Gisma University of Applied Sciences

Department of Computer and Data Sciences

Uncertainty-Aware Deep Learning for Early Detection of Alzheimer's Disease Using MRI Images Data

Jason Joel Pinto

Submitted in partial fulfillment of the requirements for the degree of

MSc in Data Science, AI and Digital Marketing

Under supervision of

Prof. Dr. Mazhar Hameed

December 2024

Abstract

In this research paper, an robust, effective and efficient deep learning model is developed. And aims to improve the intepretablity of predictions for clinical applications. This paper included the devel and assessment of a deep learning model for the categorization of Alzheimer's disease stages using MRI images. Building on a well performing Kaggle model, the model adds many improvements, such as the ability to estimate uncertainty to produce prediction confidence scores. These confidence scores help doctors or clinicians to make well-informed decisions by offering insightful information about the model's decision-making process. The algorithm successfully classifies MRI pictures into the four stages of Alzheimer's disease that is Mild Demented, Moderate Demented, Non-Demented, and Very Mild Demented in achieving an accuracy of 99.55% on the test dataset which was a slight improvement over the base model. By measuring prediction uncertainty estimation using Monte Carlo Dropout improves the model's robustness. This helps the clinicians to get the idea about the confidence of the model Even though introduction of uncertainty estimation leads to longer prediction time, the suggested model shows improvement in training and assessment times when compared to the base Kaggle model. The model's better performance and ability to provide confidence scores makes up for the extra calculation expenses. A web application has been developed to increase the model's accessibility even further. Users can input MRI pictures to receive predictions and confidence scores. This intuitive interface allows simple integration into clinical settings, helping doctors in efficiently evaluating the model's results. The findings show how integrating uncertainty-aware deep learning models into medical diagnostics will improve Alzheimer's disease patient care, clinical desicion making, and early detection.

Acknowledgements

I sincerely thank Prof. Dr. Mazhar Hameed for all of his guidance and encouragement during the writing process of my thesis. His knowledge in machine learning and his helpful criticism have greatly influenced the quality and trajectory of this work. I want to express my gratitude to the faculty and staff at Gisma University of Applied Sciences for offering me the materials and space I needed to carry out my research. Their understanding and motivation have been crucial in helping me in accomplishing this goal. I express utmost gratitude for my friends and peers who encouraged me along this journey by sharing their knowledge and providing unwavering support. Particularly in trying times, their assistance and companionship have been a source of strength. I am truly and incredibly grateful to my family for their constant unconditional love, understanding, and patience. Their constant support for me and my goals has been a continual source of inspiration and has been essential in enabling me to overcome every challenge. In the end, I extend my sincere gratitude and dedicate my work to everyone who have helped and motivated , directly or indirectly. The contribution has had an enormous influence on my academic and personal growth as well as to this thesis.

Jason Joel Pinto
21 December 2024

Contents

List of Figures

List of Tables

Chapter 1

Introductions

Alzheimer's is a neurological problem that starts with a build up of proteins called amyloid plaques and neurofibrillary tangles. This leads to the brain cells to die over time and this leads to the shrinkage of the brain size[4]. Studies show us that about 5.4 million people in the States have Alzheimer's. And the studies show upward trend in the newer generations in getting Alzheimer's Disease. It is expected to increase by 10 million people. As of today every 68 second and new person is being diagnosed by Alzheimer's Disease. And the studies show that by 2050 this is going to decrease to 33 seconds[1]. Between the year 2000 and 2017 the deaths reported due to alzheimer's have been increased by 143 %[3]. Alzheimer's is a disease which get's worse over the years, it keeps on progressing and the worst thing it is not reversible. Once worsened a lot of time and efforts go into the alzheimer's diseace. Which might end up being one of the costliest disease for any countries economy[23]. Hence it becomes very important that it is diagnosed as a early stage so that it can be controlled and treated. If it is detected early then doctor's with the help of medications can make sure that brain damage is not worsened. Currently magnetic resonance imaging (MRI),functional magetic resonance imaging(fMRI), computer tomography and positron emission tomography (PET) are used in order to early detect the Alzheimer's disease. These technologies are very expensive, very time consuming task and not very accessible for everyone as it Requirement sophisticated machines.

