Image Classification Neural Networks for Horizontal Brain MRI

Report by Zhuo Cao, Tse-Wei Lee, Longpeng Xu, Yingqi Zhang

November 1, 2023

Contents

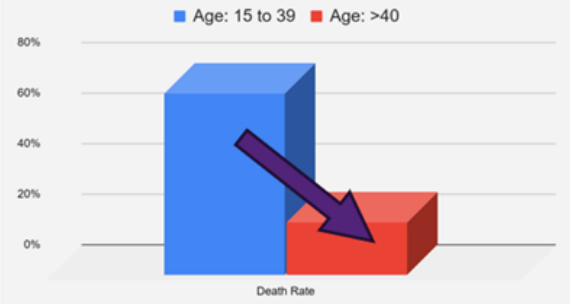
We give consent for this to be used as a teaching resource.

1 Problem and Significance

The brain is a complex organ in the human body where it processes and controls major functions that are essential for human survival such as memory, sleeping, temperature regulation, motor and sensory coordination. It works through billions of cells that are lined with bone, fluids, and fats within the cranial cavity to ensure the central nervous system can function appropriately by the physiology of the human body.

Tumours are caused by uncontrolled growth of cells and they might affect normal brain activities by destroying normal cells causing structural and chemical dysregulation. Some tumours are slow-growing or fast-growing, while some are aggressive. Regardless of the type or size, it is crucial to detect any brain lesion as early as possible. While the five-year survival rate for brain tumour is 72% for the age group between 15 to 39 years, the survival rate drops to 21% for the age group older than 40 years old.

Figure 1: The 5-year survival rate of brain tumor by age group



Despite the survival rate, brain tumour is the 9th most common cause of death from cancer in Australia, and approximately 300,000 people globally (ASCO, 2023) are diagnosed with brain tumour each year. Therefore, it is essential to identify the best possible diagnosing method that can be utilized and managed by the medical practitioner.(Australian Government, 2022)

The existing diagnosing method is to use Magnetic resonance imaging (MRI). It plays an essential role in diagnosing, analysing brain tumours. MRI is superior in characterising brain tumours as it employs different parameters such as T1, T2, FLAIR, and diffusion-weighted imaging sequences based on the time the radiofrequency pulse creates the MRI signals, which is dependent on the tissue composition of the tumour. Therefore, each sequence is useful in providing specific information regarding the tumour characteristics.(Amin et al., 2021)

Early brain tumour detection and accurate segmentation are important. Manual segmentation can be time-consuming for the radiologist, especially with subtle or complex brain tumours. Hence, computer-aided diagnosis using machine learning has been rapidly developed in recent days for precise tumour detection. Varied algorithms are developed to offer accuracy in the detection and classification of brain lesions.(Amin et al., 2019)

This report identifies several common machine learning algorithms with Pytorch including basic CNN, RestNet50, DenseNet and VGG16. The image datasets are obtained from Kaggle, a well-known dataset resource. Through the dataset and various algorithms, there are promising outcomes.

Despite the efforts and promising outcomes in medical imaging analysis with machine learning, consistent results with correct segmentation and characterisation remain a challenging task for brain tumours due to the variability in individuals, location, size, and types of tumours. More datasets and research need to be continued to improve diagnostic accuracy and predictability. Recent works have invested in automated techniques to classify between tumour and non-tumour MRI in clinical and research settings. This would be helpful, especially in complex cases to not only the radiologists but also medical and radiation oncologists in tailoring the best possible care for their patients.(Huang et al., 2022)

Therefore, radiologists assume that computerized methods can improve diagnostic abilities based on automated machine learning methods.

2 Three types of CNNs

2.1 Data Preprocessing

Initially, a dataset by Hamada (2021) is selected for the following reasons. (1) The brains are imaged from the same perspective (one of horizontal, coronal, and sagittal). (2) The images are with binary labels which indicate with-/without tumor(s). (3) The resolutions of the images are generally finer than 150×150, ensuring the quality of model training. (4) The images are of mixed imaging contrasts (T1, T2, or diffusion weighted), ensuring the generalizability of neural networks.

For preprocessing, firstly, we conducted resizing and checking for intensity. Setting a uniform size for each image improves consistency of the dataset. Intensity carries information about tissue properties, and normalization produces unified images for highly predictive models (Yildirim et al., 2022). The intensity of the sampled images are strictly within range [0,1], meaning that the dataset sourced from Kaggle has been preprocessed (Figure 2 left), and normalization like Gaussian normalization has subtle advantages on MRI images (Ellingson et al., 2012) is not required here.

Secondly, we conducted data augmentation, where flippings and Gaussian noise were introduced to transformed images. For Gaussian noise, a class AddGaussNoise configured the module torchvision.transforms.Compose. These improved the generalizability of neural networks. The pixel value distribution (Figure 2, right) and results images (Figure 3, lower) after adding Gaussian noise presents less skewness than those before (Figure 2, left; Figure 3 upper).

Figure 2: Pixel value distribution before (left) and after (right) adding noise

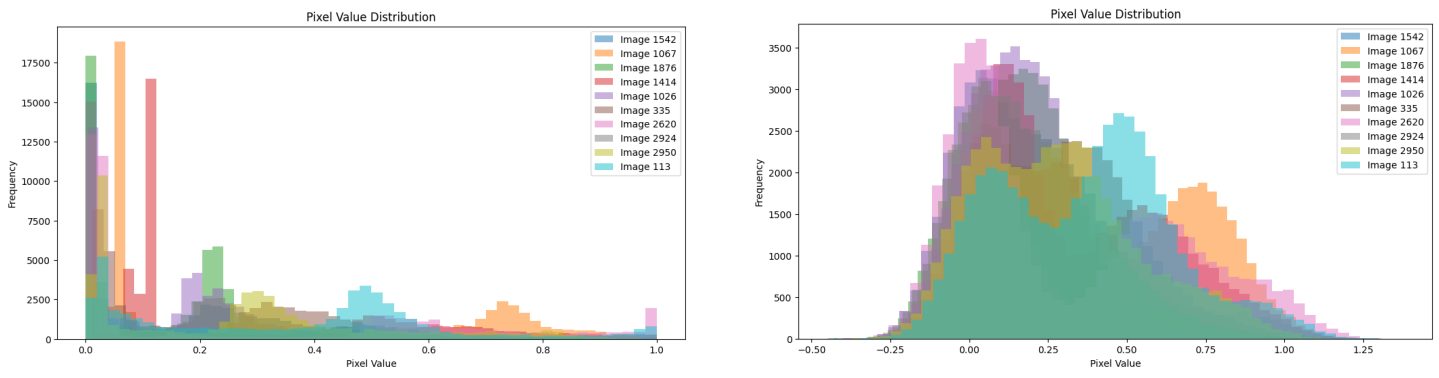
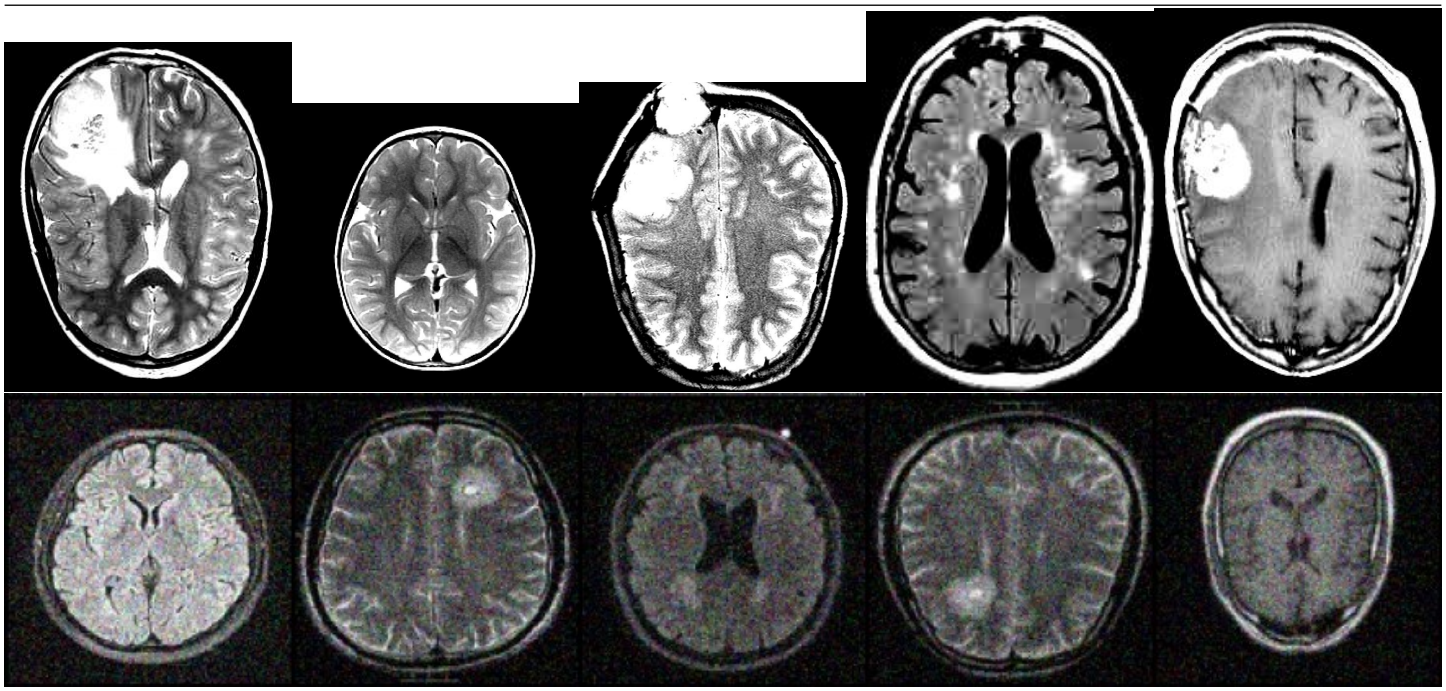


Figure 3: MRI images before preprocessing (upper) and after preprocessing (lower)



2.2 ConvNet

2.2.1 ConvNet Introduction

The first model we built is ConvNet, short for Convolutional Neural Network. We built the model architecture manually by defining a class object in Python, and the architecture and key components can be summarized as follows:

Feature Extraction

The feature extraction component forms the foundation of the model, responsible for capturing essential patterns and features from input images.

Key components within this segment include:

- Multiple Convolutional Layers: These layers utilize 3x3 kernels to perform convolution operations on the input data. The number of filters is determined by the net_width parameter, allowing for flexibility in controlling model capacity.
- Activation Functions: The model supports various activation functions, including 'Sigmoid,' 'ReLU' (Rectified Linear Unit), and 'LeakyReLU.' Activation functions introduce non-linearity into the network, enabling it to learn complex relationships in the data.
- Normalization Layers: Different normalization techniques, such as 'BatchNorm,' 'LayerNorm,' 'InstanceNorm,' 'GroupNorm,' and 'None,' can be applied. These layers enhance training stability and convergence by controlling the distribution of feature values.
- Pooling Layers: The model offers various pooling options, including 'MaxPooling,' 'AveragePooling,' and 'None.' Pooling layers downsample the feature maps, reducing their spatial dimensions while retaining essential information.

Classification Layer

- Following feature extraction, the output is flattened to create a vector of features.
- The model employs a fully connected (linear) layer for classification, where the number of output units corresponds to the specified num_classes parameter. This layer is responsible for making class predictions and determining the final classification results.

We load the dataset using the ImageFolder class and apply the following transformations:

(1) Resize images to a fixed size of 128x128 pixels using the 'transforms.Resize' operation.

(2) Apply random horizontal flipping to the images using 'transforms.RandomHorizontalFlip'. This operation creates mirror images to augment the dataset.

(3) Convert the images into PyTorch tensors with 'transforms.ToTensor'. This step prepares the data in a format suitable for machine learning.

(4) Add Gaussian noise to the images with parameters specifying a mean of 0.0 and a standard deviation of 0.1. This operation is custom and is designed to introduce controlled noise, following a Gaussian distribution $N(0, 0.1)$.

2.2.2 ConvNet Solution

To train the defined ConvNet and fine-tune the hyperparameters, we defined eight different combinations of hyperparameters as shown in the table below.

