iNeuron

Alzheimer Disease Classification

Project Report

By

Jatindra Paul

Table of Contents

I. Introduction 3
II. Analysis of existing deep neural networks 3
III. Analysis of designed deep neural networks 6
IV. Analysis of the Training Process 7
I. Analysis and Performance evaluation of DNN 11
VI. References 13

I. INTRODUCTION

In this coursework, medical image datasets, Alzheimer MRI scans are used to create deep learning models for classification problem. This models will be deployed as a web application for end-users, where they can choose the desired classification page and provided models for classification of their uploaded images.

Alzheimer Data set is a collection of MRI images of Brain for identifying any abnormalities. It is originally categorised as normal, benign and malignant but due to low resolution images it is simplified into a binary problem by combining normal and benign as positive and malignant as negative.

A quick overview of both the datasets is given in the Table 1. below:

Table 1. OVERVIEW OF ALZHEIMER IMAGEAND BREASTMNIST DATASETS

Property		Alzheimer MRI Images
Channels		1 (grayscale images)
Task		Binary classification (2 classes) - '0': 'malignant', - '1': 'normal, benign'
Total images		780 split with a ratio of 7:1:2 into training, validation and test set - Train: 546 - Validation: 78 - Test: 156

We will be developing two deep learning models for each of the classification problem, one model would be based on any existing deep neural network and the other model would be our own deep neural network.

The web application and all the models were developed in Python using mainly the following important packages:

- Keras for deep learning models
- Flask for web application
- Numpy, Scikit-learn, Matplotlib – provide visualisation and other utilities

II. ANALYSIS OF EXISTING DEEP NEURAL NETWORKS

Deep learning is a subset of machine learning consisting of neural networks with three or more layers, these networks attempt to mimic human brain. These neural networks are commonly known as deep neural networks, capable of training themselves by automatically determining features from the input that are most important to solve a given task.

Computer vision tasks such as image classification, object detection and segmentation. Convolutional neural networks (CNN) among the other different kinds of neural networks such as recurrent neural networks (RNN), long short-term memory (LSTM) and artificial neural network (ANN) are most popular for solving computer vision problems.

CNN are inspired by the visual cortex of eyes; it consists of three main types of layers:

- **Convolutional layer**
  These capture spatial features in an image by using filters and matrix dot product on the input image. These layers can be followed by additional convolutional layers or pooling layers, but ultimately a fully connected layer is the last layer. ReLu is the default activation function in these layers.

- **Pooling layer**
  These are responsible to down-sampling, in order to reduce complexity and dimensions which assists to avoid overfitting and improve efficiency. ReLu is also the default activation function in these layers as well. It has two main types:
  - Max Pooling: As the filter moves across the input, it selects the maximum value pixels.
  - Average Pooling: As the filter moves across the input, it calculates an average pixel value.

- **Fully connected layer**
  These layers perform classification task based on the features extracted, each output node is connected directly to nodes in the previous layers. It usually leverages 'softmax' or 'sigmoid' activation functions to classify inputs appropriately.

CNNs can thus classify using images or video frames effectively, as the input data progresses through the layers it detects more and more complex patterns that helps to distinguish or predict accurately.

There are many such popular existing DNN models such as LeNet, AlexNet, VGG, Inception, Xception etc. These existing models can be utilised in the following two ways:

- Training from scratch with our dataset to perform classification.
- Applying transfer learning where a trained model a similar computer vision task helps to extract features for our classifier.

We chose to apply transfer learning using **VGG16** for Alzheimer Image

Neural networks with deep architectures generally need large datasets and may take days or weeks to train. It's hard to obtain such datasets in practical applications and uneconomical to train such deep models from scratch, this gave way to the concept of transfer learning. It involves re-using model weights from pre-trained models that were developed for standard computer vision benchmark datasets, such as ImageNet.

VGG16 model was developed by the Visual Graphics Group (VGG) at Oxford (2014). It is an extremely homogeneous architecture that only performs 3x3 convolutions and 2x2 pooling from the beginning to the end. Its main contribution was showcasing that the depth of the network is critical for a good performance.

VGG16 uses smaller convolutional filters 3x3 which reduces the possibilities of overfitting, it also represents an optimal size as smaller filters will be unable to detect any left-right or top-down information. Moreover, multiple smaller convolutional layers provides more non-linear activations which improves the network efficiency as well as converges quickly.