Machine Learning can be used to find if there is a way where one can find if there is a way models can find patterns which makes it possible to detect the disease at even earlier stage. As there are various studies that are being conducted all around the world, there are various data that are available which can be used by the machine learning models to analyze and early detect the disease. Based on prior researches, the researchers have numerical data's related to a persons MRI's and biomarkers in a persons body. With help of these they find whether a person is demented or not. If these data are used and ML model be developed which will be able to predict the will be lead to inexpensive diagnosis and can be made easily available to more and more people. In addition to that the automated model will be more accurate and also removes all human errors.

A person with Alzheimer's disease will not be aware that he might be having a disease can ultimately lead to be a very dangerous disease. This is because the symptoms of Alzheimer's are very hard to find. It can be different to different people. The person can still do all the regular chores he does day to day without any problems. Having said that there will be some problems he might be facing while doing these activities, like for example not able to remember words, or not remembering names etc. People around him will definitely notice that this person has trouble remembering stuff. This is the reason why doctor as a first step usually conducts a medical interview where the person will be asked some questions. With the help of this he will be able to get a ball park idea whether he might have Alzheimer's or not. Some of the symptoms of Alzheimer's as follows,

1. Forgetting things, misplacing things, and missing meetings.

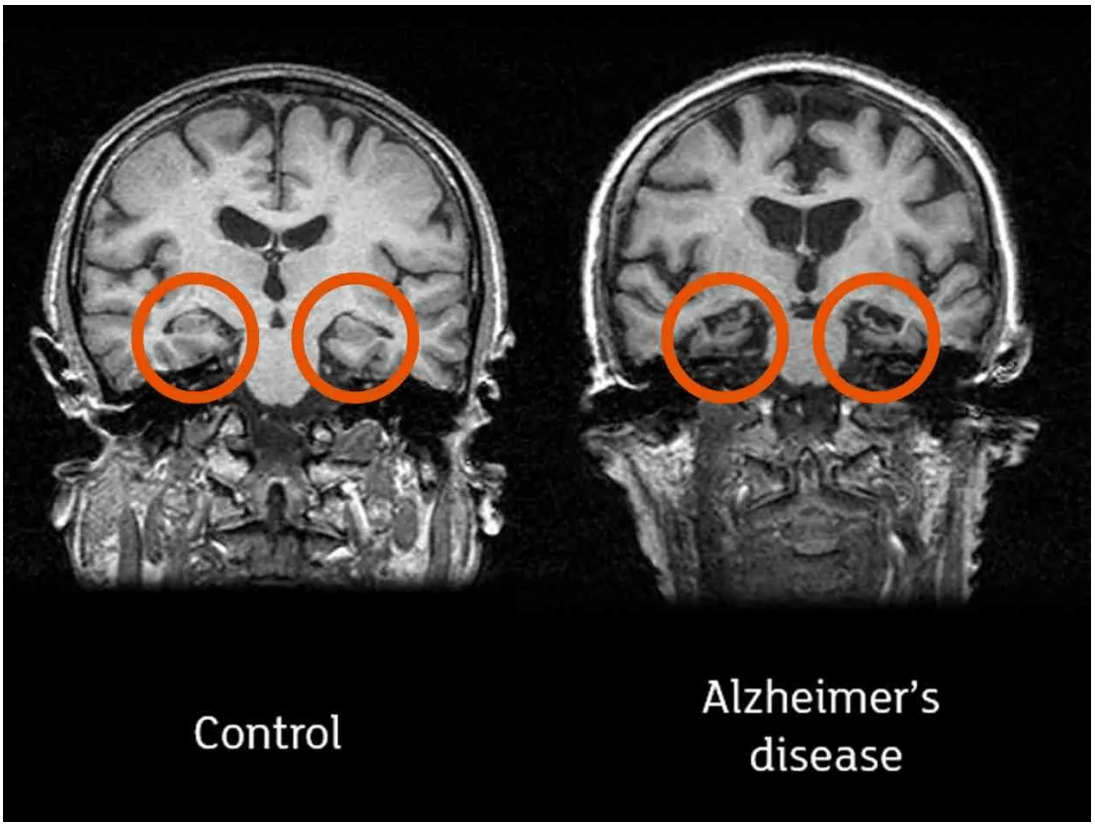


Figure 1.1: Comparison of a normal brain (left) and an Alzheimer's patient's brain (right). Images credit: Professor John O'Brien, University of Cambridge and Newcastle University

2. Forgetting names and events.
3. Not able to recall incidents happened in the past.
4. Problems with audio visual memory.
5. Being absent minded.
6. Agitation and restlessness

The symptoms of Alzheimer's becomes more evident when the disease progresses and one grows old. The person starts notices that the symptoms that he previously had are slowly slowly getting worse and worse. At later stages of AD it becomes very difficult to even communicate with the person as he does not remember anything. He forgets to even do the basic things that he has been taught to do since his childhood. This the stage when they will need continuous Assistance.

Chapter 2

Motivation

Figure 2.1: Person suffering from Dementia