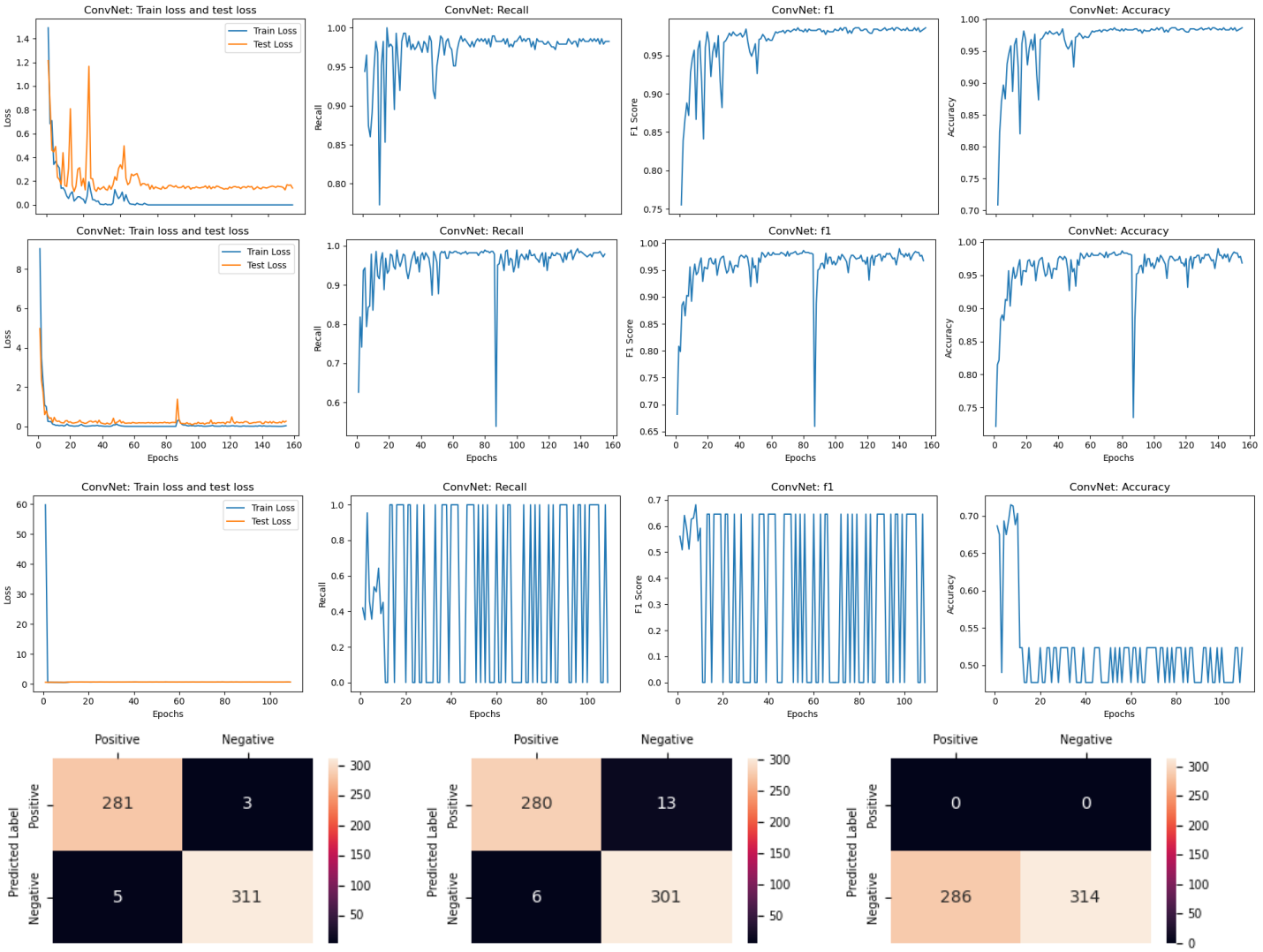
Table 1: Combinations of Hyperparameters (ConvNet)

No.	Learning Rate	Activation	Optimizer	Pooling
1	0.001	ReLU	Adam	Maxpooling
2	0.01	ReLU	Adam	Maxpooling
3	0.1	ReLU	Adam	Maxpooling
4	0.001	Sigmoid	Adam	Maxpooling
5	0.001	LeakyReLU	Adam	Maxpooling
6	0.001	ReLU	SGD	Maxpooling
7	0.001	ReLU	Adam	Avgpooling
8	0.001	ReLU	Adam	None

2.2.3 ConvNet Findings

First, we chose three different learning rates 0.001, 0.01 and 0.1and kept other hyperparameters the same (Model No. 1, 2, 3). We trained the model and generated a set of plots containing the train and test loss vs epochs, recall, f1 and accuracy score vs epochs, together with the confusion matrix for each model. The plots are shown as follows:

Figure 4: Metrics Visualization and Confusion Matrices of Combination No. 1, 2, 3 (ConvNet)



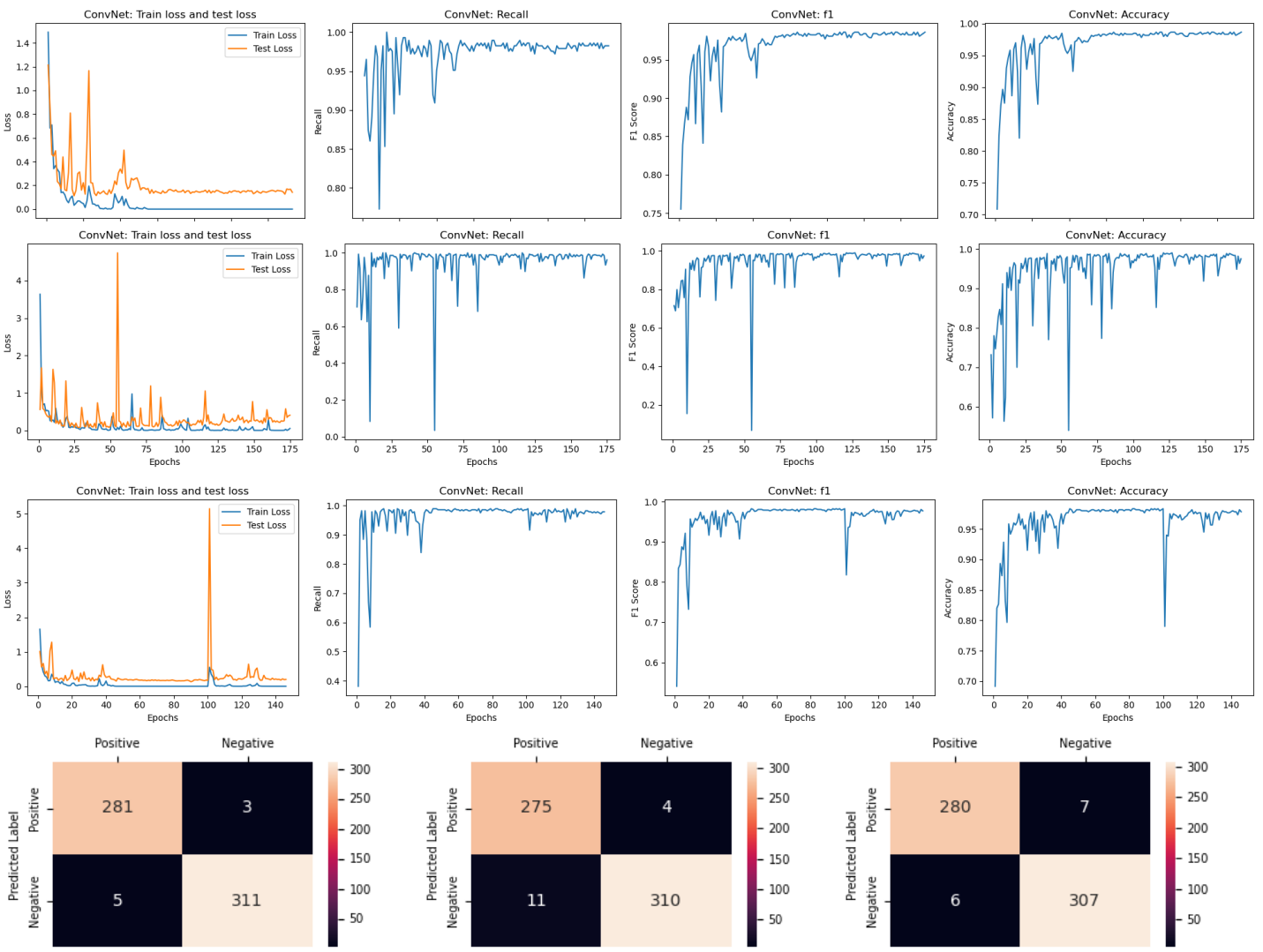
From the result shown above, as the learning rate increased from 0.001 to 0.1, The train and test loss converged much faster. However, the graphs of three metrics indicate that even if the convergence is faster, the recall, f1 and accuracy score did not return a satisfactory result. The three metrics converged to a range around 0.96 to 0.98 and did not increase further. Besides, the curves are not stable as compared to the learning rate equals 0.001 and there are more fluctuations as well. Note that there was an anomaly point shown in the graph with significantly low metrics scores. The reason is unknown yet and may be due to some randomness during training. As the learning rate increased to 0.1, the model was not actually "learning" at all. The graphs show that the recall and f1 score did not converge at all and the accuracy score converged to around 0.5, which is like random guesses. The confusion metrics tell the same story. The model No. 1 achieved the best performance with the lowest number of false predictions. Model No. 2 has a higher number of false positive cases as compared to No. 1. Model No. 3 contains only negative predictions which correlate with the results of metrics. The accuracy, recall and f1 score of model No. 1 are 0.987, 0.983 and 0.986.

The results can be explained as follows: A high learning rate may lead to overshooting the optimal model parameters and prevent the model from converging to an accurate solution (Chen et al. 2022). In this case, the learning

rate of 0.1 led to unstable training, preventing the model from effectively learning the underlying patterns in the data, and essentially making random predictions. On the other hand, a learning rate of 0.001, which is smaller and more gradual, allowed for a more stable convergence, resulting in a model (No. 1) that achieved significantly better performance in terms of accuracy, recall, and F1 score.

To further train the model and tune hyperparameters, we tried two different activation functions: "Sigmoid" and "LeakyReLU" and kept other parameters unchanged. The metrics plots and confusion matrices are shown as follows:

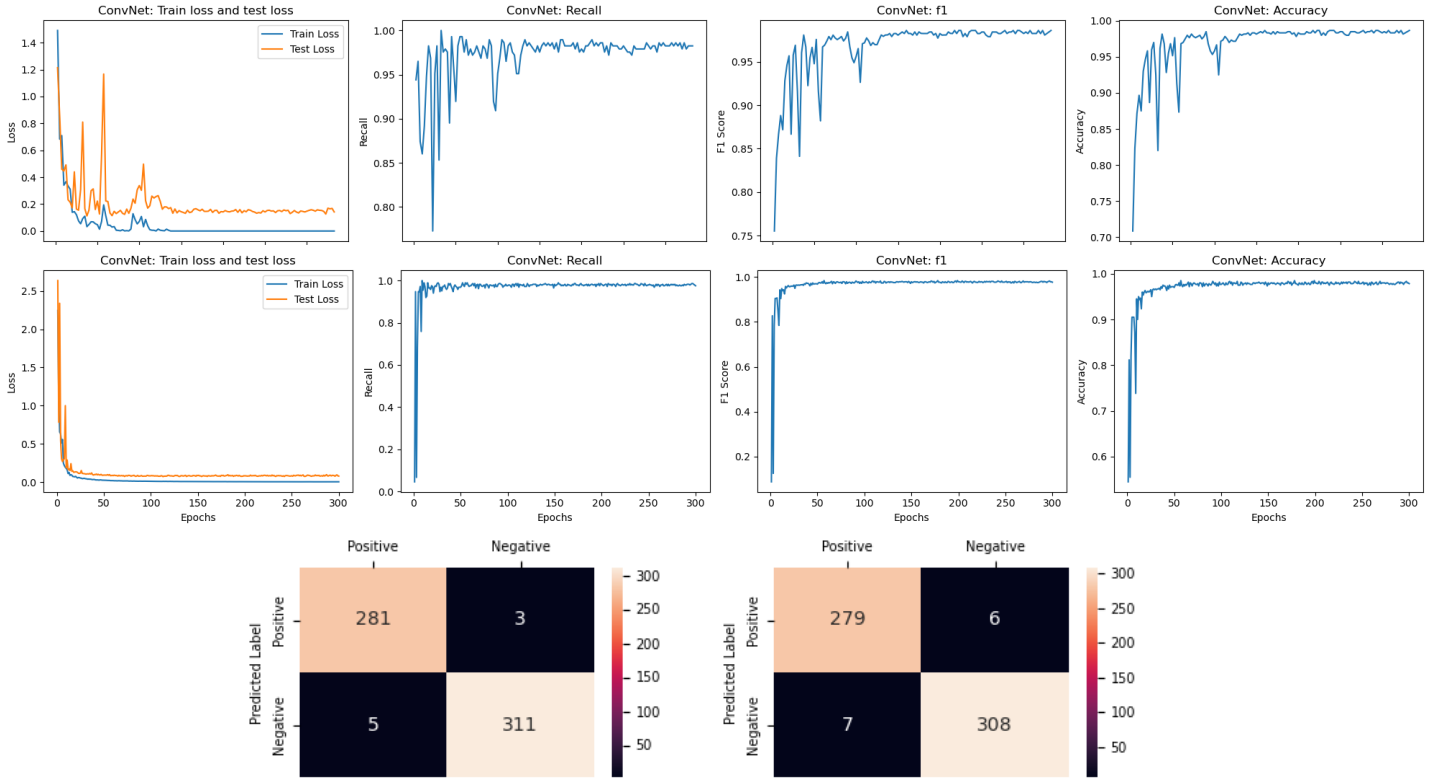
Figure 5: Metrics Visualization and Confusion Matrices of Combination No. 1, 4, 5 (ConvNet)



The line graphs show that the model with "ReLU" (No. 1) and with "LeakyReLU" (No. 5) has better performance than the model with "Sigmoid" activation function (No. 4). The two models converged more quickly and remained more stable with fewer fluctuations than model No. 4. Note that there is an anomaly point with low metric scores in model No. 5 which seemed to affect its stability to some extent. The confusion matrices tell the same story. Model No. 1 achieved the best performance among the three models with the lowest number of false prediction cases. Model No. 5 ranked after No. 1 and model No. 4 ranked the last. The accuracy, recall and f1 score of model No. 1 were the same as mentioned before.