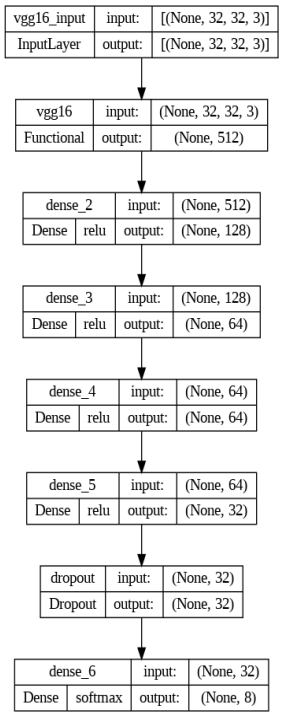

Fig 1. VGG-16 used for Transfer Learning for Alzheimer

Quick outline of our overall VGG16 transfer learning model used for Alzheimer Image(refer to Fig 1):

- **Input –** A 32x32x3 image input is passed to this model, for this we have padded our dataset images accordingly.
- **Convolutional layer blocks –** VGG16 uses the smallest possible receptive field of 3×3, also 1×1 convolution filter is applied as the input's linear transformation. Rectified Linear Unit Activation Function (ReLU) is used in these blocks since it's not affected by vanishing gradients and computationally efficient.
- **Fully connected layers –** We excluded the VGG16 classifier, and added our own dense layers with ReLu with a dropout layer (dropping 50% connections to help avoid overfitting) for classification, the final output layer has 8 nodes integrated with a 'softmax' activation function.

Xception model by Google (2017) on the other hand takes CNNs to the next level by utilising the concept of depth-wise separable convolution, where a spatial convolution is performed independently over each channel of input, followed by a pointwise convolution, i.e., a 1x1 convolution, projecting the channels output by the depth-wise convolution onto a new channel space.

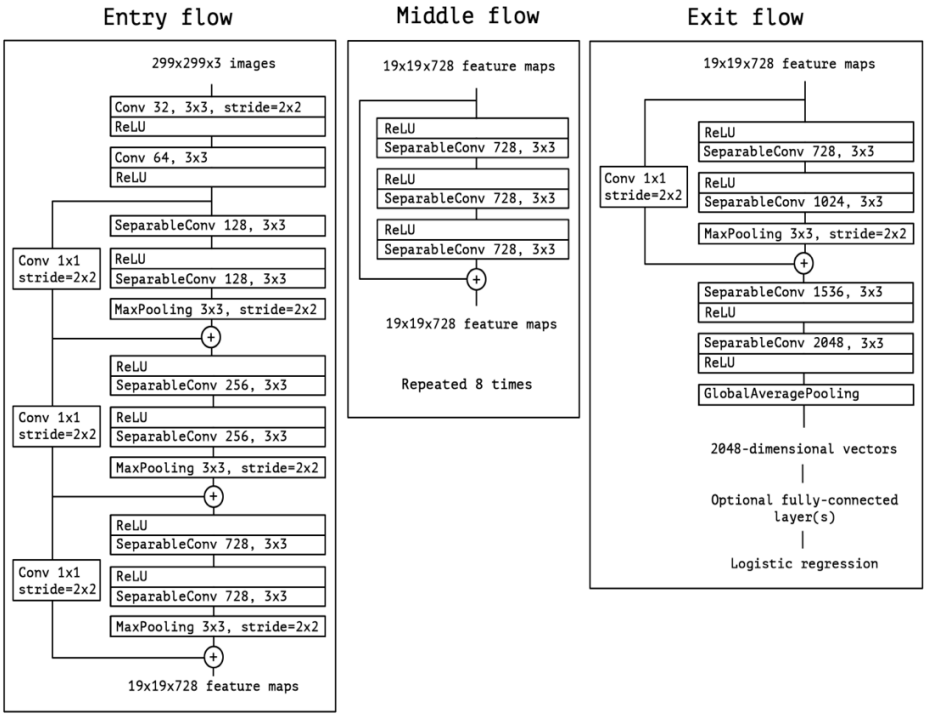

Fig 2. Xception architecture (Source: Image from the original paper[12])

A quick overview of both the existing DNNs is given in the Table 2. below:

Table 2. OVERVIEW OF VGG16 AND XCEPTION DEEP NEURAL NETWORK

Property	VGG16	Xception
Overview	VGG16 follows a simple convolutional neural network structure with emphasis on making denser networks for more accuracy and using smaller filters to avoid overfitting.	Xception stands for extreme inception, it follows the concept of separable convolutions by doing

		depth-wise followed by pointwise convolutions reducing the overall computation cost and memory.
Developers	Visual Graphics Group (VGG) at Oxford (2014)	François Chollet at Google Inc. (2017)
Depth/Layers	16	81
Trainable parameters	138.4M	22.9M
Benchmark performance (ImageNet)	Top-1 accuracy = 71.3% Top-5 accuracy = 90.1%	Top-1 accuracy = 79.0% Top-5 accuracy = 94.5%

VGG16 was one of the top models from the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) - 2014 competition, and remains an attractive but yet a simple deep neural network architecture

III. ANALYSIS OF DESIGNED DEEP NEURAL NETWORKS

The Allzheimer datasets are image datasets, using convolutional neural networks that perform well to train and predict these datasets. One such convolutional neural network is designed for each of the image datasets. The data is pre-processed in a way that is compatible with feeding the images to the network. The architecture of the designed models is discussed below:

- **Model**: A Sequential model is used to train the neural network. Adding more convolutional layers on top of each layer helps the network identify complex patterns and features in the input image.
- **Input Layer:** The first layer is the input layer, which takes the size of the input image as a parameter to understand the incoming image dimensions.
- **Convolutional Block:** Convolutional blocks containing combinations of multiple convolutional layers, max-pooling and dropout layers with a different number of nodes, activation layers and filter sizes are added to the network to help the designed network train and predict the dataset.
- **Dropout:** When the network is complex and the size of the dataset is small, there is a good chance that the network tries to learn noises in the data and eventually overfits the dataset. Dropout layers are added to regularize the network and to handle overfitting. Dropout layers randomly drop a few calculated outputs from the layers, so it gets difficult for the network to overfit the data by learning from noises in the data.[1]

A. AlheimerImages:

- Alzheimer Image has input images of size 28x28x1 that is it has grayscale images of breast scans. A combination of convolutional, maxpooling, dense and dropout layers are used to build a neural network to train and predict the Breast dataset. Designed architecture is shown in the figure.

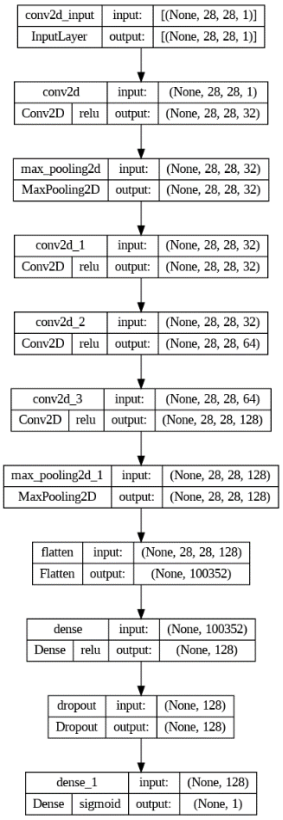

Fig 4. Architecture of model built from scratch on Alzheimer dataset.

Both models use different input shapes, are built using a different number of layers and are trained on different numbers of trainable parameters. The table lists all the key features of the layers used to design the models for both datasets.

Table 3. Design of the layers used in both models

Property		**Alzheimer Images**
Input shape		28x28x1
Number of layers		10
Trainable parameters		12,947,233
Activation function		sigmoid

IV. Analysis of the Training Process

The below flow chart explains the process of training followed in a convolutional neural network. Both data sets- Alzheimer Imageand BreastMNIST according to their corresponding models pretty much go through a similar kind of flow.

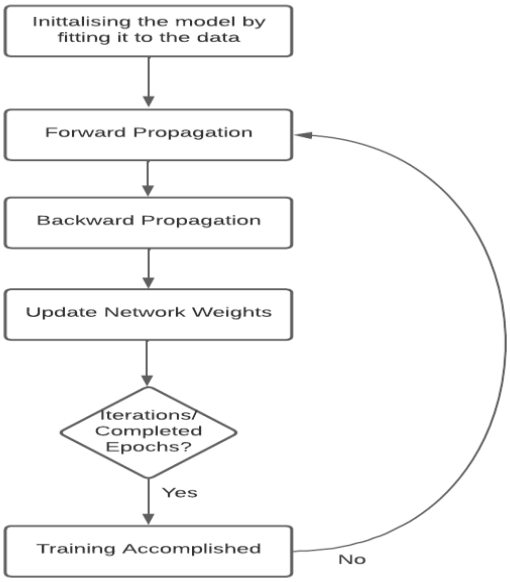

Figure 5. Flowchart of the training process

For the data to go through the above flow, there needs to be a couple of steps followed on the data such as pre-processing, class imbalance correction and early stopping. The importance and analysis of these steps are discussed below:

A. *Pre-Processing:*

1) **Scaling:** The pixel values in any image range from 0 to 256 each number represents a colour code and when we pass this through the network it could get a little complex. Hence, we divide all the values by 255 to bring it down to the 0 to 1 range.

2) **Data/Image Augmentation**: More data always results in better training of the model; however, it is quite redundant and difficult to find and keep adding unique images to the dataset. This is where data augmentation comes in place. By taking a pre-existing image we make simple changes such as rotation, cropping, flipping and so on to provide our model to train on different examples. Instead of augmenting and storing images in a numpy array, Keras provides a class called ImageDataGenerator which lets you transform and augment the image while data is training. Arguments in this class are rotation_range which defines the range for random rotations of the image and is set to a value of 10 and horizontal_flip is set as true to flip the image horizontally.

3) **One Hot Encoding:** In order to make the models understand the multiclass labels better we use this method. ML models work best on integer data than categorical, however, the Alzheimer Imagedata set consists of 8 [0-7] labels that outputs can map to, and we need to convert them to an array of binary values. To achieve this we used a utility library in Keras to_categorical(). It takes in a vector that contains integers (0-7) and gives out an array consisting of binary values. The parameters included in One Hot Encoding are the data set labels (train_dataset.labels, val_dataset_lables) and the number of classes (8).

4) **Padding**: For the Alzheimer Image dataset, we have applied a Transfer Learning model called VGG16 to train the data. This transfer learning model only accepts an image with size 32*32, however, the dataset images are of size 28. To resolve this issue, we have applied a padding mechanism (nipped) which takes NumPy arrays and pads them accordingly. The parameters of this function include: array with images (new_train_dataset.img), pad_width which is values of beginning and ending that are padded to the edge of the axis and mode function which is given as ‘constant’ (default value), pads the data with a constant value.

The dimension of the dataset is sample size*28(image dimension 1)* 28(image dimension 2)*3(Colourscale-RGB). The first value is padded by zero, the second and third is padded by (2,2) which is 28+4=32, and the last value by 0. Hence the resolution of the image changes to (32,32,3). [2].

B. Class Imbalance:

Any dataset that is into classification might have an imbalance in the proportion of data between the classes. For instance, the Alzheimer dataset has two classes and the, it might be very well possible that one of the classes has more sample frequency than the other. In such cases, the model will have enough data to predict the class with high frequency and the other with less frequency will not have enough data. This issue of change in frequencies is called a class imbalance. Most of the algorithms assume that the data is evenly distributed among the classes and hence create a problem of being more biased towards predicting the class with the highest frequency, algorithm, in this case, will not have enough data to understand the patterns of low-frequency classes thereby misclassifying the data.

To rectify the above problem, we can train the algorithm to understand that the distribution is skewed. This can be achieved by giving the different classes different weights. The function used to achieve this is called 'class_weight' in Sklearn. First, the class weights are computed by counting the unique classes in the dataset and giving the class weight attribute as 'balanced'. Balanced- automatically arranges the class weights in an inversely proportional way to their respective frequencies.

C. Compiling of the model:

Before training any dataset, we compile it. This basically defines the model architecture. The compilation step asks for a definition of the loss function, optimiser and metric. These are the important arguments for the compile() method in Keras.

1. **Loss Function:** This gauges the consistency between the network's output predictions made through forward propagation and the provided ground truth labels. for binary (Alzheimer) we use 'binary_crossentropy' as attribute values.
2. **Optimiser:** This value optimises the input weights in accordance with the loss function. For the datasets here we use Adam as the optimiser. It is chosen as it specifically changes learning rates for each network weight and is usually the best optimiser to use.
3. **Metrics**: These are the functions that are used to gauge the performance of the model. Here we choose to give 'accuracy' as the metric to understand how accurate the model is.