Alzheimer's is considered as one of the most challenging disease in the modern medicine research. And this disease not only effects the individual but also impacts the people around him/her that is family, doctors and also the hospitals. As the people effected by this disease are increase at a rapid rate around us, there is a need to develop a better way to diagnose and prevent this disease. Particularly, there is a high demand for a system that will detect the disease at a early stage. So that the medical team can stop the disease from getting worse at the old age which might cause a big economic burger for the patients.

The motivation to write this thesis came from the fact that there are sophisticated diagnostics options available at the moment which are very expensive, time-consuming and they are not accessible for of people. But the main problem is even after spending a lot of money on these diagnostic procedures, it cannot detect it at a very earlier stage. Doctors use MRI's and PET scans which are very accurate but they machines used for this are very special and needs an expert to use them. Additionally, even after analyzing a persons brain in these scans, the disease is not detected at a early stage because the small changes that are happening in the brain are very different to find when

compared to the normal aging of a persons brain.

With the advanced Machine Learning techniques we can try to fix these problems that are currently there. By using the datasets that available online the advanced Machine Learning models can find patterns in the datasets which is hard to find using a naked human eyes. This will help in detecting the Alzheimer's disease at a very earlier stage.

Personally for me this paper is driven by the fact that I have seen many people around me struggling with Alzheimer's disease in their old age. I have seen it first hand how much time, money and effort does it take for their family to take care of them in that situation. As it is a disease that gradually gets worse, they slowly start losing their memory and identity. Having witnessed this I have seen how much effort goes into taking care of a Alzheimer's patient. This has motivated me to provide my Contribution to this communicate and make a difference. Hence, my aim in this research is to create a model will be able to predict the possibility of the disease at a earlier stage.

Chapter 3

Contribution

The contribution of this thesis paper is to use the advanced Machine Learning algorithms to find a solution to problems of early detection of the Alzheimer's disease. This is to provide a novel extension to the community of deep learning models to the Alzheimer's stage classification.This will include implementing various different advance machine learning models to improve the predicting of the disease based on the MRI images that we are using for this research. The main contributions are such as:

