on natural scene images and the other was pre-trained on bladder images, and the experimental results indicated that the feature extractors trained on bladder images achieved better performance. It indicates the promise potential of end-to-end trainable deep learning model in the future with large training data. Wald, N., et al [4] Uses statistical analysis by performing Fourier Transform on Infrared Images of the cells. It relies on different infrared spectral features allowing automated identification of cell types. FTIR spectroscopy has similar detection power as immunohistochemistry. Shen, Wei et al. [5] authors

*https://www.kaggle.com/c/histopathologic-cancer-detection/leaderboard

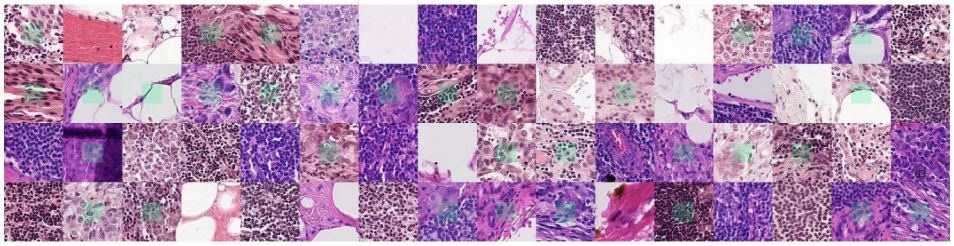
propose a hierarchical learning framework - Multi-scale Convolutional Neural Networks (MCNN) to capture nodule heterogeneity by extracting discriminative features from alternatingly stacked layers. They use CT scan images. Steiner, David F., et al. [6] Utilize a proof of concept assistant tool, this study demonstrates the potential of a deep learning algorithm to improve pathologist accuracy and efficiency in a digital pathology workflow.

2.2 Clever Approaches

Hu, Zilong, et al. [7] mentions in one study that the performance of brain tumor segmentation using a deep learning model suffered moderate decrease when trained with multi-institutional data, so the current transfer learning approach mostly trained on non-medical images. It also noted that a big challenge using CNN models for cancer detection was the size variation of the target objects. To overcome this problem, several studies proposed to train the same models using different scales of image data and then fusing the outputs to gain the final result. The study by Shen, Wei et al. [8] introduced a multi-crop pooling operation to replace the traditional pooling layer to directly capture multi-scale features from the image. Several studies also investigated the potential of combining multimodality information for cancer detection. The study by Xu et al. [9] combined extracted deep features and non-image multimodal clinical information to train a classifier. Danaee, Padideh, et. al. [10] used a Stacked Denoising Autoencoder (SDAE) to deeply extract functional features from high dimensional data and verified its usefulness by combining with the use of supervised learning methods.

3 Dataset and Features

We trained our model using a subset of the PCam dataset in which duplicate images had been removed. The PCam dataset is a 10x undersampling of the Camelyon16 Challenge Dataset which contains 400 hematoxylin and eosin (H&E) stained whole slide images of sentinel lymph node sections, collected in the Radboud University Medical Center and University Medical Center Utrecht, and digitized at 2 different centers using a 40x objective[†].



Example images taken from the PCam Github page[*]. Green boxes indicate tumor tissue in center region, dictating a positive label.

In total, the project dataset consisted of 220,025 color images (96px x 96px) with a corresponding binary label indicating whether or not the image contained metastatic tissue. The class distribution is about a 60:40 split between negative and positive examples. We split this dataset into 80% training examples (176,020), 10% dev examples (22,002), and 10% test examples (22,003). We sped up the learning process for our model by normalizing all input images with a mean and standard deviation of 0.5 for each channel. We added a transformation to the input images, giving them a 50% chance of being flipped horizontally and a 50% chance of being flipped vertically, to reduce overfitting on our training set.

4 Methods

4.1 Framework and Hardware

We built our model using PyTorch, an open-source Python machine learning library. We chose PyTorch because, when compared to other libraries/frameworks, was easier to ramp-up on and use,

[†] camelyon16.grand-challenge.org/Background/
[*] https://github.com/basveeling/pcam

granting the ability for rapid prototyping, without compromises in speed and performance*. We trained our models using Microsoft Azure's Standard_NC6 Deep Learning Virtual Machine, powered by a single NVIDIA Tesla K80 graphics card.

