

DEEP LEARNING AND AUGMENTATION ARCHITECTURES FOR IMAGE CLASSIFICATION IN ALZHEIMER'S DIAGNOSIS

Jiawei Zhang¹, Xin Zhang² and Xinyin Miao³

¹ Senior Investment Analyst, PRA Group (Nasdaq: PRAA), Norfolk, Virginia, USA

² Data Scientist, PRA Group (Nasdaq: PRAA), Norfolk, Virginia, USA

³ Senior Data Analyst, American Airlines Group Inc (Nasdaq: AAL), Dallas, Texas, USA

ABSTRACT

This paper utilizes four cutting edge deep learning architectures, namely VGG19, Xception, InceptionV3, and ResNet50, with transfer learning, image augmentation and two layers of regularization to be able to accurately predict the Alzheimer's Disease classes under 33,982 MRI images with a 0.9563 accuracy, 0.9972 roc_auc, and 0.9559 F1 score in the testing scenario. By investigating the internal neural network structures and comparing the prediction performance, it provides the insight of how various deep learning architectures work differently with corresponding deep learning structure layouts. The main contribution of this article also includes the study of image augmentation and deep learning regularization methodologies to enhance architecture performance and reduce transfer learning over-fitting in the Alzheimer's disease image-based classification.

KEYWORDS

Deep learning, Transfer Learning, Image Classification, Neural Network Architecture, Regularization, Augmentation

1. INTRODUCTION

As one of the key reasons causing disability and dependency for elderly people, dementia has been increased by around 10 million new cases annually and influenced over 57 million people worldwide as of 2025 [1]. Contributing to 60-70% dementia, Alzheimer's disease could cause serious consequences including memory loss, concentration and thinking, routine activities disabilities, and even personality and behavioral change [2]. To detect and prevent Alzheimer's diseases early, brain imaging, such as MRI and PET, has been recently developed to help diagnose clusters of amyloid proteins or tau associated with Alzheimer's diseases [3]. However, diagnosing Alzheimer's disease (AD) using brain imaging has its limitations including missing constant atrophic patterns for AD diagnosis, difficulty in distinguishing between subtypes, high cost and dependency on doctor's clinic experience and judgement [4]. Therefore, machine learning and especially deep learning architecture can help to increase the sensitivity of subtype detection while reducing the cost and subjectivity of human diagnosis based on larger amount of training data [5].

This article is specifically designed to make several unique contributions to the transfer learning application in the AD diagnosis as below:

1. Explain the four CNN architectures differences in the neural network structure and use the layout visualization to open the black box for further deep learning study.
 2. Compare the deep learning performance before and after image augmentation to provide a clearer picture of how augmentation would influence image classification in Alzheimer's diagnosis.
 3. Implement two layers of regularization, namely early stopping and Dropout, to apply the regularization and show their influence on transfer learning using CNN architectures.
- By explaining the architecture internal structures, augmentation and regularization in detail, this article will reveal the main manipulations in architectures transfer learning and techniques that can better help in AD diagnosis using MRI brain imaging data.

2. LITERATURE REVIEW

In the AD diagnosis field, many recent researchers have focused on finding various ways to implement machine learning algorithms to it. For example, in the study [6], researchers have implemented the Decision Tree and Support Vector Machine models to detect AD based on numeric data and realized a 0.85 accuracy. In addition to the traditional Machine Learning models, in the study [7], researchers also used the 2D CNN to detect the brain imaging and realized a 87.9% accuracy compared to 85.5% accuracy based on traditional machine learning models including SVM and Random Forest. In the study [8], researchers have utilized the 3D CNN with the bi-directional LSTM algorithm through ensemble methods including simple voting, weighted voting, and stacking, realizing a 96.3% accuracy of prediction through 2,847 patients' data points.

Besides the learning models exploration, data preprocessing and augmentation also contribute to the imaging classification in AD diagnosis as well. For instance, in the study [9], through implementing image augmentation, researchers have improved the F1 score from 53.4% to 88% using vision transformer, of which the performance was higher than Forward Neural Network and Support Vector Machine based on 82 cognitively normal patients and 149 patients with AD as dataset. In the study [10], researchers implemented the image resizing and normalization to standardize the image input size and numeric values, while used the SMOTE technique to create the similar data input to handle the imbalanced data.

3. METHODOLOGY

The purpose of this article is to conduct the cutting-edge CNN architectures, namely ResNet50, VGG19, InceptionV3 and Xception, along with transfer learning, image manipulation, augmentation, and regularization methodologies to handle the data imbalance, increase the prediction accuracy and minimize the overfitting issue during the AD deep learning prediction process.

In this analysis, we've used 6,400 MRI images from Kaggle (original dataset), which were segmented into 4 target diagnosis classes, including NonDemented, VeryMildDemented, MildDemented, and ModerateDemented [11].