1. **Inclusion of Uncertainty-Aware Framework:** In this research patient's MRI images will be analyzed which are divided into various stages of Alzheimer's and build a Classification Model using these images. Many high performing models are already presend online but in this thesis we will looking one of them and which performed the best and implement Monte Carlo Dropout Uncertainty calcucation so that the confidence of the model can be analyzed. This will help in knowing how confident the model is when predicting the class.
2. **Improved Model Interpretation:** The addition made to the code to include the Uncertainty indication will help in knowing how confident the model is when predicting the classes. This will also help in the clinical environment where the professionals looking at the results of the models they will not blindly accept the stage of alzheimer's disease, they can check if the model was confident when predicting the class. If not they can furthur review with their senior doctors about the case.
3. **Addition of Uncertainty Score in Predicted Results:** In this thesis a very much necessary steps in the deep learning models when it comes to medical usage are being looked at and included. These additional metrics will ensure that models performance is not evaluated only based on the accuracy but important factors like Uncertainty or confidence is considered so that cases with high uncertainty can be taken for further studies instead of just labeling whatever the model is classifying it as. A well structure visualization framework is put into action to highlight uncertainty for each and every individual predicted images. These information on the visualization will allow clinicians to find and focus on unique and unknown cases, which will allow them to make more informed decision-making.
4. **Utilization of Publicly Available Datasets:** The MRI images used for this study is a cleaned dataset which is availabel on kaggle which is sourced from ADNI websites which are the MRI images of actual people with true labels. Since it's is very reliable source this study uses real world data to train the model at the same time following all the ethical guidelines.
5. **Visualization of Uncertainty in Predictions:** By focusing on machine learning approaches, the research aims to provide scalable and cost-effective solutions for early AD diagnosis. This could significantly improve

accessibility, especially in resource-constrained settings where sophisticated imaging technologies are not viable.

6. **New addition will make it possible for this to be applied in a clinical setup:** By introduction Uncertainty into account when classifying the Alzheimer's Disease into multiple classes this thesis reseach will help reducing the rist of wrong diagnosis in a medical setup who are ready to introduce AI into for their diagnostics setup. It is especially important in medical field as misdiagnosis might cause a major problems in a persons life.
7. **Contribution to the AD community as well as open sourcing these projects:** The code used for this thesis will be open to everyone to view and as this will allow to access and reproduce this models for further studies. By adding Uncertainty to these already well performing models this reasearch further improves the the communities acceptence of these new technologies into their system, as it also allows human proffessionals to take things into their hands the models is not so sure how to react the unexpected situations.

Chapter 4

Literature Review

In the recent years have seen a impressive amount of progress in the utilization of the deep learning models for the Alzheimer's Disease detection based on the MRI data. And alot of advancement focus on how to improve, how accurately the model predicts the Disease of how accurately it classifies the images or data into different classes. Whereas the integration of Uncertainty calculation is something that is not explored too much, nevertheless. In this Literature review the previous research papers are being studied and will be compared and seen if there are any gaps in the study that can be full filled.

4.1 Overview of techniques used on MRI

All thanks to the impressive technology of the Magnetic Resonance Imaging(MRI), Alzheimer's disease (AD) can be detected at a very early stage compared to earlier. The 3 Dimensional high quality images of brain captured using MRI machines provide a view of our brain with which we can find the patterns in the parts of the brain like temporal regions and hippocampal region which helps in diagnose of the Alzheimer's Disease. By taking a look at the white matter integrity and functional connectivity, the help of diffusion tensor imaging (DTI) and functional magnetic resonance imaging (fMRI) make amazing contributions to this. Methods such as cortical thickness measurement and voxel-based morphometry have been widely used for quantitative analysis of structural changes and brain volume[35].

Recently, there have been studies that are going on to create synthetic longitudinal MRI images for Alzheimer's disease using a conditional diffusion model. Diffusion models are are excellent at producing high-quality images with consistent training, which are used in this strategy. The models can produce realistic target pictures that mimic the course of the disease by using the time interval and the source MRI scan as conditioning elements. This strategy may be able to get around the drawbacks of current techniques like GANs and VAEs, which can have unstable, undiversified, or hazy results. The suggested model offers a useful tool for Alzheimer's disease research and clinical applications, producing high-fidelity synthetic MRI scans with encouraging findings[7].

