Deep Learning Final Project
Technical Report

Classification of Alzheimer Disease stages in MRI Modality using Transfer learning

Batoul Khalil

1 Department of Computer Science and Telecommunications, National and Kapodistrian University of Athens

Abstract

The role of medical imaging has expanded beyond visualization and disease diagnosis to encompass simulation, treatment planning, extracting information on anatomical structures, and monitoring disease progression. The growth in medical image data is attributed to the rapid development of acquisition devices. Our research focuses on identifying Alzheimer's disease stages through MRI images. We assessed the effectiveness of transfer learning with three pre-trained models (VGG16, ResNet50, EfficientNetB1) using metrics such as balanced accuracy, MCC, F1-score, confusion matrices, and precision. Cross-validation was employed to ensure the generalization of our model to unseen data. Notably we utilized Grad-CAM to comprehend the model's decision -making process in classification tasks. Our results underscore the significance of employing deep learning in addressing critical problem- solving challenges.

Introduction:

Alzheimer's disease (AD) is the most common type of dementia and typically manifests through a progressive loss of episodic memory and cognitive function [1]. Globally, more than 50 million people are estimated to be living with Alzheimer's disease, and this number is projected to grow significantly [2]. Even though scientists have spent considerable amounts of time, money, and effort researching early AD diagnosis using a variety of methods, post-mortem examination is still considered to be the only definitive way to confirm a diagnosis [3]. Magnetic resonance imaging (MRI) is a popular imaging technique for diagnosing and treating AD [4]. The pathogenesis of AD remains not fully elucidated and no available therapy can cure AD or completely stop disease progression. Amnestic mild cognitive impairment (MCI) is a transitional stage between cognitively normal aging and AD, and patients with MCI are more likely to develop AD than age-matched healthy cognition (HC) [5]. Early detection of AD by screening MCI is crucial both for effective management and care strategies and for developing new drugs and measures to prevent further deterioration of the disease [7]. MRI allows for the visualization of brain structure and the detection of structural changes that may occur in AD, such as the shrinkage of certain brain regions, changes in brain tissue density, and the accumulation of certain substances in the brain [8]. We propose a novel lightweight deep learning model using MRI images for the accurate detection of AD. The impact of the proposed model is significant, as it provides a lightweight and efficient approach for accurately detecting AD from MRI images.

In this study we aim to assess the effectiveness of prominent computer vision algorithms in the context of Alzheimer’s stages detection, utilizing a moderately-sized academic dataset focused on neuroimaging. While the original dataset’s publication extensively applied deep learning methods for classification [8], our objective is to conduct a comparative analysis of diverse models’ performance specifically within the realm of Alzheimer’s disease detection.

In the medical domain, acquiring extensive labeled data for deep learning algorithms can be challenging due to data scarcity. To overcome this limitation, we explore the potential of transfer learning, a powerful tool that leverages knowledge from related but distinct source domains. By doing so, we aim to mitigate the reliance on vast quantities of target-domain data, providing a practical solution to the challenge of data scarcity in Alzheimer’s stages detection. Rather than constructing a new model and training it from the beginning, transfer learning allows us to capitalize on pre-existing model’s knowledge and repurpose it for our specific task. The primary advantages we gain from employing transfer learning include a reduction in training time, enhanced performance of neural networks, and the ability to achieve meaningful results without requiring an extensive amount of data.

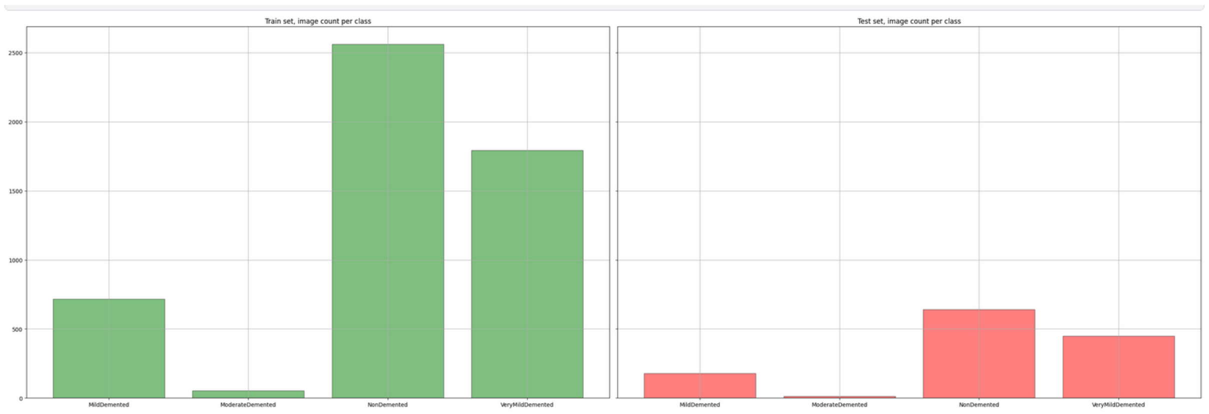
Currently, individuals with symptoms typically receive a diagnosis through a collaborative effort between general practitioners and specialized professionals like neurologists or neuropsychologists. The diagnostic process usually entails gathering the patient’s medical history, conducting physical exams, administering diagnostic tests, carrying out neurological and neuroimaging assessments, and performing mini-mental state evaluations. These manual techniques not only consume a significant amount of time but also demand expertise. The reliance on visual inspection or semi-quantitative approaches in these manual methods is subjective and may lead to disagreements among examiners/readers. However, employing computer- aided diagnosis systems to identify the underlying neurological cause of brain disorders can offer a more accurate and precise diagnosis in the early stages of the disease continuum.[9]

Methodology:

Dataset exploration and description:

In this project, we employed brain MRI scans obtained from the Alzheimer’s Disease Neuroimaging Initiative (ADNI) dataset, accessible through [10]. The dataset encompasses two subdirectories, namely the training and testing sets, combined containing 6400 pre-processed MRI images categorized into four types: Non-Dementia, Very-Mild Dementia, Mild Dementia, and Moderate Dementia. The image size in the dataset is standardized at 256 by 256 pixels, with the training and testing datasets constituting 90% and 10% of the total data, respectively.

Figure1: Distribution of MRI images in the dataset.



Upon observing the distribution of the training and testing sets, a noticeable imbalance was identified, and it was observed that both sets shared the same distribution. Two approaches were considered for further analysis: the first involved using the training set for training and the test set for testing. However, after exploring the results, it was evident that the model's performance was suboptimal due to a degradation in complexity and quality between the training and testing sets. Consequently, we opted to merge the two initial directories and re-split the dataset, resulting in a more robust model that outperformed the initial approach. Following the merger of the two sets, the weight distribution for each class was 14%, 1.0%, 50%, and 35% for Mild Dementia, Moderate Dementia, Non-Dementia, and Very-Mild Dementia, respectively.

Figure2: the distribution of the merged datasets.

