Comparative study of transfer learning on Brain Tumour MRI image classification

Asish Binu Mathew
Department of Computing and Information Sciences
University of Strathclyde
Glasgow, Scotland
0000-0003-2340-3606

***Abstract*—A brain tumor is a growth of cells in the brain that multiplies in an abnormal, uncontrollable way. More than 11,000 people are diagnosed with a primary brain tumor in the UK each year, of which about half are cancerous. Magnetic resonance imaging or MRI is the most common and initial steps in the diagnosis of brain tumor. Therefore, it is vital to correctly identify presence of brain tumors from MRI images. Deep Learning models can help medical professionals in early detection of the presence of brain tumors. This paper aims to compare several deep learning algorithms and how accurately they can detect the presence of brain tumor depending upon the parameters on which they are trained. We have used pretrained Resnet50 and VGG16 with batch normalization models with varying parameters. The dataset we have taken comes from Kaggle. We tried adjusting the learning rate, optimization functions like Stochastic Gradient Descent with Momentum and Adam optimization Algorithm.**

Keywords—cnn, vgg16bn, Resnet50, optimizer comparison, deep learning, transfer learning

I. INTRODUCTION

MRI is the most common and initial steps in the diagnosis of brain tumour [1]. This paper aims to compare several deep learning algorithms and how well they can detect the presence of brain tumour depending upon the parameters on which they are trained. We will be studying how two different deep learning algorithms tries to learn the data while changing several things. This helps to 1) identify brain tumour from MRI images 2) get an understanding on how different models learn depending on their hyperparameters like loss function, loss optimizer, learning rate among many other.

There are several studies conducted for classifying or detecting brain tumour from MRI images. One of them is the paper from 2009 by Zacharaki [2] which uses binary support vector machines to achieve 85% to 87% accuracy for various classes of brain tumour.

Another study by Abiwinanda in 2019 [3], Tumour Classification Using Convolutional Neural Network achieved 84.19% accuracy on test set on their own CNN architecture. This model was able to learn even complex features despite being only an image classification task.

In 2019 Irmak used a pretrained GoogLeNet model [4] to achieve an accuracy of 98% using transfer learning.

II. METHODOLOGY

A. Dataset Description

The dataset I have taken comes from Kaggle [5]. The dataset has only 253 samples in total. They are separated into two folders Yes and No with Yes folder having 155 samples and No folder having only 98 samples. I split this into training, validation and test set by a ratio of 0.8, 0.2 and 0.2 respectively.

B. Pre-processing

Pre-processing images helps to generalize the data more. Pre-processing often introduces random changes to existing image and helps the model generalize more. Resnet50 and VGG16BN uses same following pre-processing steps. These were extracted from Pytorch's pretrained weights [6]. They are

- Crop Size of 224,224
- Resize of 232,232
- Mean of [0.485, 0.456, 0.406]
- Standard Deviation of [0.229, 0.224, 0.225]
- Interpolation of Bilinear

C. Deep Learning Architecture

1) Resnet 50

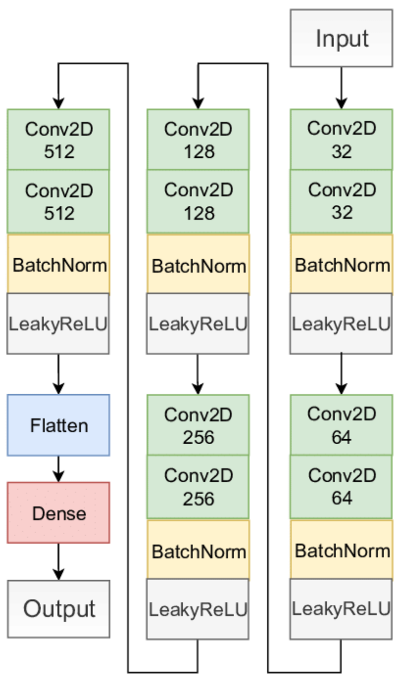
Resnet50 was specifically picked because of its ability to solve vanishing gradient problem with residual blocks [4] . Resnet 50 has 48 convolutional layers 1 max pool layer and 1 average pool layer, forming together 50 layers. Information is retained till the end of the 50^{th} layer because of the residual block.

I have used pretrained weights for this model. The last classifier layer is changed from 1000 features to 2 which is the number of our output classes. Only the last single fully connected layer in resnet50 was used for training the rest were frozen.