Based on the results just mentioned, the "ReLU" activation function can be regarded as the most appropriate for the ConvNet model. To further tune the rest of the hyperparameters, we changed the optimizer to "SGD" and kept other parameters the same (model No. 6). As compared to model No. 1, the results are shown as follows:

Figure 6: Metrics Visualization and Confusion Matrices of Combination No. 1, 6 (ConvNet)

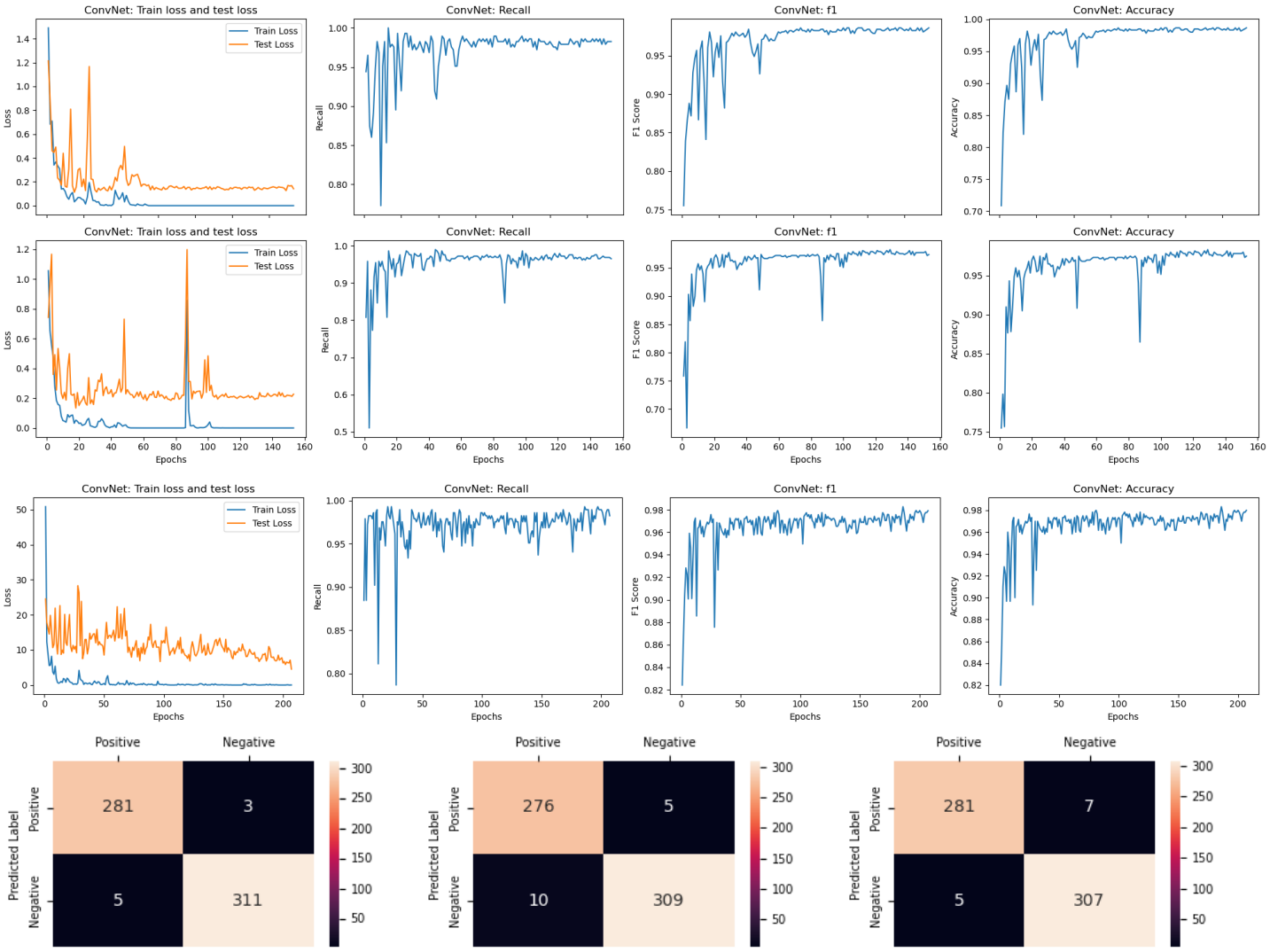


The results in the figure above show that when the activation function was set to "SGD", the metrics converged much quicker than model No. 1, and remained stable close to one with fewer fluctuations. However, the confusion matrices tell the opposite story, even if model No. 6 seems to have better convergence, it has a slightly higher number of false predictions than model No. 1. The metrics scores of model No. 1 are the same as mentioned above.

The explanations of the above observation are as follows: Convergence speed and stability of metrics during training do not always directly correlate with the model's ability to generalize and make accurate predictions. While model No. 6 may have converged quickly and appeared stable in terms of metrics, it might have fit the training data too closely, leading to overfitting and poorer generalization on unseen data. In this case, the "ReLU" activation function can be regarded as more appropriate for the ConvNet.

To finalize the model and find the best pooling methods, we tried two new approaches: "Average pooling" and without pooling. The plots of evaluation metrics and confusion matrices are shown as follows:

Figure 7: Metrics Visualization and Confusion Matrices of Combination No. 1, 7, 8 (ConvNet)



From the plots shown above, it can be found that model No. 1 achieved faster and more stable convergence of evaluation metrics. Model No. 7 has an anomaly point in the graphs while model No. 8 converged much slower than the other two models. The confusion matrices tell the same story. Model No. 1 achieved the best performance, followed by model No. 8 and model No. 7. The result indicates that the Maxpooling method can reduce the time of convergence and achieve better model performance. The evaluation metrics of model No. 1 are the same as mentioned above, and the "Maxpooling" method can be regarded as the most appropriate for ConvNet.

Summarize the results of all the comparisons above, model No. 1 achieved the best model performance in terms of accuracy, recall and f1 score. Model No. 1 is selected to be a candidate to compare with other neural net models.

2.3 DenseNet

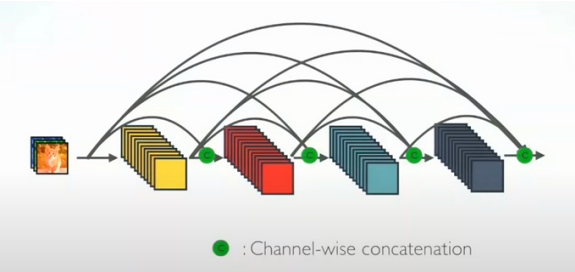
2.3.1 DenseNet Introduction

One key component of DenseNet is the dense block. A Dense Block is a fundamental building block in the DenseNet architecture, which is a deep neural network architecture designed for image classification tasks. The Dense Block is a novel approach to constructing deep convolutional neural networks, and it addresses the vanishing

gradient problem by establishing dense connections between layers within a block.

In a Dense Block, a typical structure consists of multiple convolutional layers grouped together (figure below). Each layer within the block takes as input the feature maps produced by all preceding layers in the same block. These feature maps are concatenated together, and the resulting tensor is passed to the subsequent layers within the block. This dense connectivity ensures that each layer has direct access to the features extracted by all the previous layers, creating a highly interconnected and feature-rich block (Pandey and Wang, 2020).

Figure 8: Dense Block in DenseNet



The key components of a Dense Block are as follows:

- Batch Normalization: Used to normalize the inputs to each layer to speed up training and improve convergence.
- ReLU (Rectified Linear Unit) Activation: Applied to the output of each layer to introduce non-linearity.
- Convolutional Layers: Convolutional operations are performed to extract features from the input feature maps.

Other key characteristics of the DenseNet architecture include:

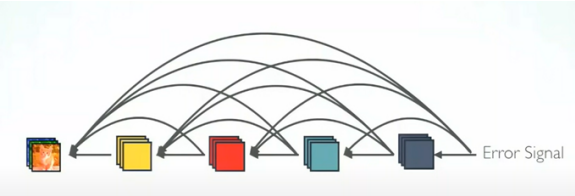
- Transition Layers: Transition layers are used to control the spatial dimensions of feature maps and reduce the number of channels, which helps manage the computational complexity of the network.
- Global Average Pooling: Instead of using fully connected layers at the end of the network, DenseNet typically uses global average pooling to reduce the number of parameters and improve model generalization.
- Bottleneck Layers: In many versions of DenseNet, bottleneck layers are used, which consist of a 1x1 convolution layer followed by a 3x3 convolution layer. These bottleneck layers help in reducing the number of input channels to the 3x3 convolution layer, which saves computational resources.

The DenseNet architecture consists of multiple dense blocks and transition layers, and it offers various configurations to suit different computational and accuracy requirements. DenseNet has demonstrated strong performance on a variety of image classification tasks and is known for being highly parameter efficient.

As compared to the previously introduced ResNet, The Advantages of DenseNet are as follows:

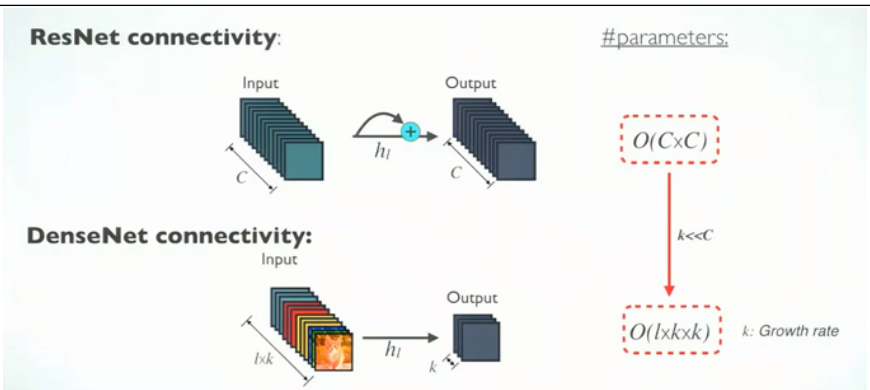
(1) **Strong Gradient Flow and Implicit "Deep Supervision":** DenseNet's dense connectivity pattern provides strong gradient flow and implicit deep supervision. The error signal can be easily propagated to earlier layers more directly, addressing the vanishing gradient problem. This feature is akin to implicit deep supervision, allowing earlier layers to receive direct supervision from the final classification layer (Dolz et al. 2018).

Figure 9: Implicit "Deep Supervision"



(2) **Parameter & Computational Efficiency:** Compared to other architectures like ResNet, DenseNet excels in parameter and computational efficiency. In ResNet, the number of parameters is directly proportional to $C \times C$ (the number of channels), whereas in DenseNet, it is directly proportional to $l \times k \times k$ (the number of layers and the growth rate) (figure below). Given that the growth rate (k) is typically much smaller than C, DenseNet results in a significantly smaller model size, making it more memory and computation-friendly (Huang et al. 2016).

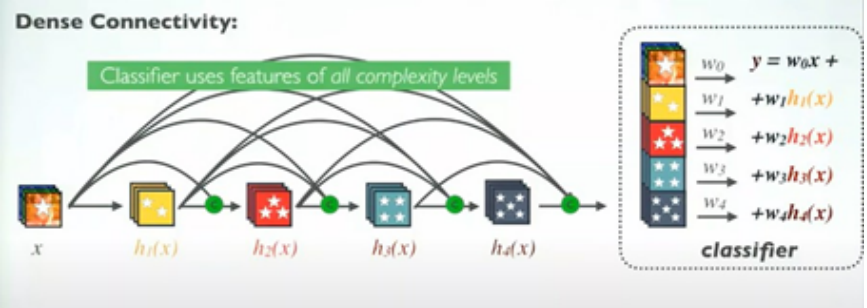
Figure 10: Parameters for ResNet and DenseNet



(3) **More Diversified Features:** DenseNet's unique dense connectivity ensures that each layer receives inputs from all preceding layers. This results in more diversified features within the network, encouraging the extraction of richer and more complex patterns, which enhances the model's capability to capture various image characteristics.

(4) **Maintains Low Complexity Features and Smooth Decision Boundaries:** DenseNet leverages features of all complexity levels (figure below). This tendency to use features of varying complexities leads to more smooth decision boundaries. It explains why DenseNet performs well, even when the training data is limited or insufficient. The ability to utilize low-complexity features can be particularly valuable in scenarios with limited training data.

Figure 11: DenseNet



In summary, DenseNet's dense connectivity pattern, strong gradient flow, parameter efficiency, diverse feature extraction, and the utilization of low complexity features contribute to its effectiveness in image classification tasks and have made it a favoured choice for deep learning practitioners. These advantages are especially pertinent in situations where computational resources are limited, and rich feature representations are essential for success.

2.3.2 DenseNet Solution

The data was loaded and preprocessed in the same way as described in the ConvNet Implementation. Based on prior knowledge of DenseNet, we set a series of combinations of hyperparameters as shown in the table below. We train the DenseNet model from scratch on each set of hyperparameters with 300 epochs, implementing an early stopping strategy. The model will stop training when the training loss does not drop more than $10-3$ in the last 100 epochs. We generated graphs to visualize the changes in training loss and test loss, recall accuracy and F1 score through each epoch. We also created confusion matrices to compare the model performance under different combinations of hyperparameters.