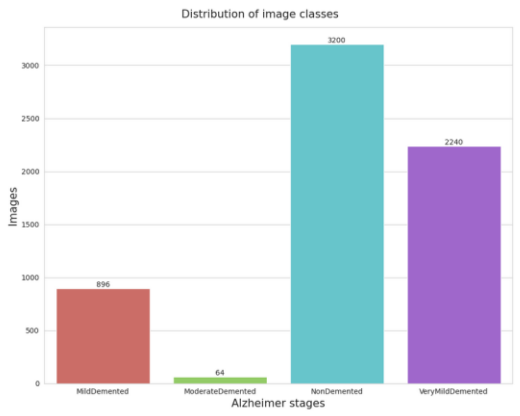
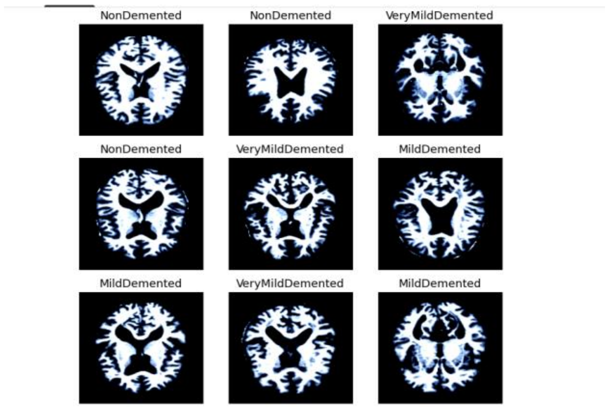


Figure3: Example microscopy image per class



Deep Learning:

In recent years, deep learning techniques have been successfully applied to a wide range of medical imaging tasks [11,12,13,14]. Including the detection of AD [15, 16]. The deep learning approach has attracted a lot of attention in exploring new imaging biomarkers for AD diagnosis and prediction, which requires no prior knowledge to extract biologically meaningful information from subtle changes [22].

A general deep learning architecture contains a combination of some neural layers like input layers, convolution, fully connected layers, sequence layers, activation layers, normalization, dropout, pooling, output layers and many more. Only the input and output layers' values are easily accessible by us, the rest are called hidden layers. It is in these hidden layers where the majority of important work happens and because of them complex data may be modelled. In this study two specialized kinds of neural networks are tested, three convolutional (ConvNets) and one Vision Transformer. Each type of model is based on different kinds of layers and architectural patterns in order to tackle the image classification task. The steps we followed to train and test our models on mycological images are described in detail in this section.

Data Set Preprocessing for Deep Learning

To menage preprocessing and dataset-related tasks, we leverage two fundamental built-in PyTorch classes: datasets and Dataloaders [22].

Specifically, we applied the ImageFolder method to directories containing a total of 6,300 RGB images, organized into subdirectories corresponding to their respective classes. This approach facilitated the creation of an iterable dataset containing both image data and their corresponding labels. Additionally, the dataset allowed the definition of preprocessing transformation applied to the images.

For the preprocessing transformation, we opted for the standard approach used in most pre-trained models, involving:

1. Resizing the images according to the input size of the model we use using bilinear interpolation.
2. Transforming the image into a tensor.
3. Adjusting the channels by subtracting 0.485, 0.456 and 0.406 from each channel, followed by dividing the results by 0.229, 0.224 and 0.225 for each respective channel to standardized them.

Subsequently, we initialized Dataloaders to manage our two primary training approaches: a stratified cross-validation training aimed at establishing an average estimate of the model's performance across different datasets and the final training on s standard training-validation split to create the ultimate model.

For cross-validation, we employed the scikit-learn library to create an 85% training and 15% testing data split. We then implemented a stratified 5-fold split on the training dataset, yielding the necessary indices for segmenting the training dataset into five distinct subsets. Separate data loaders were constructed for each split, each with a batch size of 32, and to prevent model bas during backpropagation, we introduced data shuffling into the training data loaders, ensuring that the images were presented in a randomized order.

For the ultimate model training, we established an additional split, comprising training and validation sets allocated at 85% training dataset. These sets were then loaded into two separate data loaders, using the same configuration parameters employed during the cross-validation phase.

It is noteworthy that, for the training dataset, we adopted an alternative preprocessing transformation to introduce data augmentations, thereby instilling variability in the dataset and mitigating the risk of overfitting on the training set. The steps for this approach during a distinct final model training session were as follows:

1- Resize the images to match the input size of the model using bilinear interpolation.
2- Horizontally flip the image (50% chance).
3- Randomly rotate the image up to 10 degrees left or right.
4- Adjust the brightness and contrast by up to 20%, and the saturation and hue by up to 10%.
5- Transform the image into tensor.
6- Adjust the channels by subtracting 0.485, 0.456 and 0.406 from each channel, followed by dividing the result by 0.229, 0.224 and 0.225 for each respective channel to standardize them.

The custom Pytorch training loop

The schematic representation of the custom PyTorch training loop we employed is depicted in Figure 4. Initially, the model is moved to the GPU device (P100), and a forward pass is executed using iterative batches of 32 images from the training dataset. This involves predicting labels for each batch and calculating the training loss utilizing the cross-entropy loss function:

$$L(y, p) = -\sum_{i=1}^{N} y_i \,.\log(p_i)$$

Here, y_i represents the true probability over distribution over classes (a one-hot encoded vector indicating the true class), and p_i is the predicted probability distribution over classes after applying a SoftMax function to the model's logits output. Notably, the logits are individually weighing by 1.7857, 25.0000, 0.5000, and 0.7143, respectively. This weighting strategy is implemented to incorporate class balance-sensitive learning.

Subsequently, backpropagation is initiated from the model's known output layer value, using the chain rule to compute gradients for the weights for all layers. The Adam optimizer with a learning rate of 0.0001 is employed, to iteratively update the model's weights, aiming to minimize the loss function.

Afterwards, the current model's weights are frozen, and the validation Dataloader provides image batches for the model to make predictions on the validation set. From these predictions, a validation loss is calculated in the same manner as the training step. The current model weights are saved in memory, and we assess whether the validation loss has improved compared to previous epoch. If the validation loss remains unchanged or increases for five consecutive epochs, the training process is terminated.

Upon completion of training, predictions on the test set are made using the same methodology as employed for validation set. This enables the calculation of classification metrices such as F1 score, Balanced Accuracy, and Mathews Correlation coefficients (MCC).

For cross-validation, the mentioned training methodology is independently applied to each of the five stratified splits, utilizing the Dataloaders ae described in the preprocessing section. However, after each split, the model is reset by loading its default weights from training on the ImageNet dataset. This reset ensures that the model begins training from the same initial weights for each validation fold.

Cross Validation:

Cross validation is a technique used in machine learning to evaluate the performance of a model on unseen data. It involves dividing the available data into multiple folds or subsets, using one of these folds as a validation set and training the model on the remaining folds. This process is repeated multiple t, each time using a different fold as the validation set. Finally, the results from each validation step are averaged to produce a more robust estimate of the model's performance. Cross validation is an important step in the machine learning process and helps to ensure that the model selected for deployment is robust and generalizes well to new data.

We use cross validation to prevent overfitting, which occurs when a model trained too well on the training data and performs poorly on new, unseen data. Evaluating the model on multiple validation provides a more realistic estimate of the model's generalized performance and its ability to perform well on new, unseen data.

Transfer learning:

Utilizing pretrained models on extensive dataset such as ImageNet, which comprises 1.2 million images across 1000 categories, for initialized or as fixed features extractors in specific tasks has gained widespread popularity in image classification [18]. Nowadays, training large models from scratch is infrequent, given the availability of numerous pretrained models that exhibit superior accuracy across a diverse range of tasks.

Transfer learning can be implemented in two ways:

1_ Fine Tuning the ConvNet:

Instead, of initializing the weights randomly, the network is initialized with pretrained weights and then trained for a few epochs on the dataset at hand using the standard training loop. Using a learning rate with low value to assure not losing the information we gained from the pretrained model with also giving the model the ability to learn from our different problem.