D. Fitting of the model:

The model.fit() method trains the model for a fixed number of Epochs. Epoch is basically one pass over the dataset. The arguments that we included in the code are explained below:

1. **Input data:** This will be the pre-processed and augmented train dataset including the labels.
2. **Epochs:** Epoch is one forward and one backward propagation of data through the algorithm. We have set the epoch value as 100 for our models to find out the best performance.
3. **Callbacks**: Call back is an object that can customise the training loop of a model. In the call back instance, we specify the 'EarlyStopping'. The goal of every training is to minimise the loss, at the end of every epoch the model.fit() will check whether the loss is decreasing, if not 'EarlyStopping' terminates the iterations. Another attribute under callbacks is the 'patience', it is set to 10 for our models and it means that 10 epochs with no improvements will stop the training.
4. **Validation data:** A part of the dataset with labels used to evaluate the loss and metrics. This data is not used to train the model, thus it is not affected by layers.
5. **Shuffle**: The value for shuffle is set to True in our fir function for all models. This enables the input data to be shuffled for every epoch resulting in the model being trained with new data every time avoiding biased results.
6. **Class_weights**: This maps to the class imbalance discussed above. The argument value is given as weights which correspond to the adjusted weights of the classes in the dataset.
7.

V. Analysis of Designed Interactive Application and GUI

The designed application allows user to navigate between different pages – one for Alzheimer Imageand one for BreastMNIST and to upload blood images or breast scan images based on what page they are on. Streamlit is used to design the UI and integrate it with the designed deep learning models [2]. Once the user opens the app, they land on Homepage, (shown in fig)where they are given the option to navigate through different pages using streamlit's sidebar option on the left-hand side of the application.

When the user selects one of the two datasets, they are navigated to their respective page, which allows the user to upload an image. An example of how users can navigate through screens and use the application is shown below:

1. The user lands on a homepage once the application is loaded:

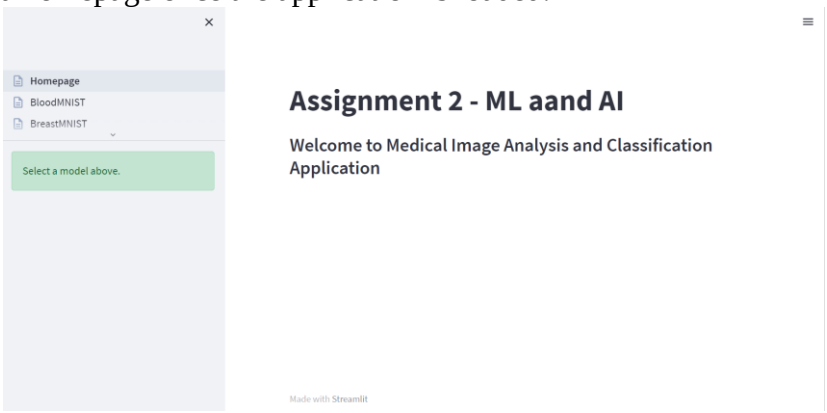

Fig 6. Homepage of the application

2. The sidebar on the left-hand side allows user to navigate to other pages and be able to upload images and get predictions accordingly. For example, if the user selects BloodMNIST, the page is navigated to its respective page.

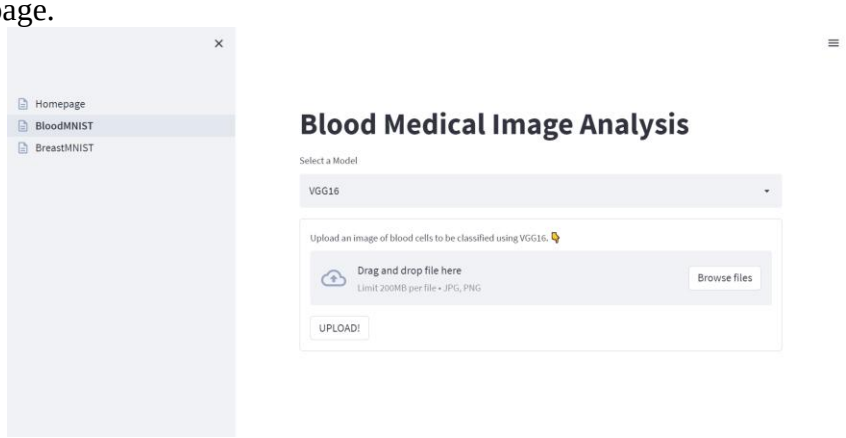

Fig 7. Blood image analysis homepage

3. On this page, a dropdown is displayed on the screen from which user can chose between two models designed for that particular dataset. For BloodMNIST, we have VGG16 and the model designed from scratch.