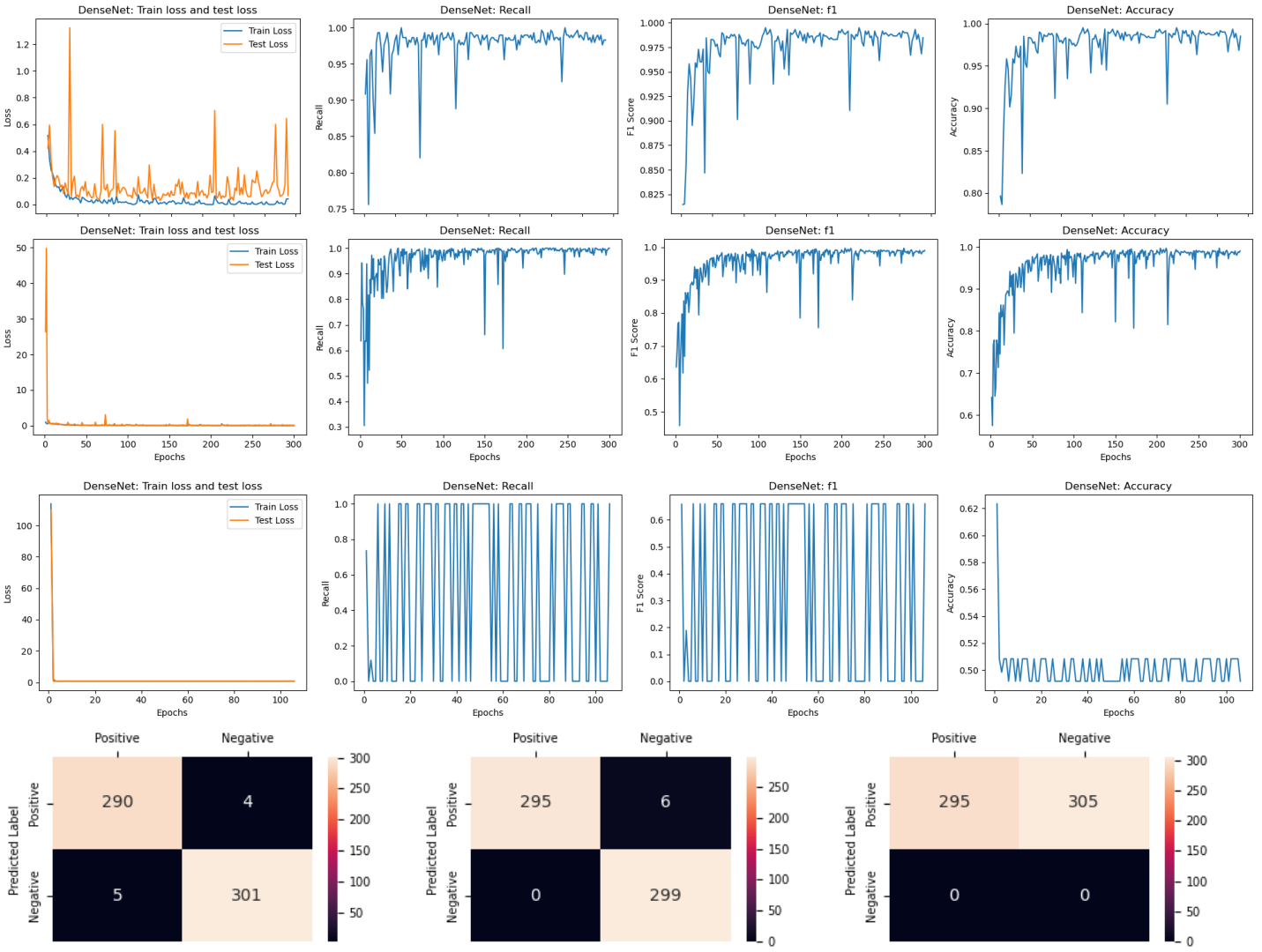
Table 2: Combinations of Hyperparameters (DenseNet)

No.	Learning Rate	Dropout	Activation	Optimizer
1	0.001	0.5	ReLU	Adam
2	0.01	0.5	ReLU	Adam
3	0.1	0.5	ReLU	Adam
4	0.001	0.3	ReLU	Adam
5	0.001	0.1	ReLU	Adam
6	0.001	0.5	Sigmoid	Adam
7	0.001	0.5	Tanh	Adam
8	0.001	0.5	ReLU	SGD

2.3.3 DenseNet Findings

The graphs of metrics and confusion matrices of No. 1, 2 and 3 are shown as follows:

Figure 12: Metrics Visualization and Confusion Matrices of Combination No. 1, 2, 3 (DenseNet)



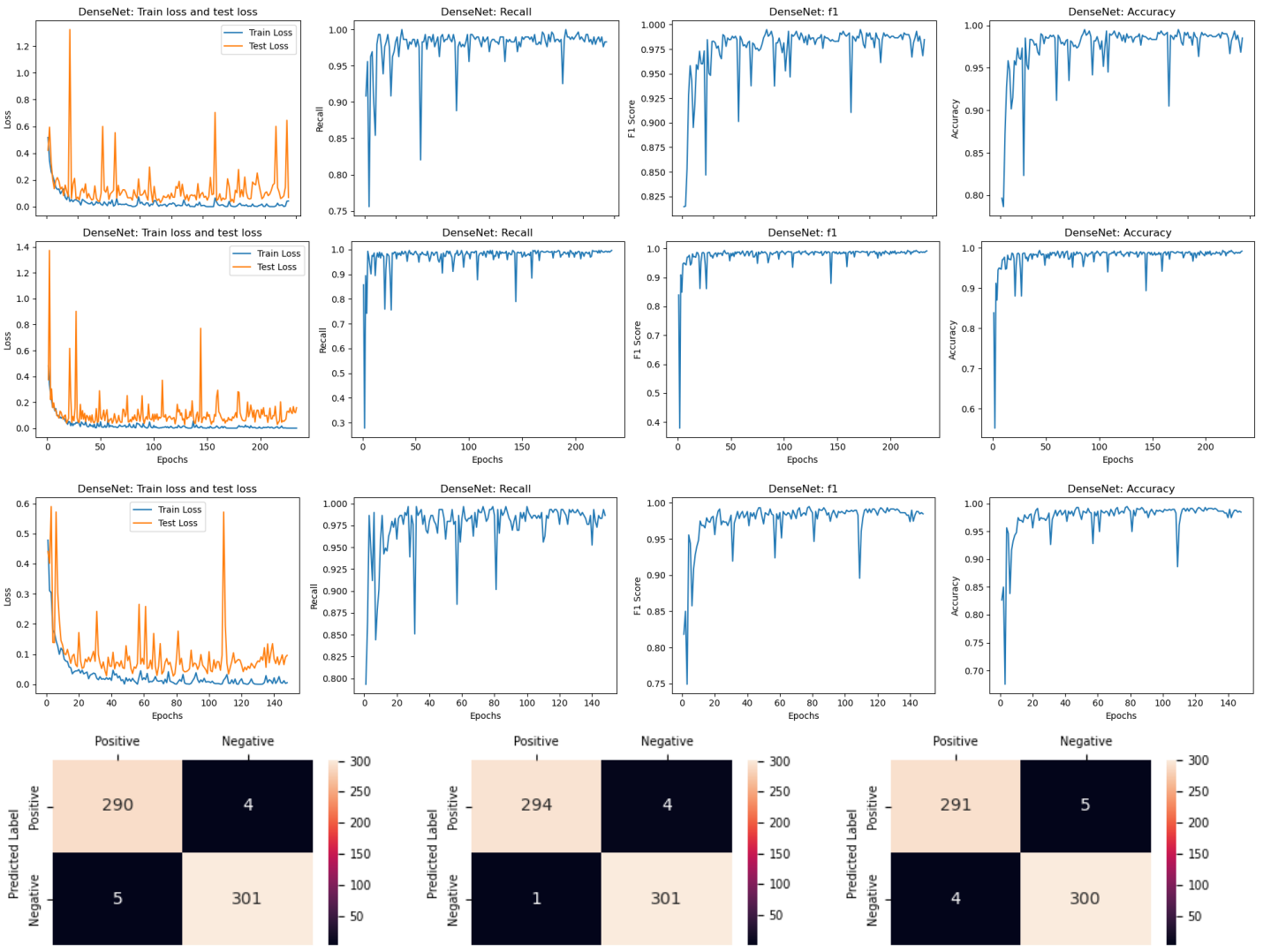
From the result shown above, as the learning rate increased from 0.001 to 0.1 and other hyperparameters remained the same, the training loss and test loss decreased dramatically and remained steadily close to zero. The recall, f1- score and accuracy converged faster close to one and there were fewer fluctuations at the same time. Based on the confusion matrices, as the increment of learning rating, the number of false negative cases decreased from five to zero, which is a good sign since the brain tumor detection models need to lower the false negative cases as much as possible. The model's accuracy score increased from 0.985 to 0.99, recall increased from 0.983 to 1.0 and the f1 score increased from 0.985 to 0.989. The increment of all the metrics indicates the improvement of model performance. However, as the learning further increased from 0.01 to 0.1, the plots indicate that the model is not "learning" anything. The recall and f1 fluctuate between zero and one, and the accuracy score is close to 0.5, which is nearly random guesses. The confusion matrix further proved that, with all test data predicted positive cases.

The changes in the observed pattern in the model's performance can be explained as follows: as the learning rate increases from 0.001 to 0.1, it allows the model to converge faster, resulting in lower training and test losses and improved metrics such as recall, F1 score, and accuracy. This suggests that a higher learning rate may help the

model learn more efficiently. However, when the learning rate is increased further to 0.1, the model experiences a sharp decline in performance, as indicated by erratic metrics and confusion matrices. This could be attributed to the learning rate becoming too large, causing the model to overshoot the optimal solution and making it difficult to converge. The optimal learning rate lies in a delicate balance, and this observation underscores the significance of hyperparameter tuning and the need to avoid extremes in hyperparameter values to ensure model stability and performance.

To further tune the hyperparameters, we set the dropout value to be 0.5, 0.3, 0.1 and keep other parameters the same, recorded in table as No. 1, 4 and 5. The graphs of metrics and confusion matrices these three combinations are shown as follows:

Figure 13: Metrics Visualization and Confusion Matrices of Combination No. 1, 4, 5 (DenseNet)



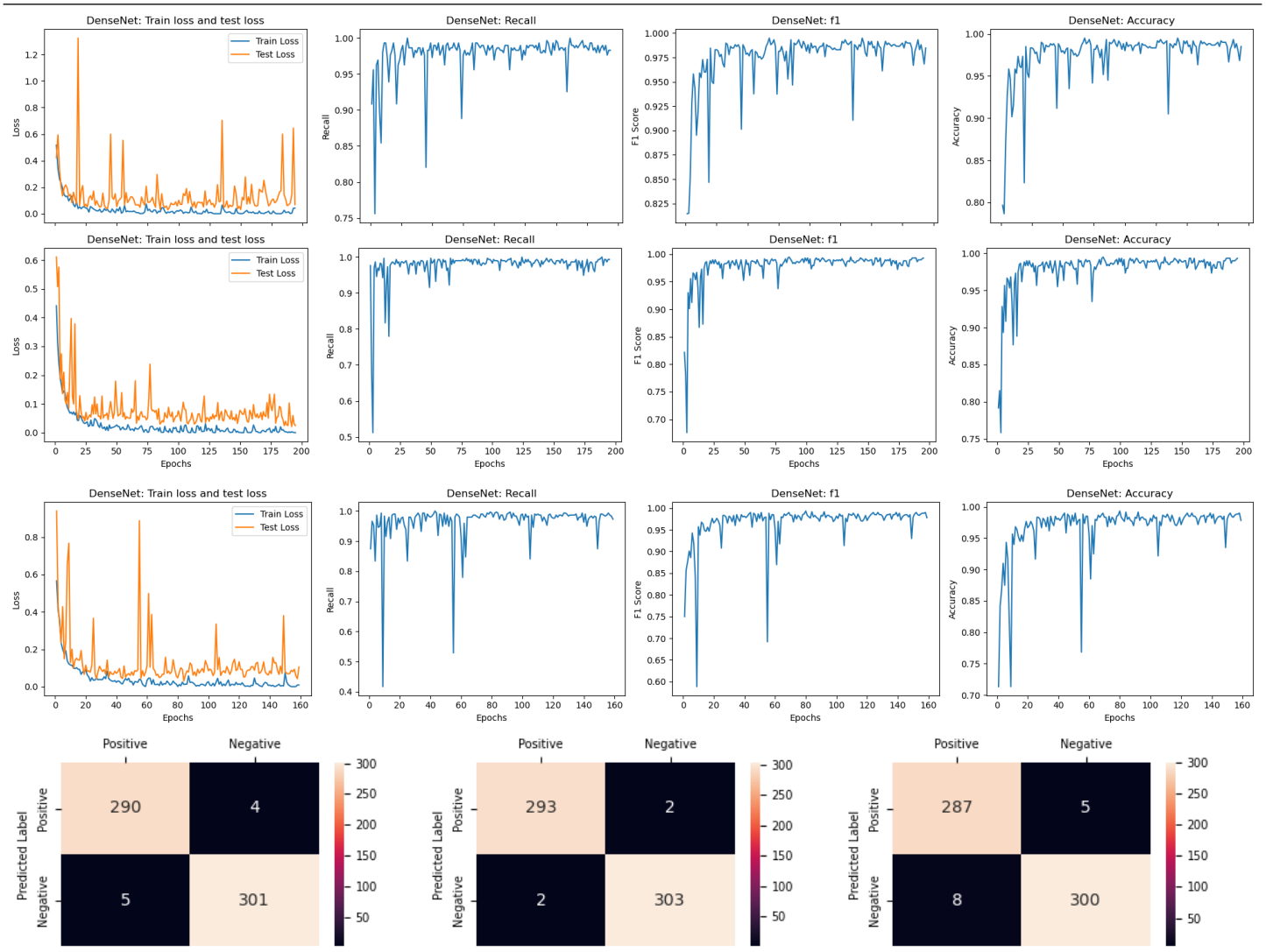
As the visualization results shown above, when the dropout value decreases from 0.5 to 0.3, although all those metrics converge fast for both settings, there are much fewer fluctuations and the recall, f1 and accuracy remain more steadily close to one. The confusion matrix tells the same story, the number of false negative cases decreased from five to one and the number of false positive cases remained the same. The accuracy score increased from 0.985 to 0.991, the recall increased from 0.983 to 0.997 and the f1 score increased from 0.985 to 0.992. The result indicates

that the model performance increased the dropout value decreased from 0.5 to 0.3. However, as the dropout further decreased from 0.3 to 0.1, the result shows that there are few improvements as compared to the result of No.1.

The observed trends in the model's performance based on varying dropout values shows the trade-off involved in selecting the appropriate dropout rate. When the dropout value decreased from 0.5 to 0.3, there was a noticeable improvement in the model's stability and performance, with reduced fluctuations and higher values for metrics like recall, F1 score, and accuracy. This implies that a moderate dropout rate helped in regularizing the model and improving its ability to generalize. However, when the dropout rate was further decreased to 0.1, the model's performance did not have substantial improvement, which indicates that excessively low dropout rates might not provide any additional benefit.