Because the original dataset is imbalanced in that only 64 images fall into ModerateDemented class and 896 images fall into MildDemented class, we've also used the augmented dataset provided in Kaggle. Image augmentation is the technique to generate similar image inputs by methodologies such as reshaping, resizing, rotation, brightness adjustment, and flipping to better balance the data distribution. The augmented dataset (augment dataset) contains 33,982 MRI images with 6,464 ModerateDemented images and 8,958 MildDemented images to handle the data imbalance issue [11].

3.1. Image Preprocessing

The inputs are image files, which need to be manipulated into standardized and constant inputs regardless of their original figure size and color for a more stable deep learning model building process. To realize a better data transformation process, five preprocessing steps have been conducted:

1. Image Resizing

We've first resized all input images to be 224*224-pixels width and length. The reason is that such image size is the optimal input for ResNet50 and VGG19 performance and is compatible with most of CNN architectures due to its ability to smoothly transforming the figure into 7*7-pixels within 5 neural network pooling layers.

2. Image Channel Split

The original images were split into "RGB" channels after image resizing to separate the information from three color layers to ensure more details could be captured during the deep learning process.

3. Image Normalization

Because the original values per pixel ranges are positive numbers ranging from 0 to 255, which will constrain the direction of backpropagation and significantly slow down the training convergence [12], the values are normalized to be centered at 0 and represented by their standard deviation for a better convergence performance.

4. One Hot Encoding

The original target variable, which contains 4 classes including NonDemented, VeryMildDemented, MildDemented, and ModerateDemented, has been one hot encoded as 2-dimension matrix so that it will avoid the shortage of ordinal encoding leading a false interpretation of quantitative relationship among different target classes (ie. Labelling 4 categories as 1-4 will lead to a false quantitative interpretation that the fourth category is four times greater than the first category in the deep learning dense layer).

5. Batch

Due to the fact that original dataset contains 6,400 images while augment dataset contains 33,982 images before flattening, the inputs into neural networks are batched as 16 images per batch. The reason is that batching the original data will increase the convergence speed by differentiate all components per batch together. To avoid internal covariate shift due to batch technique, batch normalization was embedded in the neural network framework and will be further explained in the following sections.

After the image preprocessing steps mentioned above, we've created the normalized quantitative image inputs with 3 color channels and 224*224 pixels shape included in a batch size of 16. Such processed data was randomly split into 80% training data and 20% validation data to avoid data leakage before training process.

3.2. Functional Architectures

In this article, we've implemented four cutting-edge CNN functional architectures, namely VGG19, ResNet50, Inception3 and Xception. Although all of them have higher than 90% accuracy on complex image classification work such as ImageNet data validation [13] [14], the architecture and image processing sequences are different due to their internal structures.

To better understand the image processing logic within the black box, we've visualized the beginning parts of their internal layout to better understand their differences and support the performance comparison in our Alzheimer's Disease prediction analysis.

1. VGG19

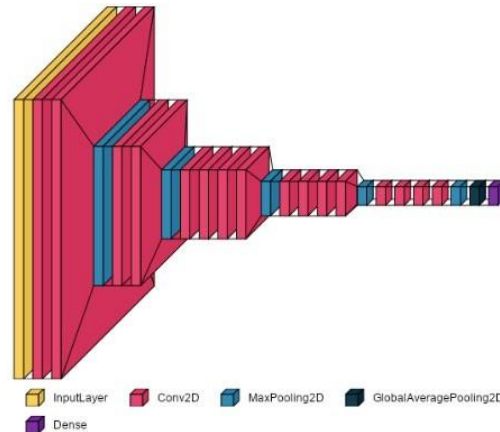


Figure 1. VGG19 Architecture

As shown in Figure 1, VGG19 (structural source code provided in [15]) starts with 2 layers of 2D convolutional layers (Conv2D), then directly down samples into smaller dimension through a 2D max pooling hidden layer, of which the window shape is 2*2 to quickly reduce the dimension from previous layers by half. After two more 2D convolutional layers, another global average pooling layer was introduced to compress the input from previous layers by half, followed by layers of 2D convolutional layers and pooling layers until the final dense layer.