4.2 Using Deep Learning for Alzheimer's Detection

The most common type of dementia that causes a lot of trouble for millions of people worldwide is Alzheimer's disease (AD). In order to manage symptoms and maybe reduce the progression of the disease, early recognition of AD is essential[20].Conventional diagnosis techniques depend on radiologists' subjective and time-consuming manual review of brain imaging data. And that is where Deep Learning brings an serious advantage over the time

consuming tasks. Deep Learning models can predict these kind of tasks with the mater of seconds.

With the use of deep neural networds we have see an amazing results in the recent years. Using the MRI data, and deep learning neural networks in prediction of the early detection of the Alzheimer's Disease has been a very promision method with great acccuracy. The convolutional neural networks as well as the recurrent neural network have have drastically improve the prediction for this desease when compared to the traditionaL Machine Learning alternatives. This is because the Deep Learning models are able undertand the complex nature of the high dementional data that are used in training these models and to capture the brain image[15]. Recent improvements in the this field, specially deep leaning have changed the landscape of the Alzheimer's Detection. It is able to provide an objective and efficient solution.

Convolutional Neural Networks (CNNs) is a kind of deep learning model which has showed the world it's abilities with great results in prediction of Alzheimer's with the help of MRI images. Very high resolution images or high dimensional data's can be trained using these models, which will find the existing patters in these data, and helps in detecting Alzheimer's at a very early stage. By drastically reducing the time and efforts required in the diagnosis process of the Alzheimer's Disease and being able to predict with great precision,these deep learning models perform impressively better than than the conventional machine learning models.

Let's check some of the recent studies. The summary of the studies is provided in the Table 4.1 :

Table 4.1: Comparison of Studies on Alzheimer's Disease Detection

Study	**Dataset**	**Model**	**Accuracy**	**Limitations**
Sharma, Sarang (2022) [28]	Kaggle	Hybrid Deep Learning Model	91.75%	No uncertainty estimation, limited interpretability
Desai, Maitri (2024) [5]	OASIS	Multi-Task CNN	91.0%	Limited generalizability, lacks uncertainty
Ahmed, Gulnaz (2022) [2]	OASIS	DAD-Net	99.22%	No reliability or uncertainty analysis
Sarraf et al. (2016) [27]	ADNI	CNN (LeNet-5)	98.84%	High false negatives
Farhana Islam et al. (2023) [13]	Custom MRI Dataset	ResNet-50	98.71%	No uncertainty estimation
Santos et al. (2023) [8]	Kaggle	Neural Network	80.6%	Moderate accuracy, lacks optimization
El-Latif et al. (2023) [19]	Kaggle	Lightweight Deep Learning Model	95.9%	No uncertainty quantification
Isunuri et al. (2023) [14]	Kaggle	Transfer Learning with CNN	97.32%	Limited interpretability
Murugan et al. (2023) [24]	Kaggle	DEMNET	95.23%	No uncertainty assessment
Gupta et al. (2023) [10]	NRC Korea	DEMNET	96.89%	No reliability metrics
Mohammad. (2023) [10]	Kaggle	Xception	99.6%	No uncertainty calculation

