Big Brain Energy: Brain Tumor Segmentation and Prediction

Gia-Khang Ernest Nguyen
Jeffrey Lam Yuk Tseung
Inés Gonzalez Pepe

Presented to Professor Andrew Delong

COMP432
Section DD

Date due: December 7th, 2020

Abstract

Machine learning will soon have an enourmous impact in the health sector. Our team is exploring ML's potential in identifying the presence of tumours given an MRI brain scan. In doing so, our team was able to compare the performance of convolutional neural networks and logistic regression when applied to tumour identification based on performance metrics and computational complexity of each model. Through our continuous, methodical experimentation consisting of training and improvement, we were able to meet our goal and determine that while in the short run the Logistic Regression model is better, the Convolutional Neural Network has more room to improve in the long run.

1 Introduction

The objective is to design the optimal tool that will help radiologists find tumours in their patients' MRI brain scans more easily, thus allowing them to dedicate their time to more complex tasks. We will be comparing the performances and exactness of a ConvNet model vs a Logistic Regression model as well as the practicalities associated with creating and running the aforementioned models in terms of time, computational resources and costs. To this end, we are basing our project off the Brain Tumor Detection Using Convolutional Neural Network paper by Hossain et al [2]. We aim to build the best version of each model that we can in order to compare their features, given the practical constraints of the situation. [1]

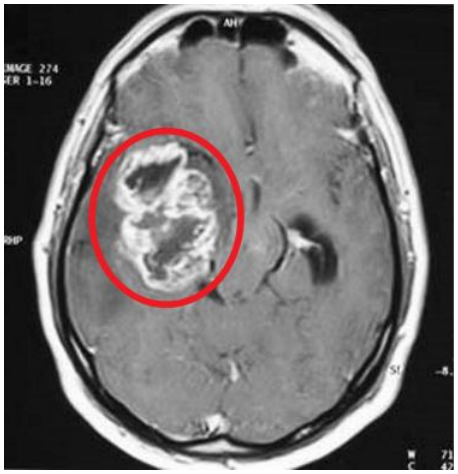

Figure 1: Desired Model Prediction: 1

2 Methodology

In terms of input data[2], our first dataset was the Brain MRI Images for Brain Tumor Detection found on Kaggle that was used in the BRaTS (Brain Tumour Segmentation) challenge [1]. This dataset was composed of png, jpg and jpeg images of MRI scans mostly in a top-down view which have been classified into 2 distinct files; "yes" and "no", (the names of which refer to whether or not the brain scan contains a tumour). Our second dataset Brain Tumor Classification is also from Kaggle and contains many similar images. The main difference between the datasets is that in the second dataset, the scans are categorized by presence and type of tumour (glioma tumours, meningioma tumours, pituitary tumours and no tumours) [3].

In order to have a dataset that fit into our project's scope, we extracted

[1] We had limited experience, time and resources to develop a full scale project.

[2] As seen in Figure 1, 1 is equal to True aka a tumour is present, 0 is equal to false there is no tumour.

data from the glioma, meningioma and no-tumour categories in dataset 2 and put them in copies of the "yes" and "no" categories of the first dataset. We ultimately decided to omit the pituitary category because too many tumorous MRI scans biased the model and pituitary tumours are best identified from a portrait view MRI instead of the common top down view as in most of our images.

In terms of size, with both initial datasets combined, we had an overall dataset of approximately 800 training samples and 200 testing samples. This dataset had already been sorted into its categories, but we filtered the data by removing duplicate images and MRI scans that were not in the top-down view. Therefore, our final dataset containing 2 folders (*yes and no*) had the following ratio:

Yes:	514	52%
No:	474	48%
Total:	988	100%

Afterwards, the images were preprocessed and uniformly converted to grayscale, resized to the same (128,128) dimensions using nearest-neighbor interpolation and shuffled for good measure. This was done to decrease computational cost and any variation among data that might confuse the model. Background cropping (using image contour identification) was also implemented for both as it improved performance.