2. ResNet50

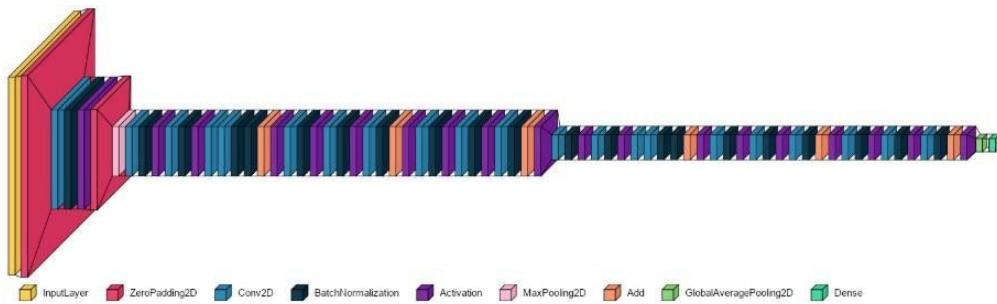


Figure 2. ResNet50 Architecture

As shown in Figure 2, ResNet50's (structural source code provided in [16]) dimensions were reduced by the convolutional 2D layer with filter equal to 64 and kernel size equal to 7, meaning that this layer shrinks the inputs by 7 times while keep the input for next layer to be 64. It proceeded by implementing the batch normalization and relu activation layer to focus on normalization and transformation in early stage of proceeding.

3. Inception V3



Figure 3. Inception V3 Architecture

As shown in Figure 3, the beginning part of Inception V3 (structural source code provided in [17]) starts with a dimension reduction by a 3×3 kernel 2D convolutional layer, which reduces the image inputs dimension by 3 times with padding set up to be valid. The immediate dimension reduction is followed by 8 layers of batch normalization, Relu activation, and convolutional kernel transformation, which is longer than in ResNet50 and VGG19.

4. Xception

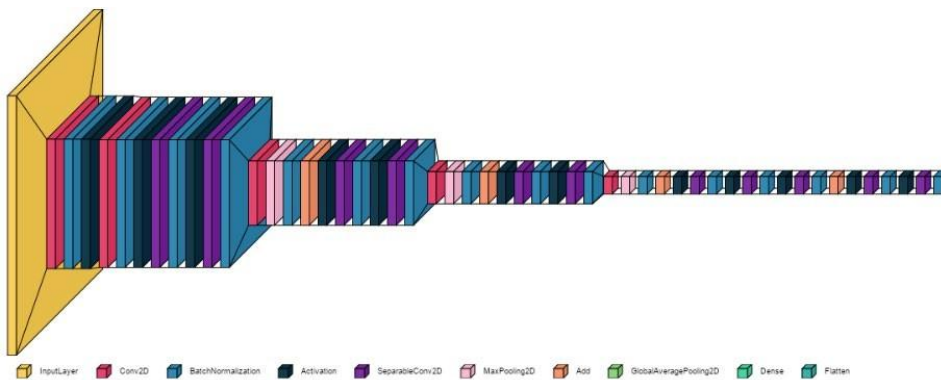


Figure 4. Xception Architecture

As shown in Figure 4, Xception (structural source code provided in [18]) starts with a dimension reduction based on a 2D convolutional layer with 3×3 kernel. Similar to the Inception V3, the dimension reduced input is followed by 8 batch normalization, activation and convolution layers, except 2 additional separable 2D convolution layers, which provides a filter as high as 728 and 1024 to give more kernel transformation and is usually better when learning multiple patterns from the input images.

3.3. Transfer Learning

In all the transfer learning strategies of this article, we've applied three transfer learning hyperparameters (1) weights: "ImageNet" (2) pooling: "avg" (3) first layer frozen: True to all architectures to keep their pre-trained weights based on ImageNet dataset and keep the first layer frozen.

In order to transfer the results of architecture into our use case, we've also conducted six additional approaches to translate the output of architectures into our multi class target variables.

1. Result Flattening

As the original output of architecture are 1,000 classes, we've first flattened the outputs so that such high dimension classification results will generate a transformed new input as into our final prediction layers.

2. Batch Normalization

Because the flattened input will greatly expand the amount of data into our final prediction, we've introduced three batch normalization layers with each layers positioned before a dense layer to increase the convergence efficiency and avoid the internal covariate shift within each mini batch [19].

3. Adam optimization

We've used the Adaptive Moments (Adam) methodology with learning rate equal to 0.001 to make sure a gradual convergence during the training due to the fact that larger learning rate could escape or diverge from the optimal avoid smaller learning rate that could make the convergence slow.

4. Loss Function

Because we've encoded our target classes as one hot encoding pattern to avoid unintended quantitative translation of target variables under ordinal encoding methodology, the loss function is categorical cross entropy of which the formula is as below to align the one hot encoded prediction results:

$$L = - \sum_{i=1}^C y_i \log \hat{y}_i$$

Where C is the number of classes, y_i is true labels (1 or 0), and \hat{y}_i is the predicted probability from deep learning.

5. Gradual Dense

Instead of directly dense the flattened output of 1,000 classes into 4 classes classification, we added three dense layers, of which the outputs are 512, 256, and 4, to gradually compress the inputs into lower dimension without losing too much information caused by aggressive dimension reduction. To align with the multi-class target, we've used Relu as first two dense activation function and softmax as the third activation function.