To further tune the hyperparameters, we tried two new activation functions Sigmoid and Tanh as comparison to the ReLU function, and keep other parameters the same, recorded in the table above as No. 1, 6 and 7. The graphs of metrics and confusion matrices these three combinations are shown as follows:

Figure 14: Metrics Visualization and Confusion Matrices of Combination No. 1, 6, 7 (DenseNet)



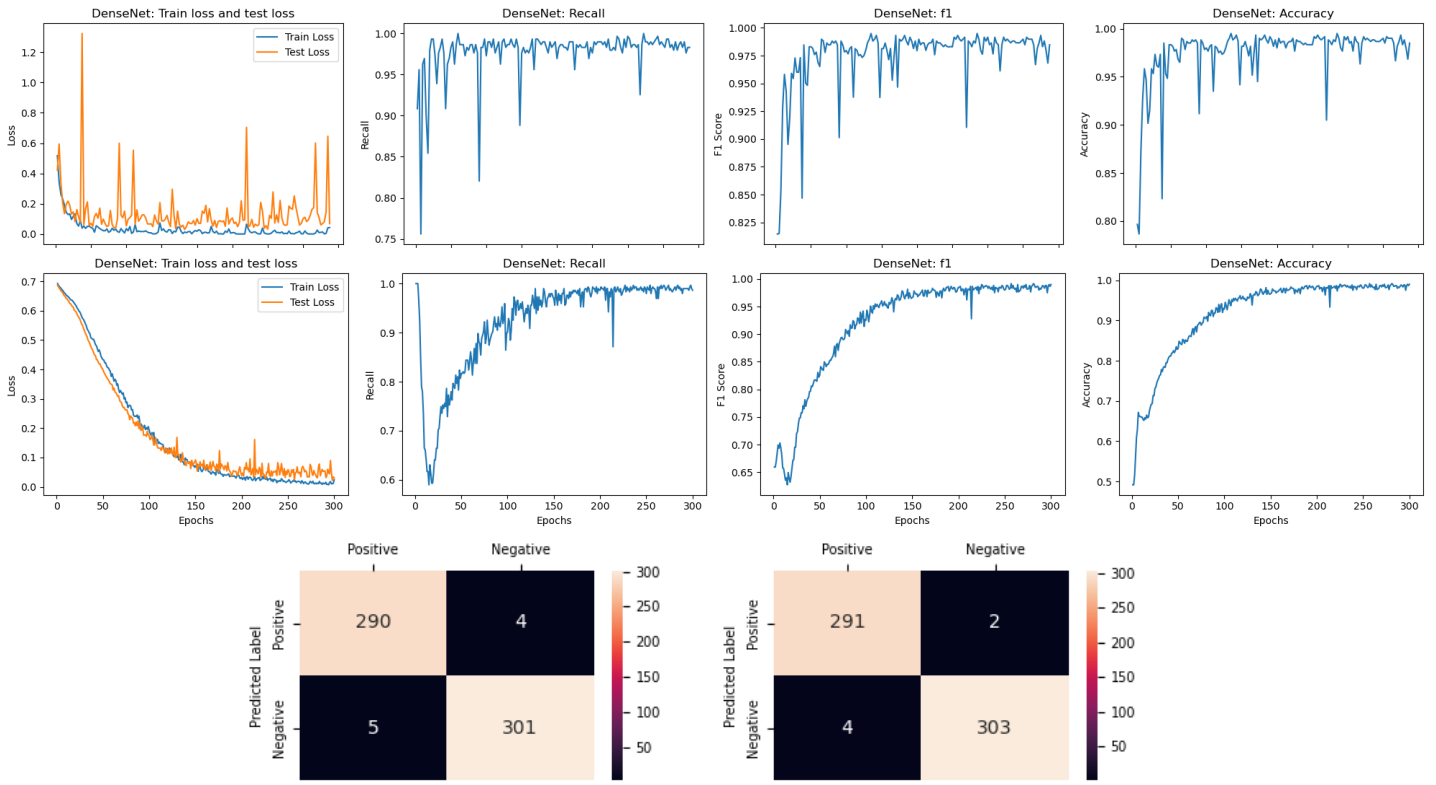
From the results shown above, the model achieved the best performance when the activation function was set to be

"Sigmoid". The metrics converged faster and remained steadier with fewer fluctuations. The confusion matrices tell the same story. The model with the "Sigmoid" activation function has the lowest number of false positives and false negatives. The models' performance can be ranked as follows: Sigmoid > ReLU > Tanh. The calculated accuracy, recall and f1 of No.6 with Sigmoid function are 0.993, 0.993, 0.993. The result indicates that model No. 6 achieved the best performance.

The observation can be explained as follows: The advantage of the Sigmoid activation function can be attributed to its ability to squash the model's outputs into a bounded range, making it well-suited for binary classification tasks like brain tumor detection. It promotes stable and rapid convergence, as evidenced by the metrics and confusion matrices, which demonstrate fewer false positives and false negatives in this project implementation.

To further tune the hyperparameters, we set a new optimizer to be 'SGD' which is short for Stochastic Gradient Descent, and keep other parameters the same, recorded in table above as No. 1 and 8. The graphs of metrics and confusion matrices of these three combinations are shown as follows:

Figure 15: Metrics Visualization and Confusion Matrices of Combination No. 1,8 (DenseNet)



From the results shown above, changing the optimizer resulted in a significant change in the learning pattern. The model with SGD optimizer converged much slower than the model with Adam optimizer. However, after convergence, the metrics have much less fluctuation. The confusion matrices show that the model with SGD has slightly better performance with a slightly lower number of false positive and false negative cases. The calculated accuracy, recall and f1 of No.8 with SGD optimizer are 0.99, 0.986 and 0.99.

The result shows a potential trade-off between convergence speed and final stability in model training. The "SGD"

optimizer displayed a slower convergence rate compared to "Adam." However, the metrics for the SGD-optimized model exhibited greater stability and fewer fluctuations once convergence was achieved. The advantage of the SGD optimizer can be seen in the slightly improved performance, characterized by lower numbers of false positive and false negative cases in the confusion matrices.

Recall that we are building a brain tumor detection model, our aim is to find a model which can achieve a higher score on both accuracy and recall. We expect the model can classify those MRI images correctly as well as lower the number of false negative cases as much as possible. Due to these requirements, even if a couple of DenseNet models achieved satisfactory metrics scores. We decided on DenseNet model No. 2 as the candidate since it has achieved a high accuracy score (0.99) and zero false negative cases.

2.4 ResNet

2.4.1 ResNet Introduction

ResNet is a kind of deep neural network (DNN) featured by recursive structure via residual connections. ResNet aims to alleviate not only gradient issues but performance degradation on training data as the network gets deeper. The issue is caused by fitting identity mappings rather than overfitting (He et al., 2015). Specifically, if $h(x)$ is the model to be learned, we don't find a DNN $f(x)$ approximating $h(x)$ directly, but an $f(x)$ approximating the $h(x) - x$ such that

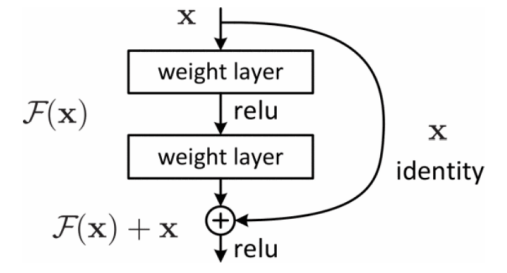
$$h(x) = x + (h(x) - x) = x + f(x),$$

and the approximation process is done recursively:

$$x_i = x_{i-1} + f_i(x_{i-1}).$$

In addition, ResNet is composed of numerous residual units, and the structure of a residual unit is shown in Figure 16 (He et al., 2015).

Figure 16: One residual unit



2.4.2 ResNet Solution

ResNet-18 is composed of 17 convolutional layers and 1 Softmax output layer, with 2 pooling layers. The latter 16 convolutional layers are partitioned into 8 residual blocks, each of which has two layers. Within each block, batch normalization is conducted right after convolution. This is followed by skip connection, where the input x is added to the output of batch normalization, which is followed by ReLU activation. The structure of ResNet-18 is shown in Figure 17 from Li (2022). Furthermore, we made modifications to the output layer to fit the binary classification of brain MRI images, from a fully connected layer to a new sequence which includes three linear layers with ReLU activations and Dropout in between. The modified parameters are shown in Table 3.

Figure 17: The structure of original ResNet-18

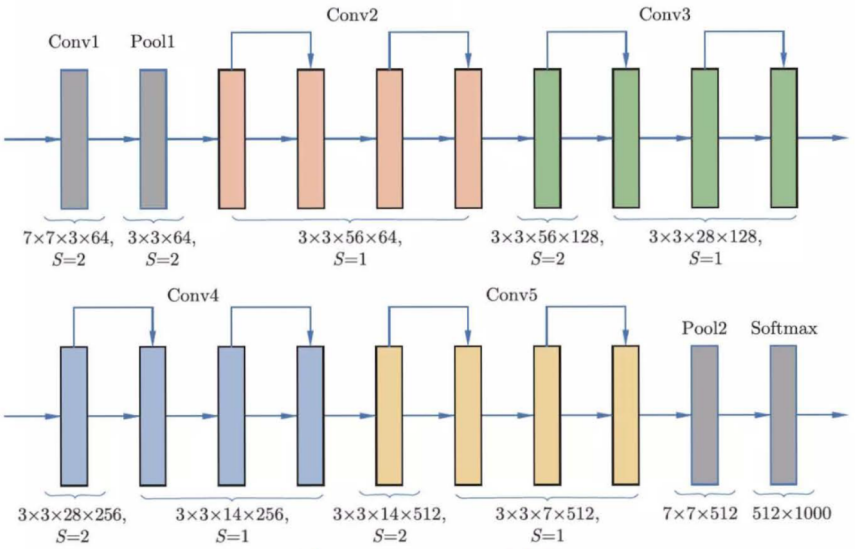


Table 3: The parameters of modified ResNet-18

Layer	Hyperparameter	Output size
Conv1	F = 7x7x3, S = 2	112x112x64
Pool1, Max	F = 3x3x64, S = 2	56x56x64
Conv2.1, Conv2.2, Conv2.3, Conv2.4	F = 3x3x3, S = 1	56x56x64
Conv3.1	F = 3x3x56, S = 2	28x28x128
Conv3.2, Conv3.3, Conv3.4	F = 3x3x28, S = 1	28x28x128
Conv4.1	F = 3x3x28, S = 2	14x14x256
Conv4.2, Conv4.3, Conv4.4	F = 3x3x14, S = 1	14x14x256
Conv5.1	F = 3x3x14, S = 2	7x7x512
Conv5.2, Conv5.3, Conv5.4	F = 3x3x7, S = 1	7x7x512
Pool2, Mean	F = 7x7x512	1x1x512
Output: (FC+ReLU+dropout)×2+FC	$W_1 = 512 \times 128,\ W_2 = 128 \times 32,\ W_3 = 32 \times 2$	2x1

For the training process, it was set to go through 300 epochs with early stopping. Besides, parameter tuning was involved, where 9 sets of parameters were used, covering learning rates (0.001, 0.01, 0.1), dropout (0.1, 0.3, 0.5), activation function (ReLU, sigmoid, tanh), and optimizer (Adam, SGD, RMSProp). The training for each set outputs performance plots (performance versus epochs), confusion matrix and training log.

2.4.3 ResNet Findings

Parameter set 5, from Table 4, evidences the highest accuracy 0.9883 and the lowest test loss 0.0576, with learning rate = 0.01, drop out = 0.5, sigmoid activation, and SGD optimizer. The setting makes the model well generalized, striking a balance between precision and recall. A high recall means cases where an MRI image with tumor is predicted as no tumor are rare, which is cardinal for clinical application and the lifelong wellbeing of patients.

For learning rate and dropout, it seems that learning rate = 0.01 is the sweet spot for the horizontal MRI dataset,

resulting in the best and the second best f1 scores from sets 5 and 4. On average, the highest learning rate 0.1 is the worst regarding accuracy and test loss. On average, a dropout of 0.1 has the highest performance, meaning that the model is less likely to overfit, even when fewer samples are dropped.

Activation functions and optimizers. Sigmoid stands out in modified ResNet-18, which is again supported by set 5 and is within expectation. It is intuitive that compared to ReLU, sigmoid and tanh are more suitable for binary classification problems, for their binary property when taking a limit of both sides of infinity. For optimizers, SGD outperforms Adam and RMSProp. Adam has more stable performance, regardless of changes in the learning rate and the dropout, than SGD and RMSProp.