Size	vCPU	Memory: GiB	Temp storage (SSD) GiB	GPU	GPU memory: GiB	Max data disks	Max NICs
Standard_NC6	6	56	340	1	8	24	1

Microsoft Azure's Standard_NC6 Deep Learning Virtual Machine Size†.

4.2 Loss Function

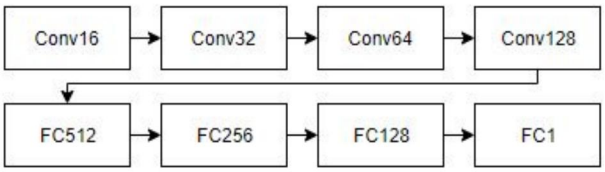
Since this is a binary classification problem, our loss function is Binary Cross-Entropy Loss (BCE Loss). Specifically, we utilized PyTorch's BCEWithLogitsLoss function (shown below) which combines the Sigmoid and the BCE Loss calculation into one single class. We chose this version because, by combining the operations into a single layer, PyTorch takes advantage of the log-sum-exp trick**. Therefore, it is more numerically stable than if we were to use a plain Sigmoid followed by a BCELoss.

$$L(X,Y) = -\frac{1}{n}\sum_{i=1}^{n} y_i log(\sigma(x_i)) + (1-y_i) log(1-\sigma(x_i)) \tag{1}$$

For the function shown above, x_i is the output of our model for the i^{th} input image and y_i is the corresponding target label. As previously stated, the Sigmoid function is applied to x_i as part of the loss function.

4.3 Model

Below we outline the architecture of our model. For each convolutional layer, we apply batch normalization, ReLU, and then max pooling (in that order) to get the output. For each fully connected layer (excluding the output layer), we apply batch normalization and ReLU (in that order) to get the output and also add drop-out regularization $(p = 0.5)$ to address over-fitting. For the final version of our model, we used a learning rate of 0.0005, batch size of 16, and a hybrid approach of training with both the Adam and Standard Gradient Descent optimizers. Explanations for our hyperparameter values and the hybrid approach are included in the next section.



Model architecture. Conv'N' denotes a convolutional layer with N filters and FC'M' denotes a fully connected layer with M outputs.

5 Experiments/Results/Discussion

The experiments below are shown in chronological order of how we developed our final model. Each experiment builds off the previous experiments' results.

5.1 Hyperparameter Tuning

Although it's commonly stated that batch size selection should be dependent upon the memory of the GPU being used, *Revisiting Small Batch Training For Deep Neural Networks* [11], by Dominic Masters and Carlo Luschi, proposes that smaller batch training provides improved generalization

*https://wrosinski.github.io/deep-learning-frameworks/

†https://docs.microsoft.com/en-us/azure/virtual-machines/windows/sizes-gpu

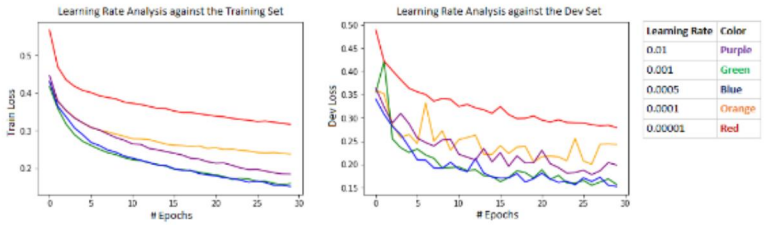
**https://makarandtapaswi.wordpress.com/2012/07/18/log-sum-exp-trick/

performance. We tested the findings of the paper using several different batch-sizes and a learning rate of 0.001. For each batch size, we trained a fresh version of our model until it reached 97% training accuracy and then evaluated them against the dev and test sets. Aligning with the paper's results, we found that although training with larger batch sizes typically took less epochs, models trained with smaller batch sizes generalized better. We determined that a batch-size of 16 gave the best trade-off between time-to-train and dev/test set accuracy.