6. Early Stopping

Due to the fact that this training dataset is relatively small, we've implemented early stopping with monitor equal to loss function and patience of performance improvements to be 5 iterations. Therefore, over the 100 maximum iterations we've set up for all modelling strategies, training will stop when no additional training loss gained after 5 iterations.

3.4. Image Augmentation

As mentioned in dataset description section, the original dataset is highly imbalanced with the distribution shown as below:

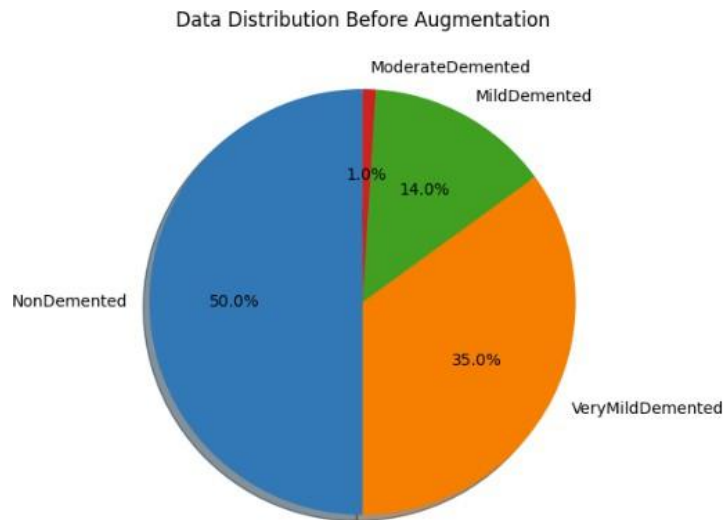


Figure 5. Data Distribution Before Augmentation

As we can see in Figure 5, out of the 6,400 original dataset images, only 1% (64) images are ModerateDemented images and 14% (896) images are MildDemented images.

By using the augmented input images, we've got a much more balanced dataset as shown in Figure 6.

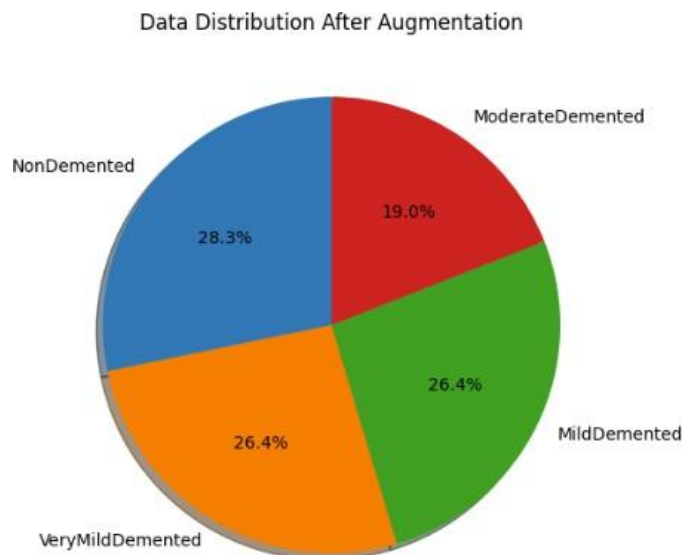


Figure 6. Data Distribution After Augmentation

As we can see in Figure 6, by generating more augmented images, we've got 33,982 images as augment dataset, of which 19% (6,464) are now ModerateDemented images and 26.4% (8,958) are MildDemented images.

3.5. Modelling Strategy

In this article, we'll explore 4 architectures under three scenarios, namely imbalanced dataset, augmented dataset, and augmented dataset with regularization. The imbalanced dataset, as previously shown in Figure 5, is the original dataset without any image generation. The augmented dataset generated similar images to balance the original dataset as shown in Figure 6. To avoid the overfitting on the augmented dataset, we've also introduced a regularization situation where we add additional Dropout layers with drop out rate equal to 10% to regularize the training process in augment dataset. The list of strategies is listed in Table 1:

Table 1. Modelling Strategies

Strategies	Architecture	Training	Regularization	Validation	Testing
RN_Ori	ResNet50	Original Training	N/A	Original Validation	Original Testing
VG_Ori	VGG19				
ICE_Ori	Inception V3				
XCE_Ori	Xception				
RN_Aug	ResNet50	Augment Training	N/A	Augment Validation	Original Testing
VG_Aug	VGG19				
ICE_Aug	Inception V3				
XCE_Aug	Xception				
RN_Aug_Do	ResNet50	Augment Training	Dropout	Augment Validation	Original Testing
VG_Aug_Do	VGG19				
ICE_Aug_Do	Inception V3				
XCE_Aug_Do	Xception				

As shown in Table 1, the training and model evaluation of those 12 strategies are split into two major scenarios

- (1) Trained on 80% of original data and validate / test on 20% of original dataset
- (2) Trained on 80% of augment data, validate on 20% of augment data and test on 20% of original data

Using the same testing original data for all strategies could help test the modelling strategy performance on real original images while avoiding the situation to be tested on artificially generated images and making all strategies to be comparable under the same testing standards for a better comparison.

