# Deep Learning in Brain Tumor Classification - A Comparative Analysis

**Jiaheng Cui**[1] **Abdulaziz Suria**[1] **Danya Gordin**[1]

[1]Khoury College of Computer Sciences
Northeastern University
cui.jiah@northeastern.edu, suria.a@northeastern.edu, d.gordin@northeastern.edu

## Abstract

It's important to diagnose brain tumors early and accurately from MRI images. Recent research has demonstrated that deep learning methods, especially Convolutional Neural Networks (CNNs), are fast and accurate at diagnosing brain tumors. In this project, we attempted to discover different ways to train an image classification model. We discovered the aspects of getting more data and also developing complex model architectures. We started from a baseline model which is trained on a single dataset and then a broader combined dataset. We derive conclusions from this baseline model and further created better approaches such as proper data augmentation and transfer learning to deal with the problems and finally prepared a framework that detects the locations of tumors that worked really well on the dataset.

## Introduction

Brain tumors comprise 2% of all cancers, but are disproportionately responsible for cancer-related deaths. Glioblastoma is the most common form of malignant brain tumor with a 5-year survival rate of only 4.7% while the overall 5-year survival rate for any brain tumor (benign or malignant) is 34.4% (Neugut et al. 2019). In fact, cancer in the central nervous system is the 10th leading cause of death for men and women in the United States (Rouse et al. 2015). Thus, early and accurate detection of brain tumors is imperative for maximizing patient survivability.

The most common medical imaging technique for diagnosing a brain tumor is magnetic resonance imaging (MRI). Studies have shown that accuracy of physicians using conventional MRI for diagnosing intracranial tumors can vary wildly depending on the tumor type (Yan et al. 2016). In order to assist radiologists and neuropathologists with diagnostic accuracy, there has been a flurry of research into automated tumor identification and classification (Díaz-Pernas et al. 2021).

Recent research has demonstrated that deep learning methods, especially Convolutional Neural Networks (CNNs) are in some cases faster and more accurate at diagnosing brain tumors than physicians (Hollon et al. 2020). Our goal is to further contribute to this growing body of work by developing and testing a few different CNNs to classify the presence or absence of a brain tumor in MRI images.

The primary goal of our project is to optimize the accuracy of brain MRI image classification using a variety of modifications to the data and models. We built a CNN model that would take brain MRI images as input and output if a patient has a brain tumor or not. We then improved the original model by using augmented data and experimented on the factors which affected augmentation results. Finally, we compared the accuracy of this model built from scratch to a pre-trained EfficientNet B4 model (Tan and Le 2019) on the ImageNet dataset (Deng et al. 2009) and built a detection framework that can report the location of the tumors based on YOLOV5 (Jocher et al. 2020).

## Background

We used Python as the main programming language and implemented all the models with Pytorch and Keras.

### CNN

The bulk of our work involves building and fine-turning a CNN model from scratch. As shown in Figure 1, a typical classification CNN consists of convolutional layers and activation functions for feature extraction, fully connected layers for classification, and a softmax layer for output. We followed this architecture and explored how many layers each part should have, and the specific parameters of each layer. As we should gradually move from simple to complex, we decided to start with a simple architecture with minimal layers and hyperparameters.

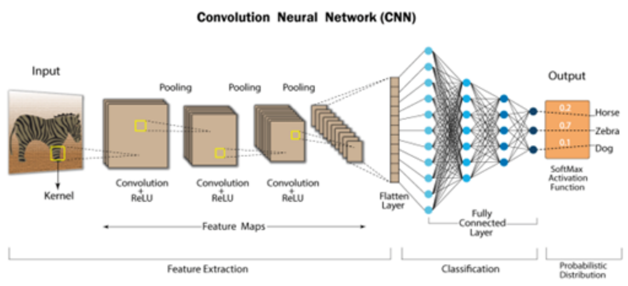


Figure 1: Typical architecture of a classification CNN

## Transfer Learning

Transfer learning refers to the scenario where the learning in one setting is exploited to improve generalization in another context. This technique is particularly powerful in situations where data volume is low or when there exists a domain overlap between the two settings by leveraging another model that has previously been trained on massive amounts of related data (Figure 2). More specifically, transfer learning works by using the new data to retrain the latter layers of a network. The early and middle layers transferred from the original network often describe more general image features such as edge detection and shape and thus don't really require much retraining.

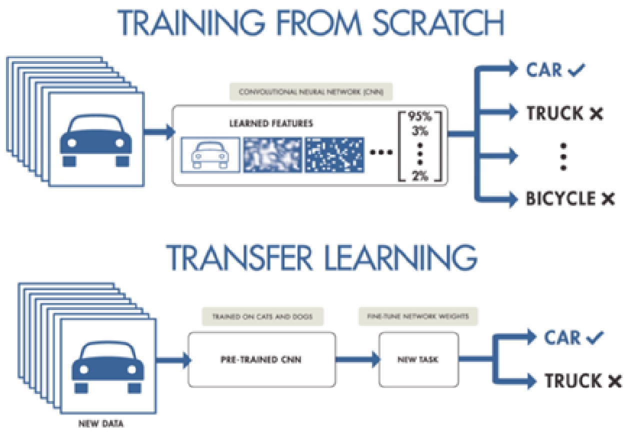


Figure 2: Learning From Scratch vs Transfer Learning

Our project harnesses transfer learning by retraining two powerful deep learning architectures to greatly increase performance metrics: EfficientNetB4 and YOLOv5.

**Brief Introduction to EfficientNet** The EfficientNet family of neural nets have received a flurry of interest recently due to their cutting-edge accuracy while using an order-of-magnitude less parameters than their competition. This is mostly in part to their innovative scaling method that uses a compound coefficient to uniformly scale the dimensions of depth, width, and resolution as seen in Figure 3. This methodology lays in stark contrast to the more conventional random scaling approach which often yields small performance boosts. The logic behind the compound scaling approach is based on the idea that larger inputs will correspondingly need more layers in order to increase the receptive field and capture finer patterns.

**Brief Introduction to YOLO** You Only Look Once (YOLO) is a state-of-the-art real-time object detection system which departs from the traditional two-stage protocol that most other object detection algorithms follow by using an end-to-end fully convolutional neural net that makes predictions of bounding boxes and class probabilities all at once (Figure 4).

YOLO essentially operates by dividing an image into a grid system, and each grid cell detects objects within itself.

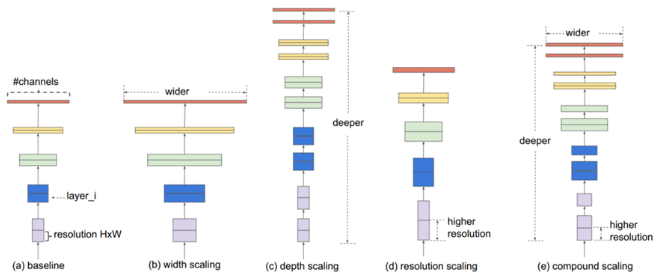


Figure 3: Scaling Methods in Neural Nets, EfficientNet uses (e) as it combines methods from (b), (c) and (d)

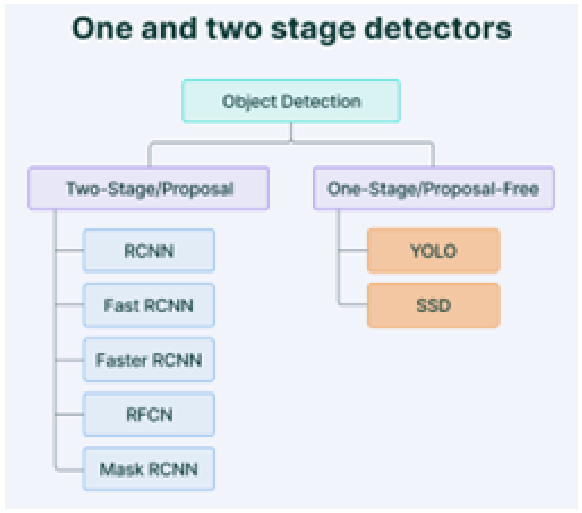


Figure 4: Popular two-stage and single-stage objective detection algorithms.

More technically, it applies $1 \times 1$ detection kernels on feature maps of three different sizes. The resultant feature map (grid cells) has detection attributes that include bounding box coordinates, object scores, and class scores (Figure 5).

Since both detection and recognition occur in each grid cell, multiple cells can predict the same object just with different bounding boxes. In order to identify the best bounding box for each object, YOLO applies a non-maximal suppression (Figure (**?**)) approach that suppresses any bounding boxes that have a large IOU (Intersection Over Union) with the highest probability bounding box.

## Metrics

We used a variety of metrics and visualization techniques in order to evaluate our models including Loss vs Epoch plots, F1 score, confusion matrix and Precision-Recall curves (Tharwat 2020). In this particular project, false negatives are extremely undesirable because misdiagnosing a person with a brain tumor as "normal" may prove fatal. Therefore, the most important metrics to maximize is F1 score.