In recent studies, various different techniques have been used to predict the alzheimers disease. Out of all of them deep learning models have shows a lot of potential and promise with great results. Deep learning is widely used acrross various fields or study, but the stream where we excel the most is classification of the images. As in recent times people are able to get hands on the powerful hardware as it has become accessible, and more research are being conducted on Deep Learning models there has been alot of improvement in the performance and hence the reliability of these models have been increased [32]. Particularly for alzheimer's there have been several studied that have been conducted. In this study we will be looking at few of them. In a paper by Sharma et al. [28] they propose HTLML. It is a hybrid deep learning model that uses the Kaggle dataset to predict the Alzheimer's Disease. In their implementations they were able to achieve an accuracy of 91.75%. Which is a very good performace. But what it lacks is the implementation of uncertainty estimation. When it comes to a clinical environment it would be very advantageous for clinicians to know how confident the model is. The lack of uncertainty estimation is a rather common gap found in many existing studies, limiting them to consider for a clinical deployment.In the reseach conducted by Desai et al. [5], they use a Multi-Task CNN model using the OASIS dataset to diagnose Alzheimer's. Their model achieved an accuracy of 91.0%, but limitations of the model is it's generalizability. The authors themselves mention in the paper that the model performs good on the OSIS dataset but it might struggle to perform well in the real-world clinical data. Generalizability is considered to be a very common issue with the deep learning models. Specially when there is not enough data to be trained on or there is no diverse patient data. Or when tested to perform with the real world data. In a paper by author Ahmed et al. [2] has experimented with DAD-Net, it is also a deep learning-based model for Alzheimer's detection. Here they have used ADASYN to oversample the data as the dataset they used was imbalanced. Using the deep leaning network they managed to get an impressive accuracy of 99.22%. This model again however does not incorporate Uncertainty calcucation. Even through the model prediction is high, without the model confidence/ Uncertainty implementation it's implementation in a clinical environment will be very low because even when the model is struggling the clinicians needs to know it so that they can take a informed desicion. the model does not incorporate uncertainty estimation, which is a crucial factor for clinical decision-making. While the high accuracy is promising, the lack of uncertainty estimation limits the model's applicability in high-stakes environments such as healthcare. Sarraf et al. [27] use CNNs and LeNet-5 for the classification of Alzheimer's classification with the ADNI image dataset. They achieved an accuracy of 98.84%. However, the study is criticized for its high false-negative rate, which can lead to misdiagnosis in clinical applications. Additionally, this work does not include any discussion of uncertainty estimation, further limiting its clinical relevance. Islam et al. [13] proposed a method using the ResNet50 feature extractor and an SVM classifier for diagnosing Alzheimer's disease from MRI images. This study achieved an accuracy of 98.71%, but it lacked assessment of the model reliability. Since there is no assessment with the unseen data the reliablity of the model is hard to know with the real world data. Even in this paper the Uncertainty calcucation is not considered. Further in the study of neural network model for Alzheimer's prediction by Santos et al. [8], they employed a neural network model on the Kaggle dataset, and were able to get a accuracy of 80.6%. While the model is able to provide the diagnosis and classification the performance of the model could be improved compared to other models. In the conclusion the authors suggested that the model could be improved through further research, but uncertainty estimation is not part of their investigation, which is a limitation. El-Latif et al. [19] presented a very lightweight deep learning model for Alzheimer's disease detection using MRI data. The model achieved an accuracy of 95.9%, but

like the previous case the authors acknowledged that the performance could be improved further. Again the lack of uncertainty estimation is a limitation. Isunuri et al. [14] employ transfer learning and a CNN for Alzheimer's severity classification, achieving an accuracy of 97.32%. Even though the model demonstrates very good performance performance, there is no mention of uncertainty estimation, which could add a layer of reliability and interpretability to the model's predictions. The authors also note that the model outperforms competing models in several metrics, yet its generalizability to other datasets is unclear. Murugan et al. [24] introduced DEMNET, again it a type of deep learning model itself for the sake of early diagnosis of Alzheimer's disease from MRI images. The model achieved an accuracy of 95.23%, but the authors acknowledged that the performance of the model could have been be improved. While the model is promising, it does not incorporate uncertainty estimation, which is a critical aspect for clinical use as earlier mentioned where decisions based on model predictions can have serious consequences. Gupta et al. [10] use combined features from voxel-based morphometry and cortical, subcortical, and hippocampus regions of MRI T1 brain images for early Alzheimer's diagnosis. With an accuracy of 96.89%, the study demonstrates a robust method for Alzheimer's detection. But similar to other studies, it lacks a mechanism for assessing uncertainty, which is important for practical deployment in healthcare environments. One of the model which particularly caught my attention was a kaggle model where with the kaggle dataset it was able to achieve a accuracy of 99.6%. I will be selecting this model as my based model. The reason why I am selecting this model is because it was able to achieve this high accuracy score, even though the model was not have a complex architecture. This make the model very efficient and easy to deploy and maintain as a final product. But having said that there were some limitations in these model.