Additional preprocessing such as canny edge detection was considered and was implemented on the logistic regression model as we found it increased the final results, likewise for the ConvNet model. We provide the performance metrics of the both models, with and without the image enhancement in the Appendix. Afterwards the data was split as seen in the figure below:

training:	80%
validation:	10%
testing:	10%

Using Tensorflow's `tf.keras.Model` object to build the model, the subsequent layers were created (consult Figure 3 & 4 in Appendix for overview of model architecture):

1. A **Zero Padding** layer with a pool size of (2, 2). It was included to control the shrinkage of dimensions after applying filters and to avoid losing information at the boundaries.

2. A **Convolutional** layer with 32 filters of size (7,7) and a stride of 1.

3. A **Batch Normalization** layer to normalize pixel values and ultimately speed up computation time.

4. A **ReLU** activation layer

5. Two **Max Pooling** layers with f = 4 and s = 4 in order to lower the computation cost.

6. A **Flatten** layer to transform the 3-dimensional matrix into a one-dimensional vector.

7. A **Dense** fully connected layer with one neuron and a sigmoid activation that will return our final output unit of 0 or 1.

The convolutional network was trained using an Adam optimizer which combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can

handle sparse gradients on noisy problems and is relatively easy to configure. It also uses the Binary Cross-Entropy loss function which is particularly useful when performing binary classification. We tested our model with epochs ranging from 10 to 20 and batch sizes of 30 to 60 and determined the final values of 10 and 20 for batch size and epochs respectively through observation and comparison of the validation f1-score at the last epoch.

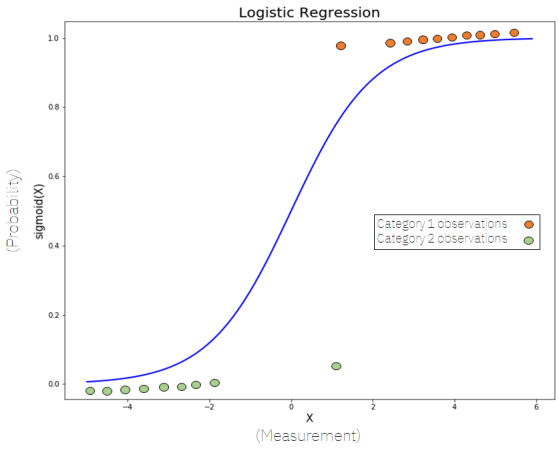

Figure 2: Logistic Regression Function [4]

Scikit-learn was used to build the Logistic Regression model as well as GridsearchCV to automate the search for the optimal parameters for the model. The final values selected were elasticnet for penalty as it combines the L1 and L2 penalty and will minimize both the size of all coefficients as well as the absolute coefficient values. An L1 ratio of 0.5 was also selected to accompany the elasticnet penalty and the warm start was set to true to prevent weight re-instantiation and cut down computational time. The saga solver was used because it works fast on large datasets with as many features as samples and works well with the selected penalty.

Once the model was constructed, the maximum number of iterations of 1500 was manually selected after testing and comparison showed it was where performance improvement started to conclusively plateau.

3 Analysis & Results

Table 1: ConvNet Metrics

	Training	Validation	Testing
Accuracy	0.995	0.879	0.84
Precision	0.995	0.875	0.84
Recall	0.995	0.907	0.84
F1	0.995	0.891	0.84

After constructing both models to our satisfaction, we started the comparison. As stated earlier, we experimented with different mini-batch sizes and number of epochs. Generally, we noticed that the f1-score (for validation especially), tends to be larger as we decrease the mini-batch size and increase the number of epochs. With image cropping and edge detection activated, our best-performing model was able to reach a validation f1-score of 89.09% and a testing f1-score of 84%, with training metrics nearly reaching 100% (see Table 1). It is important to note that these were stabilized using random seeding for Python, Numpy, sci-kit learn and TensorFlow's random generator (all set at 0), and that changing any of those random seeds could influence the computed metrics or even change the hyper-parameters that maximizes the model's validation f1-score. However, the result's stability will only be guaranteed per machine, since some metrics may not have the same values

across all machines.