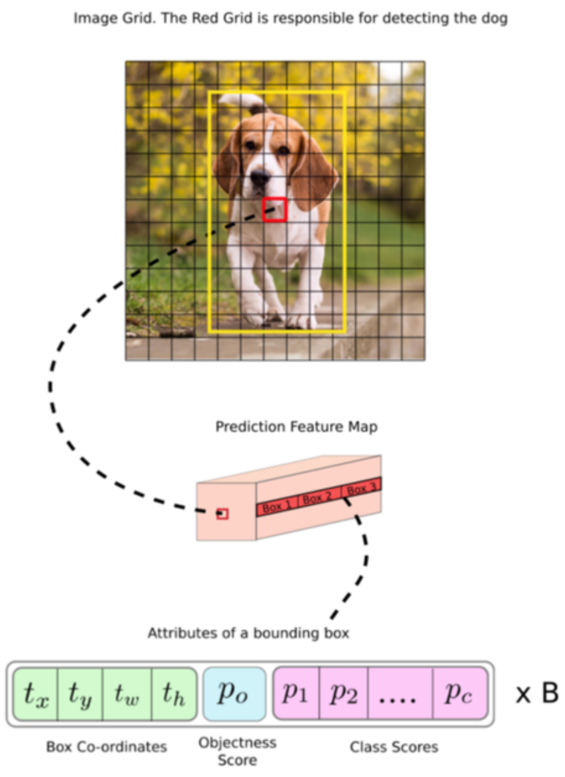


Image Grid. The Red Grid is responsible for detecting the dog
Prediction Feature Map
Box 1
Box 2
Box 3
Attributes of a bounding box
$t_x$ $t_y$ $t_w$ $t_h$ $p_o$ $p_1$ $p_2$ .... $p_c$ x B
Box Co-ordinates
Objectness Score
Class Scores

Figure 5: YOLO illustrated example

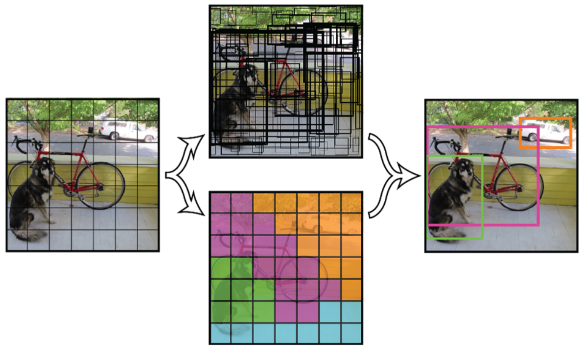


Figure 6: YOLO non-maximal suppression illustration

## Data

We used 4 different datasets of MRI brain images to train and test our models. In total, all 4 datasets combined contain approximately 10000 images. The basic information of our datasets are listed in Table 1. Sample images are shown in Figure 7.

Furthermore, we aim to also create a much larger set of augmented images to solve the "data insufficiency problem" associated with building models from scratch. Augmented data is created by performing various operations on the original data. Please see the details in the Approach section.

Finally, the transfer learning approach uses EfficientNetB4 which is pretrained on the well-known ImageNet dataset, and YOLOV5 which is pretrained on the COCO dataset (Lin et al. 2014).

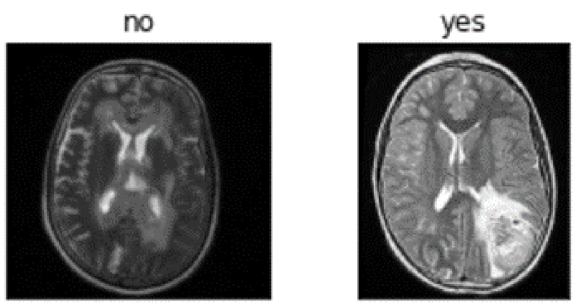


Figure 7: Sample images from the negative and positive group

## Related Work

The development of machine learning and artificial intelligence greatly impacts the medical imaging field. Brain tumor classification is now typically done by deep learning-based algorithms that allow for automatic detection/diagnosis without the need for extensive medical experience. Hence, these techniques have drawn the interest of scientists and researchers and have proven incredibly useful.

CNN-based models have been successfully applied to brain tumor classification problems (Charron et al. 2018). A CNN architecture was able to extract features from brain MRI in the work by (Pashaei, Sajedi, and Jazayeri 2018), but suggested limitations in the discrimination capability. A modified architecture called CapsNet was later proposed by (Afshar, Plataniotis, and Mohammadi 2019). Still, the improvement in performance was not sufficient.

Data augmentation in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It typically acts as a regularizer and helps reduce overfitting when training a machine learning model. Typical augmentation techniques for image data include rotation, mirroring, random cropping, skewing, gray-scale, etc (Gron 2017). Another solution is the sourcing of entirely new, synthetic images through various techniques, for example, the use of generative adversarial networks (GANs) (Goodfellow et al. 2020) to create new synthetic data for augmentation (Shorten and Khoshgoftaar 2019). However, this synthetic approach usually takes place particularly for biological data, which tend to be high dimensional and scarce. Image recognition algorithms typically show improvement when transferring from images simply generated using traditional approaches in virtual environments to real-world data (Bird et al. 2020).

More recently, transfer learning has been dominating image classification problems. (Zhou et al. 2019) used a pretrained InceptionV3 model for differentiating benign and malignant renal tumors on CT images. (Deniz et al. 2018) proposed a classifier for breast cancer on histopathologic images. The accuracy measures reported in the transfer learning-based algorithms were superior to those traditional

| Name | Positive | Negative | Total Count |
|---|---|---|---|
| Br35H: Brain Tumor Detection (Cheng 2017) 2020 | 1500 | 1500 | 3000 |
| Brain Tumor Classification from NIT Durgapur (Bhuvaji 2019) | 2400 | 800 | 3200 |
| Open Access Brain MRI Dataset (Bohaju 2020) | 3700 | 1200 | 3700 |
| Open Access Brain MRI from Kaggle (Chakrabarty 2018) | 155 | 98 | 253 |

Table 1: Basic information of our datasets

methods. (Yang et al. 2018) proved that GoogLeNet proved superior to AlexNet on grading of glioma from MRI images. (Jain et al. 2019) used a pre-trained VGG-16 network for diagnosis of Alzheimer's disease from MRI. These experiments concluded that a deep transfer learning model can be adapted to medical image classification, with minimum preprocessing.

## Approach

### Deep Convolutional Neural Network

The first approach is by building a CNN from scratch. As we are gradually moving from simple to complex, we decided to start with a simple architecture with minimal layers and hyperparameters. These types of architectures will be simpler than the advanced neural networks like VGG or EfficientNet. As for image feature extraction, CNN and MaxPooling are the most efficient approaches. We decided to combine 2 layers of CNN with 1 layer of MaxPooling2D.

We further experimented with the number of layers and dimensionalities for the hidden layers and finalized the following architecture. The input images of the dataset are varying in sizes ranging from 200 to 800 pixels. We decided to cap the image size to 128 by 128 pixels as this allowed us to effectively capture the essence of the image. This further was used to decide the dimensions of the input layer of the image.

The basic information of our datasets are listed in Table 2. The following architecture was finalized with training with 100 epochs, batch size = 128 and learning rate = 0.001 using Adam Optimizer (Kingma and Ba 2014). This set of hyperparameters and configurations gave us the best result so far. The activation function between the layers used is ReLU activation function.

We've experimented training our baseline architecture on a single dataset Br35H as well as the entire dataset. The test results can be found in the Results section.

The most significant challenge in this project is the relatively small amount of data available to train on, also known as the "data insufficiency problem." Therefore, while considering the improvement of our current model, we have taken two main approaches to solve this problem.

### Data Augmentation

The second approach is called "data augmentation". Data augmentation is a widely-used technique that involves generating new, synthetic training data from existing data by applying random transformations such as rotation, cropping, and flipping, etc. This allows the model to be trained on a larger and more diverse dataset, which can improve its performance on the test set and reduce overfitting. This allows us to compare the classification accuracy of a CNN trained on a small amount of real data vs a CNN trained on a large amount of augmented data. Our hypothesis is that the model trained on a large amount of augmented data will perform better. In our experiments, we employed data augmentation to enhance the classification power of our model.

In our second experiment, we preserved the CNN used in our first approach, but only added more synthetic data by using data augmentation techniques. To implement data augmentation in our experiments, we used a library called Augmentor (Bloice 2019) in our deep learning framework. It can apply six kinds of augmentation techniques (perspective skewing, distortion, rotation, shearing, zoom, flip) and easily generate and store samples to our PC. The reason we use Augmentor instead of Pytorch's built-in transforms is that we value the ability to store any number of images easily and directly for future use.

**Types of Augmentation Techniques** In our experiments, we applied three data augmentation techniques - rotation, flip, and zoom - to our training dataset. We selected these techniques because they do not significantly alter the natural shape of the tumor in the images. As shown in Figure 9, techniques such as perspective skewing, distortion, or shearing can drastically change the shape of the tumor, resulting in images that do not reflect the true appearance of the tumor. This could potentially confuse the model and affect its performance.

By using only rotation, flip, and zoom, we were able to generate augmented training data that closely resembled the original images and preserved the natural shape of the tumor. This allowed our model to learn from a more diverse and realistic dataset, ultimately improving its performance on the validation and test sets.

