

Practicum Sprint

Lila James & Nhu Nguyen

kjames64@gatech.edu & nhu@gatech.edu

Abstract—With the advent of neural networks, expanding on existing datasets to bolster accuracy is possible. The detection of image-based medical illnesses is notoriously difficult and time-consuming: causing unnecessary patient distress. By taking from Fei Fei Lee’s key insights about data, and applying state-of-the-art techniques in data manipulation, the authors seek to build and test a data-enhancement strategy to edge neural network detection of human diseases.

1 IMPLEMENTATION AND LINKS

1.1 External Links

Below is a list of our project deliverables:

- **Presentation:**
 - Video: <https://youtu.be/1Cb6sLW3ZXI>
 - Powerpoint:
https://docs.google.com/presentation/d/1WZYAPlb7Q_PxUy6KivuS10xYvDI4sR9LveOVpuRAX4A/edit?usp=sharing
- **Sample Test Images**
 - https://gtvault-my.sharepoint.com/:f/g/personal/kjames64_gatech.edu/Et4y0wy4qKtPrb0nSViqxzsBFKn0TwDsOrgXyxWV3ou7SA?e=5mtOtF
- **Frontend**
 - URL: <https://main.dfe4ie22i5jo7.amplifyapp.com>
 - Github: <https://github.gatech.edu/qnguyen324/cancer-detection-api>
- **Backend API (main.py)**
 - URL: <http://ec2-52-90-57-19.compute-1.amazonaws.com/>
 - Github: <https://github.gatech.edu/qnguyen324/cnn-api>
- **Training Program: (app/utlis/train_model.ipynb)**
 - Github: <https://github.gatech.edu/qnguyen324/cancer-detection-ui>

- NeuralNetwork:
https://github.gatech.edu/qnguyen324/cancer-detection-api/blob/main/app/utils/neural_net.py
- ImageProcessing:
https://github.gatech.edu/qnguyen324/cancer-detection-api/blob/main/app/utils/load_images.py
- TrainingData:<https://www.kaggle.com/datasets/mohammadamireshraghi/blood-cell-cancer-all-4class>

1.2 Final Task List

Week	Task	Due Date
1	Set up API server	10/20/24
1	Deploy API server	10/24/24
1	Create image manipulation class Create Data analytics class	10/20/24
1	Research into creating a deep-wide convolutional neural network with existing frameworks	10/20/24
2	Create a working Neural Network in pytorch	10/27/24
2	Train with blood cancer dataset: 4 categories to classify	10/27/24
2	Test with blood cancer images	10/27/24
2	Add Angular frontend	10/27/24
2	Add user management	10/27/24
2	Deploy frontend application	10/27/24
3	Test Neural Network	11/04/24
3	Integrate backend server with frontend server	11/11/24
3	Optimize Docker image	11/18/24
3	Finetune the CNN	11/18/24
3	Restructure image pre-processing to include more types of noise	11/18/24
3	Experiment with different Kernal sizes, activation	11/18/24

	functions, and optimizers in the CNN	
3	Add component to upload photo	11/18/24
4	Integrate processing the photos on the backend	11/27/24
4	Try to make the current model more accurate by adding in new data and trying: zooming & different type of noise	11/27/24
4	Test in production	11/27/24
4	Add component to display result	11/27/24
5	Prepare & record video	12/1/24
5	Prepare final report	12/1/24

2 PROJECT SUMMARY AND DISCUSSION

2.1 Background

Accurate and timely disease detection is important for providing effective healthcare. Traditional methods have relied on human expertise which can be time-consuming and prone to errors (Lee et al. 2013). Missed diagnoses can have severe consequences for patients, as early intervention is often crucial for successful treatment (Cush 2021).

In recent years, advancements in machine learning have shown significant potential for improving the accuracy and efficiency of disease detection. Neural networks, a type of machine learning algorithm inspired by the human brain, have demonstrated success in various domains, including image recognition and classification. Neural networks are made of node layers. Each node is connected to the next and has a weight and threshold value. When the output is above the threshold value, the node is activated and the data is sent to the next layer to eventually produce an output (Li et al. 2014). By learning from large and complex datasets, neural networks can extract relevant features from medical images to identify abnormalities.