4.3 Limitations of Existing Approaches

Table 4.1 all the previous existing papers and research have been summarized, it also highlights the study's limitations. Many papers and research have models that do exceptionally well with accuracy, but then all the paper lacks the attempt to include uncertainty score as an output which might turn out to be invaluable and crucial in a clinical environment in decision-making. Furthermore, some of the models had issues with high false-negative rates, less generalizability, and dataset bias still exists. These problems support a requirement or necessity for the inclusion of uncertainty estimation so that the clinical usability and reliability of the model can be improved. When the stakes are high in clinical settings, it is challenging to trust models that lack uncertainty measurements. Adding uncertainty estimates to deep learning models can improve their interoperability and give medical professionals important information about the accuracy of the model's predictions. By using uncertainty-aware deep learning to enhance the applicability of Alzheimer's diagnosis models, this study fills a significant gap in the body of existing literature.

4.4 Novelty of the Proposed Approach

The integration of uncertainty estimation into deep learning models addresses critical gaps in existing research. By providing confidence scores alongside predictions, clinicians can make informed decisions, reducing the risks associated with false negatives or overfitting to biased datasets. This approach also enhances model interoperability,

a vital factor for clinical adoption.

Chapter 5

Methodology

In this chapter the methodology used in this thesis for training, developing, evaluating and developing an uncertainty-aware deep learning model will be discussed. This sections will be taking you into the deep dive into why certain things are chosen over others. Each and every step from the loading of the dataset, data augmentation, model building, evaluation metrics, uncertainty awareness in the model and as well building an web application to predict Alzheimer's which can be used in a clinical environment.

5.1 Dataset used for this research

5.1.1 About Dataset

For this study the dataset used from a publicly available dataset. The dataset is available here https://www.kaggle.com/datasets/uraninjo/augmented-alzheimer-mri-dataset. This dataset is derived from the MRI dataset that is available on the ADNI website https://adni.loni.usc.edu/. This dataset consists of high quality 2D Magnetic Resonance imaging(MRI) scans images. Magnetic Resonance imaging scans are usually 3D, but this particular images dataset that is available on Kaggle is being processed and converted into high quality 2D images. This dataset is best suited for multi class classification task. The images are being divided into 4 different classes based on the stages of the alzheimer's the patients are in. And the folder names are the true labels of the images:

1. **NonDemented** – This consist of MRI images of brains which does not show any sign of having Alzheimer's Disease or in other words cognitive impairment. There are a total of 9600 images of the Non Demeted brains in this dataset.

2. **VeryMildDemented** – This consists of MRI images of the brains which shows slight early sign's of patient progressing towards Alzheimer's Disease. There are a total of 8960 images of the Very Mild Demented brains in this dataset.

3. **MildDemented** – This consists of MRI images of the brains which show mild structural brain changes of patient progressing towards Alzheimer's Disease. These patients will start having noticeable memory problems and cognitive problems. There are a total of 8960 images of the Mild Demented brains in this dataset.

4. **ModerateDemented** – This consists of MRI images of the brains which shows significant structural brain changes of patient Alzheimer's Disease. This is a advanced stage of the Alzheimer's disease. These patients