**Settings of Augmentation** In our experiments, we applied four individual settings of data augmentation techniques to the training dataset in order to see which technique was helpful for training. The four settings we used were:

- Only do a forced rotation in a clockwise or counterclockwise direction by less than or equal to 10 degrees.
- Only do flipping, we have a probability of 0.5 to do a left-right flip, and another probability of 0.5 to do a top-down flip. These two flips are independent.
- Only do a zoom-in by applying a random cropping with probability 0.8, preserving an area that is 0.8 times the size of the original image.

| Layers | Positive |
|---|---|
| Convolution 2D | $1 \times 128$(kernel size $= 3$) |
| Max Pooling 2D | $2 \times 2$ |
| Convolution 2D | $128 \times 32$(kernel size $= 2$) |
| Fully Connected | $62 \times 128$ |
| Fully Connected | $128 \times 64$ |
| Flatten | $126796$ |
| Fully Connected (Output) | $126796 \times 2$ |

Table 2: Architecture for our CNN

- A pipeline that combined rotation, flip, and zoom in this order using the parameters described above.

We performed each augmentation independently, generating 1000 augmented images for each technique. As a result, the training set now has 5881 positive results (size increased by 20.48%) and 4258 negative results (size increased by 30.69%). Figures 10 and 11 some sample images generated using these four settings.

To evaluate the effectiveness of data augmentation, we trained two separate models using different augmented datasets. The first model was trained using the fourth augmentation technique (pipeline). The second model was trained using the first augmentation technique (rotation). We didn't perform training on the other two datasets due to limited time.

Note that we don't apply augmentation on validation and test sets. So we evaluated all models with the same validation set during training, and evaluated their overall performance on the same test set. The results of both models are shown in the Results section.

## Transfer Learning with EfficiencyNetB4 and YOLOv5

**Transfer Learning with EfficiencyNetB4** Our final analysis employed transfer learning to address a few of the issues encountered by the first two approaches. Firstly, it helps resolve the "data insufficiency" problem since a large amount of data is not as necessary as most of the parameters are already pre-trained. Second, since there are fewer trainable parameters, the training time is usually less and more focus can be devoted to fine-tuning hyperparameters. Finally, these models' higher level of sophistication allow us to explore problems outside of just classification such as object detection or segmentation.

The first model we utilized was EfficientNetB4 which is a 300+ layer architecture that is originally trained on the ImageNet dataset. EfficientNet operates on the concept of compound scaling while increasing model size in order to optimize accuracy gains. Most of the modifications to the model involved replacing the last layer with a batch normalization layer, a regularized dense layer, a dropout layer, and a softmax output with an overall learning rate of 0.001 using Adam Optimizer.

**Object Detection with YOLOv5** The second model we used for our transfer learning approach was YOLOv5, a state-of-the-art object detection architecture that is very popular in the field of computer vision due to its speed and accuracy. We ultimately decided to use YOLO because it offered a few key advantages over our other methodologies. Firstly, this being an object detection algorithm enabled us to go beyond simple classification and actually identify the exact location of the tumor in the image. This capability becomes particularly useful in the context of diagnosing the tumor type as its location is closely linked to its disease characteristics. The second benefit is YOLOs' hyper-fast processing which enables the model to perform objection detection in real-life time such as with a videofeed. Proving that YOLO works well in the context of static medical images would be an important proof of concept to justify its usage in continuous medical imaging such as an ultrasound.

The majority of the pre-processing involved converting bounding box annotations to YOLO format. Specifically, YOLO annotations are represented by a class value and 4 values describing the bounding box: [class, x_center, y_center, width, height]. All training was performed using the default hyperparameters found in the .yaml file. These hyperparameters were originally optimized on the COCO dataset that YOLOv5 was originally trained on.

Due to the computational intensity of this type of training, we elected to use Google Colab Pro with a P100 GPU. This means we were able to use more aggressive training settings to improve performance. The image training size was chosen to be 640 as this was the native resolution of COCO and we had the training run for 400 epochs to help mitigate overfitting. Finally, the hardware allowed us to use a relatively large batch size of 85 to greatly increase performance. Please refer to the Results section to see the results of YOLOv5.

## Experiments and Results

### Model 1: CNN trained from scratch

**Using Single Dataset and Baseline Architecture** In the first attempt, we used the Br35H brain tumor dataset as our target dataset. We decided to maintain the architecture from Table 2.

The loss vs epoch curve (Figure12) didn't show much fluctuations apart from the regular noise it receives while calculating the gradient descent.

The training and validation F1 scores were 0.932 and 0.895, respectively. This further assured that our model actually learned from the data.

On the test dataset, the model performed accurately well enough with around 0.844 F1 score. Adjusting the hyperparameters didn't appear to further improve the accuracy for the test dataset. As it is possible that the current model was not able to understand the more hidden features of the image. The visualizations for the confusion matrices are displayed in Figure 13 with the PR curves in Figure 14.

The results obtained from this initial approach resulted into further approaches which could be defined as follows:

- Improve model by optimizing and utilizing data
- Improve model by adding more complex layers

**Using Additional Datasets on the Same Architecture** As we are attempting to gain more accuracy over the simple model, we decided to involve more datasets to see the results in effect.

After again adjusting the hyperparameters for the model, we selected the epoch count to be 100 and the batch size to be 128 again, so we can replicate the behavior observed in the single dataset model and verify the impact of additional data to the given model. The loss vs epoch curve (Figure 18) had some noise at a few iterations where the loss function value increased but it again got back on track and started behaving as expected.

The training and validation F1 score on this dataset are 0.912 and 0.848 respectively. This further assured that our model actually learned almost similar features from the data. On the test dataset, the model performed with almost the same F1 score as it did on a single dataset, i.e., 0.857. Adjusting the hyperparameters didn't appear to further improve the accuracy of the test dataset. The visualizations for the confusion matrices are displayed in Figure 16 with the PR curves in Figure 17.

There are two possible explanations for this scenario which are as follows:

- As the datasets were from different sources, it is possible that the model was not able to capture the most differentiating features, and hence may require more data. This will further be resolved with the help of data augmentation. We achieve these results using Model 2.
- The second reason is pertaining to the complexity of the model. Maybe we need more layers and nodes in order to capture the more differentiating factors in an image and hence, we will further resolve this issue using transfer learning. We achieve these results using Model 3.

## Model 2: CNN Training with Augmented Data

**Model Trained Using Augmentation Pipeline** Please note that the two models were trained using the same hyperparameters as the first approach.

Our first step was to examine the performance of the model when all available techniques were combined (the fourth dataset). We hypothesized that the use of multiple techniques would result in a more diverse and comprehensive augmented dataset, which would improve the performance of the model to its best compared to other datasets. We show our training loss (Figure 18), F1 scores, confusion matrices (Figures 19, 21) and PR curves (Figure 20, 22) for training, validation and testing.

During the model tuning process, we unexpectedly found that the model trained with only 4 epochs achieved very good results, already surpassing the same model without augmented data and comparable to EfficientNet. Results showed that the training loss went down drastically before 4 epochs and the F1 scores on training set, validation set and test set reached 0.969, 0.954, 0.961.

Furthermore, the final model is trained using 100 epochs. The results were astonishing: F1 score on the training set reached 1.0, which suggests that the model learned the distribution of our training set (including the augmented data) perfectly with not even one single mis-classification. The F1 scores on validation set and test set also increased to 0.982 and 0.977 respectively. This experiment demonstrated that data augmentation increased the model's ability to generalize well to both acquired and unseen data. Overall, the results of the training are striking and show the effectiveness of data augmentation in enhancing the classification power of our model.

**Model Trained Using Only Rotation Augmentation** We also explored another model that was trained using original data and data augmented by only rotation. We astonishing found that this model is actually better than our first model, which suggests that our hypothesis that the use of multiple techniques would improve the performance of the model is wrong. We still show our training loss (Figure 23), F1 scores , confusion matrices (Figure 24) and PR curves (Figure 25) for training, validation and testing.

The results show that the training loss decreases more slowly than the first moel, becoming small after about 30 epochs (comparable to the first model at 4 epochs). The F1 scores on training set is still 1. Though the performance on validation set decreased to 0.975, it actually got a better F1 score on the test set at 0.982. This experiment demonstrates that while data augmentation can improve model performance, using all available techniques without consideration may not necessarily result in the best performance.

One potential reason is that rotation may have been better suited to the specific characteristics of the training data, leading to more effective regularization and improved generalization. After using rotation-based data augmentation, the model learned to identify brain tumors as circular shapes, allowing it to accurately detect tumors in brain images regardless of their orientation. It is also possible that using only rotation for data augmentation may have reduced the risk of overfitting, since the F1 score for validation set decreased while the score for testing increased. Therefore, using only rotations may result in a model that better captures the underlying patterns in the data.

Another possible reason could be that the additional data augmentation techniques (zoom and flip) added in the pipeline approach may have introduced additional noise or variability into the training data, making it more difficult for the model to learn effectively. This may be because using additional data augmentation techniques such as flipping and magnification can make it more difficult for the model to accurately identify brain tumors. For example, flipping an image may cause non-tumor shapes to appear similar to tumors, and excessive zoom may cause tumors to lose their characteristic shape or only be partially visible. These factors could potentially lead to reduced performance.