High-quality and annotated medical images are critical for training neural networks to produce accurate results. Dr. Fei-Fei Li's research on data augmentation can significantly improve model performance (Lin 2021). This

project aims to explore data-enhancement techniques to advance the neural network's capability to detect image-based medical illnesses based on Dr. Li's research.

2.2 Justification

Previous studies have revealed a significant rate of missed diagnoses in patients, with up to 38% of cases going undetected (Xavier et al., 2005). Timely and accurate diagnosis, especially when it comes to cancer, is crucial for providing effective treatment. Certain machine-learning models can achieve diagnostic accuracy comparable to medical experts (Liu et al., 2019). The objective of this project is to investigate various data augmentation techniques to improve diagnostic accuracy in disease detection. Having an accurate and timely diagnosis can increase the chances of a successful treatment.

2.3 Solution

We have built a full-stack web application allowing users to POST a cell image to GET an instant prognosis from a model exposed by a custom API. This medical imaging prediction is made by a deep-wide convolutional neural network, which has been pre-trained on an augmented dataset.

The dataset has randomly selected 30% of the images, in which copies are created and modified to increase spatial awareness through flipping and rotating images, combat overfitting with random noise or blur being applied to said copy, and all images have been scaled from a 64 by 64-pixel image to a 128 by 128 (Xu et al.:2022; Yamashita et. al: 2018).

The POSTed image is run through this pre-trained CNN. The Network is considered wide as the node width of the input layer is double the width relative to the original input data. We scaled the input data to the size of the input layer to take full advantage of this increase.

The network is considered deep as it is 20 layers. The idea is that the wide layers at the top learn what features are unique in an image, while the deeper layers hone in and specialize on said features. In a way, the idea is to shoehorn in as much data as possible. To handle this influx, drop-out rates increase as the model depth increases. This is a signal that we believe at deeper levels learning is

unnecessarily repeated (Nguyen et. al: 2021). As such, 50% of “what is learned” is dropped from our deepest layer compared to 20% at the first layer!

2.4 Technical Design

Below, a basic layer is shown. In the convolution method, stride denotes the number of pixels where the kernel will be, and padding is the pixel width of padding. This is more important when using kernel sizes over 3x3 due to division and the shrinking of input.

```
layers = [
    # Convolution Block 1
    nn.Conv2d(self.input_channels, 128, kernel_size=3, stride=1, padding=1),
    nn.BatchNorm2d(128),
    nn.LeakyReLU(negative_slope=0.01),
    nn.MaxPool2d(kernel_size=2, stride=2),
    nn.Dropout2d(0.2),
```

The Convolution technique used is very basic. Simply, a window of K by K size is the convolution of I, J, which is cross-multiplying the vectors, with pixels at I+M and J+N. The location of m,n is predecided by the stride, i.e., 1 pixel away.

$$Y(i, j) = \sum_{m=0}^{K-1} \sum_{n=0}^{K-1} X(i + m, j + n) \cdot W(m, n)$$

This sliding kernel creates a feature map of the image, which then has each pixel normalized in comparison with the whole kernel rather than the image. This may be useful as the normalization is relative to the kernel rather than the image. The main idea is to help highlight and allow for the preservation of unique features. Mathematically speaking, R represents all pixels in the kernel: R = KxK.

$$Y(i, j) = \frac{1}{|\mathcal{R}|} \sum_{(m,n) \in \mathcal{R}} X(i \cdot s + m, j \cdot s + n)$$

Once these kernels are normalized, the greatest feature in each kernel is extracted by a simple max function operating with the same sliding kernel logic.

$$Y(i, j) = \max_{(m, n) \in \mathcal{R}} X(i \cdot s + m, j \cdot s + n)$$

From here we use a leaky Rectified Linear Unit. The idea is to introduce non-linear learning and avoid dying neurons. This is done by introducing the -0.1 alpha for comparison, see leakyrelu function -0.1 slope. The negative alpha considers negative input while learning, which helps keep neurons alive via more connections, and non-linearity is introduced by literally bending the vectors as data points approach 0 on the X-axis.