Table 2: Logistic Regression Metrics

	Training	Validation	Testing
Accuracy	1	0.81	0.90
Precision	1	0.78	0.86
Recall	1	0.88	0.96
F1	1	0.82	0.91

In contrast, the logistic regression model was simpler to create as well as to run and train once the parameters were selected through GridSearchCV. When we looked at the performance metrics with edge detection and background cropping activating, we saw that it achieved high scores of 100% training accuracy with 81% and 90% accuracy for the validation and test data. Further examining other metrics of performance like precision, recall and F1 measure in the table above, we saw similarly high rates of performance, thus relieving our fears of over-fitting and demonstrating the confidence of the classifier's decisions.

On a side note, it was interesting that while edge detection and background cropping did not significantly impact the overall results, it appeared to lower the logistic regression performance with the validation data and improve it with the test data even though the test data was not introduced until after the model had been fully tuned. This is odd given that validation data is often used as an estimate of what the test results will be; a possible explanation for is hyper-parameter over-fitting.

For further details on the performance of each model, please consult the Appendix performance metrics for each model and the GitHub Output folder.

As can be observed, the logistic regression model performs best at identifying tumours as its performance metrics outperform that of the ConvNet except when observing the validation dataset. However, it has a consistently longer run-time than the ConvNet when comparing similar epochs/iterations, despite having the simpler architecture. Moreover, the ConvNet has more potential as it can be adapted to multi-class identification (such as identifying the type of tumour as well as its presence), while logistic regression is better suited to binary classification. While one can implement the one vs all methodology to adapt the logistic regression model to multi-class identification, this would require additional computations that would make the model lose the advantage of being "simple" to implement in comparison to the ConvNet and make its run-time significantly longer.

In brief, the logistic regression model was the best performer when identifying the presence of tumours, but the ConvNet is likely better when it comes to identifying and classifying tumours in a more computationally efficient manner.

4 Conclusion

To summarize, at the start of the semester we had set out to build a ConvNet tool that identified and classified tumours. We explored options involving a pre-trained ResNet50, however this original objective did not pan out. Therefore, we adapted our objective to compare the performance according to metrics and computational resources of

the logistic regression model vs the ConvNet model.

Based on our experiments regarding both models, we were able to conclude that the logistic regression model performed slightly better when it came to identifying our MRI dataset from a purely quantitative perspective. The ConvNet model, while more complicated to set up, was faster and more suited to being expanded upon to not only identify tumours but to classify them into categories. Therefore, should we expand on this project, our updated objectives would be to add in capabilities to classify tumours and to boost accuracy metrics in order to have more confidence in the ConvNet's predictions [3].

[3]In order to be a useful application that assists medical professionals, we would have to have extremely high certainties about our predictions, which we cannot as of yet guarantee.

References

[1] Navoneel Chakrabarty. *Brain MRI Images for Brain Tumor Detection: Version 1*. URL: https://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor-detection. (accessed: 12.02.2020).

[2] T. Hossain et al. "Brain Tumor Detection Using Convolutional Neural Network". In: *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*. 2019, pp. 1–6. DOI: 10.1109/ICASERT.2019.8934561.

[3] Sartaj. *Brain Tumor Classification (MRI): Classify MRI images into four classes: Version 2*. URL: https://www.kaggle.com/sartajbhuvaji/brain-tumor-classification-mri. (accessed: 12.02.2020).

[4] Jaime Zornoza. *Logistic Regression Explained*. URL: https://towardsdatascience.com/logistic-regression-explained-9ee73cede081. (accessed: 12.06.2020).