To sum up, this experiment highlights the importance of carefully choosing and fine-tuning data augmentation techniques for a given dataset and model.

## Model 3: Transfer Learning

**Transfer Learning with EfficientNet** The model was originally planned to train for 100 epochs, but quickly plateaued after only 25 epochs to attain F1 scores of 0.97 as seen in Figure 26.

The follow-up predictive analysis (Table 3) on the test set revealed a similarly strong performance as evidenced by an average F1 score of 0.982. In particular, the confusion matrix (Figure 27) shows very few false negatives which is extremely important as a missed diagnosis is much more dangerous than an incorrect diagnosis. Though the amount of misclassifications were quite low, it is still an important exercise to investigate which images the model seemed to struggle with. Figure 28 addresses this very question by showing a breakdown of the percentage of misclassifications the EfficientNetB4 model had for each dataset, by class. We can see that dataset 3 was disproportionately misclassified which ultimately is unsurprising given the low quality of the MRI images.

**Object Detection with YOLOv5** The results of the training, despite the low amount of data, look to be quite promising with a mean average precision of nearly 0.8 (Figure 29) which suggests that the predicted bounding boxes are mostly overlapping with their true locations.

Furthermore, we can see from the predictions made on images with tumors in Figure 30 that the model is quite good at locating the tumor despite a range of shades, sizes, and locations.

**Webapp** With the transfer learning models working so effectively at classifying and detecting tumors, we opted to take an end-to-end approach with this project by developing an interactive web app where users could upload images of brain MRIs to be evaluated by these models. Though a simple prototype, the goal is to have it serve as a useful example of how machine learning engineers can easily deploy their models to the public domain.

The app was built using Streamlit, a free and open-source Python-based framework that aims to help machine learning engineers rapidly develop and share machine learning and data science web apps (Figure 8).

The current functionality allows users to upload a brain MRI which is then classified based on the presence of a tumor using a bar plot that shows the output probabilities. This transparency is important to communicate that in some cases our model may still not be confident about a result. We hope to expand this tool by building additional predictive models to accept a larger variety of medical images.

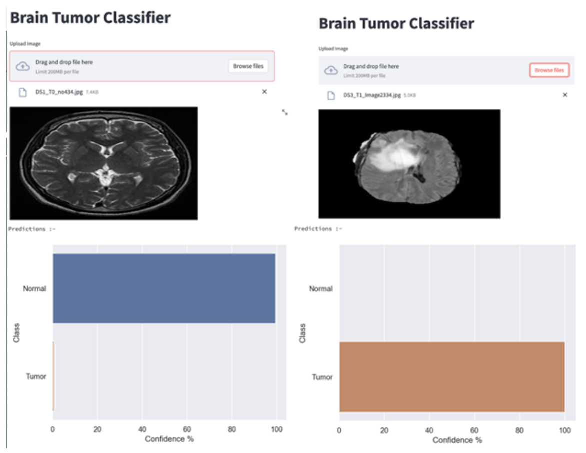


Figure 8: User Interface of the web app

## Conclusion and Future Work

### Conclusion

We did a comparative analysis of the factors affecting the performance of brain tumor classification. Started from a baseline model which is trained on a single dataset, we moved on to a broader combined dataset. This procedure didn't produce effective results. Also tuning the hyper parameters lead to over-fitting the training dataset. We derived conclusions from this baseline model that either we need to refine the data used or built a more complex model.

We then tried appropriate data augmentation, which led to improvement in F1 scores. We found that for this task, using rotation alone to perform data augmentation was better than using a pipeline of rotation, flip, and zoom. So when are using data augmentation, we must take into account the characteristics of the task but not simply combine every technique.

We finally explored transfer learning models to deal with the problems. With the help of EfficientNet, we required fewer training epochs to achieve better performance than the previous models even on the original dataset. We finally prepared a framework that detects the locations of tumors that worked really well on the dataset.

Our work can be found at https://github.com/theturbokhemist/BrainTumorClassificationDL.

### Future Work

In reality, there are many different types of brain tumors. Generally, doctors need to identify their type to determine the precise way to help them. For example, Glioma is generally located in the center of the brain, Meningioma is located at the border of the brain, and Pituitary is located in the medial part of the brain and is more difficult to be detected. We can try performing multiple classifications of brain tumors to separate them by detecting their specific location. If we have done that, we can add it to our webapp.

| Name | Precision | Recall | F1 score | Support |
|---|---|---|---|---|
| Normal | 0.980 | 0.974 | 0.973 | 408 |
| Tumor | 0.982 | 0.991 | 0.982 | 611 |
| Macro Avg | 0.981 | 0.981 | 0.981 | 1019 |
| Weighted Avg | 0.981 | 0.982 | 0.982 | 1019 |

Table 3: Classification Report

## Individual Work

**Build CNN from scratch:** Abdulaziz
**Determine training strategy:** Abdulaziz
**Data preprocessing:** Danya
**Transfer learning:** Danya
**Data augmentation:** Jiaheng
**Augmentation-based model training:** Jiaheng
**Comparison, Final Report, and Presentation:** Danya, Abdulaziz, Jiaheng

## References

Afshar, P.; Plataniotis, K. N.; and Mohammadi, A. 2019. Capsule networks for brain tumor classification based on MRI images and coarse tumor boundaries. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1368–1372. IEEE.

Bhuvaji, S. 2019. Brain Tumor Classification (MRI) Kaggle Dataset.

Bird, J. J.; Faria, D. R.; Ekárt, A.; and Ayrosa, P. P. 2020. From simulation to reality: Cnn transfer learning for scene classification. In *2020 IEEE 10th International Conference on Intelligent Systems (IS)*, 619–625. IEEE.

Bloice, M. D. 2019. https://augmentor.readthedocs.io/en/stable/.

Bohaju, J. 2020. Extracted features for brain tumor.

Chakrabarty, N. 2018. Brain MRI Images for Brain Tumor Detection.

Charron, O.; Lallement, A.; Jarnet, D.; Noblet, V.; Clavier, J.-B.; and Meyer, P. 2018. Automatic detection and segmentation of brain metastases on multimodal MR images with a deep convolutional neural network. *Computers in biology and medicine*, 95: 43–54.

Cheng, J. 2017. Br35H: Brain Tumor Detection 2020, Version 5.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Deniz, E.; Şengür, A.; Kadiroğlu, Z.; Guo, Y.; Bajaj, V.; and Budak, Ü. 2018. Transfer learning based histopathologic image classification for breast cancer detection. *Health information science and systems*, 6(1): 1–7.

Díaz-Pernas, F. J.; Martínez-Zarzuela, M.; Antón-Rodríguez, M.; and González-Ortega, D. 2021. A deep learning approach for brain tumor classification and segmentation using a multiscale convolutional neural network. In *Healthcare*, volume 9, 153. MDPI.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2020. Generative adversarial networks. *Communications of the ACM*, 63(11): 139–144.

Gron, A. 2017. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 1st edition. ISBN 1491962291.

Hollon, T. C.; Pandian, B.; Adapa, A. R.; Urias, E.; Save, A. V.; Khalsa, S. S. S.; Eichberg, D. G.; D'Amico, R. S.; Farooq, Z. U.; Lewis, S.; et al. 2020. Near real-time intraoperative brain tumor diagnosis using stimulated Raman histology and deep neural networks. *Nature medicine*, 26(1): 52–58.

Jain, R.; Jain, N.; Aggarwal, A.; and Hemanth, D. J. 2019. Convolutional neural network based Alzheimer's disease classification from magnetic resonance brain images. *Cognitive Systems Research*, 57: 147–159.

Jocher, G.; Stoken, A.; Borovec, J.; STAN, C.; and Changyu, L. 2020. ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, 740–755. Springer.

Neugut, A. I.; Sackstein, P.; Hillyer, G. C.; Jacobson, J. S.; Bruce, J.; Lassman, A. B.; and Stieg, P. A. 2019. Magnetic Resonance Imaging-Based Screening for Asymptomatic Brain Tumors: A Review. *The Oncologist*, 24(3): 375–384.

Pashaei, A.; Sajedi, H.; and Jazayeri, N. 2018. Brain tumor classification via convolutional neural network and extreme learning machines. In *2018 8th International conference on computer and knowledge engineering (ICCKE)*, 314–319. IEEE.

Rouse, C.; Gittleman, H.; Ostrom, Q. T.; Kruchko, C.; and Barnholtz-Sloan, J. S. 2015. Years of potential life lost for brain and CNS tumors relative to other cancers in adults in the United States, 2010. *Neuro-oncology*, 18(1): 70–77.

Shorten, C.; and Khoshgoftaar, T. M. 2019. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1): 1–48.

Tan, M.; and Le, Q. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, 6105–6114. PMLR.

Tharwat, A. 2020. Classification assessment methods. *Applied Computing and Informatics*.

Yan, P.-F.; Yan, L.; Zhang, Z.; Salim, A.; Wang, L.; Hu, T.-T.; and Zhao, H.-Y. 2016. Accuracy of conventional MRI for preoperative diagnosis of intracranial tumors: A retrospective cohort study of 762 cases. *International Journal of Surgery*, 36: 109–117.