2_ ConvNet as a fixed features extractor:

The model's weights are frozen for the entire network, except for the final fully connected layer, which is replaced with random numbers. During training, only these final weights are updated to fit the specific dataset.

In our implementation, we employed Three pretrained models directly from Pytorch, loading their default pretrained weights. The architectures of the models remained unchanged, except for the output layers, which were reset to have five neurons, corresponding to the number of the classes in our dataset. We opted to fine-tuning each model to our training set, as the fixed feature extractor yielded suboptimal performance in our case. The steps we followed to load utilize the pretrained models are as follows.

1. Load the appropriate model using the torchvision module and the default pretrained weights.

2. Reset the final layer according to the number of classes in our dataset.
3. Apply the transforms used during the initial training of the model to preprocess our dataset.

The following are brief description of the three models we tested.

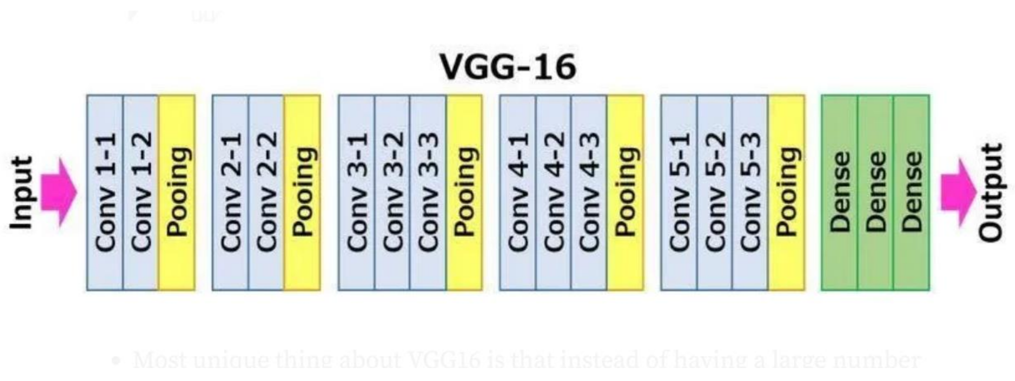
VGG16:

VGG16 is a convolutional neural network architecture introduced by the Visual Geometry Group (VGG) at the University of Oxford in 2014 [17]. Its development aimed to explore the impact of depth in visual representations.

The VGG16 version consists of 16 weight layers, comprising 13 conventional and fully connected layers (refer to Figure (5). Notably, the architecture incorporates a series of convolutional filters with a small receptive field (3*3), enabling an increase in the model's depth. These architectural features contribute to a total of 138 million trainable parameters.

In its design, VGG16 utilizes max-pooling layers to down-sample feature map, but these layers are not applied after every convolutional layer. This intentional design choice ensures superior pattern recognition while maintaining computational efficiency.

Figure 5: the architecture of VGG16 [21]



EfficientNet:

EfficientNet introduced by Google AI in 2019, It's a powerful convolution neural network (CNN) architecture that addresses the challenges of scaling CNN models. Traditionally, CNN is initially developed at a fixed resource cost and later scaled up to improve accuracy with more resources. However, existing scaling techniques are often random, leading to inconsistent results. EfficientNet introduces compound scaling method that uniformly scales width, depth, and mage resolution with fixed coefficients, providing a more ethical and effective approach to model scaling.

Compound Model Scaling involves balancing width, depth, and image resolution dimensions using a constant ratio. The method is backed by a systematic study on the impacts of each scaling technique, revealing the balancing of all three dimensions improves overall model performance. The scaling technique, combined with AutoML, resulted in the development of seven EfficientNet models of various dimensions, surpassing state-of-the=art accuracy and efficiency of most CNNs.

The architecture of EfficientNet is based on the AutoML MNAS framework, featuring a mobile inverted bottleneck convolution similar to MobileNet v2 but larger due to increased FLOPS. The basline model is scaled up to create the family of EfficientNets [23]. In our project we utilized EfficientNetsb1 which achieved a 79.1% Top1 Accuracy in the ImageNet dataset with 7.8M parameters.

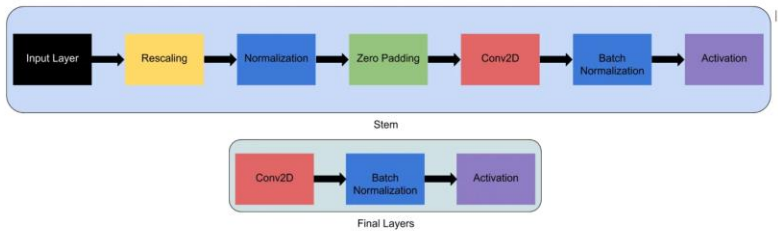
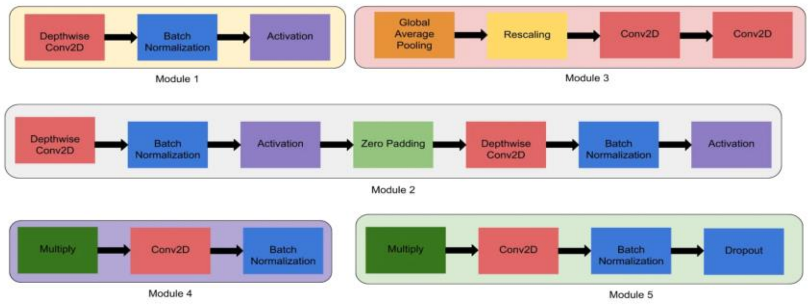
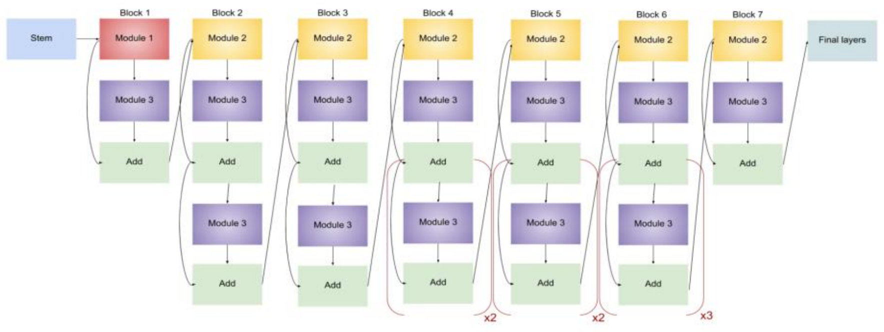


Figure (6) stem and Final layers architecture [24]



[7] Models' architecture in EfficientNetB1 [24]



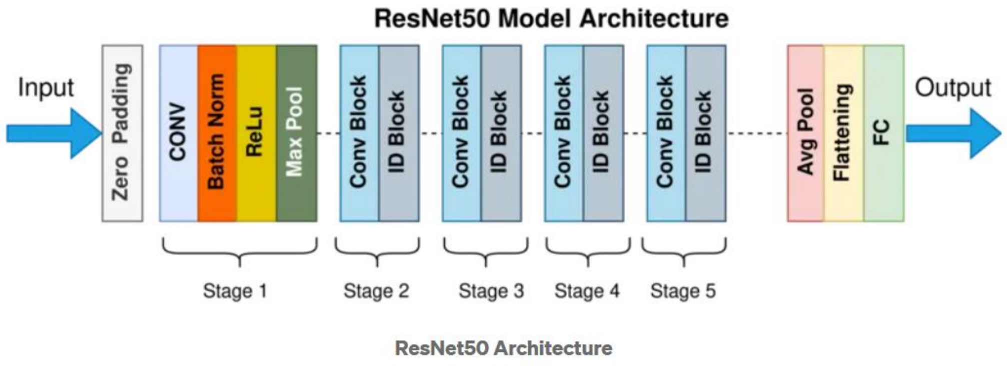
[8] Architecture for EfficientNet-B1 [24]

ResNet50:

ResNet50 is a deep convolutional neural network (CNN) architecture that was developed by Microsoft Research in 2015. It is a variant of the popular ResNet architecture, which stands for "Residual Network." The "50" in the name refers to the number of layers in the network, which is 50 layers deep [19].