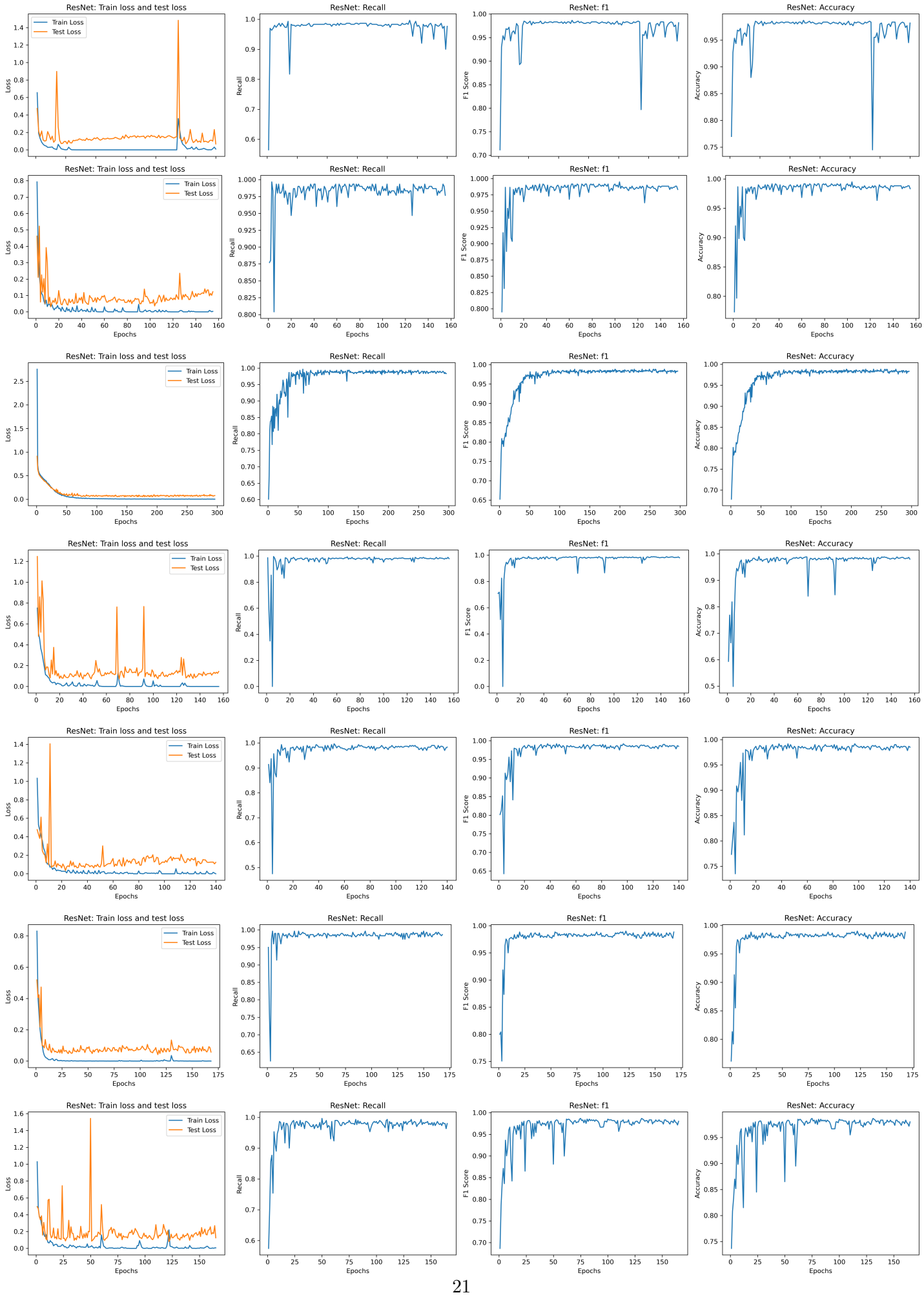
In addition, Table 4 shed light on the tradeoff between efficiency and performance. Parameter set 8 is the most time-efficient at 23.267 minutes, but without the best performance metrics. Set 5, while not the fastest, strikes the balance between performance and efficiency. Moreover, the dispersed epochs indicate the importance of parameter tuning.

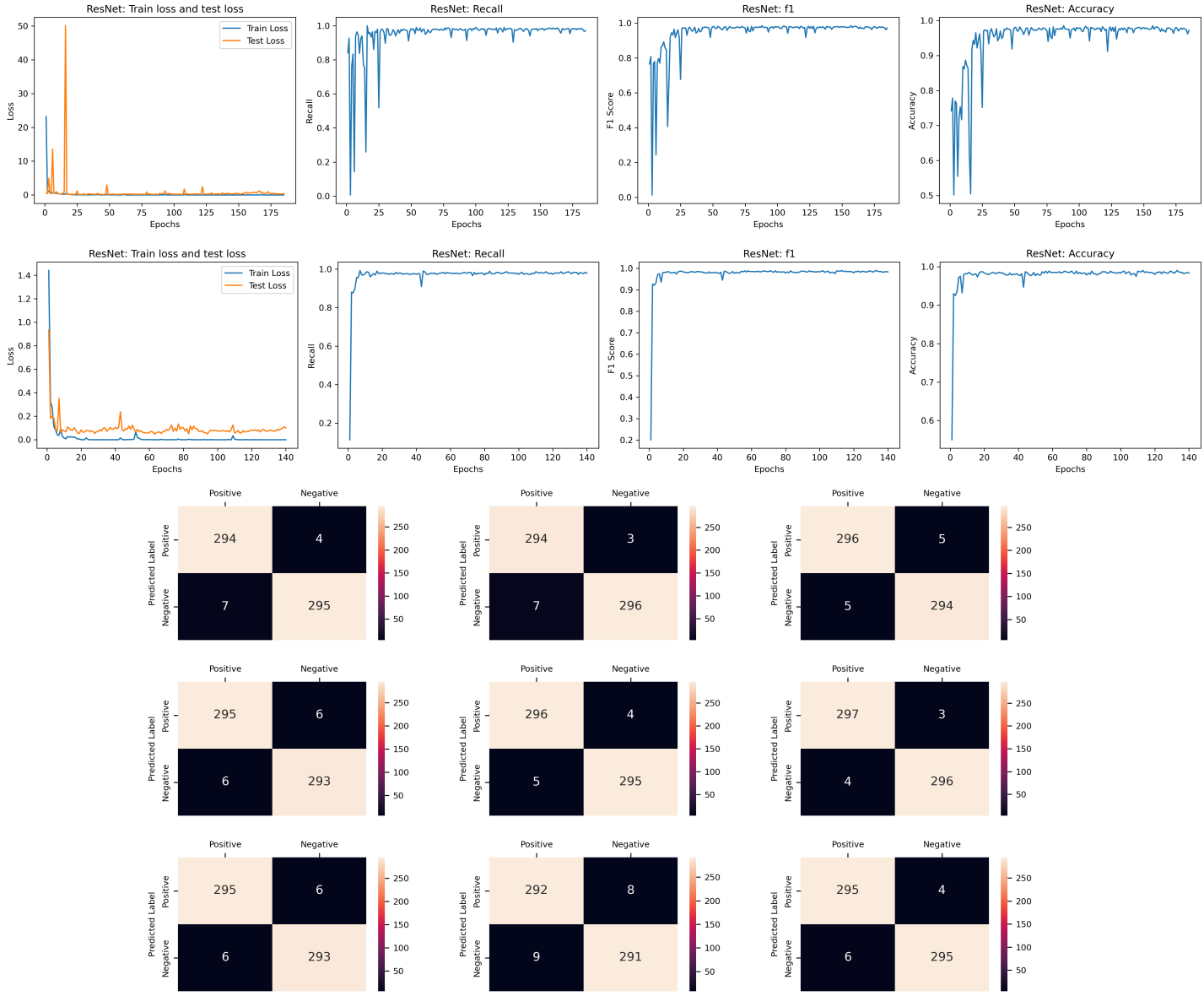
Table 4: Modified ResNet-18: Sets of Hyperparameters, Performance Metrics, and Costs

No.	**Hyperparameters**				**Performance metrics**				**Costs**	
	lr	**dropout**	**activ**	**optim**	**test loss**	**recall**	**f1**	**acc**	**epoch**	**time (mins)**
0	0.001	0.5	relu	adam	0.0674	0.9767	0.9816	0.9817	120	20.083
1	0.001	0.3	relu	rmsprop	0.1228	0.9767	0.9833	0.9833	155	25.800
2	0.001	0.1	relu	sgd	0.0775	0.9834	0.9834	0.9833	296	49.483
3	0.01	0.3	tanh	adam	0.1440	0.9801	0.9801	0.9800	156	26.000
4	0.01	0.1	relu	rmsprop	0.1235	0.9834	0.9850	0.9850	140	23.367
5	0.01	0.5	sigmoid	sgd	0.0576	0.9867	0.9884	0.9883	168	27.850
6	0.1	0.1	relu	adam	0.1251	0.9801	0.9801	0.9800	165	27.417
7	0.1	0.5	tanh	rmsprop	0.3650	0.9701	0.9717	0.9717	185	31.017
8	0.1	0.3	sigmoid	sgd	0.1013	0.9801	0.9833	0.9833	140	23.267

From Figure 18, it is well established that the model performs decently in terms of test loss, recall, f1 and accuracy. The training losses decrease smoothly, meaning that the model keeps learning for the train set. Spikes in test loss occur in sets 0, 3, 6 and 7, the majority of which have an Adam optimizer. The reasons may fall in the pixel values outside [0, 1] due to adding Gaussian noise, and adaptive learning rates in Adam, where some parameters were adjusted more than aggressively. Nevertheless, the spikes are not a fatal issue in case of early stopping which makes all the metrics converge at a satisfactory level finally.

Figure 18: Modified ResNet-18: Performance plots for each set of parameters (same sequence as in Table 4)





3 Transfer Learning on VGG16

3.1 VGG16 Introduction

VGG16 originated from the Visual Geometry Group of the University of Oxford and stands as one of the benchmarks of CNN architectures. The reasons why VGG16 stands out in CNN are below.

(1) Depth: VGG can boost the achievable depth of the network without significantly raising the number of parameters. The development of VGG established the stage for deeper architecture like ResNet.

(2) Uniformity: A consistent 3x3 filter is utilised throughout the network. The uniform approach constructed a cleaner design and the tuning of hyperparameters easier.

(3) ImageNet Pre-training: VGG16 is available as a model with pre-trained weights from ImageNet, so it is more straightforward to initialize the network. Using a pre-trained VGG16 model facilitates transfer learning, offering improved generalization and faster training, especially in situations with limited data.(datagen, 2023)

In the implementation section, the report demonstrated three different training strategies.

(1) Pre-trained with unfrozen layers

(2) Pre-trained with frozen layers

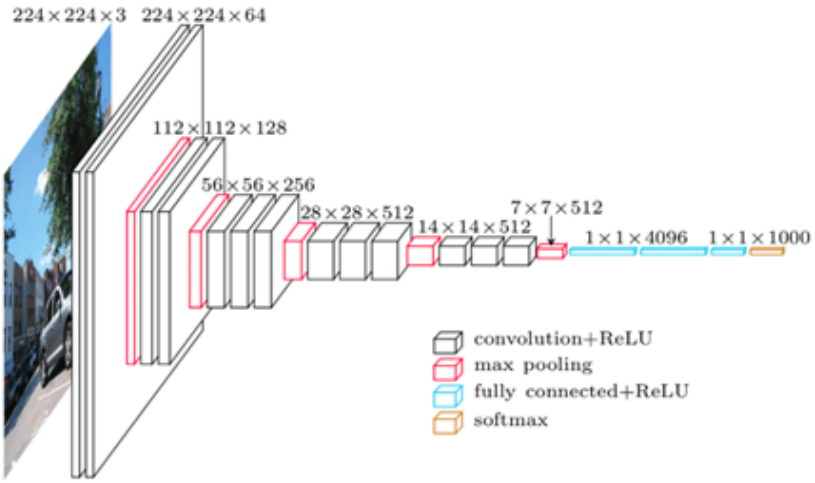
(3) Train from scratch

VGG16 requires input images of 224x224 size, because it was originally trained on images of that size, so it is essential to resize the input image. The train-from-scratch version is based on the VGG16's architecture, which is introduced below.

3.2 VGG16 Architecture

All the features are built on the unique VGG16 architecture, which comprises 16 layers including 13 convolutional layers followed by 3 fully connected layers. VGG16 utilizes the deeper architecture to focus on hierarchical feature extraction across multiple layers. Each layer's features are introduced as below. (datagen, 2023)

Figure 19: The architecture of VGG16



(1) Convolutional Layers: 3x3 convolutional filters allow the model to capture spatial hierarchies and patterns in input data. Multiple 3x3 convolutional layers are stacked sequentially, increasing the depth of the network, and its capacity to learn complex features.

(2) Max-Pooling layers: It is utilized with certain convolutional layers to decrease the spatial dimensions of the output volume, thereby decreasing the computational complexity.

(3) Dropout layers (optional): It is introduced with fully connected layers as a regularisation method to reduce overfitting.

(4) Fully Connected layers: Following the convolutional layers, the architecture has fully connected layers to help in making final predictions. In the implementation section, the report built its own fully connected layers with two classification outputs.

(5) Softmax layer: The final layer that uses a softmax activation to produce probability distributions. In the implementation, the softmax layer is internally applied in the loss function. (nn.CrossEntropyLoss)

(6) Activation Function: It is common for VGG16 to use ReLU (Rectified Linear Unit) activation function to introduce non-linearity without affecting the receptive properties of convolutional layers. In the implementation section, the report also tried to tune with different activation functions such as Tanh and Sigmoid.

3.3 VGG16 Findings

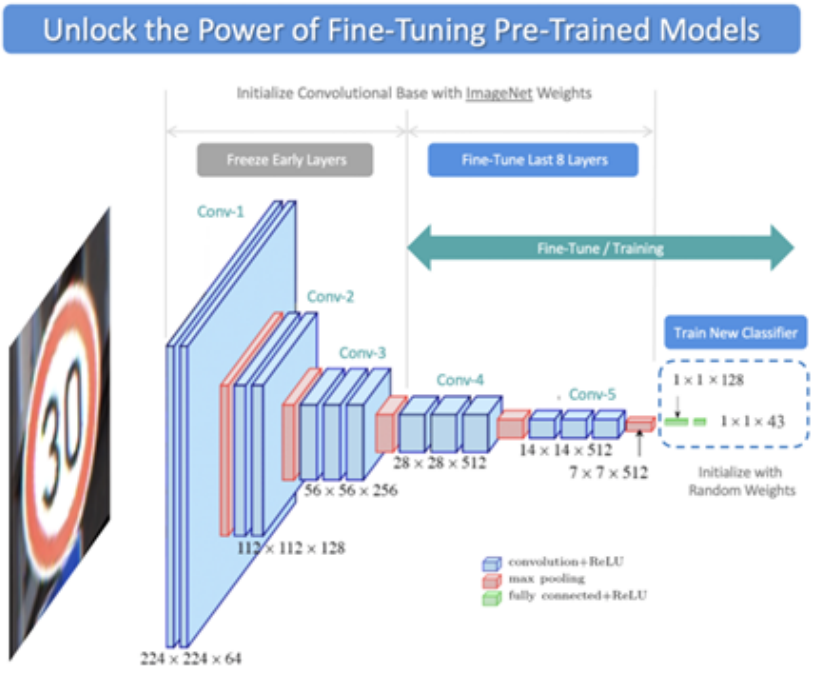
The report performed three major versions with transfer learning (pre-trained) or from scratch. In this section, we discuss the difference between each version and their implementation.