Yang, Y.; Yan, L.-F.; Zhang, X.; Han, Y.; Nan, H.-Y.; Hu, Y.-C.; Hu, B.; Yan, S.-L.; Zhang, J.; Cheng, D.-L.; et al. 2018. Glioma grading on conventional MR images: a deep learning study with transfer learning. *Frontiers in neuroscience*, 12: 804.

Zhou, L.; Zhang, Z.; Chen, Y.-C.; Zhao, Z.-Y.; Yin, X.-D.; and Jiang, H.-B. 2019. A deep learning-based radiomics model for differentiating benign and malignant renal tumors. *Translational oncology*, 12(2): 292–300.

## Appendix: Supportive Images

Please see all supportive images starting from the next page. We moved them to the back because they are too large in size and will interfere with the reading of the text.

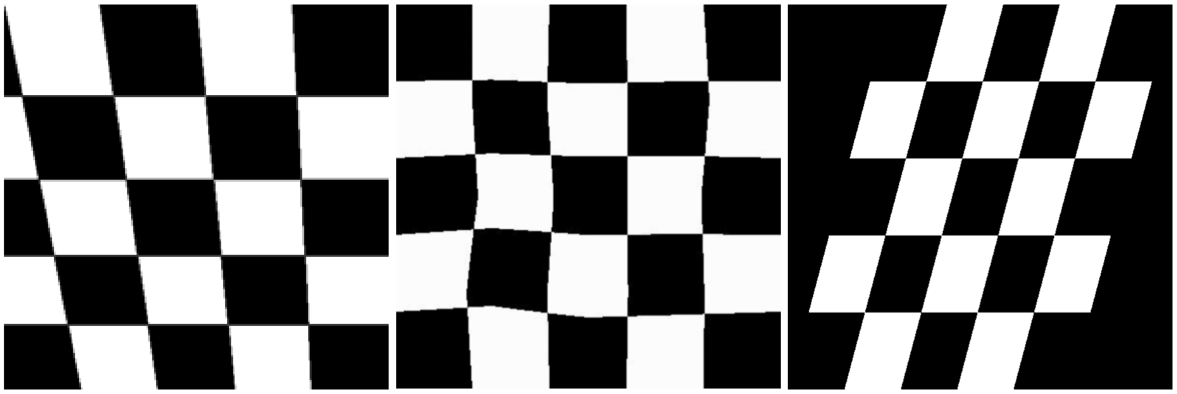


Figure 9: Effects of skewing, distortion and shearing on a grid

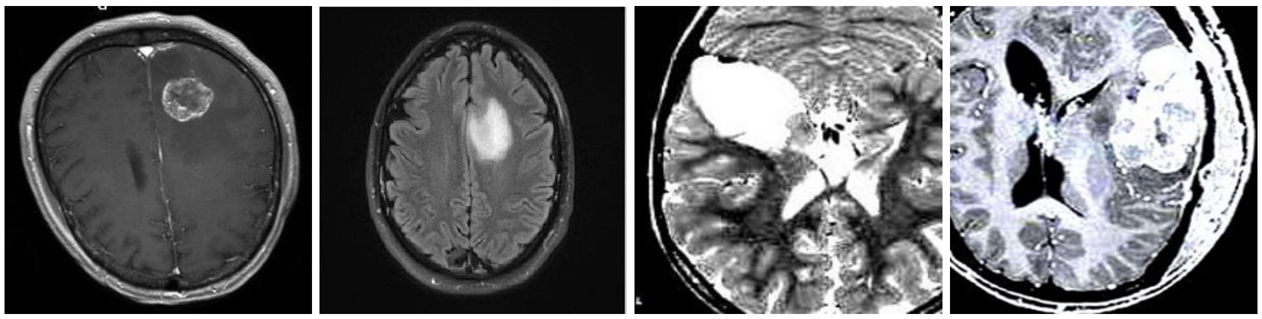


Figure 10: Images with tumors that are generated by: rotation, flipping, zoom, pipeline

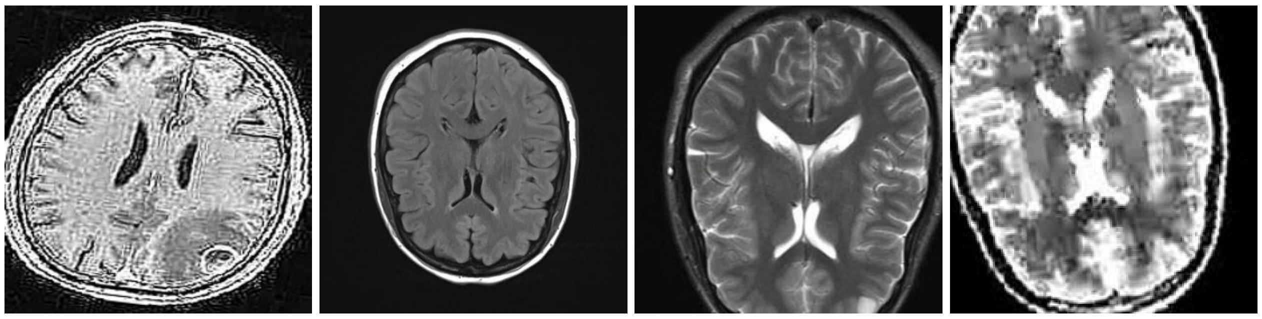


Figure 11: Images without tumors that are generated by: rotation, flipping, zoom, pipeline