3.6. Evaluation Metrics

To compare the training, validation and testing performance, we've introduced the metrics as below to compare the convergence speed, validation accuracy, and testing accuracy as below:

- (1) Number of Iteration

As mentioned in 3.3.6, with early stopping set up for a maximum of 100 iterations, the number of iterations were used to compare the convergence time across 12 strategies.

- (2) Loss

Loss in this article refers to the loss metrics, namely categorical cross entropy, that we used in the

deep learning loss function to tune the models. Formula of Loss is shown as in 3.3.4.

(3) Accuracy

Accuracy is the metric to measure how much percentage of testing target is correctly predicted by the classification model.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} * 100\%$$

(4) F1

F1 score is the harmonic mean of precision and recall measuring the model's performance on positive prediction, especially if the target variable is imbalanced.

$$F1 = \frac{2 * TP}{(2 * TP + FP + FN)} * 100\%$$

(5) ROC

ROC_AUC (Receiver Operating Characteristic – Area Under the Curve) is the area under the ROC curve to represent the true positive rate against the false positive rate at various classification thresholds.

(6) Recall

Recall is the metric to measure how much percentage of positive results are correctly labelled by the classification model as positive.

$$Recall = \frac{TP}{TP + FN} * 100\%$$

(7) Precision

Precision is the metric to measure how much percentage of positive predictions are actually true positive results.

$$Precision = \frac{TP}{TP + FP} * 100$$

4. RESULTS

4.1. Convergence Performance

Using early stopping to constrain the overfitting iterations, the convergence number of iterations are shown in Figure 7.

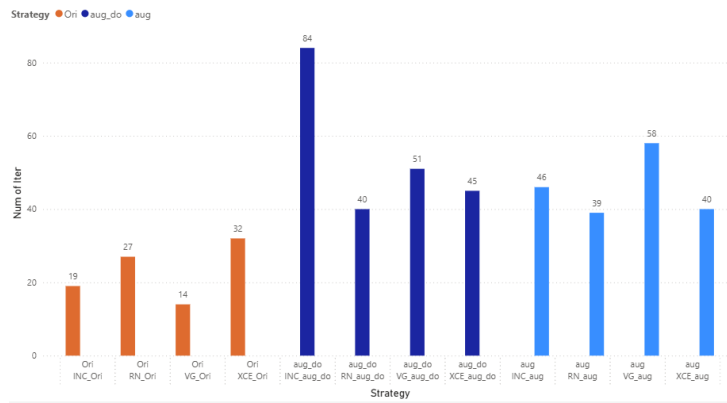


Figure 7. Number of iterations needed to converge

- (1) “ori” refers to original training dataset. (2) “aug” refers to augment training dataset.
- (3) “aug” refers to augment training dataset with dropout layer added

As shown in Figure 7, out of three training scenarios, iterations trained on original training dataset converges faster than on augment training dataset by 22 iterations on average due to the fact that original training dataset is much less than augment training dataset.

Under the same amount of augment training data, adding drop out hidden layer between the dense layers additionally increases the convergence iterations by 9 on average, indicating the time increase due to the additional drop out complexity and computational cost.

4.2. Validation Performance

To compare the validation performance, we’ve calculated the training accuracy, training loss, validation accuracy and validation loss as shown in Figure 8.

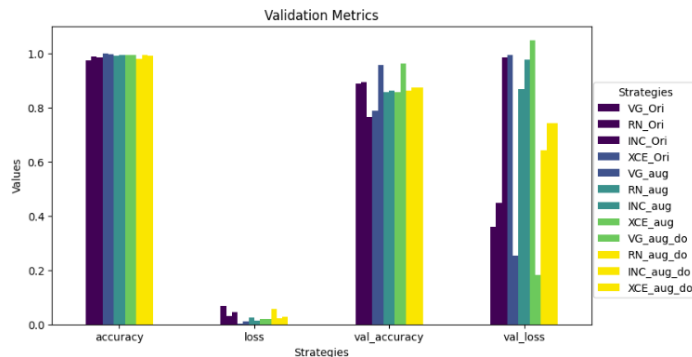


Figure 8. Validation loss and accuracy comparison

As shown in Figure 8, while validation accuracy is comparable among original, augment, and

augment drop out training scenarios, the validation loss is overall lower when conducting drop out strategy compared to augment strategy. The detailed validation performance comparison is shown in Table 2.