2) VGG16 with Batch Normalization

Instead of regular VGG16 architecture I have used a slightly modified VGG16. This one has batch normalization layer added to it. This variant of VGG 16 is said to have 10x training speed and reduces overfitting [7] .

I have used pretrained weights for this model too. In the case of VGG 16 BN the last classifier layer is changed from 1000 features to 2 which is the number of our output classes. Apart from this I have used the full classifier section for training. Meanwhile only the single fully connected layer in resnet50 was used for training the rest were frozen.



D. Error Function

I have used cross entropy loss as the loss function for all the models. Since I was comparing only the optimizer and learning rate, I felt it was better to use the same error function to make it easier to compare against models.

E. Optimizer Function

The optimizer function helps us to minimize the loss function and converge faster. It does this by changing the parameters during training to reach the optimal ground. There are a lot of optimization functions, I have used Stochastic Gradient Descent with Momentum and Adam optimization function.

1) SGD with momentum

Stochastic Gradient Descent with momentum is an extension of vanilla SGD or Stochastic Gradient Descent algorithm with Mini Batch processing [8]. It has all the advantages of SGD without their disadvantages. SGD with momentum converges faster, it does a better job at moving away from local minima or maxima than vanilla SGD or SGD with mini batch.

$$x_t = x_{t-1} - \alpha \sum_{i=0}^{t-1} \rho^{t-1-i} (1-\rho) \nabla f(x_i) \quad (1)$$

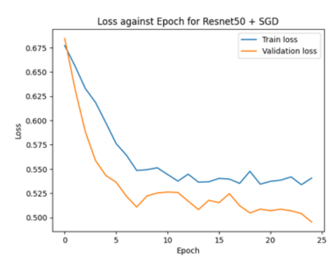
2) Adam Optimization Function

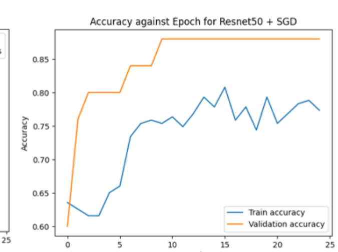
Adam optimizer takes advantage of momentum and Root Mean Square Propagation (RMSP). Momentum takes the 'exponentially weighted average' of the gradients to converge towards the minima faster. The RMSP takes the 'exponential moving average' of the gradients. Adam inherits all the goodness of the above two methodologies to reach to the minima faster [9].

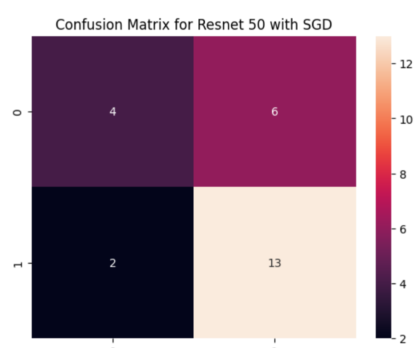
III. Results

A. Resnet 50 with SGD

This configuration had a test accuracy of 68%, the model was able to learn some basic features, maybe training few more epochs might have increased the test accuracy score. From the accuracy vs epoch graph we can see the validation accuracy has hit a roof. This is because our test set only had 25 samples, looking at the confusion matrix, we can see that 8 images were incorrectly classified. The validation graph remains as a straight line from epoch 10 till the end, which suggests no additional information was learned from training from epoch 10 till the end. This suggests that the model is not able to generalize well due to the lack of sufficient data.