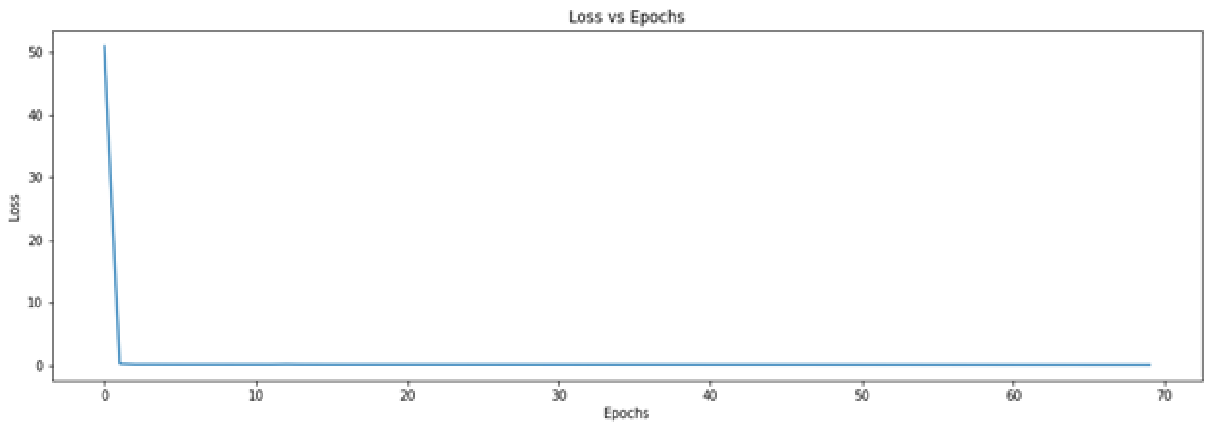


Figure 12: Loss vs Epochs plot of our model trained on Br35H

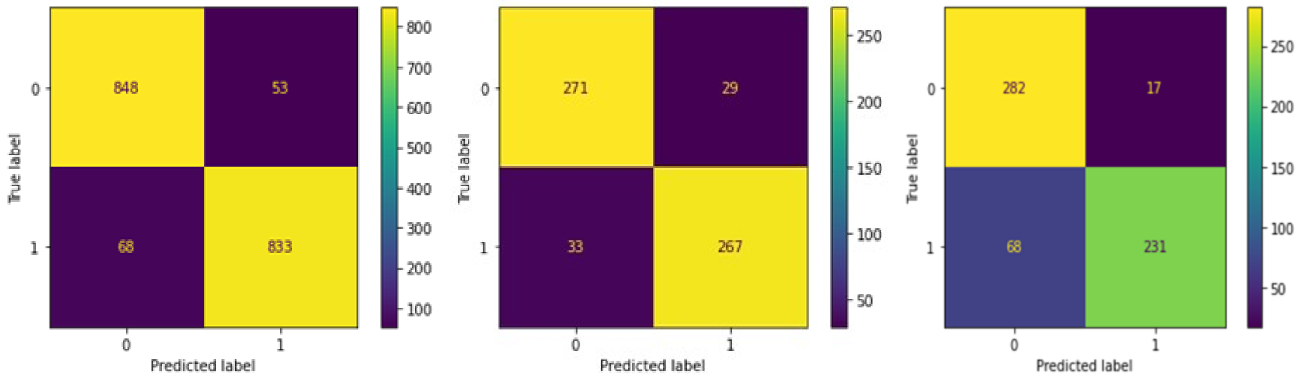


Figure 13: Confusion matrices obtained with using Br35H only

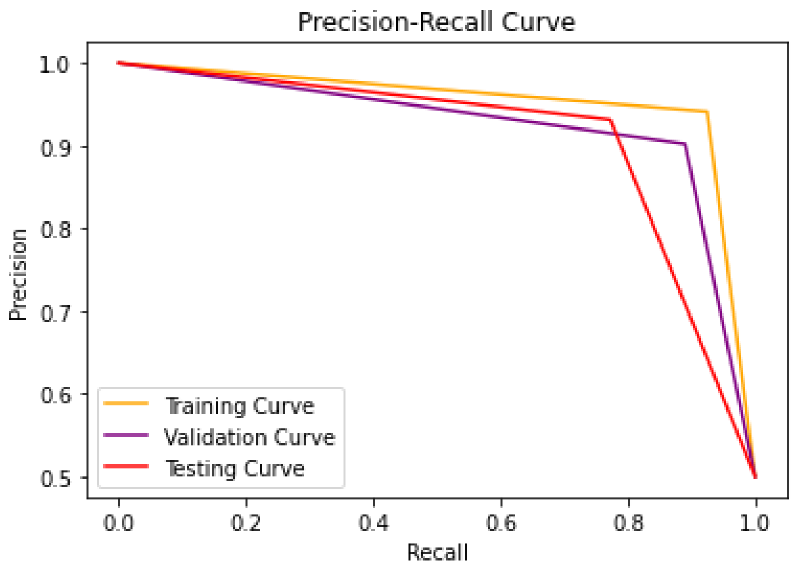


Figure 14: PR curve during training and testing of the model trained on Br35H

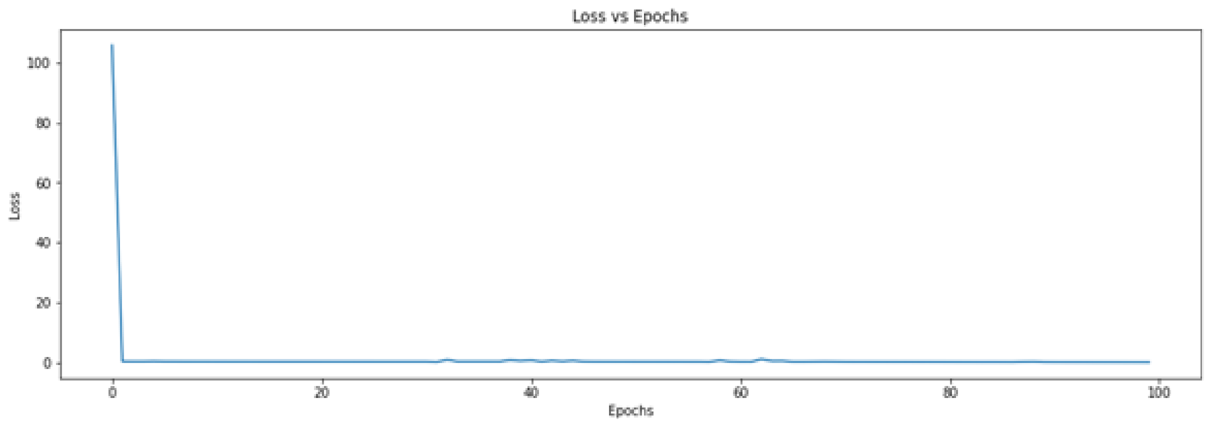


Figure 15: Loss vs Epochs plot of our model trained on all datasets

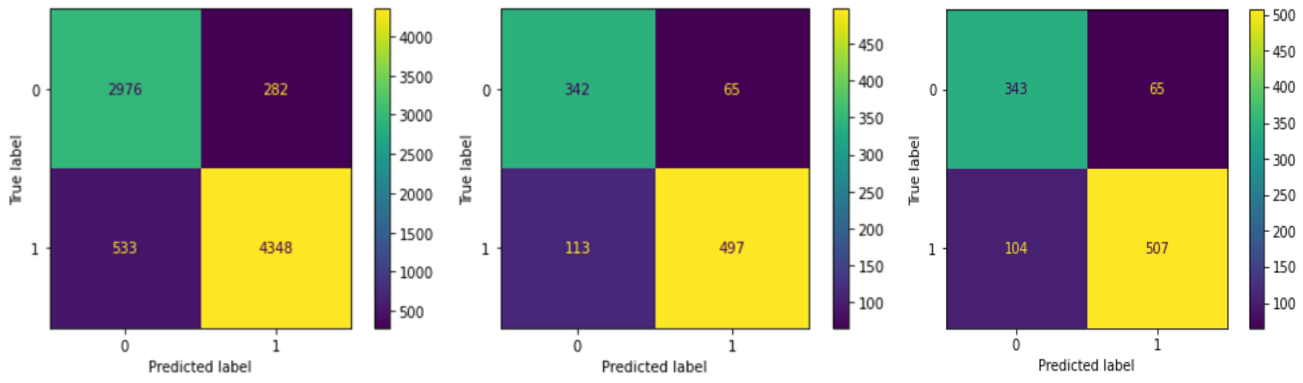


Figure 16: Confusion matrices obtained with using all datasets

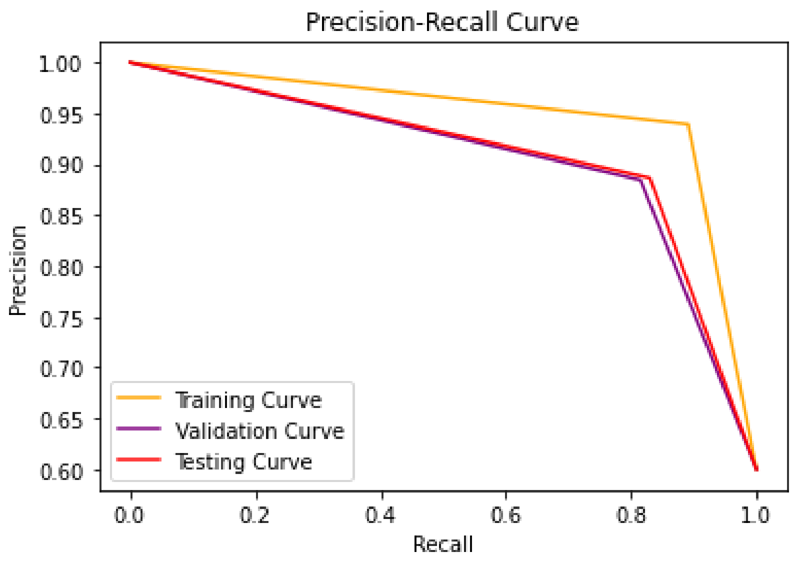


Figure 17: PR curve during training and testing of the model trained on all datasets

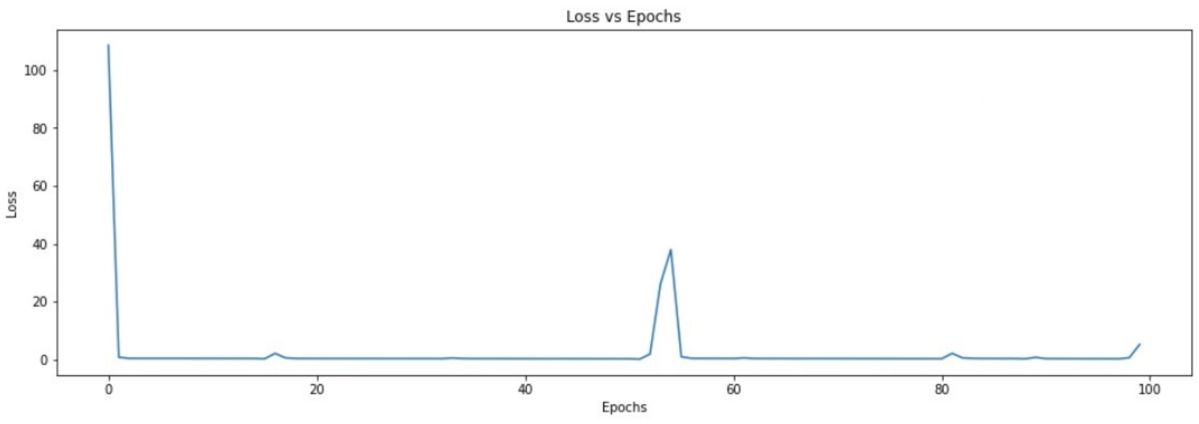
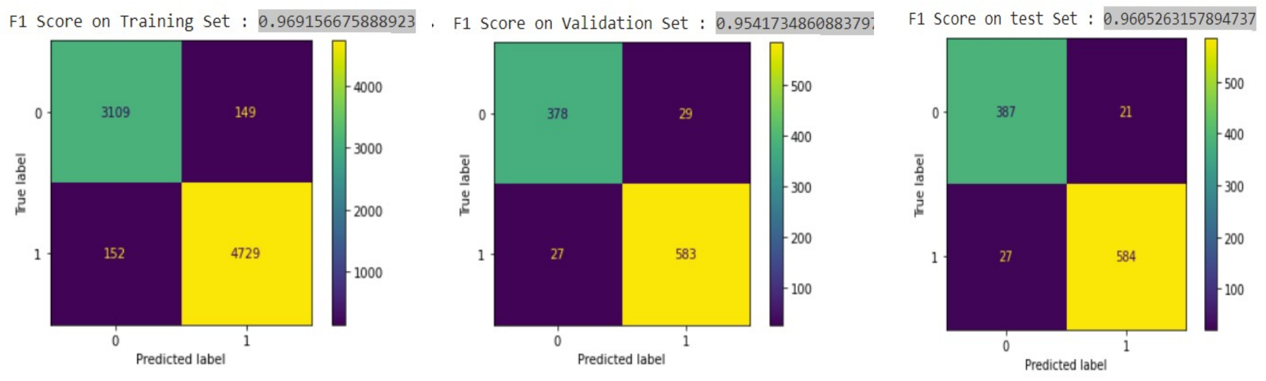