Using the optimal batch size of 16, we compared the effects of several different learning rates by training our model with each specific learning rate for 30 epochs and then plotting the train and dev loss. We started with 0.01 and kept reducing the rate by an order of one magnitude. From 0.01 to 0.001, it seemed that smaller learning rates would continue to provide better results. However, when we hit 0.0001, there was a large increase in the loss value. We tried 0.00001 to verify that learning rates smaller than 0.001 would have a negative impact on our model. Noticing that the best learning rate value seemed to lie somewhere between 0.001 and 0.0001, we decided to try 0.0005 and found that it gave the best results for minimizing train and dev loss. Using a learning rate of 0.0005 and a batch size of 16, our model reached a test set accuracy of 96.7%.

Batch Size	Train Accuracy	Dev Accuracy	Test Accuracy
8	97.0%	95.2%	96.3%
16	97.1%	95.1%	96.4%
32	97.0%	95.6%	95.1%
64	97.1%	93.8%	94.6%
128	97.0%	93.4%	94.8%
256	97.1%	94.4%	93.9%
512	97.1%	94.3%	93.5%

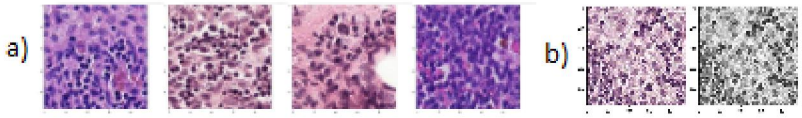
Batch Size Analysis Table



Loss plots for learning rate analysis.

5.2 Grey-scale Input

We explored the possibility of augmenting the data set by converting the training images to grey-scale. H&E staining is the most popular and commonly used histological staining procedure. The hematoxylin stains the DNA blue/violet and the eosin stains the cytoplasm pink/red. We interviewed a few more knowledgeable personnel [13] about the importance of the color in H&E stains and learned that, although color in H&E stains could provide useful information (i.e. some researchers will stain a slide more to indicate the stage of cancer), it may not be a reliable component because there is no standardized procedure for performing H&E staining. Every researcher has their own protocols which introduces variability to the degree and saturation of the stain color. This variability can be seen when sampling a few examples from our dataset since the images come from two different centers.



a) Showcases the differences between H&E stains from two different centers. b) Colored image (left) converted to grey-scale (right)

When researching the PCam dataset, we could not find any indication that the staining was used to show extra information besides the morphological features. Therefore, we considered grey-scale input images a viable approach that would not only reduce computation costs, but also possibly generalize better. Although the former holds true, the latter did not hold for our data set. When taking grey-scale images as input, the model achieved only 93.5% test set accuracy, compared to the previous color input model which achieved 96.8% test set accuracy. We conclude that this is because, although the PCam dataset does not explicitly state this, the researchers who developed the dataset do use color to show some sort of extra information in images with cancer.

5.3 SWATS

In their paper **SW**itching from **A**dam **T**o **S**GD (SWATS), Keskar, Nitish Shirish, et al. [12] describe how despite superior training outcomes, adaptive optimization methods such as Adam have been found to generalize poorly compared to Stochastic Gradient Descent (SGD). They explore a

hybrid approach where they switch their model from training on Adam to SGD when a triggering condition is satisfied. We experimented with this idea by switching to the SGD optimizer once our model met the early stopping conditions, and trained it with SGD until early stopping was invoked again. Matching their findings, our hybrid model achieved the highest project test accuracy of 97.2%.

Previous Model

N = 22,003	True Positive	True Negative
Predicted Positive	8,489	224
Predicted Negative	473	12,817

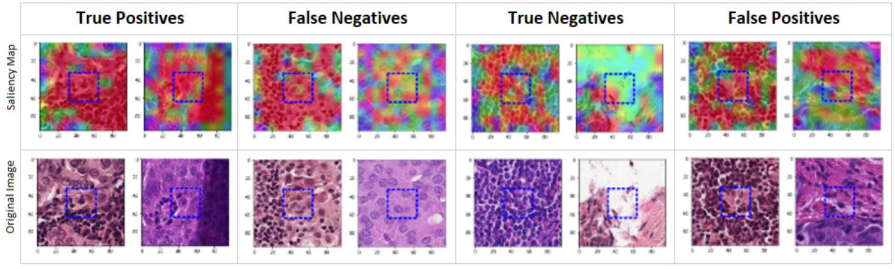
Hybrid Model

N = 22,003	True Positive	True Negative
Predicted Positive	8,580	234
Predicted Negative	382	12,807