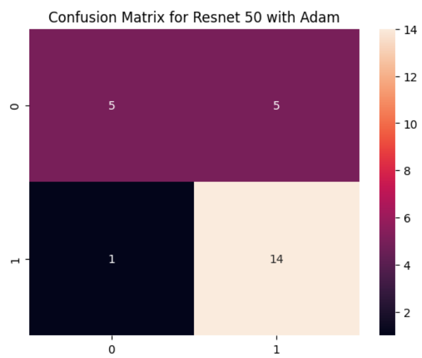






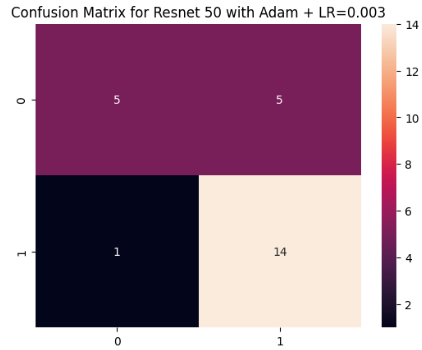
B. Resnet 50 with Adam

This configuration had a test accuracy of 76%, this is much better than the Resnet model with SGD. Another interesting thing is the loss vs epoch graph. They are converging earlier than the previous one. This is an expected behaviour since we were using Adam optimizer instead of SGD. The flatlining for the validation line in accuracy vs epoch graph came early at epoch 5. This suggests that the problem is with the dataset, but the current configuration-based model was able to learn features faster despite that.

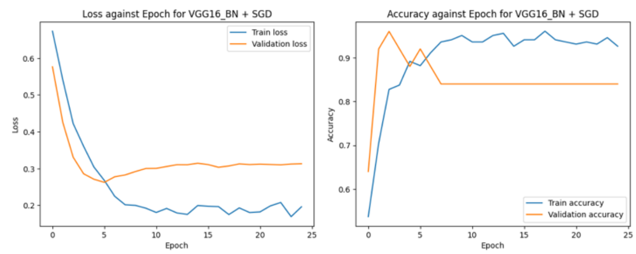


C. Resnet 50 with Adam and LR=0.003

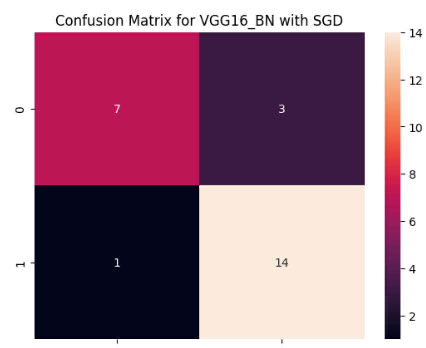
This configuration had a test accuracy of 76%, this time we trained the last model for 50 epochs and a bit lower learning rate of 0.003. All other configurations were same. The Loss vs Epoch graph shows that the model was able to reduce its loss much earlier because of the lower learning rate to around 0.2 at epoch 6 and kept roughly the same value till epoch 50. I would say the model was overfitted and can be justified if you look at Accuracy vs Epoch graph where train accuracy is higher than validation accuracy after epoch 10. This is because of the infrequency of data.



D. VGG16 Batch Normalized with SGD

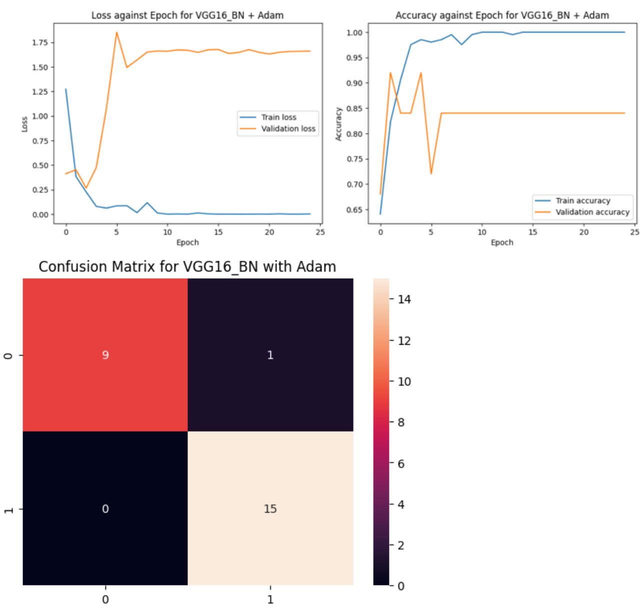


This configuration had a test accuracy of 84%, We trained the VGG 16 BN model with an SGD optimizer while keeping everything else the same. The Loss vs Epoch graph shows that the model was able to converge the loss slowly at epoch 5 and starts to overfit from there. The slow convergence when compared to the Adam optimizer is expected. I would say the model was overfitted and can be justified if you look at Accuracy vs Epoch graph where train accuracy is higher than validation accuracy after epoch 6. This is because of the infrequency of data.