Appendix

```
Model: "BrainDetectionModel"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 128, 128, 1)]     0
_________________________________________________________________
zero_padding2d_1 (ZeroPaddin (None, 132, 132, 1)       0
_________________________________________________________________
conv0 (Conv2D)               (None, 126, 126, 32)      1600
_________________________________________________________________
bn0 (BatchNormalization)     (None, 126, 126, 32)      128
_________________________________________________________________
activation_1 (Activation)    (None, 126, 126, 32)      0
_________________________________________________________________
max_pool0 (MaxPooling2D)     (None, 31, 31, 32)        0
_________________________________________________________________
max_pool1 (MaxPooling2D)     (None, 7, 7, 32)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1568)              0
_________________________________________________________________
fc (Dense)                   (None, 1)                 1569
=================================================================
Total params: 3,297
Trainable params: 3,233
Non-trainable params: 64
```

Figure 3: Model Summary

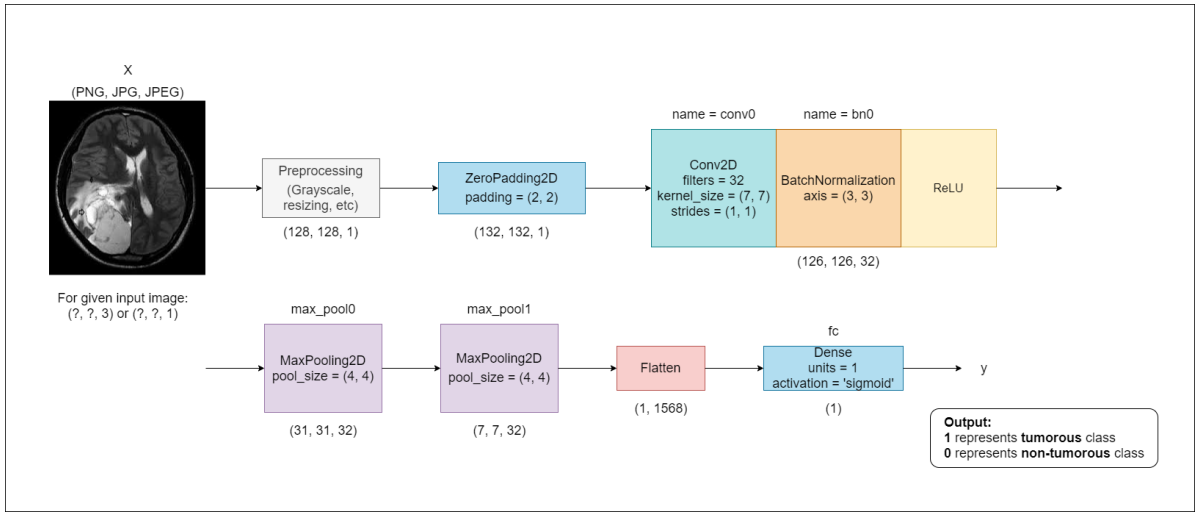

Figure 4: Convolutional Neural Network Architecture

Unfiltered vs Enhanced Logistic Regression Performance Graphed Metrics

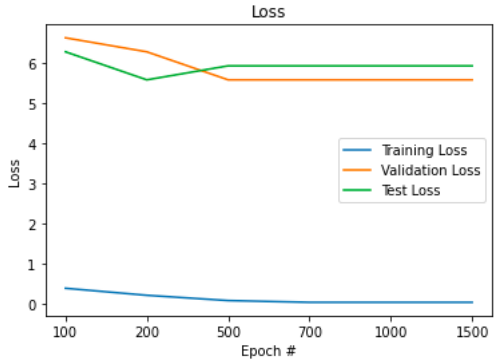

Figure 5: Unfiltered Logistic Regression Loss Function

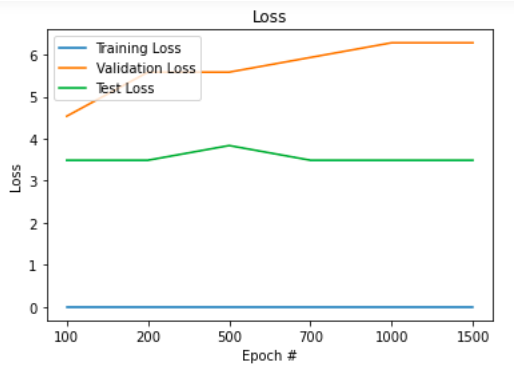

Figure 7: Enhanced Logistic Regression Loss Function

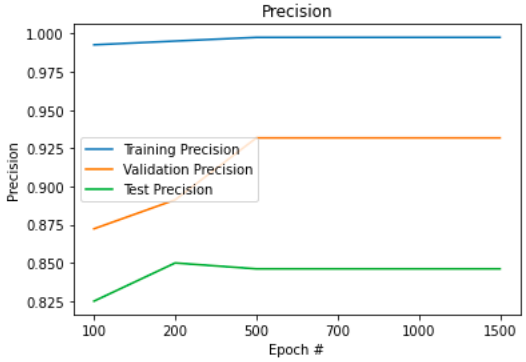

Figure 6: Unfiltered Logistic Regression Precision

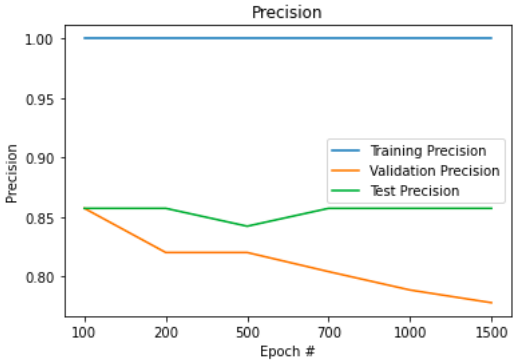

Figure 8: Enhanced Logistic Regression Precision

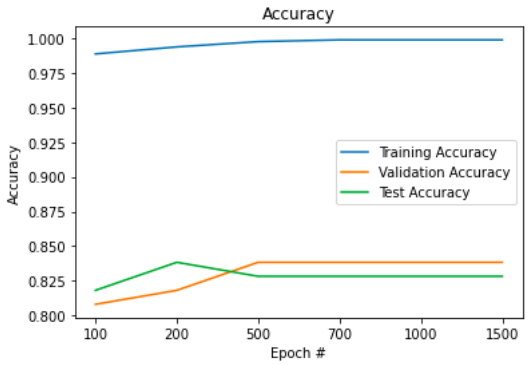