Figure 18: Loss vs Epochs plot of our first augmented model



```
F1 Score on Training Set : 0.969156675888923
F1 Score on Validation Set : 0.954173486088379
F1 Score on test Set : 0.9605263157894737
```

Figure 19: F1 score and confusion matrix obtained with only 4 training epochs

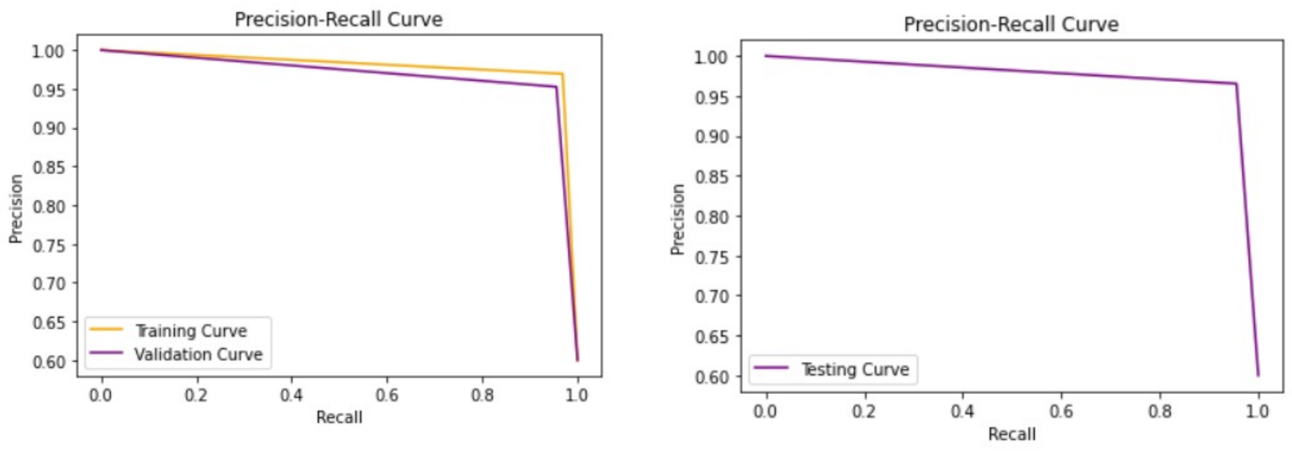


Figure 20: PR curve during training and testing obtained with only 4 training epochs

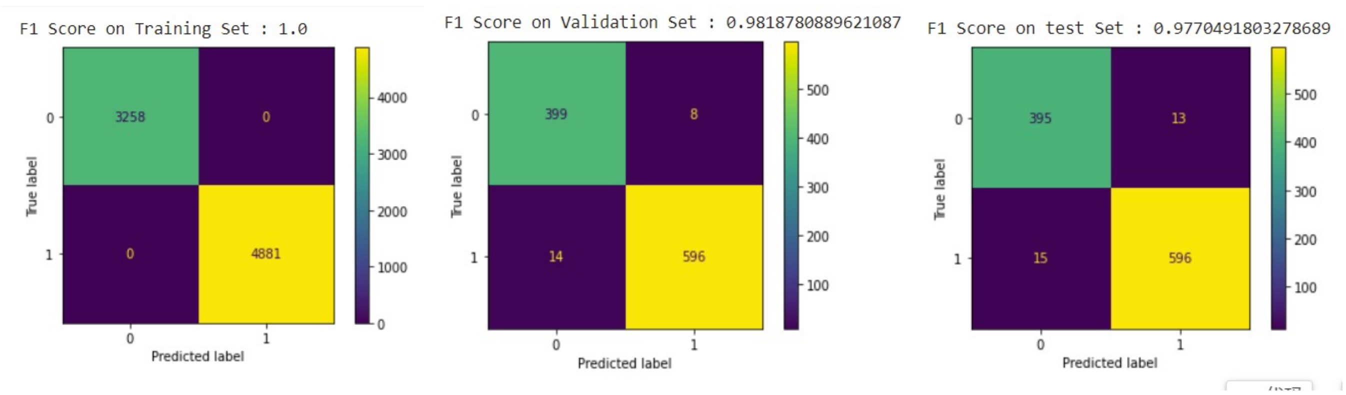


Figure 21: F1 score and confusion matrix obtained with 100 training epochs

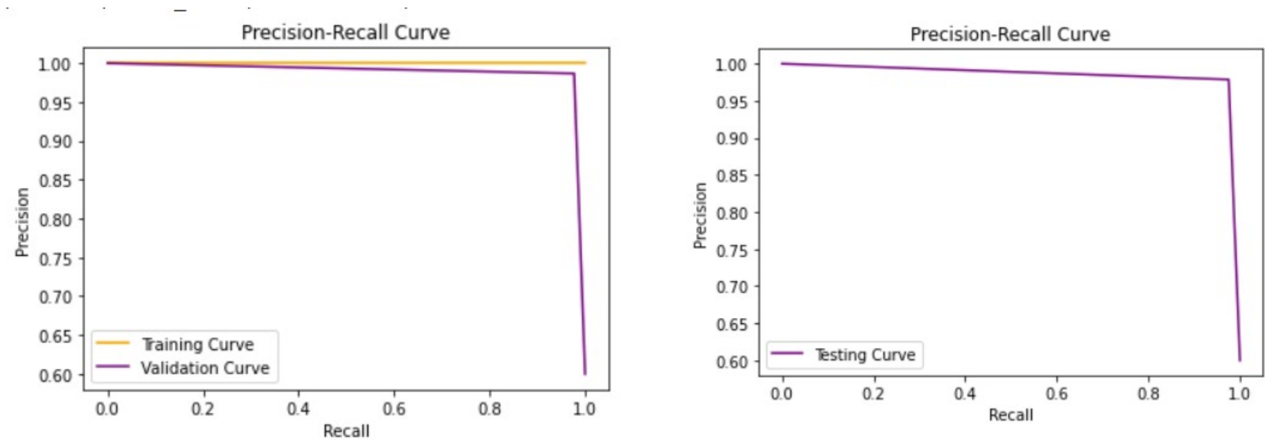


Figure 22: PR curve during training and testing obtained with only 4 training epochs