E. VGG16 Batch Normalized with Adam

This configuration had a test accuracy of 96%, We trained the VGG 16 BN model with an Adam optimizer while keeping everything else the same. The Loss vs Epoch graph shows that the model was able to converge the loss almost instantly at epoch 2 and starts to overfit from there. The fast convergence when compared to the Adam optimizer is expected. The model was overfitted and can be justified if you look at Accuracy vs Epoch graph where train accuracy is higher than validation accuracy after epoch 4. This is because of the infrequency of data.



IV. Comparison Study

Configurations with Adam Optimizer converged much faster than SGD. Adam optimizer is one of the best loss optimization functions. The lack of data makes it harder for the model to generalize. Almost all the configurations validation accuracy starts to flatline around 0.85, which strongly suggest an imbalance in the dataset, perhaps some kind of edge case feature that the model hasn't seen previously. Resnet models were expected to perform better than VGG16 BN but produced lower scores than VGG16 BN. This can also be accounted by the lack of sufficient data because of the large number of layers. Resnet34 or Resnet18 might work comparatively better than VGG16 BN models. Lowering the learning rate didn't produce any difference in the test accuracy score, the model learned faster because of the small amount of data and a good model.

Sl.NO	Model	Learning Rate	Epochs	Optimizer	Momentum	Test Accuracy
1	Resnet50	0.001	25	SGD	0.9	68%
2	Resnet50	0.001	25	Adam	NA	76%
3	Resnet50	0.003	50	Adam	NA	76%
4	VGG16BN	0.001	25	SGD	0.9	84%
5	VGG16BN	0.001	25	Adam	NA	96%

V. Discussion and Future Works

Using a better dataset would produce much better results. We didn't have enough data to learn from and it was evident from the results. Instead of considering this as classification tasks this could be considered as image segmentation problems. This would improve the results Networks like U-Net can generate the highly accurate predictions for medical images by using image segmentation [10]. The validation accuracy flatlining issues can be solved by either adding more data or using data augmentation methods.

References

[1] K. R. Maravilla and W. C. Sory, 'Magnetic resonance imaging of brain tumors', *Semin. Neurol.*, vol. 6, no. 1, pp. 33–42, Mar. 1986, doi: 10.1055/s-2008-1041445.

[2] E. I. Zacharaki *et al.*, 'Classification of brain tumor type and grade using MRI texture and shape in a machine learning scheme', *Magn. Reson. Med. Off. J. Soc. Magn. Reson. Med. Soc. Magn. Reson. Med.*, vol. 62, no. 6, pp. 1609–1618, Dec. 2009, doi: 10.1002/mrm.22147.

[3] N. Abiwinanda, M. Hanif, S. T. Hesaputra, A. Handayani, and T. R. Mengko, 'Brain Tumor Classification Using Convolutional Neural Network', in *World Congress on Medical Physics and Biomedical Engineering 2018*, Singapore, 2019, pp. 183–189. doi: 10.1007/978-981-10-9035-6_33.

[4] E. Irmak, 'Multi-Classification of Brain Tumor MRI Images Using Deep Convolutional Neural Network with Fully Optimized Framework', *Iran. J. Sci. Technol. Trans. Electr. Eng.*, vol. 45, no. 3, pp. 1015–1036, Sep. 2021, doi: 10.1007/s40998-021-00426-9.

[5] 'Kaggle Brain Tumor MRI Dataset'. NAVONEEL CHAKRABARTY. [Online]. Available: https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection/discussion

[6] M. Salvi, U. R. Acharya, F. Molinari, and K. M. Meiburger, 'The impact of pre- and post-image processing techniques on deep learning frameworks: A comprehensive review for digital pathology image analysis', *Comput. Biol. Med.*, vol. 128, p. 104129, Jan. 2021, doi: 10.1016/j.compbiomed.2020.104129.

[7] S. Ioffe and C. Szegedy, 'Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift'. arXiv, Mar. 02, 2015. doi: 10.48550/arXiv.1502.03167.

[8] S. Ruder, 'An overview of gradient descent optimization algorithms'. arXiv, Jun. 15, 2017. Accessed: Nov. 27, 2022. [Online]. Available: http://arxiv.org/abs/1609.04747

[9] D. P. Kingma and J. Ba, 'Adam: A Method for Stochastic Optimization'. arXiv, Jan. 29, 2017. doi: 10.48550/arXiv.1412.6980.

[10] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, 'Image Segmentation Using Deep Learning: A Survey'. arXiv, Nov. 14, 2020. doi: 10.48550/arXiv.2001.05566.