Table 2. Validation Performance

	iter	accuracy	loss	val_accuracy	val_loss
VG_Ori	14	0.975	0.068471	0.886719	0.358076
RN_Ori	27	0.988867	0.029454	0.892969	0.446877
INC_Ori	19	0.985937	0.045024	0.764063	0.985244
XCE_Ori	32	1	0.001949	0.786719	0.995238
VG_aug	58	0.996579	0.011012	0.957923	0.252655
RN_aug	39	0.991392	0.025056	0.856849	0.869142
INC_aug	46	0.995292	0.013993	0.861851	0.977421
XCE_aug	40	0.99393	0.017857	0.855672	1.048451
VG_aug_do	51	0.992717	0.020138	0.961454	0.181915
RN_aug_do	40	0.979989	0.056109	0.862586	0.641647
INC_aug_do	84	0.992533	0.02186	0.875092	0.742513
XCE_aug_do	45	0.989627	0.028091	0.872738	0.743111

As shown in Table 2, while comparing the same architecture between original training data and augment training data, the validation accuracy are higher for VGG19, InceptionV3 and Xception, while is 3.5% lower for ResNet50. When drop out was added, all architectures' validation accuracy were increased with a lower validation loss than normal augmentation scenarios.

4.3. Testing Performance

We've also pulled the testing results based on the same original testing data for all strategies. The testing roc_auc score, F1 score, accuracy, precision, and recall are shown in Figure 9.

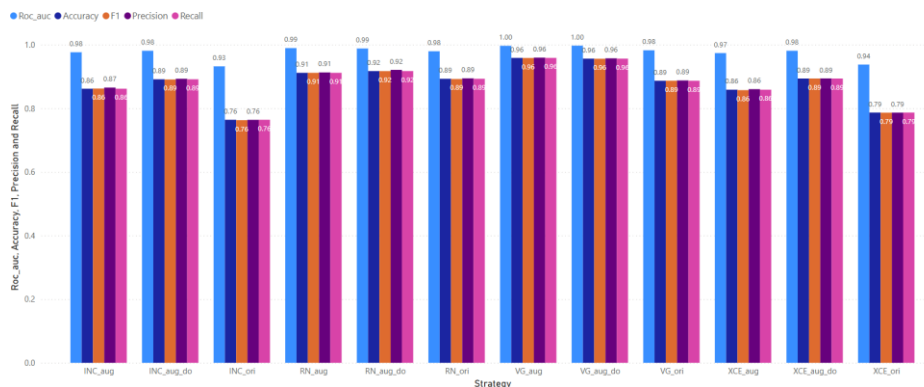


Figure 9. Testing performance comparison among strategies

As shown in Figure 9, tested on the same testing dataset, augmentation trained architectures outperform the original trained architectures in all scenarios. When introducing drop out algorithm, it further increases the roc_auc, F1 score, accuracy, precision and recall scores compared to augmentation training scenario.

The numeric testing performance is shown in Table 3.

Table 3. Testing Performance

	accuracy	roc_auc	F1 score	recall	precision
VG_ori	0.886719	0.98257	0.886829	0.886719	0.887719
RN_ori	0.892969	0.979482	0.892326	0.892969	0.894342
INC_ori	0.764062	0.932074	0.762836	0.764062	0.763878
XCE_ori	0.786719	0.937673	0.786352	0.786719	0.786516
VG_aug	0.958594	0.996675	0.958372	0.958594	0.959396
RN_aug	0.911719	0.989433	0.91175	0.911719	0.91291
INC_aug	0.861719	0.976343	0.862192	0.861719	0.865261
XCE_aug	0.858594	0.973754	0.857261	0.858594	0.860304
VG_aug_do	0.95625	0.997179	0.955932	0.95625	0.957212
RN_aug_do	0.917188	0.988118	0.917103	0.917188	0.920993
XCE_aug_do	0.89375	0.981017	0.89385	0.89375	0.894063
INC_aug_do	0.891406	0.981059	0.890732	0.891406	0.892993

As shown in Table 3, out of all strategies, VG_Aug_Do provides the highest roc_auc score as 0.9972, with 0.9562 testing accuracy, 0.9559 testing F1 score, 0.95625 testing recall, and 0.9572 testing precision.

4.4. Confusion Matrix Performance

When it comes to the detailed prediction performance, we've conducted the confusion matrix comparison in the testing dataset for all 12 strategies categorized by 3 scenarios.