The image matrix is then flattened into a 1D vector and ran through a simple linear regression, where $y=mxT+b$! Lastly, the vector is then reduced in size by adaptive pooling, averaging pixel values in the KxK sliding window, and is passed through one last linear regression to determine the cancer image category

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ -0.1x & \text{if } x \leq 0. \end{cases}$$

CNN: Full Overview

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 128, 64, 64]	3,584
BatchNorm2d-2	[-1, 128, 64, 64]	256
LeakyReLU-3	[-1, 128, 64, 64]	0
MaxPool2d-4	[-1, 128, 32, 32]	0
Dropout2d-5	[-1, 128, 32, 32]	0
Conv2d-6	[-1, 128, 32, 32]	147,584
BatchNorm2d-7	[-1, 128, 32, 32]	256
LeakyReLU-8	[-1, 128, 32, 32]	0
MaxPool2d-9	[-1, 128, 16, 16]	0
Dropout2d-10	[-1, 128, 16, 16]	0
Conv2d-11	[-1, 256, 16, 16]	295,168
BatchNorm2d-12	[-1, 256, 16, 16]	512
LeakyReLU-13	[-1, 256, 16, 16]	0
MaxPool2d-14	[-1, 256, 8, 8]	0
Dropout2d-15	[-1, 256, 8, 8]	0
AdaptiveAvgPool2d-16	[-1, 256, 1, 1]	0
Flatten-17	[-1, 256]	0
Linear-18	[-1, 128]	32,896
LeakyReLU-19	[-1, 128]	0
Dropout-20	[-1, 128]	0
Linear-21	[-1, 4]	516
...		
Forward/backward pass size (MB): 19.26		
Params size (MB): 1.83		
Estimated Total Size (MB): 21.14		

Discussion & Future Work

The future of this project is promising. Two main avenues for development are:

- 1.) Diversify activation functions to bolster more complex learning and to protect from overfitting a model.
- 2.) Expose users to the automated train model script to enable the proliferation of data and shared models.

The first point is quite simple. Different activation functions spread throughout the CNN may prove to be useful in learning more diverse traits of a cancer cell. Currently, the model is geared towards finding the main features of a cancer cell, which may not be optimal. Setting up an experiment for this would be quite simple but demands the key to point two.

A script has been created to automate the process of augmenting data and training a new CNN on a unique dataset or to continue training on an existing model. See the `loda_model` and `train_model` python notebooks for reference. Exposing this functionality requires hardware that can handle training and image augmentation- both are computationally expensive tasks.

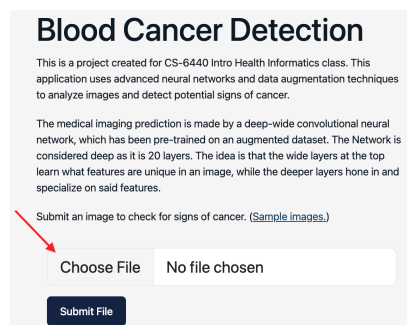
The most logical next step is to remove the blocker for points one and two. This can be done by migrating the `train_program` Docker image to a more dedicated server with idealistically access to GPUs. The current script does check if GPUs are available for use, and defaults to CPU if not. See the neural network Python file.

3. TECHNICAL DOCUMENTATION

3.1 User Manual

For this project, our main focus is the data scientist portion. We want to build a user friendly application to allow users to upload an image and see the result. Thus, the main application only has one screen to allow users to upload and display the result. Here are the steps to get the result:

1. Go to <https://main.dfe4ie22i5jo7.amplifyapp.com>
2. Create an account
 - a. Verify your account
3. In the main window, select an image to upload. ([Sample images](#))



The screenshot shows a web application titled "Blood Cancer Detection". It contains a paragraph of introductory text about the project's purpose and the underlying deep learning model. Below the text, there is a red arrow pointing to a "Choose File" button. To the right of this button is a text field that currently says "No file chosen". Below these elements is a dark blue "Submit File" button.

Blood Cancer Detection

This is a project created for CS-6440 Intro Health Informatics class. This application uses advanced neural networks and data augmentation techniques to analyze images and detect potential signs of cancer.