ResNet50 is a robust image classification model renowned for achieving state-of-the-art results on large datasets. Its key innovation lies in using residual connections. Overcoming the vanishing gradients problem and enabling the learning of deeper architectures. The architecture comprises convolutional blocks for feature processing, and fully connected layers are followed by batch normalization and RelU activation, extracting essential images features. The identity block preserves input information, while the convolutional block incorporates a 1*1 convolutional layer to reduce filter numbers. The fully connected layers produce the final classification, and a softmax activation function generates class probabilities [19].

Figure (9): ResNet50 Model Architecture [20]



Model Explainability with Grad Cam:

Grad-CAM (Gradient -weighted Class Activation Mapping) is a technique used in the field of computer vision, specifically in deep learning models based on CNNs. Addresses the challenge of interpretability in these complex models. By highlighting the important regions in an input image that contributes to the predictions.

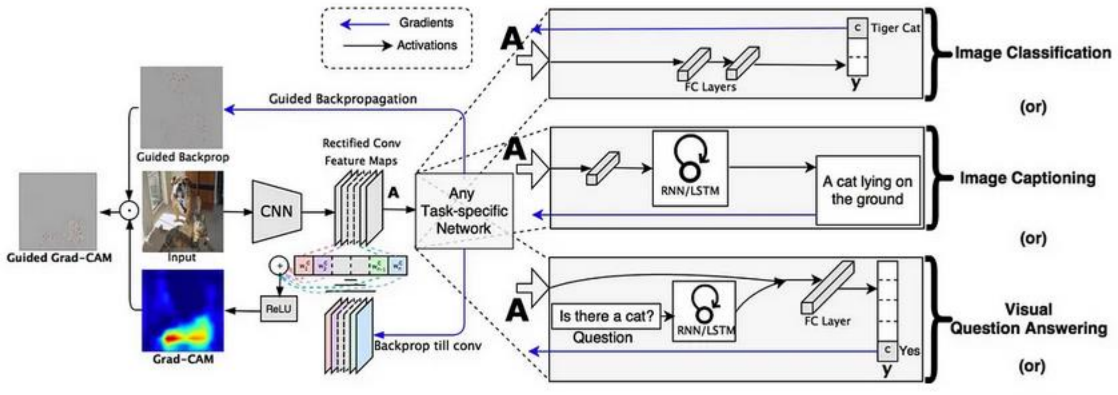


Figure (10): the architecture of grad cam

The importance of using Grad-CAM in Deep Learning

Grad -CAM is crucial in deep learning for enhancing interpretability, which is essential for understanding and trusting the decisions made by complex CNN models without compromising their accuracy. It provides insights into model predications, helping to debug, improve performance, and interpretability by offering visual explanations without altering the model's architecture. This transparency is vital in applications like medical diagnoses or autonomous vehicles, where AI decisions have significant implications. Grad-CAM's class activation maps localize important regions in an image, aiding in the visualization and understanding of model decisions [25].

Results & Discussion:

Cross-validation:

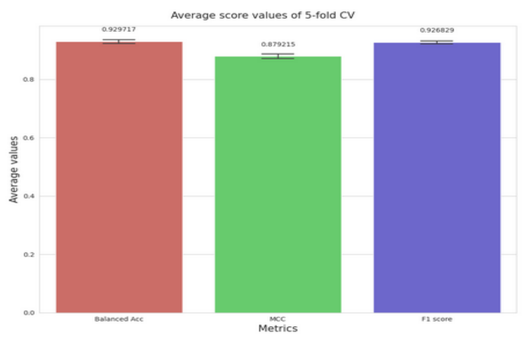
Tabel 1 and Figure (11) present a comparison of the model's 5-fold average performance on consistent train-validation folds, including standard deviation. In general, ResNet50 exhibits the top average performance. Despite these distinctions, there is minimal variation in performance across the various models.

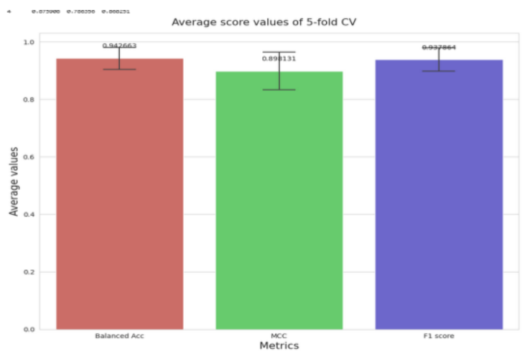
	ResNet50	Eff.Netb1	VGG16
n.parameters	23,516,228	6,518,308	134,276,932
Balanced acc	0.942	0.929	0.819
MCC	0.893	0.879	0.729

F1-score	0.937	0.926	0.825
Average Training time per split	301.570 sec	730.737 sec	800.744 sec

Table 1: Average Classification metrics of vision models on the stratified 5-fold validation splits.

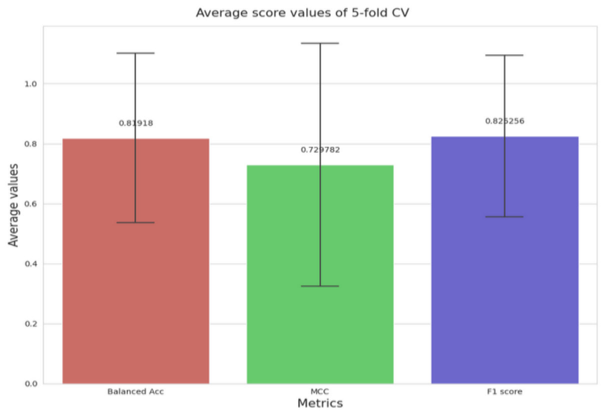
Figure (11) Average Classification metrics of vision models on the stratified 5-fold validation splits.





(A) EfficicentNetb1

(B) ResNet50



(c) VGG16

The performance of final model in the test data:

The evaluation of the final models' performance on the held-out test set, trained both without data augmentation and with data augmentation, is presented in Tables 2 and 3, respectively.

Remarkable performance is observed across all four models, with ResNet50 consistently achieving the highest scores in all three validation metrices for both the regular and augmented datasets. While the EfficientNetb1 and VGG16 show improvements across all metrices with augmented data.

The positive impact of data augmentation on learning and generalization is evident from the comparison of training curves in Figure (15). In the case of training without data augmentation, the corresponding training curves indicate faster overfitting. Conversely, in the case of training with data augmentation, these curves closely follow the training curves, reducing overfitting.

Further insight into misclassifications is gained by comparing the test set confusion matrices figure (16) of models trained with data augmentation. RseNet50 performs exceptionally well for classes “MildDemented", "ModerateDemented", while all models struggle more with classes NonDemented, and VeryMildDemented and often confused them. The similarity between images in these classes, as observed in figure (3) suggests that their features maps could resemble each other, leading to missclassifications by models.