3.3.1 Version 1: VGG16 with pre-trained weights, unfrozen layers and an additional classifier

Version 1 utilizes the weights pre-trained by ImageNet, a massive image dataset, so it has an enormous amount of prior weights about various types of images. The model can converge in a short amount of time based on trained features, and this method of using prior knowledge is also called transfer learning. The concept for this version is to initialize a convolutional base with ImageNet weights, and add a new 'Yes' and 'No' customized classifiers in the end. All parameters in layers can be trainable.

- Load VGG16 with pre-trained weights. (transfer learning)
- Modify the classifier by removing the last fully connected layers and appending new layers for the final Yes and No classifier. (class_num = 2)
- Set parameters in the layers to be trainable. (requires_grad=True)(unfrozen)

Figure 20: The training structure of VGG16 (This figure contains the fine-tuning and training, but it does not equal to actual training structure in this report)(BIG VISION LLC, 2023)



3.3.2 Version 2: VGG16 with pre-trained weights, frozen layers and an additional classifier

It provides a different approach towards leveraging the pre-train weights. In this version, the pre-trained layers are frozen, so their weights can not be updated during training. (BIG VISION LLC, 2023)

- Load VGG16 with pre-trained weights. (transfer learning)
- Discard the existing fully connected layers, and create a new fully connected layer ourselves with a pooling layer, linear layers, and activation function.
- Add binary classifier for 'Yes' and 'No' classes
- The feature extraction layers are frozen during training (requires_grad=False)

3.3.3 Version 3: VGG16 from scratch

Version 3 is a VGG16 network with a custom activation function. This model does not rely on the pre-trained VGG16.

- Initialize VGG16 without pre-trained weights (pretrained=False)
- Keep the feature extraction layers of VGG16
- Test different activation functions with different learning rates.

On top of three major versions, the report also combines different tuning hyperparameters including optimizer and learning rate.

3.3.4 VGG16 Findings

There are two best results, one is the theoretical best result, which would be explained in the later section, and the other one is the best result from the training dataset. The detailed results are demonstrated in the appendix sections.

The best result from this dataset is model #7, because the recall performance was perfect.

$$[VGG16 + Trainfromscratch + Activationfunction : Tanh + SGD + Learningrate : 0.1] =$$

$$\#7.\ Vgg16_{scratch\_tanh\_sgd\_0.01}$$

The generalizable best result is model #2, because it can utilize transfer learning and provide better generalizability.

$$[VGG16 + Pretrained + Parameters : unfreeze + Activationfunction : ReLU + SGD + Learningrate : 0.01] =$$

$$\#2.\ Vgg16_{pretrained\_p.freeze\_relu\_sgd\_0.01}$$

The result section is separated into the criteria, analysis and VGG16 conclusion section.

There are three most important criteria to identify whether the results are valid. First, the convergence is essential, whether the model can finish the iteration (epoch=300) and provide a suitable result. (accuracy>0.95). The testing model using the Adam optimizer (#3) and activation function of the sigmoid (#5) failed these criteria. It is possible that #3 model and #5 model only require further tuning to converge, but it is better to fine tune other models that reach higher initial accuracy.

The result table captures the important information to identify the best model.

Table 5

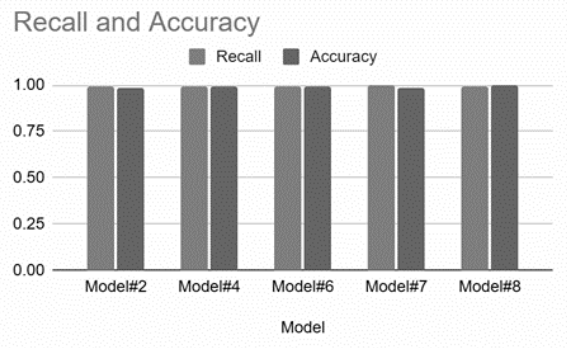
#	Model	Recall	Accuracy	Epochs
1	Vgg16_pretrained_p.freeze_relu_sgd_0.01	0.9673	0.9833	64/300
2	Vgg16_pretrained_p.unfreeze_relu_sgd_0.01	0.9935	0.9867	34/300
3	Vgg16_pretrained_p.unfreeze_relu_adam_0.01	0	0.4867	0/300
4	Vgg16_scratch_relu_sgd_0.01	0.9932	0.9933	60/300
5	Vgg16_scratch_sigmoid_sgd_0.01	0	0.4867	0/300
6	Vgg16_scratch_relu_sgd_0.1	0.9935	0.9933	55/300
7	Vgg16_scratch_tanh_sgd_0.1	1	0.9833	49/300
8	Vgg16_scratch_tanh_sgd_0.01	0.9935	0.9967	50/300

Second, recall and accuracy performance are essential for medical image diagnosis, especially the recall section is required to be as high as possible. While the recall is high, the model can capture all the possible patients with brain tumours and do not miss any of them.

Table 6

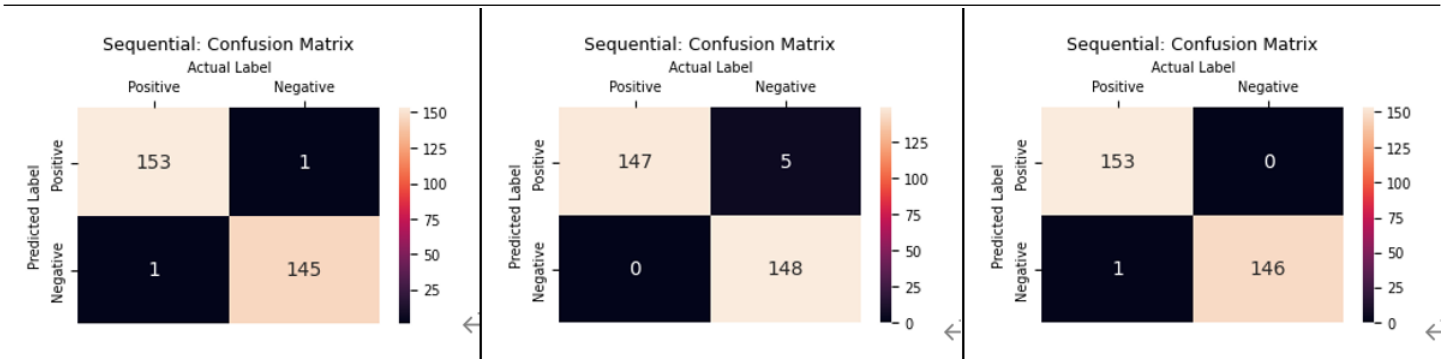
Model	Recall	Accuracy
Model#2	0.9935	0.9867
Model#4	0.9932	0.9933
Model#6	0.9935	0.9933
Model#7	1	0.9833
Model#8	0.9935	0.9967

Figure 21



By average recall and accuracy, the best three are model #6, #7, #8. The following table presents the performance metrics of each model tested:

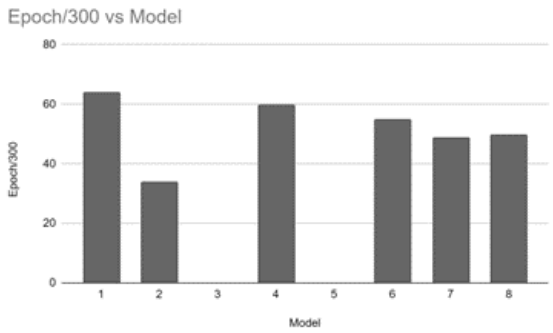
Figure 22: Confusion matrix of Model #6, Model #7, Model #8



However, both model #6 and #8 has 1 false negative, so the patient is actually positive. It is unacceptable to have misdiagnosis in the medical field, so the best selection is model 7 in this criteria.

Third, the iteration and generalizability are also necessary factors in the criteria. Within the test models, model #2 has the best performance. It completed the convergence with only 34 epochs, because model #2 utilized the transfer learning ability. It combines the initial weights to start with, so it performed much better than other models.

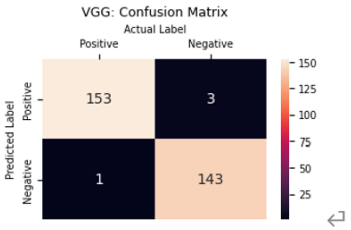
Figure 23

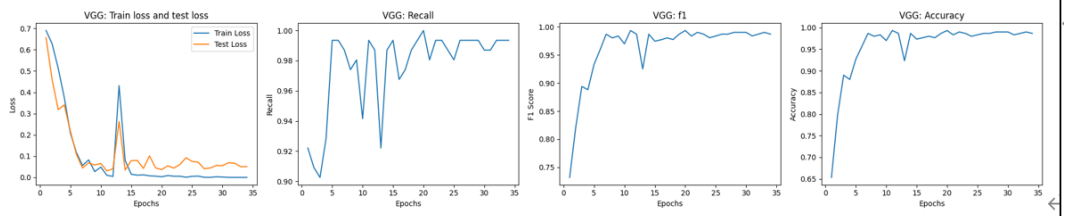


Transfer learning also benefits generalizability. The research MRI dataset is relatively simple and small, making it easy for the classifier to make judgements. However, in the real world with complex images, transfer learning can create huge benefits in learning and classifying. Therefore, the advantage of transfer learning from VGG16 model #2 is the theoretical best model.
In conclusion, Model #2 is the best model based on its transfer learning and generalizability in real life. Model #7 is the best model based on its high recall, which is essential in the medical field.

For the generalizability and potential future study, the overall best model is model #2. Model #2 evidences the fastest convergence time with transfer learning, and high performance (98%+), with learning rate = 0.01, ReLU activation, and SGD optimizer. The setting makes the model well generalized, striking a balance between generalizability and recall performance.

Figure 24

VGG Test Model 2: Vgg16_pretrained_p.unfreeze_relu_sgd_0.01		
Pretrained: Yes	Parameters: Unfreezed	Activation: ReLU
Optimizer: Sgd	Learning Rate: 0.01	Early Stopping: Yes
Result: Succeed Epoch [34/300], Training Loss: 0.0002 Epoch [34/300], Validation Loss: 0.0511 Recall: 0.9935, Precision: 0.9808, F1 Score: 0.9871, Accuracy: 0.9867 Early Stopping Triggered.		



The full results for VGG16 are listed in Section 7 Appendix.

4 Limitations

Although the four Neural Network models all achieved outstanding performance and high-level accuracy scores, we admit that this project still has several limitations from various perspectives. We will discuss them thoroughly mainly in two parts: technical limitations, ethical issues and real-world challenges.

4.1 Technical Limitations

(1) **Data Quality and Quantity**: The success of machine learning models heavily depends on the quality and quantity of data. In this project, we selected the Brain MRI image dataset from Kaggle and used it for model training, evaluation, and prediction. The dataset contains 3000 Brain MRI images with equally 1500 images having brain tumors and without tumors. Even if all the models performed well enough on this dataset, the limited number of training data may still affect the models' real-world performance. Other data quality issues such as potential bias, noise and limited diversity may also affect models' generalizability to a wider population.

(2) **Model interpretability**: While the models achieved high accuracy scores, the interpretability remains a challenge. Deep learning models like CNNs, ResNet and DenseNet are often considered as "black box" models, making it difficult to understand and explain why a particular decision or prediction was made. In real-world tumor detection applications, Experts are more inclined to believe the predictions given by interpretable models. The lack of transparency could be a limitation when trying to explain the models' prediction to medical professionals or patients.

(3) **Model Generalizability**: Although all the models performed well on the evaluation data, their ability to generalize to new unseen image data, especially those from different medical centres or MRI machines,

remains uncertain. These models should be further validated with data from more diverse sources to ensure their reliability in real-world clinical settings.

4.2 Ethical Issues

(1) **Data Privacy and Security**: Medical data can be sensitive since they are collected from real patients and may contain sensitive personal health information. Any mishandling, breach, or unauthorized access to the data can have severe consequences. Even if the dataset we used in this project was published on Kaggle with only images and labels and does not contain any personal information. It's always necessary to consider data privacy issues when implementing this project. Our aim is to build powerful models to assist in the early diagnosis of brain tumors without leaking any sensitive information.

(2) **Informed Consent**: When collecting sensitive brain BMI images for research purposes, prioritizing informed consent principles is crucial. Respecting the autonomy and privacy of individuals whose data is being utilized is not only a legal requirement but a fundamental ethical obligation (Hushmandi et al. 2023). Again, although the dataset is published and collecting it does not require any consent, it's always essential to carefully consider any potential ethical issues from various perspectives to ensure the ethical soundness of further implementation of this project.

4.3 Real-world Challenges

(1) **Clinical validation**: While the project's objective is to assist in early brain tumour diagnosis, deploying the model in real clinical settings involves complex challenges. The high-level performance on the Brain Tumor MRI dataset cannot guarantee satisfactory performance in real-world brain tumor detections. Those models' real-world generalizability still requires further clinical validations. We have to admit that even if those models seem to be quite powerful in handling brain image classification tasks, they are completely different from human brains, which are considered the most powerful and partially unexplainable neural networks in the world. These models are designed to assist in brain tumor detection and not to make any decisions for humans. We can use them as an assistant instead of relying on them completely.

(2) **Misclassification Consequences**: Brain tumor misclassification can have serious consequences. False negative cases may lead to incorrect diagnosis and delayed treatment, while false positives can cause unnecessary stress and medical interventions. Although all the models achieved high accuracy scores with only a handful of false prediction cases, all the misclassified images should be treated seriously and justified whether the misclassification was caused by a data quality issue or because of the model itself. They are all valuable insights for further improvements of those models.

5 Conclusion

In this project, we implemented various advanced machine learning techniques, and trained four neural network models for brain tumor detection MRI images published on Kaggle. The models include a 3-layer CNN, DenseNet-121, ResNet-18, and a transfer learning model VGG-16. The model training is based on a range of hyperparameters, such as learning rates, dropout rates, activation functions, and optimizers, to assess their impact on model performance. The model performance is evaluated based on various metrics like accuracy, recall, f1 score and confusion matrix. The results are well-recorded and visualized, and we also highlight the importance of data preprocessing and augmentation to enhance model generalizability. In summary, after the fine-tuning

process of hyperparameters, all four models achieved outstanding performance on the validation dataset, and a comprehensive comparison between models' performances is also provided in the report.