Figure 9: Unfiltered Logistic Regression Accuracy

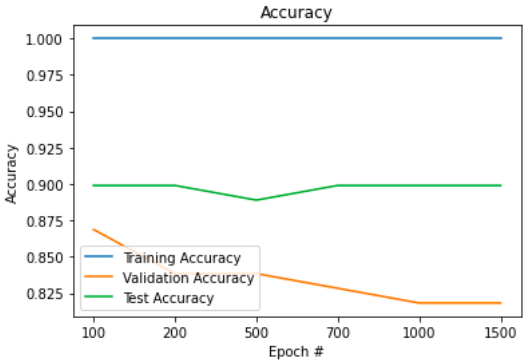

Figure 12: Enhanced Logistic Regression Accuracy

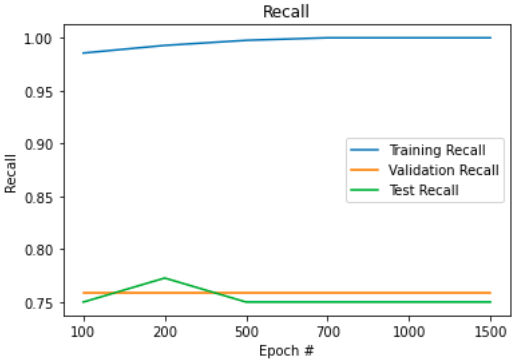

Figure 10: Unfiltered Logistic Regression Recall

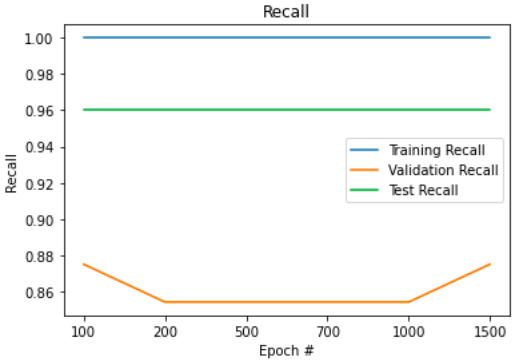

Figure 13: Enhanced Logistic Regression Recall

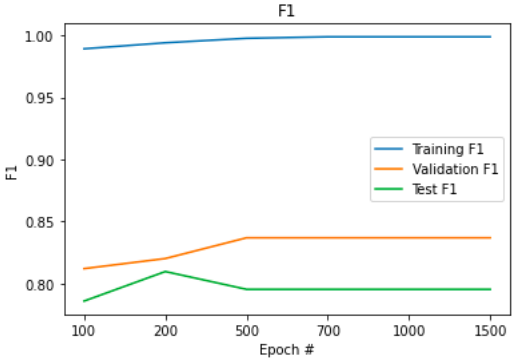

Figure 11: Unfiltered Logistic Regression F1 Measure

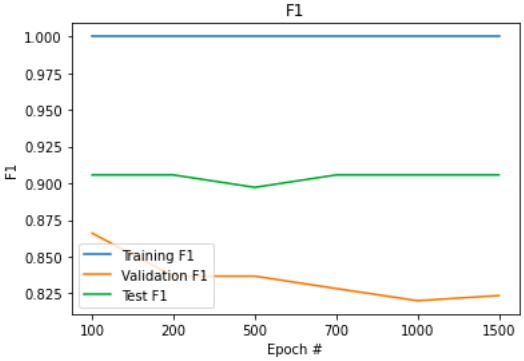

Figure 14: Enhanced Logistic Regression F1 Measure

Logistic Regression Performance Numerical Metrics

Table 3: Enhanced Logistic Regression Metrics

	Training	Validation	Testing
Loss	0.99×10^{-15}	4.88	6.28
Accuracy	1	0.86	0.82
Precision	1	0.89	0.81
Recall	1	0.85	0.77
F1	1	0.87	0.79

Table 4: Unfiltered Logistic Regression Metrics

	Training	Validation	Testing
Loss	0.044	6.63	5.93
Accuracy	1	0.84	0.83
Precision	1	0.93	0.85
Recall	1	0.76	0.75
F1	1	0.84	0.80