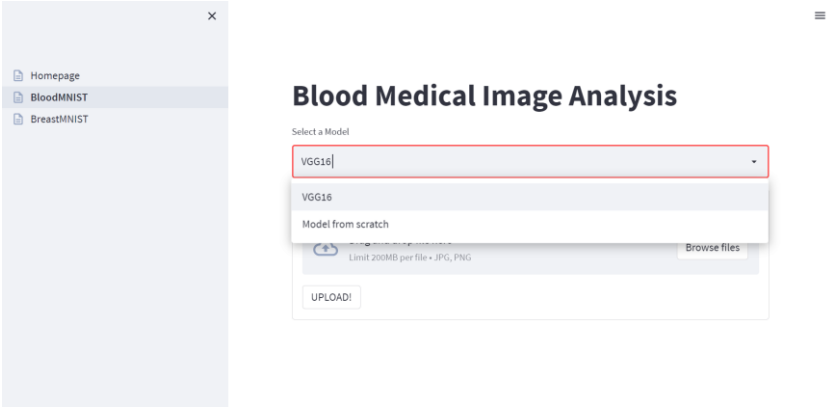

Fig 8. Dropdown to choose from models.

4. Once the user selects the model of choice, the application allows the user to upload an image, a blood cell image in this case and click on the upload button. The model is then loaded and a prediction is made on the image uploaded.

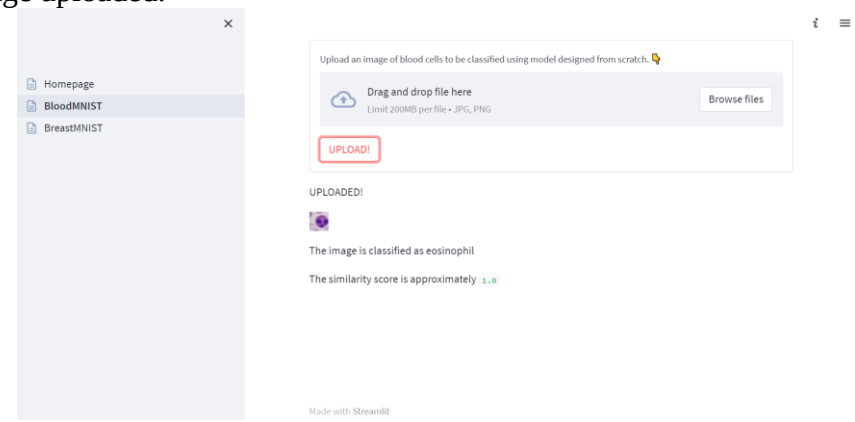

Fig 9. Classification of the uploaded image.

VI. Analysis and Performance Evaluation of DNN

Table 5. Alzheimer METRICS SUMMARY

Names of Metrics		Model from Transfer Learning-VGG16
Accuracy		0.87
Precision		0.91
Recall		0.91
F1 Score		0.91
ROC AUC		0.89

The calculation of the below metrics is made onto the test dataset to understand the performance of each model.

A. Accuracy:

Accuracy is a classification score. It sees how accurately the predicted set of labels matches the ground truth. The accuracy of VGG16 and in the Alzheimer dataset is at 0.87

Precision:
Precision refers to the ratio of the number of true positives to the sum of true and false positives. The best score of precision is 1 denoting the model is performing a 100%. Alzheimer which is at 0.91.

B. Recall:

Recall refers to the ratio of a few true positives to the sum of true positives and false negatives. The best score for the recall is also 1, it happens when the numerator and denominator are equal and false negatives become 0. Like the above metrics Alzheimer models at 0.91.

C. F1 Score:

The F1 score is a mean value of precision and recall. The best and worst values of F1 are 1 and 0 respectively. Alzheimer models at 0.91.

D. ROC AUC Score:

The ROC AUC score also helps in visualising how well a classifier performs. This however is for only binary classification, best fit for BreastMNIST data as it has only 2 classes. The best and worst values of ROC AUC are 1 and 0 respectively. Both models perform well at a similar score of 0.89.

E. Confusion Matrix:

.