The findings of the four models can be summarized as follows: The 3-layer CNN demonstrates exceptional efficiency and satisfactory performance, making it an excellent choice for time-sensitive applications or resource-constrained environments. The modified ResNet-18 achieved decent performance across all the sets of parameters while illustrating the trade-off between efficiency and performance, and the effect of the initial conditions. DenseNet requires more computational resources and time, and stands out as the model with the highest accuracy, making it ideal for scenarios where maximizing accuracy is a priority. For model VGG-16, transfer learning can utilize pre-trained weights from ImageNet, allowing it to identify approximate correct parameters more quickly for the ideal classifier. The model can have better generalizability when facing a larger medical image dataset.

Although the four models achieved satisfactory performance on this dataset, we admit that there are a few limitations that need to be carefully considered. The technical limitations include the quality and quantity of data, model interpretability and generalizability to diverse data sources. Ethical considerations, such as data privacy and informed consent, are crucial when dealing with sensitive medical information. Additionally, we highlight real-world challenges, including the need for clinical validation and the potential consequences of misclassification in brain tumor detection.

For future research, we plan to continue enhancing these models by addressing the potential limitations. We are considering expanding the dataset with a more extensive and diverse collection of MRI images to further enhance the generalizability of the models. Additionally, We will continue to work on improving model interpretability, potentially through advanced techniques such as attention mechanisms and visualization tools. We will also share our model implementation code on Kaggle for any learning purposes and further research.

In this report, we provide a detailed implementation of four neural network models for brain tumor detection, together will a comprehensive comparison among model performances. The selection of the most suitable model should be guided by the specific requirements of the application, considering various factors such as computational resources, accuracy, and application requirements. This project contributes to the field of medical image analysis, aiming to assist in early brain tumour diagnosis. Our goal is to improve diagnostic success and ultimately improve patients' quality of life.

6 Reference

Amin, J., Sharif, M., Haldorai, A., Yasmin, M., & Nayak, R. S. (2021). Brain tumor detection and classification using Machine Learning: A comprehensive survey. Complex & Intelligent Systems, 8(4), 3161–3183. https://doi.org/10.1007/s40747-021-00563-y

Amin, J., Sharif, M., Raza, M., Saba, T., & Anjum, M. A. (2019). Brain tumor detection using statistical and machine learning method. Computer Methods and Programs in Biomedicine, 177, 69–79. https://doi.org/10.1016/j.cmpb

ASCO, Cancer. N. (2023, May 31). Brain Tumor - statistics. https://www.cancer.net/cancer-types/brain-tumor/statistics

Australian Government, C. A. (2022). Brain cancer. Brain cancer in Australia statistics | Cancer Australia. https://www.canceraustralia.gov.au/cancer-types/brain-cancer/statistics

BIG VISION LLC, L. (2023, September 8). Unlock the power of fine-tuning pre-trained models in Tensorflow & Keras. https://learnopencv.com/fine-tuning-pre-trained-models-tensorflow-keras/

Chen, Liang, Haoming Jiang, Simiao Zuo, Pengcheng He, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Tao Zhao. 2022. "No Parameters Left behind: Sensitivity Guided Adaptive Learning Rate for Training Large Transformer Models." arXiv (Cornell University), February. https://doi.org/10.48550/arxiv.2202.02664.

datagen, datagen. (2023, May 22). Understanding VGG16: Concepts, architecture, and performance. Datagen. https://datagen.tech/guides/computer-vision/vgg16/

Dolz, José, Kaundinya Gopinath, Jing Yuan, Hervé Lombaert, Christian Desrosiers, and Ismail Ben Ayed. 2018. "HyperDense-Net: A Hyper-Densely Connected CNN for Multi-Modal Image Segmentation." arXiv (Cornell University), April. https://doi.org/10.48550/arxiv.1804.02967.

Ellingson, B. M., Zaw, T., Cloughesy, T. F., Naeini, K. M., Lalezari, S., Mong, S., Lai, A., Nghiemphu, P. L., & Pope, W. B. (2012). Comparison between intensity normalization techniques for dynamic susceptibility contrast (DSC)-MRI estimates of cerebral blood volume (CBV) in human gliomas. Journal of Magnetic Resonance Imaging, 35(6), 1472–1477. https://doi.org/10.1002/jmri.23600

Hamada, A. (2021, November 14). BR35H :: Brain tumor detection 2020. Kaggle. https://www.kaggle.com/datasets/ah tumor-detection

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition (arXiv:1512.03385). arXiv. https://doi.org/10.48550/arXiv.1512.03385

Huang, Gao, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. 2016. "Densely Connected Convolutional Networks." arXiv (Cornell University), August. https://doi.org/10.48550/arxiv.1608.06993.

Huang, J., Li, H., Yan, H., Li, F.-X., Tang, M., & Lu, D.-L. (2022). The comparative burden of brain and central nervous system cancers from 1990 to 2019 between China and the United States and predicting the future burden. Frontiers in Public Health, 10. https://doi.org/10.3389/fpubh.2022.1018836

Hushmandi, Kiavash, Sam Saghari, Abdorrahman Harif Nashtifanii, Mohammad Arad Zandieh, and Rasoul Raesi. 2023. "Respecting Autonomy, Privacy, and Information in Maternity Care: A Study of Midwifery Personnel in Mashhad University of Medical Sciences (2022)." The Open Public Health Journal 16 (1). https://doi.org/10.2174/18749445-v16-e230927-2023-22.

Li, H. (2022). Machine Learning Methods. Tsinghua University Press. http://www.tup.tsinghua.edu.cn/booksCenter/b

Keerthana, A., Kavin Kumar, B., Akshaya, K. S., & Kamalraj, S. (2021). Brain tumour detection using machine learning algorithm. Journal of Physics: Conference Series, 1937(1), 012008. https://doi.org/10.1088/1742-6596/1937/1/012008

Pandey, Ashutosh, and DeLiang Wang. 2020. "Dense CNN with Self-Attention for Time-Domain Speech Enhancement." arXiv (Cornell University), September. https://doi.org/10.48550/arxiv.2009.01941.

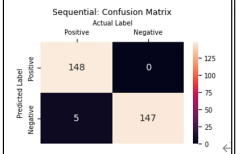
Yildirim, Z., Hançer, E., Samet, R., Mali, M.T., & Nemati, N. (2022). Effect of Color Normalization on Nuclei Segmentation Problem in H&E Stained Histopathology Images. 2022 30th Signal Processing and Communications Applications Conference (SIU), 1-4.

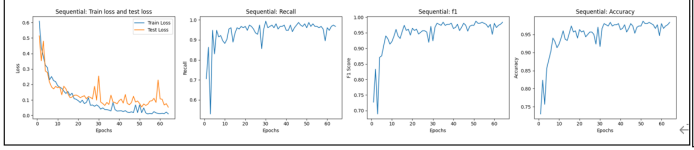
7 Appendix

The full results of VGG16

VGG Test Model 1: Vgg16_pretrained_p.freeze_relu_sgd_0.01		
Pretrained: Yes	Parameters: Freezed	Activation: ReLU
Optimizer: Sgd	Learning Rate: 0.01	Early Stopping: Yes

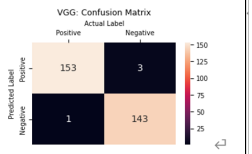
Result: Succeed
Epoch [64/300], Training Loss: 0.0102
Epoch [64/300], Validation Loss: 0.0532
Recall: 0.9673,
Precision: 1.0000,
F1 Score: 0.9834,
Accuracy: 0.9833,
Early Stopping Triggered.

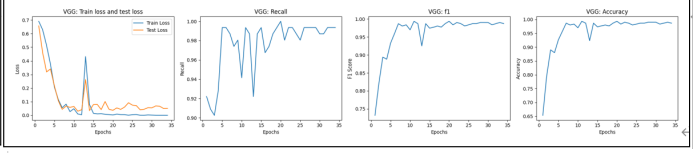




VGG Test Model 2: Vgg16_pretrained_p.unfreeze_relu_sgd_0.01		
Pretrained: Yes	Parameters: Unfreezed	Activation: ReLU
Optimizer: Sgd	Learning Rate: 0.01	Early Stopping: Yes

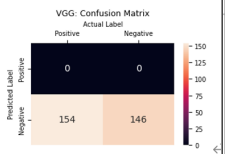
Result: Succeed
Epoch [34/300], Training Loss: 0.0002
Epoch [34/300], Validation Loss: 0.0511
Recall: 0.9935,
Precision: 0.9808,
F1 Score: 0.9871,
Accuracy: 0.9867
Early Stopping Triggered.

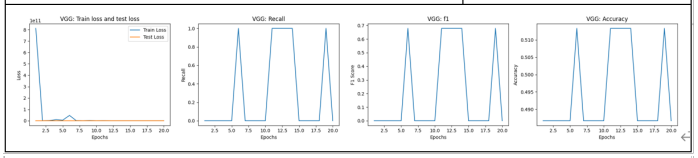




VGG Test Model 3: Vgg16_pretrained_p.unfreeze_relu_adam_0.01		
Pretrained: Yes	Parameters: Unfreezed	Activation: ReLU
Optimizer: Adam	Learning Rate: 0.01	Early Stopping: Yes

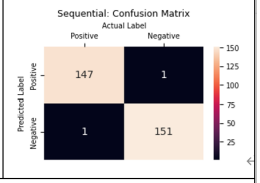
Result: Failed
Epoch [20/300], Validation Loss: 0.6933
Recall: 0.0000,
Precision: 0.0000,
F1 Score: 0.0000,
Accuracy: 0.4867
Early Stopping Triggered.

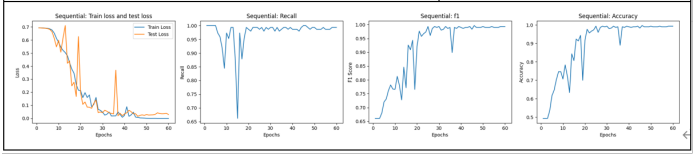




VGG Test Model 4: Vgg16_scratch_relu_sgd_0.01		
Pretrained: No	Parameters: No	Activation: ReLU
Optimizer: Sgd	Learning Rate: 0.01	Early Stopping: Yes

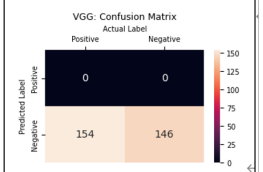
Result: Succeed
Epoch [60/300], Training Loss: 0.0001
Epoch [60/300], Validation Loss: 0.0298
Recall: 0.9932,
Precision: 0.9932,
F1 Score: 0.9932,
Accuracy: 0.9933
Early Stopping Triggered.

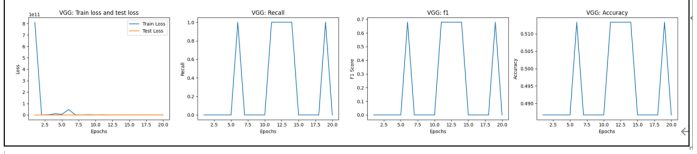




VGG Test Model 5: Vgg16_scratch_sigmoid_sgd_0.01		
Pretrained: No	Parameters: No	Activation: Sigmoid
Optimizer: Sgd	Learning Rate: 0.01	Early Stopping: Yes

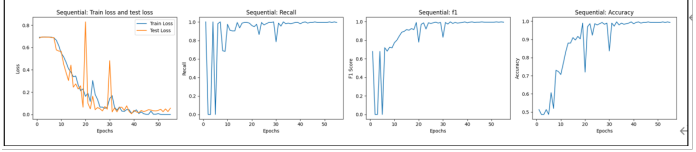
Result: Failed
Epoch [11/300], Training Loss: 0.6968
Epoch [11/300], Validation Loss: 0.6983
Recall: 0.0000, Precision: 0.0000,
F1 Score: 0.0000, Accuracy: 0.4867
Early Stopping Triggered.





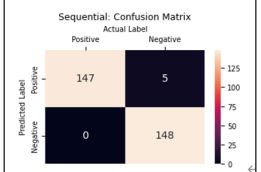
VGG Test Model 6: Vgg16_scratch_relu_sgd_0.1		
Pretrained: No	Parameters: No	Activation: ReLU
Optimizer: Sgd	Learning Rate: 0.1	Early Stopping: Yes

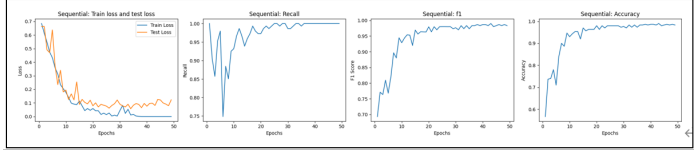
Result: Succeed
Epoch [55/300], Training Loss: 0.0000
Epoch [55/300], Validation Loss: 0.0563
Recall: 0.9935,
Precision: 0.9935,
F1 Score: 0.9935,
Accuracy: 0.9933
Early Stopping Triggered.



VGG Test Model 7: Vgg16_scratch_tanh_sgd_0.1		
Pretrained: No	Parameters: No	Activation: tanh
Optimizer: Sgd	Learning Rate: 0.1	Early Stopping: Yes

Result: Succeed
Epoch [49/300], Training Loss: 0.0000
Epoch [49/300], Validation Loss: 0.1231
Recall: 1.0000,
Precision: 0.9671,
F1 Score: 0.9833,
Accuracy: 0.9833
Early Stopping Triggered.





VGG Test Model 8: Vgg16_scratch_tanh_sgd_0.01		
Pretrained: No	Parameters: No	Activation: tanh
Optimizer: Sgd	Learning Rate: 0.01	Early Stopping: Yes

Result: Succeed
Epoch [50/300], Training Loss: 0.0011
Epoch [50/300], Validation Loss: 0.0214
Recall: 0.9935,
Precision: 1.0000,
F1 Score: 0.9967,
Accuracy: 0.9967
Early Stopping Triggered.