Unfiltered vs Enhanced ConvNet Performance Graphed Metrics

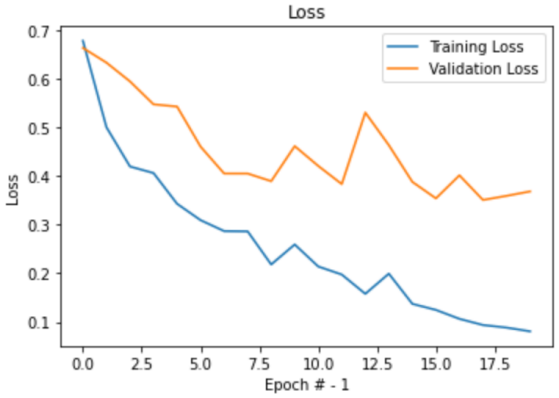

Figure 15: Unfiltered ConvNet Loss Function

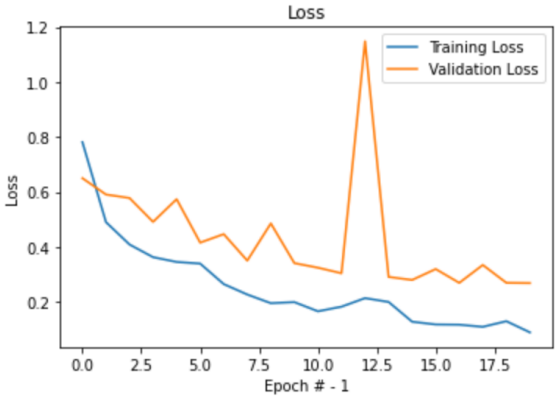

Figure 17: Enhanced ConvNet Loss Function

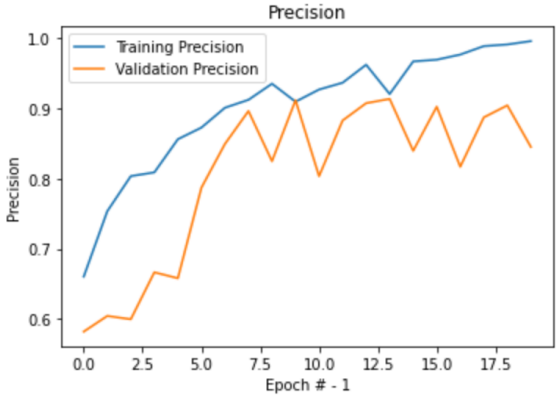

Figure 16: Unfiltered ConvNet Precision

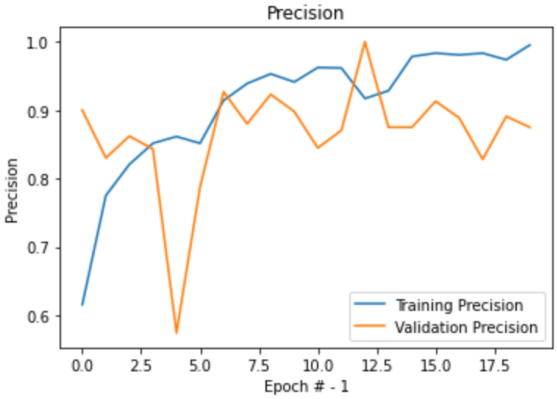

Figure 18: Enhanced ConvNet Precision

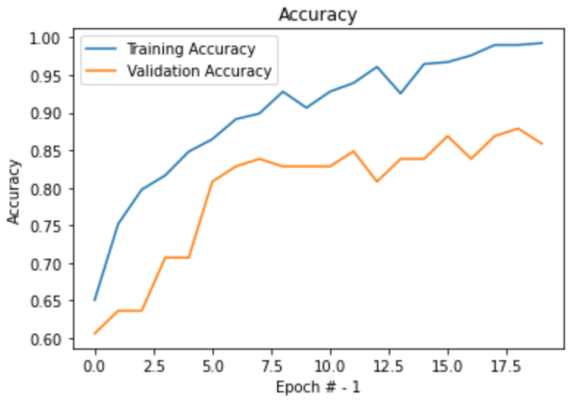

Figure 19: Unfiltered ConvNet Accuracy

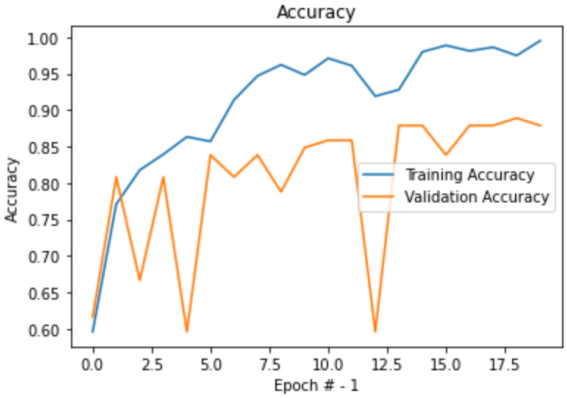

Figure 22: Enhanced ConvNet Accuracy

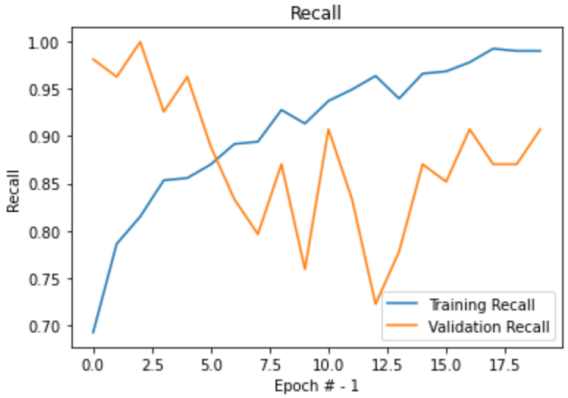

Figure 20: Unfiltered ConvNet Recall