1. Original Strategies

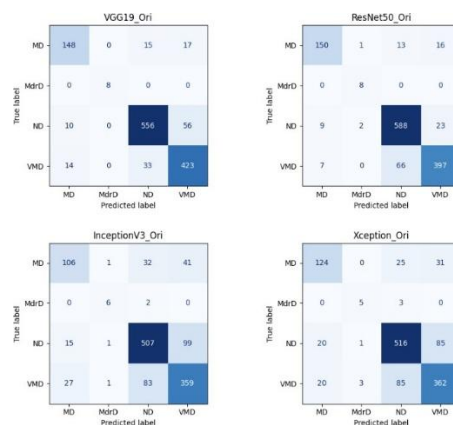


Figure 10. Original Strategies Testing Confusion Matrix

As shown in Figure 10, original strategies are able to accurately predict around 80-90% of major classes such as NonDemented (ND) and VeryMildDemented (VMD). However, due to the imbalanced training data, the architectures of InceptionV3 and Xception cannot predict with a

higher than 80% accuracy for minor classes such as MildDemented (MD) or ModerateDemented (MdrD).

2. Augment Strategies

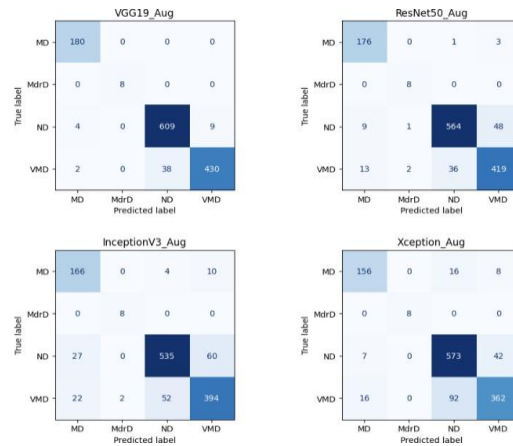


Figure 11. Augment strategies testing confusion matrix

As shown in Figure 11, augment strategies highly improved the testing prediction accuracy in minor classes such as MD and MdrD, where we can see all 4 architectures are able to accurately predict more than 80% of MD and all of MdrD.

Additionally, the augmentation is able to improve the major class prediction in VGG19 by increasing the accurate prediction of ND from 556 to 609 and VMD from 423 to 430.

3. Augment – Dropout Strategies

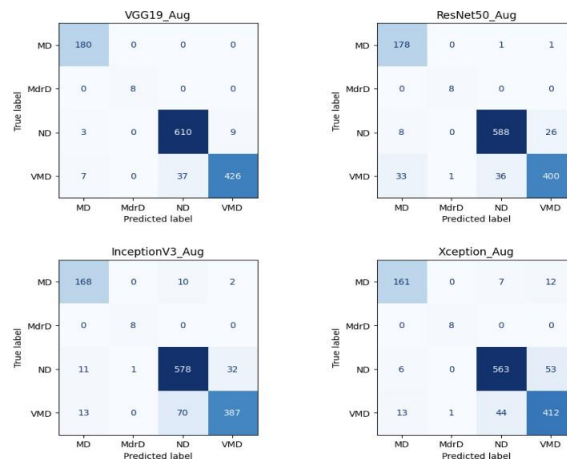


Figure 12. Augment – dropout strategies testing confusion matrix

As we can see in Figure 12, augment – dropout strategies are able to further improve the major class and minor class predictions across all architectures. In all augment-drop out strategies, VGG19 outperforms the other architectures by accurately predicting 100% of MD, 100% of MdrD, 98.07% of MD, and 90.64% of VMD in the testing scenario.

5. CONCLUSIONS

This study compares the 4 cutting edge architectures and through implementing the transfer learning, image augmentation, drop out regularization, it improves the performance from simply using the architectures in original dataset and realizes a 0.9563 accuracy, 0.9972 roc_auc, and 0.9559 F1 score in the testing scenario.

However, such study has its limitation on the size of data. The dataset used in this study contains 33,982 augmented MRI images and 6,400 original MRI images from Kaggle open-source data [11]. Such experimental results could be applied to larger amount of clinical MRI images to further test and enhance the performance before considering the possibility of implementation on clinical application use cases.

In conclusion, this article makes a novel contribution by moving beyond performance-only comparisons to systematically demystify how convolutional neural networks operate and adapt in the context of Alzheimer's disease diagnosis. In addition, it provides one of the few side-by-side evaluations of classification performance before and after image augmentation. The integrated use of early stopping and Dropout as dual regularization strategies uniquely illustrates their individual and combined effects on transfer learning with CNNs. Together, these elements position the article as a methodological bridge between theory and practice, which deliver both interpretability and actionable guidance for improving CNN-based Alzheimer's disease detection using brain MRI data.