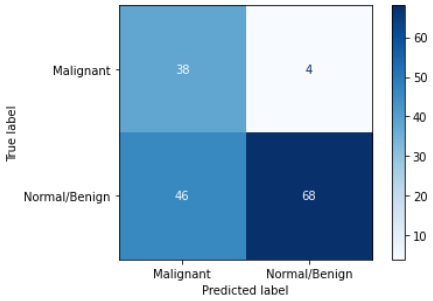

Figure 1. Confusion matrix of VGG16 on Alzheimer

Like the text, these figures are for the Alzheimer dataset. Both the models predicted the same number of true positives for both classes. However, the Normal/Benign is predicted with 38 true positives than Malignant which has only 68 true positives.

F. Classification Report:

Table 7. Classification Report of each class on Alzheimer

	Malignant	Normal/ Benign
Precision-Vgg16	0.45	0.94
Recall-Vgg16	0.90	0.60
F1-Score-VGG16	0.60	0.73
Support-VGG16	42	114

The above table represents the metrics for the Alzheimer dataset. The Normal/ Benign class has the best scores in all metrics and both models at 0.91 hence it's the best-predicted class for both models. The Malignant class is at 0.76 for all metrics, in both models. Which is not a bad score but comparatively less than the other class.

VII. REFERENCES

[1] J. Yang, MedMNIST, 2023. [Online]. Available: https://medmnist.com/. [Accessed: 16-Mar-2023].

[2] MedMNIST, "MedMNIST/MedMNIST: 18 mnist-like datasets for 2D and 3D Biomedical Image Classification: Pip install medmnist," GitHub, 2023. [Online].

[3] J. Yang, R. Shi, D. Wei, Z. Liu, L. Zhao, B. Ke, H. Pfister, and B. Ni, "MedMNIST v2 -- a large-scale lightweight benchmark for 2D and 3D Biomedical Image Classification," arXiv.org, 25-Sep-2022. [Online]. Available: https://arxiv.org/abs/2110.14795. [Accessed: 16-Mar-2023].

[4] A. Sarkar, "Xception from scratch using tensorflow-even better than inception," Medium, 22-Oct-2020. [Online]. Available: https://towardsdatascience.com/xception-from-scratch-using-tensorflow-even-better-than-inception-940fb231ced9. [Accessed: 16-Mar-2023].

[5] K. Team, "Keras documentation: Image Classification From Scratch," Keras. [Online]. Available: https://keras.io/examples/vision/image_classification_from_scratch/. [Accessed: 16-Mar-2023].

[6] A. A. Awan, "A complete guide to data augmentation," DataCamp, 23-Nov-2022. [Online]. Available: https://www.datacamp.com/tutorial/complete-guide-data-augmentation. [Accessed: 16-Mar-2023].

[7] "Tf.keras.preprocessing.image.imagedatagenerator :   tensorflow V2.11.0," TensorFlow. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator. [Accessed: 16-Mar-2023].

[8] "convolutional neural networks for visual recognition," CS231N convolutional neural networks for visual recognition. [Online]. Available: https://cs231n.github.io/convolutional-networks/. [Accessed: 16-Mar-2023].

[9] A. Tam, "Using activation functions in deep learning models," MachineLearningMastery.com, 07-Feb-2023. [Online]. Available: https://machinelearningmastery.com/using-activation-functions-in-deep-learning-models/. [Accessed: 16-Mar-2023].

[10] "Very deep convolutional networks for large-scale visual recognition," Visual Geometry Group - University of Oxford. [Online]. Available: https://www.robots.ox.ac.uk/~vgg/research/very_deep/. [Accessed: 16-Mar-2023].

[11] "Deep learning: Image recognition online class: Linkedin learning, formerly Lynda.com," LinkedIn. [Online]. Available: https://www.linkedin.com/learning/deep-learning-image-recognition/. [Accessed: 16-Mar-2023].

[12] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," arXiv.org, 04-Apr-2017. [Online]. Available: https://arxiv.org/abs/1610.02357. [Accessed: 16-Mar-2023].

[13] J. Brownlee, "A gentle introduction to dropout for Regularizing Deep Neural Networks," MachineLearningMastery.com, 06-Aug-2019. [Online]. Available: https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/. [Accessed: 16-Mar-2023].

[14] "Main concepts," Streamlit Docs. [Online]. Available: https://docs.streamlit.io/library/get-started/main-concepts#app-model. [Accessed: 16-Mar-2023].