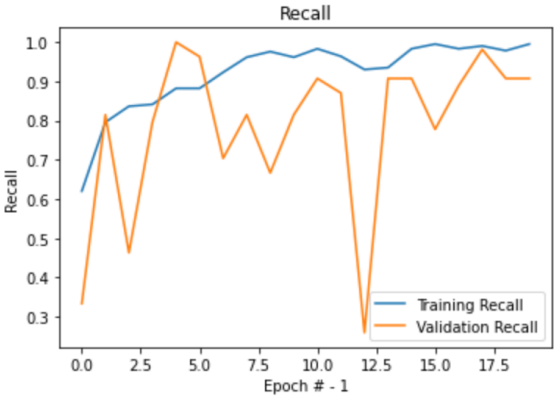

Figure 23: Enhanced ConvNet Recall

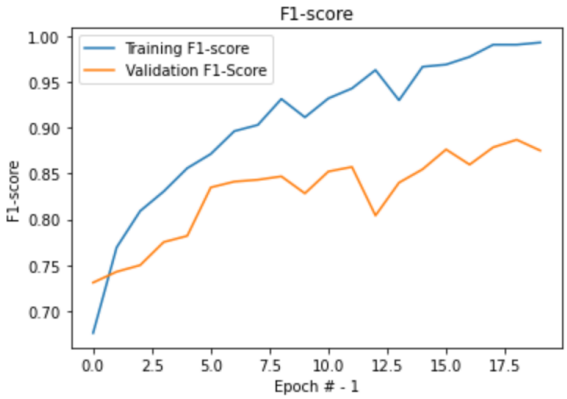

Figure 21: Unfiltered ConvNet F1 Measure

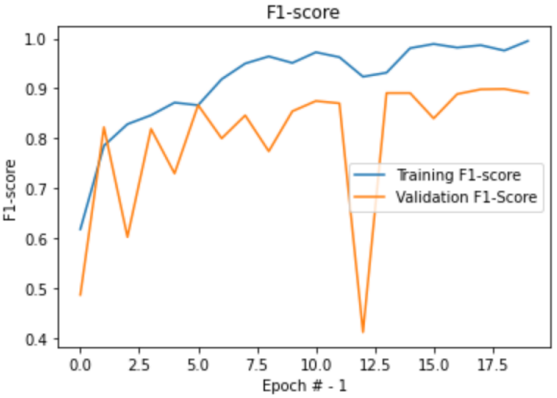

Figure 24: Enhanced ConvNet F1 Measure

ConvNet Test Metrics

	precision	recall	f1-score	support
no	0.85	0.85	0.85	55
yes	0.82	0.82	0.82	44
accuracy			0.84	99
macro avg	0.84	0.84	0.84	99
weighted avg	0.84	0.84	0.84	99

Figure 25: Enhanced ConvNet Test Metrics

	precision	recall	f1-score	support
no	0.89	0.87	0.88	55
yes	0.84	0.86	0.85	44
accuracy			0.87	99
macro avg	0.87	0.87	0.87	99
weighted avg	0.87	0.87	0.87	99

Figure 26: Unfiltered ConvNet Test Metrics