REFERENCES

- [1] World Health Organization (WHO). (2025). Dementia Retrieved from: <https://www.who.int/news-room/fact-sheets/detail/dementia#:~:text=Dementia%20is%20caused%20by%20many%20different%20diseases,of%20the%20frontal%20lobe%20of%20the%20brain.>
- [2] Mayo Clinic. (2025). Alzheimer's Disease. Retrieved from: <https://www.mayoclinic.org/diseases-conditions/alzheimers-disease/symptoms-causes/syc-20350447>
- [3] Mayo Clinic. (2024). Diagnosing Alzheimer's: How Alzheimer's is diagnosed. Retrieved from: <https://www.mayoclinic.org/diseases-conditions/alzheimers-disease/in-depth/alzheimers/art-20048075>
- [4] Saikumar Aramadaka, Raam Mannam, Rajagopal Sankara Narayanan, Arpit Bansal, Vishnu R Yanamaladoddi, Sai Suseel Sarvepalli, Shree Laya Vemula. (2023). Neuroimaging in Alzheimer's Disease for Early Diagnosis: A Comprehensive Review. Retrieved from: [https://pmc.ncbi.nlm.nih.gov/articles/PMC10239271/#:~:text=1\)%20Reduced%20hippocampal%20volume%20is,AD%20is%20difficult%20to%20find.](https://pmc.ncbi.nlm.nih.gov/articles/PMC10239271/#:~:text=1)%20Reduced%20hippocampal%20volume%20is,AD%20is%20difficult%20to%20find.)
- [5] Golrokh Mirzaei, Hojjat Adeli. (2022). Machine learning techniques for diagnosis of alzheimer disease, mild cognitive disorder, and other types of dementia. Retrieved from: <https://www.sciencedirect.com/science/article/abs/pii/S1746809421008909>
- [6] J. Neelaveni; M.S.Geetha Devasana. (2020). Alzheimer Disease Prediction using Machine Learning Algorithms. Retrieved from: <https://ieeexplore.ieee.org/abstract/document/9074248>
- [7] Maryam Akhavan Aghdam, Serdar Bozdog, Fahad Saeed & Alzheimer's Disease Neuroimaging Initiative. (2025). Machine-learning models for Alzheimer's disease diagnosis using neuroimaging data: survey, reproducibility, and generalizability evaluation. Retrieved from: <https://link.springer.com/article/10.1186/s40708-025-00252-3>
- [8] Wenyan Liu, Shukai Fan, Guifan Weng. (2025). Multi-Modal Deep Learning Framework for Early Alzheimer's Disease Detection Using MRI Neuroimaging and Clinical Data Fusion. Retrieved from: <https://annalsofappliedsciences.com/index.php/aas/article/view/34/34>
- [9] Lilia Lazli. (2025). Improved Alzheimer Disease Diagnosis With a Machine Learning Approach and Neuroimaging: Case Study Development. Retrieved from: <https://xmed.jmir.org/2025/1/e60866/>
- [10] Hiba A. Alahmed & Ghaida A. Al-Suhail. (2025). AlzONet: a deep learning optimized framework for multiclass Alzheimer's disease diagnosis using MRI brain imaging. Retrieved from:

- <https://link.springer.com/article/10.1007/s11227-025-06924-5>
- [11] Kaggle, Author: Uraninjo. (2022). Augmented Alzheimer MRI Dataset. Retrieved from: <https://www.kaggle.com/datasets/uraninjo/augmented-alzheimer-mri-dataset>
 - [12] Yann LeCun, Leon Bottou, Genevieve B. Orr & Klaus -Robert Müller. (2002). Neural Networks: Tricks of the Trade. Chapter: Efficient BackProp. Retrieved from: https://link.springer.com/chapter/10.1007/3-540-49430-8_2
 - [13] Keras applications. (2025). Keras Applications Available Models. Retrieved from: <https://keras.io/api/applications/>
 - [14] ImageNet. (2020). ImageNet Large Scale Visual Recognition Challenge Dataset. Retrieved from: <https://www.image-net.org/download.php>
 - [15] Keras. (2025). Keras Documents: VGG19.py. Retrieved from: <https://github.com/keras-team/keras/blob/v3.13.1/keras/src/applications/vgg19.py#L20>
 - [16] Keras. (2025). Keras Documents: resnet.py. Retrieved from: <https://github.com/keras-team/keras/blob/v3.13.1/keras/src/applications/resnet.py#L276>
 - [17] Keras. (2025). Keras Documents: inception_v3.py. Retrieved from: https://github.com/keras-team/keras/blob/v3.13.1/keras/src/applications/inception_v3.py#L19
 - [18] Keras. (2025). Keras Documents: xception.py. Retrieved from: <https://github.com/keras-team/keras/blob/v3.13.1/keras/src/applications/xception.py#L19>
 - [19] Sergey Ioffe & Christian Szegedy. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Retrieved from: <https://arxiv.org/pdf/1502.03167>