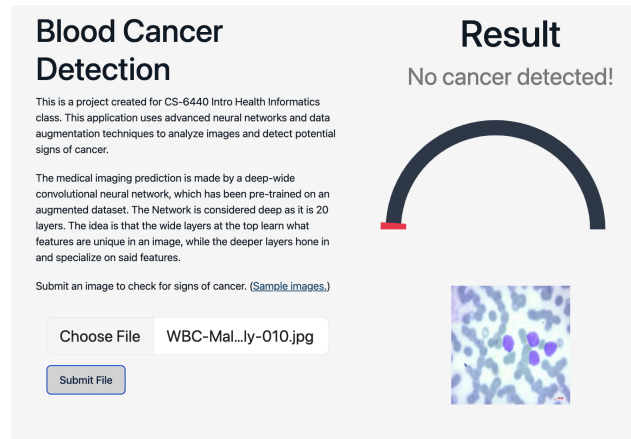
The medical imaging prediction is made by a deep-wide convolutional neural network, which has been pre-trained on an augmented dataset. The Network is considered deep as it is 20 layers. The idea is that the wide layers at the top learn what features are unique in an image, while the deeper layers hone in and specialize on said features.

Submit an image to check for signs of cancer. ([Sample images](#))

Choose File No file chosen

Submit File

4. Select “Submit file”
5. Notice the result is shown on the right side



3.2 Deployment

For this application, we created a frontend and backend server. Here are the instructions to deploy both:

3.2.1 Backend

1. Download <https://github.com/qnguyen324/cancer-detection-ui> the project
2. With Docker
 - a. Navigate to the project
 - b. Run “docker compose up” in the command line
 - c. Note you may need 20-30GB
3. Without Docker (need Python 3.12)
 - a. Create an venv (python3 -m venv blood-cancer-api)
 - b. Activate (source blood-cancer-api/bin/activate)
 - c. Specify the main app (export FLASK_APP=main.py)
 - d. Start the application (flask run --host=0.0.0.0 --port=5001)

3.2.2. Frontend

1. Download <https://github.com/qnguyen324/cancer-detection-api> the project
2. I use [AWS Amplify](#) for authentication, but it can be removed

10 REFERENCES

1. In Proceedings of the 19th International Conference on Artificial Intelligence in Education. London, United Kingdom. Springer.
2. Lee, C. S., Nagy, P. G., Weaver, S. J., & Newman-Toker, D. E. (2013). Cognitive and system factors contributing to diagnostic errors in radiology. *American Journal of Roentgenology*, 201(3), 611-617.
3. Cush, J. J. (2021). Rheumatoid Arthritis: Early Diagnosis and Treatment. *The Medical Clinics of North America*, 105(2), 355-365.
4. Li, Q., Cai, W., Wang, X., Zhou, Y., Feng, D. D., & Chen, M. (2014, December). Medical image classification with convolutional neural network. In 2014 13th international conference on control automation robotics & vision (ICARCV) (pp. 844-848). IEEE.
5. Lin, C. H., Lin, C. S., Chou, P. Y., & Hsu, C. C. (2021). An efficient data augmentation network for out-of-distribution image detection. *IEEE Access*, 9, 35313-35323.
6. Xavier, A. C. G., Siqueira, S. A. C., Costa, L. J. M., Mauad, T., & Nascimento Saldiva, P. H. (2005). Missed diagnosis in hematological patients—an autopsy study. *Virchows Archiv*, 446, 225-231.
7. Liu, X., Faes, L., Kale, A. U., Wagner, S. K., Fu, D. J., Bruynseels, A., ... & Denniston, A. K. (2019). A comparison of deep learning performance against health-care professionals in detecting diseases from medical imaging: a systematic review and meta-analysis. *The lancet digital health*, 1(6), e271-e297
8. Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: An overview and application in radiology. *Insights into Imaging*, 9(4), 611–629. <https://doi.org/10.1007/s13244-018-0639-9>
9. Xu, M., Yoon, S., Fuentes, A., & Park, D. S. (2022). A comprehensive survey of image augmentation techniques for deep learning. *arXiv*. <https://doi.org/10.48550/arXiv.2205.01656>

10. Nguyen, T., Raghu, M., & Kornblith, S. (2021). Do Wide and Deep Networks Learn the Same Things? Uncovering How Neural Network Representations Vary with Width and Depth. arXiv preprint arXiv:2010.15327. Retrieved from [arXiv:2010.15327](https://arxiv.org/abs/2010.15327).