We also observed that ResNet50 model exhibits reduced confusion between the two classes in the augmented dataset.

	ResNet50	Eff.Netb1	VGG16
n.parameters	23,516,228	6,518,308	134,276,932
Balanced acc	0.955	0.885	0.723
MCC	0.899	0.782	0.563
F1-score	0.939	0.862	0.7099
Training time 10 epochs	382.213 sec	364.321 sec	465.539 sec

Table 2: Classification metrics of vision models on the test set

	ResNet50	Eff.Netb1	VGG16
n.parameters	23,516,228	6,518,308	134,276,932
Balanced acc	0.986	0.941	0.771
MCC	0.969	0.875	0.701
F1-score	0.981	0.922	0.814
Training time 20 epochs	1058.880 sec	774.301 sec	927.131 sec

Table 3: Classification metrics of vision models with Data Augmentation on the test set

Grad Cam Heatmap:

Grad-Cam generated heatmaps are depicted in Figure (17), illustrating the results obtained from CNN-based models using one image per class from our dataset. The target for heatmap generation corresponds to the class to which each image should be classified.

Upon visualizing these heatmaps, it becomes evident that in certain cases, the models successfully identify the Alzheimer stage while disregarding the background, basing their decisions on the microorganism's shape.

However, it is apparent that in instances like class MildDemented in VGG16 model, The model struggle to recognize the Alzheimer's stage features, leading to not detecting any features.

Notably, when the feature of the stage is detected, such as in class MildDemented in ResNet50 model, the models focus on approximately the same image regions, albeit with slight variations.

These observations contribute to a better understanding of why models encountered challenges in specific classes and offer insights into examples that perplexed them. By contrasting misclassified instances with correctly classified ones, the decision- making process of the models becomes more transparent, aiding research in identifying areas for model improvement.

Conclusions:

Our discoveries affirm that Deep Learning models, particularly when employed in transfer learning scenarios, exhibit exceptional performance in image classification tasks. Additionally, the DL models demonstrate superior generalization capabilities on unseen data from the same distribution, surpassing the best ML classifier by over 40%. These outcomes align with existing literature and contemporary research findings.

A noteworthy outcome of this study is the superior performance of ResNet50 outperforms all other models when trained on both the original and augmented dataset. These findings provide a foundation for further exploration, including experimentation with diverse data augmentation strategies and model architectures.

References:

[1] Silva, M.V.F., Loures, C.d.M.G., Alves, L.C.V. *et al.* Alzheimer's disease: risk factors and potentially protective measures. *J Biomed Sci* **26**, 33 (2019). https://doi.org/10.1186/s12929-019-0524-y.

[2] 27. Li, X, Feng, X, Sun, X, Hou, N, Han, F, and Liu, Y. Global, regional, and national burden of Alzheimer's disease and other dementias, 1990-2019. *Front Aging Neurosci*. (2022) 14:937486. doi: 10.3389/fnagi.2022.937486.

[3] Houmani et al., 2018; Popuri et al., 2020; Sun et al., 2020; Huggins et al., 2021

[4] Sarasso E., Gardoni A., Piramide N., Volontè M.A., Canu E., Tettamanti A., Filippi M., Agosta F. A Multiparametric MRI Study of Structural Brain Damage in Dementia with Lewy Bodies: A Comparison with Alzheimer's Disease. Park. Relat. Disord. 2021;91:154–161. doi: 10.1016/j.parkreldis.2021.09.003

[5] Liu, S., Liu, S., Cai, W., Pujol, S., Kikinis, R., and Feng, D. (2014). "Early diagnosis of Alzheimer's disease with deep learning," in *Proceedings of the IEEE International Symposium on Biomedical Imaging* (New York, NY: IEEE).

[7] Pan, D., Zeng, A., Jia, L., Huang, Y., Frizzell, T., & Song, X. (Year). Early Detection of Alzheimer's Disease Using Magnetic Resonance Imaging: A Novel Approach Combining Convolutional Neural Networks and Ensemble Learning. Frontiers in Neuroscience. (2020). Sec. Brain Imaging Methods, Volume 14. https://doi.org/10.3389/fnins.2020.00259

[8] Simic G., Stanic G., Mladinov M., Jovanov-Milosevic N., Kostovic I., Hof P. Does Alzheimer's Disease Begin in the Brainstem? Neuropathol. Appl. Neurobiol. 2009;35:532–554. doi: 10.1111/j.1365-2990.2009.01038.x

[9] Farina FR, Emek-Savaş DD, Rueda-Delgado L, Boyle R, Kiiski H, Yener G, Whelan R. A comparison of resting state EEG and structural MRI for classifying Alzheimer's disease and mild cognitive impairment. Neuroimage. 2020;215:116795

[10]"Alzheimer mri preprocessed dataset," (Accessed on 07/15/2022).

[Online]. Available: https://www.kaggle.com/datasets/sachinkumar413/

alzheimer-mri-datase.

[11] Hammad M., Abd El-Latif A.A., Hussain A., Abd El-Samie F.E., Gupta B.B., Ugail H., Sedik A. Deep Learning Models for Arrhythmia Detection in IoT Healthcare Applications. Comput. Electr. Eng. 2022;100:108011. doi: 10.1016/j.compeleceng.2022.108011.

[12] Hammad M., Bakrey M., Bakhiet A., Tadeusiewicz R., Abd El-Latif A.A., Pławiak P. A Novel End-to-End Deep Learning Approach for Cancer Detection Based on Microscopic Medical Images. Biocybern. Biomed. Eng. 2022;42:737–748. doi: 10.1016/j.bbe.2022.05.009. [CrossRef] [Google Scholar] [Ref list]

[13] Jabeen F., Rehman Z.U., Shah S., Alharthy R.D., Jalil S., Khan I.A., Almohammedi A., Alhumaidi A.S., Abd El-Latif A.A. Deep Learning-Based Prediction of Inhibitors Interaction with Butyrylcholinesterase for the Treatment of Alzheimer's Disease. Comput. Electr. Eng. 2023;105:108475. doi: 10.1016/j.compeleceng.2022.108475. [CrossRef] [Google Scholar] [Ref list]

[14] Wani M.A., ELAffendi M.A., Shakil K.A., Imran A.S., Abd El-Latif A.A. Depression Screening in Humans with AI and Deep Learning Techniques. IEEE Trans. Comput. Soc. Syst. 2022 doi: 10.1109/TCSS.2022.3200213. [CrossRef] [Google Scholar] [Ref list]

[15] Loddo A, Buttau S, Di Ruberto C. Deep learning based pipelines for Alzheimer's disease diagnosis: A comparative study and a novel deep-ensemble method. Comput Biol Med. 2022 Feb;141:105032. doi: 10.1016/j.compbiomed.2021.105032. Epub 2021 Nov 21. PMID: 34838263.

[16] Tang Y, Xiong X, Tong G, Yang Y, Zhang H. Multimodal diagnosis model of Alzheimer's disease based on improved Transformer. Biomed Eng Online. 2024 Jan 19;23(1):8. doi: 10.1186/s12938-024-01204-4. PMID: 38243275; PMCID: PMC10799436.

[17] Karen Simonyan and Andrew Zisserman."Very deep convolutional networks for largescale image recognition". In: arXiv preprint arXiv:1409.1556 (2014).