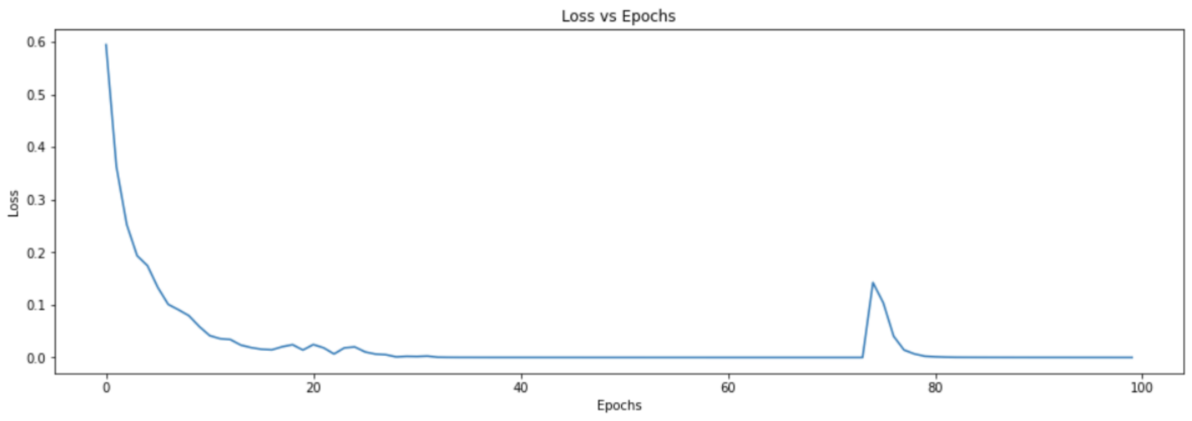


Figure 23: Loss vs Epochs plot of our second augmented model

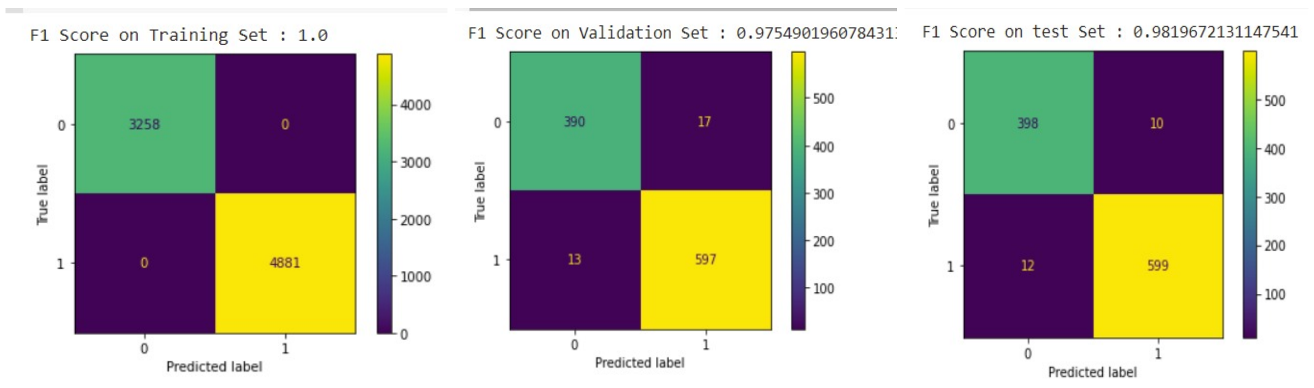


Figure 24: F1 score and confusion matrix obtained with 100 training epochs

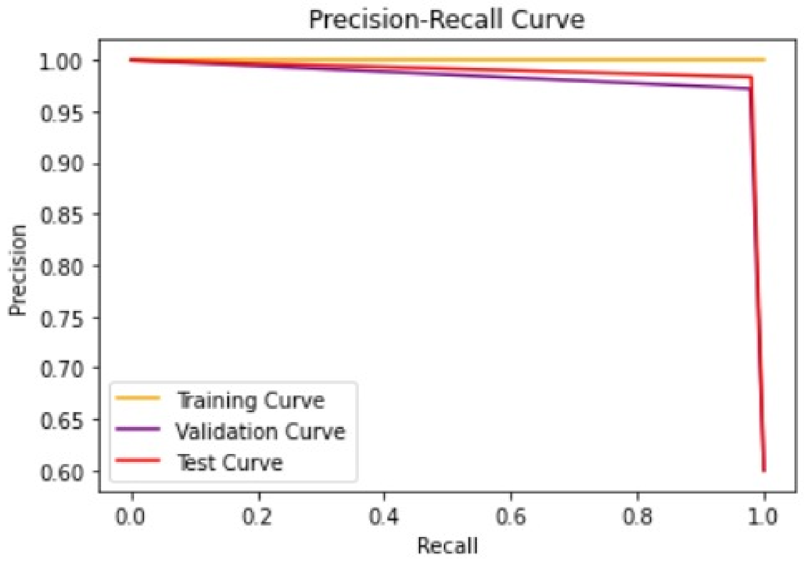


Figure 25: PR curve during Training and Testing obtained with 100 training epochs

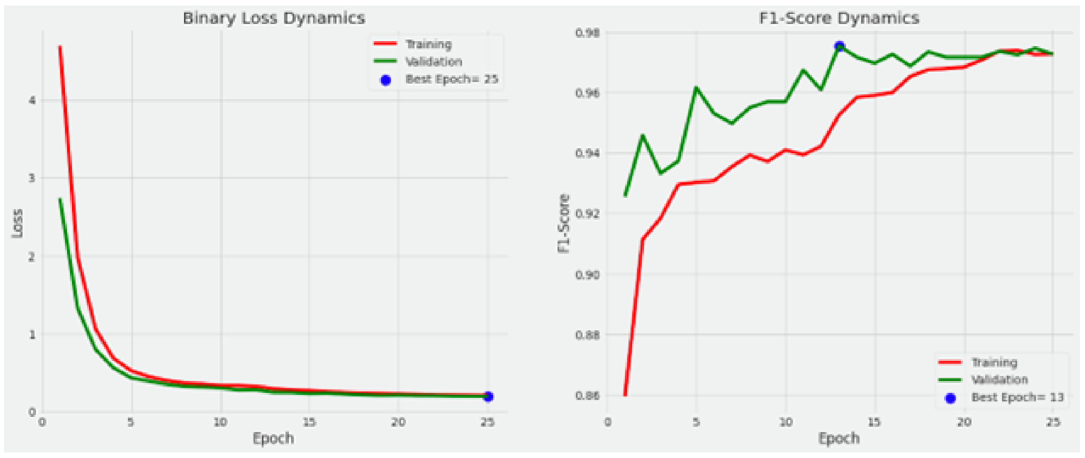


Figure 26: Training Dynamics for EfficientNet

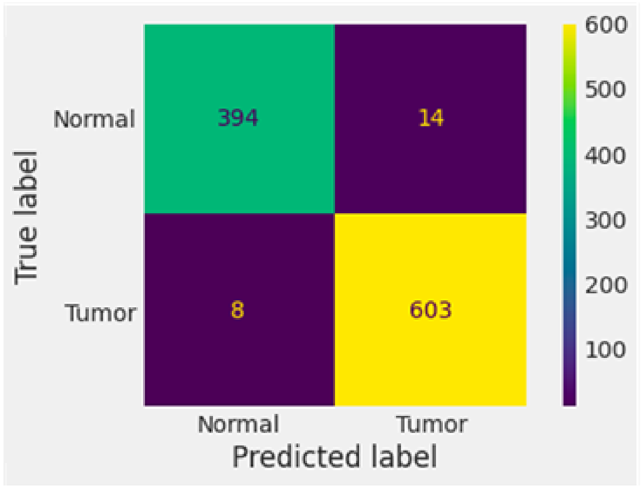


Figure 27: Training Dynamics for EfficientNet

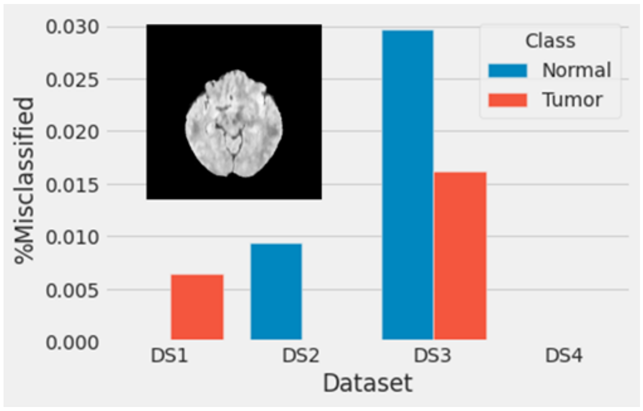
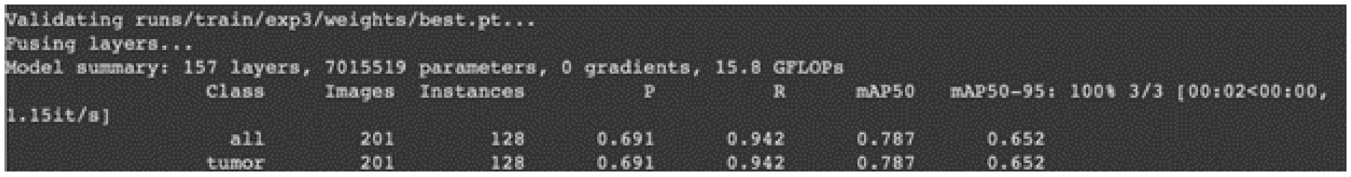


Figure 28: Misclassification Rate by Dataset and Class



```
Validating runs/train/exp3/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 3/3 [00:02<00:00,
1.15it/s]
                   all        201        128      0.691      0.942      0.787      0.652
                 tumor        201        128      0.691      0.942      0.787      0.652
```

Figure 29: Evaluation Metrics for YOLOv5 training

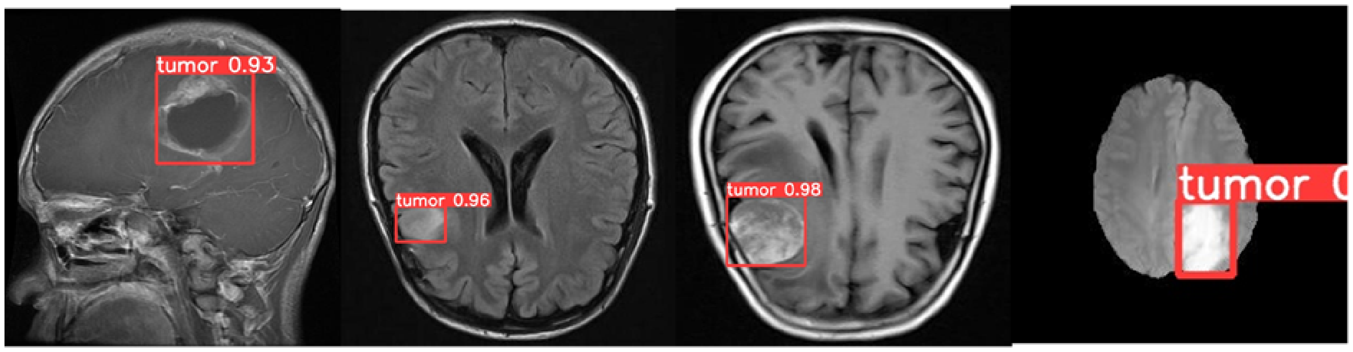


Figure 30: Output of YOLO model on test images with tumors