Confusion matrices from previous model (left) and new hybrid model (right)

5.4 Error Analysis

For error analysis, we generated saliency maps from the final convolutional layer to try and understand where our model was failing. The PCam dataset specifies that a positive label means that there is at least one pixel of tumor tissue in the center 32 x 32 pixel region of the image, so we added a square patch for that region to help us with error analysis. A feature of cancer cells is that they leave their environment to migrate to other parts of the body. When they do this, they change to a rounded morphology and detach from their neighbors. Based off the true and false positive maps, it appears that our model is looking specifically for these features. On the contrary, the true and false negative maps seem to contain more cells with a flat morphology or matrices (white between cells). Unfortunately, due to time-constraints, we were unable to capitalize on these findings.



Saliency maps

6 Conclusion / Future Work

We were able to achieve a high level of accuracy for the PCam dataset using a Deep Convolutional Neural Networks. We used the concepts taught in class like - CNNs, hyper-parameter tuning, Bias-Variance trade-offs, and error analysis to build, improve and optimize the network for the given problem. We also found class's PyTorch tutorial to be a very good reference for getting started. Given more time and resources, we would have tried more methods to improve the model's accuracy such as using transfer learning from an existing pre-trained model like AlexNET or VGG-16, testing more activation functions such as Exponential Logical Unit (ELU), or using progressive re-sizing (i.e start with the 32x32 px center of each image, train for a while, then train on the original image). We would also continue focusing on understanding the saliency maps to reduce the model's error and would maybe try some of the concepts mentioned in the 'Clever Approaches' subsection of Related Work. Besides improving the model's accuracy, something we are very interested in seeing is the effect of using our pre-trained model to apply transfer-learning to other similar problems.

7 Contributions

Every team member contributed tangibly to the project, having some part in every experiment, research, and analysis.

8 Github

https://github.com/jonathan-tan27/ML_Projects/tree/master/Final

References

[1] Bray F, Ferlay J, Soerjomataram I, Siegel RL, Torre LA, and Jemal A. "Global cancer statistics 2018: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries." CA Cancer J Clin. 2018;68:394–424. doi: 10.3322/caac.21492.

[2] Campbell, Denis. “Almost Half of Cancer Patients Diagnosed Too Late.” The Guardian, Guardian News and Media, 21 Sept. 2014, www.theguardian.com/society/2014/sep/22/cancer-late-diagnosis-half-patients.

[3] Esteva, Andre, et al. "Dermatologist-level classification of skin cancer with deep neural networks." Nature 542.7639 (2017): 115

[4] Wald, N., et al. "Identification of melanoma cells and lymphocyte subpopulations in lymph node metastases by FTIR imaging histopathology." Biochimica et Biophysica Acta (BBA)-Molecular Basis of Disease 1862.2 (2016): 202-212

[5] Shen, Wei, et al. "Multi-scale convolutional neural networks for lung nodule classification." International Conference on Information Processing in Medical Imaging. Springer, Cham, 2015

[6] Steiner, David F., et al. "Impact of deep learning assistance on the histopathologic review of lymph nodes for metastatic breast cancer." The American journal of surgical pathology 42.12 (2018): 1636-1646

[7] Hu, Zilong, et al. "Deep learning for image-based cancer detection and diagnosis - A survey." Pattern Recognition 83 (2018): 134-149

[8] Shen, Wei, et al. "Multi-crop convolutional neural networks for lung nodule malignancy suspiciousness classification." Pattern Recognition 61 (2017): 663-673

[9] Xu, Tao, et al. "Multimodal deep learning for cervical dysplasia diagnosis." International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, Cham, 2016

[10] Danaee, Padideh, Reza Ghaeini, and David A. Hendrix. "A deep learning approach for cancer detection and relevant gene identification." PACIFIC SYMPOSIUM ON BIOCOMPUTING 2017. 2017

[11] Masters, D., & Luschi, C. (2018). Revisiting small batch training for deep neural networks. arXiv preprint arXiv:1804.07612.

[12] Keskar, Nitish Shirish, and Richard Socher. "Improving generalization performance by switching from adam to sgd." arXiv preprint arXiv:1712.07628 (2017)

[13] Abimbola Adike, MD, & Man Wong, Jr. Cell Biologist at University of California, Riverside