[19] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition*. Microsoft Research. {kahe, v-xiangz, v-shren, jiansun}@microsoft.com

[18] Manali Shaha and Meenakshi Pawar. "Transfer learning for image classification". In: 2018 second

international conference on electronics, communication, and aerospace technology (ICECA). IEEE.2018, pp. 656–660.

[20] https://medium.com/@nitishkundu1993/exploring-resnet50-an-in-depth-look-at-the-model-architecture-and-code-implementation-d8d8fa67e46f.

[21] https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918

[22] Adam Paszke et al. "Automatic differentiation in pytorch". In: (2017).

[23] Tan, M., & Le, Q. v. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. 36th International Conference on Machine Learning, ICML 2019, 2019-June.

[24] https://towardsdatascience.com/complete-architectural-details-of-all-efficientnet-models-5fd5b736142

[25] Ramprasaath R Selvaraju et al. “Grad-cam: Visual explanations from deep networks via gradient-based localization”. In: Proceedings of the IEEE international conference on computer vision. 2017, pp. 618–626.

Data availability:

The dataset utilized in this study is available in this Kaggle link:

https://www.kaggle.com/datasets/tourist55/alzheimers-dataset-4-class-of-images

Figures:

Figure (12) VGG16 the loss and accuracy curves on the stratified 5-fold validation splits

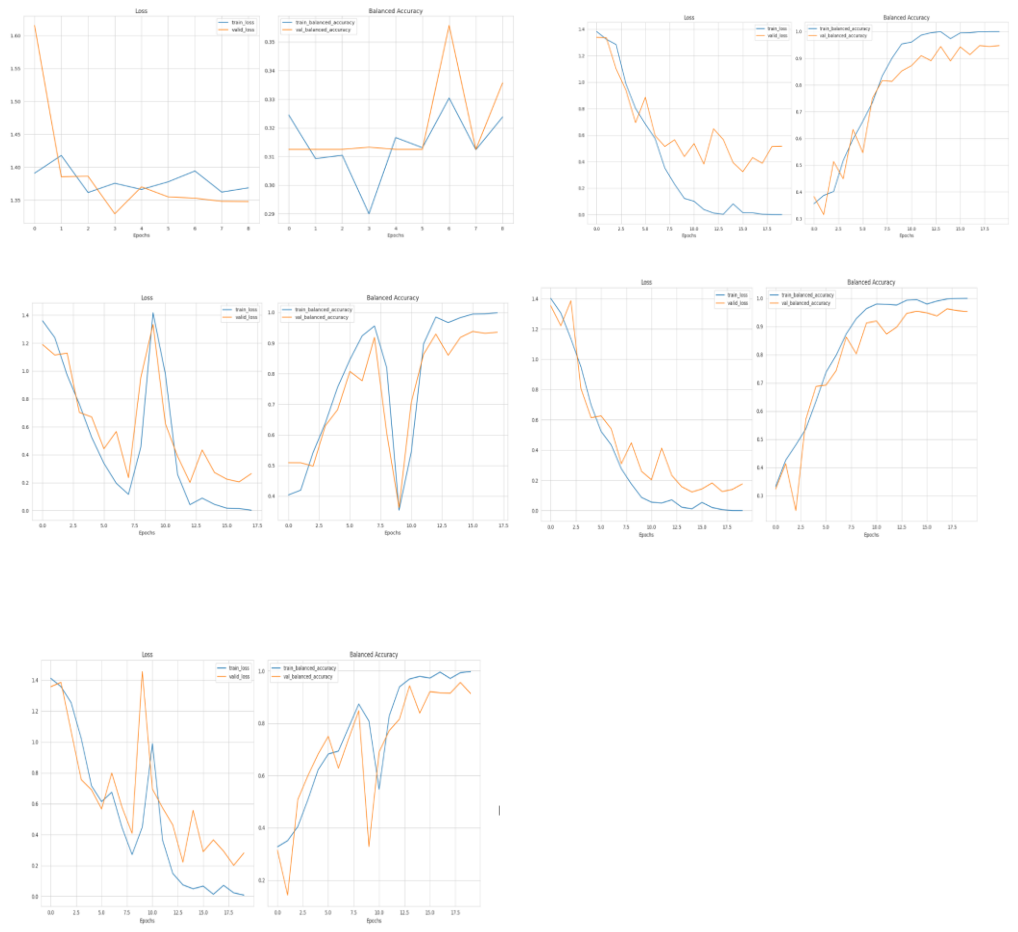


Figure (13) ResNet50 the loss and accuracy curves on the stratified 5-fold validation splits:

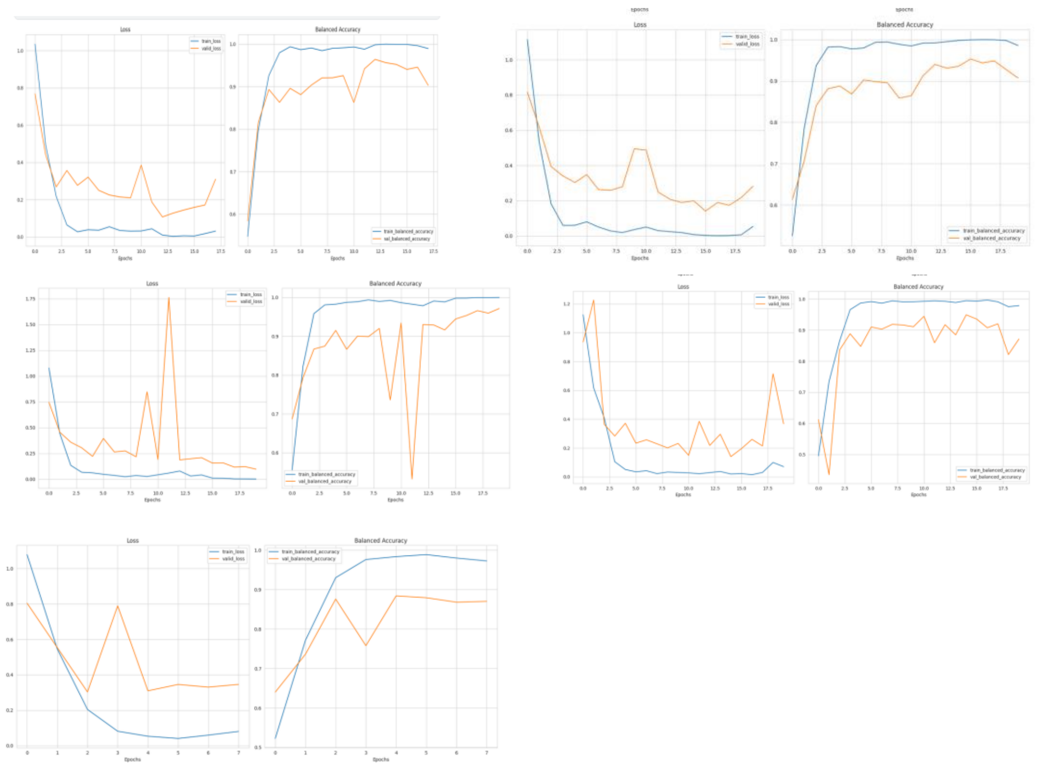
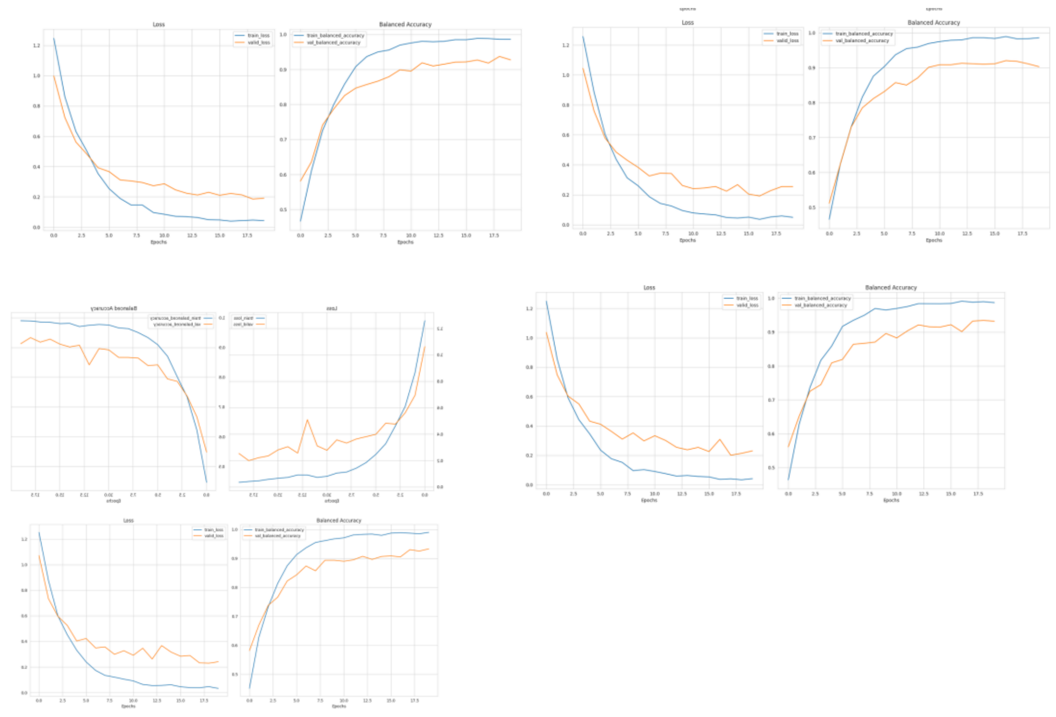
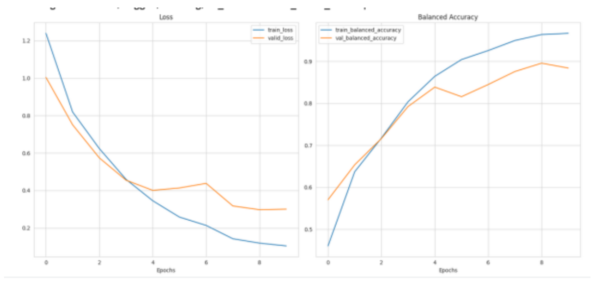


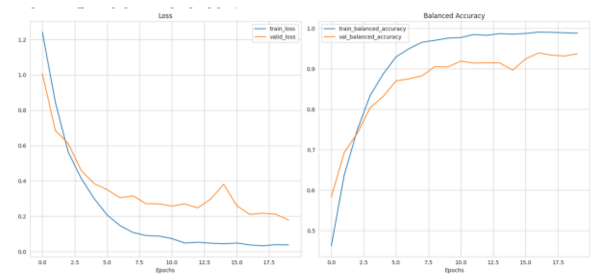
Figure (14) EfficientNet the loss and accuracy curves on the stratified 5-fold validation splits:



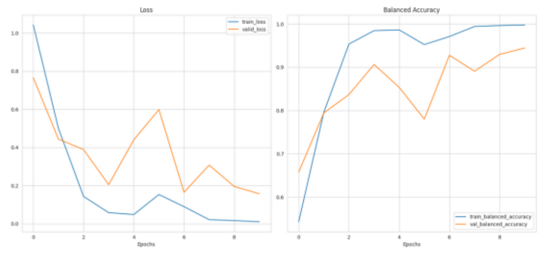
Figure(15) Comparison between the performance of the final model with and without the augmented dataset



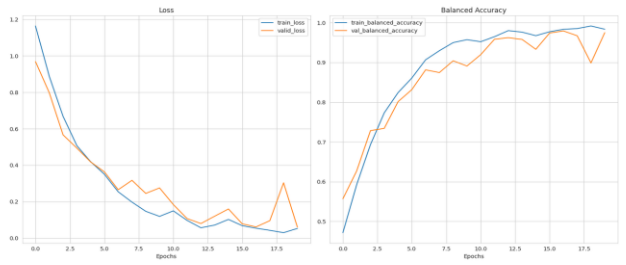
(A) EfficientNet loss and accuracy curves without augmented data



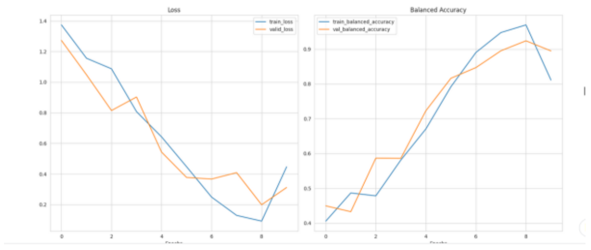
(B) EfficientNet loss and accuracy curves on the augmented data



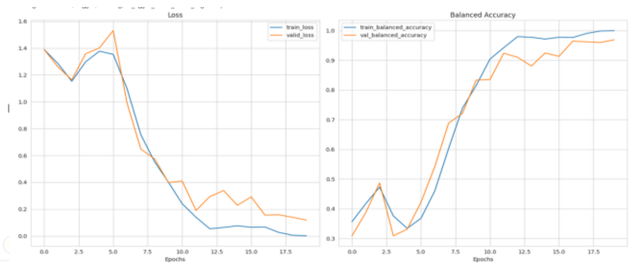
(C) ResNet50 loss and accuracy curves without augmented data



(D) ResNet50 loss and accuracy curves on the augmented data

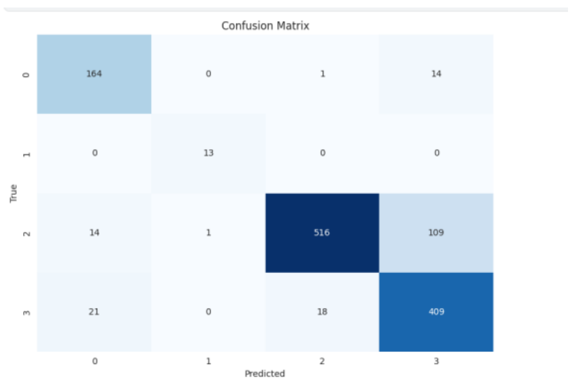


(E) VGG16 loss and accuracy curves without augmented data

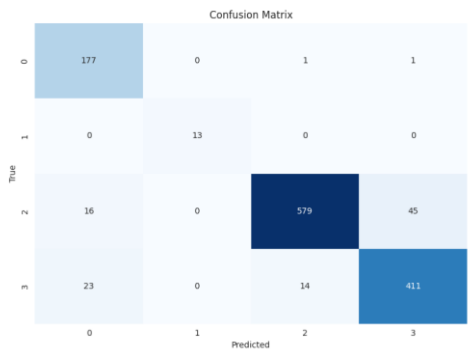


(F) VGG16 loss and accuracy curves on the augmented data

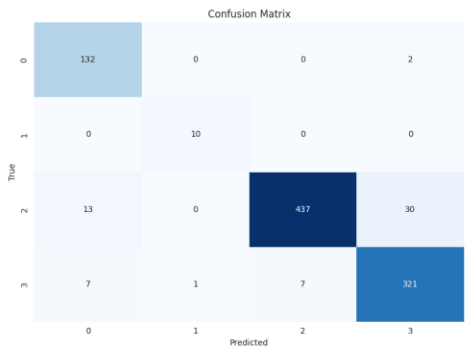
Figure(16) The comparison between confusion with and without augmented data:



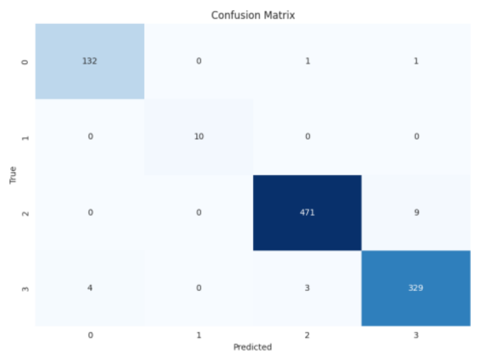
(A) EfficientNet Confusion matrix without augmented data



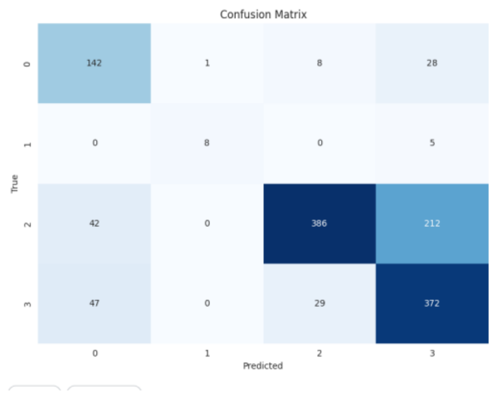
(B) EfficientNet Confusion matrix with augmented data



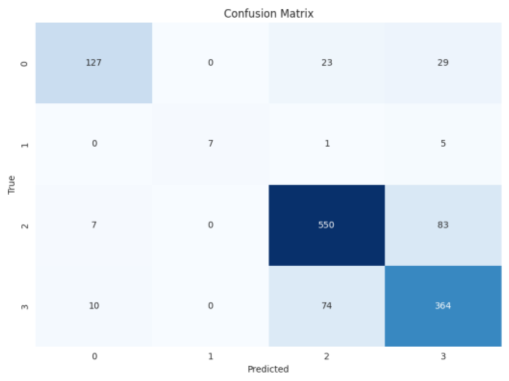
(C)ResNet50 Confusion matrix without augmented data



(D) ResNet50 Confusion matrix with augmented data

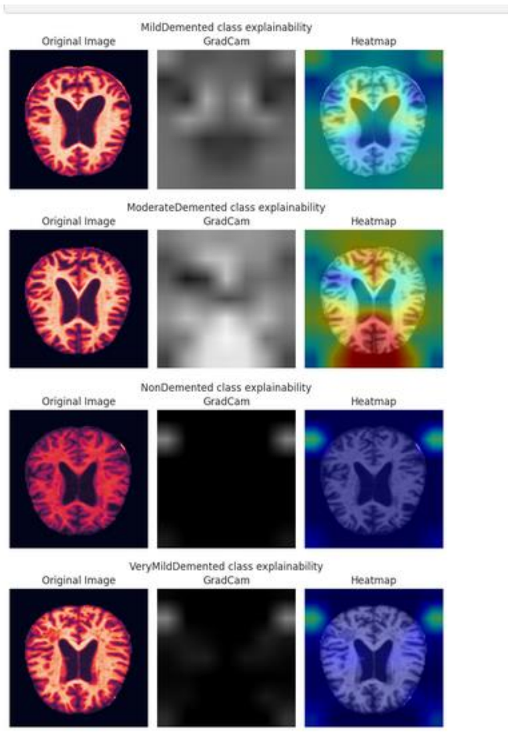


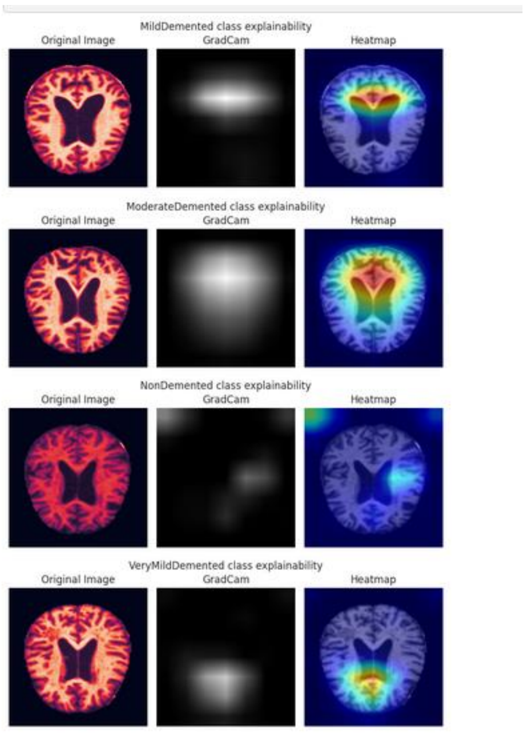
(E) VGG16 Confusion matrix without augmented data



(F) VGG16 Confusion matrix with augmented data

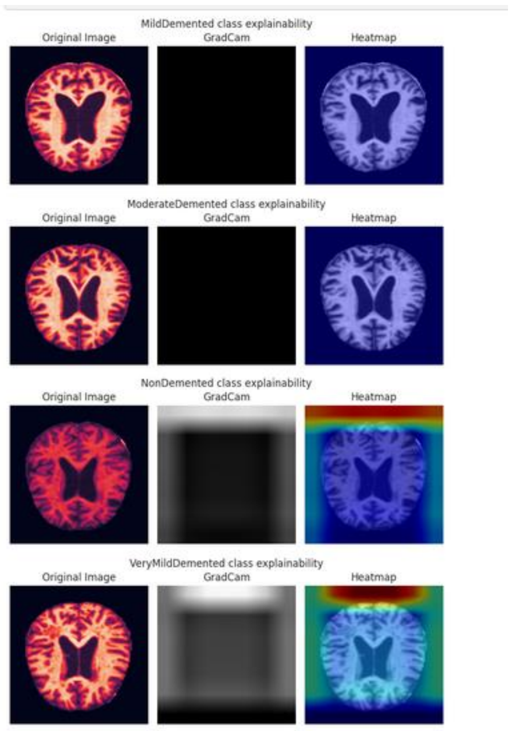
Figure(17) Visualization of the Grad-Cam Heatmaps produced on the CNN-based models.


MildDemented class explainability
Original Image
GradCam
Heatmap
ModerateDemented class explainability
Original Image
GradCam
Heatmap
NonDemented class explainability
Original Image
GradCam
Heatmap
VeryMildDemented class explainability
Original Image
GradCam
Heatmap


MildDemented class explainability
Original Image
GradCam
Heatmap
ModerateDemented class explainability
Original Image
GradCam
Heatmap
NonDemented class explainability
Original Image
GradCam
Heatmap
VeryMildDemented class explainability
Original Image
GradCam
Heatmap

(A) EfficientNet

(C) ResNet50


MildDemented class explainability
Original Image
GradCam
Heatmap
ModerateDemented class explainability
Original Image
GradCam
Heatmap
NonDemented class explainability
Original Image
GradCam
Heatmap
VeryMildDemented class explainability
Original Image
GradCam
Heatmap

(B) VGG16