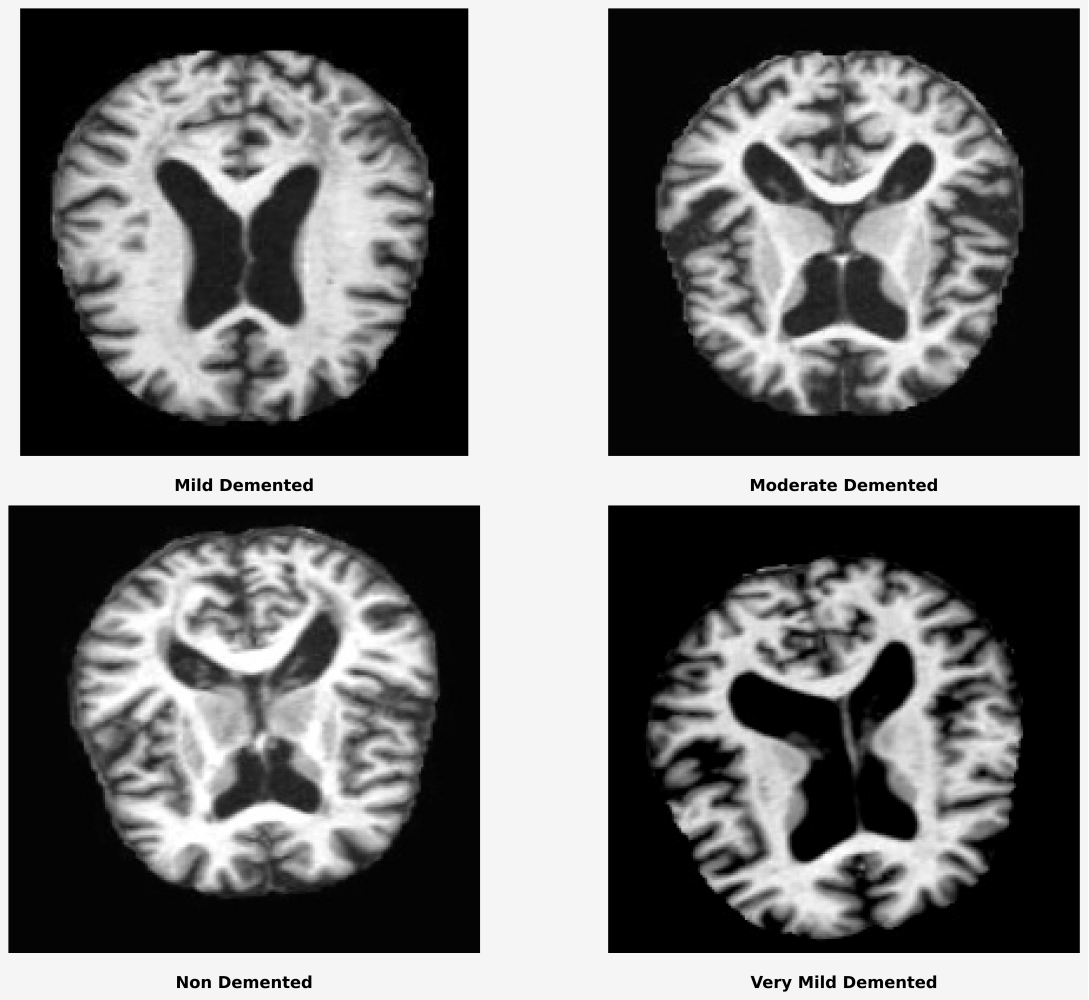


Figure 5.1: Sample images of the dataset

will start having noticeable memory problems and cognitive problems as well. There are a total of 6464 images of the Mild Demented brains in this dataset.

5.1.2 Dataset Features

Let's look at the features of the dataset that is used for this paper. There are two set's of images data in this dataset. One is the original images and the second one is the augmented images. For this research we will be taking Augmented images as the base models that we are taking from the Kaggle source has used this dataset. Hence we will be using this. The images are divided into 4 classes and images of each classes are placed in their on Folder's namely, NonDemented, VeryMildDemented, MildDemented and ModerateDemented.

1. **Size of Dataset** – The total size of the dataset is 366 megabytes. It consists of thousands of high quality 2 dimensional images for each classes. Which makes this dataset a perfect fit for the building a machine learning models by training the model using these real world datasets.
2. **Dimensions of the Dataset** – The original images of this datasets roots from the collection of MRI images from ADNI dataset. These images are originally 3 Dimensional. Having said that the dataset used for thesis

are the processed images of the ADNI dataset. The images have been processed to convert to 2 Dimensional images. This helps to reduce the complexity of the model.

3. **Class balance** – The figure 5.2 shows the class distribution in the dataset. As you can see in the figure 5.2

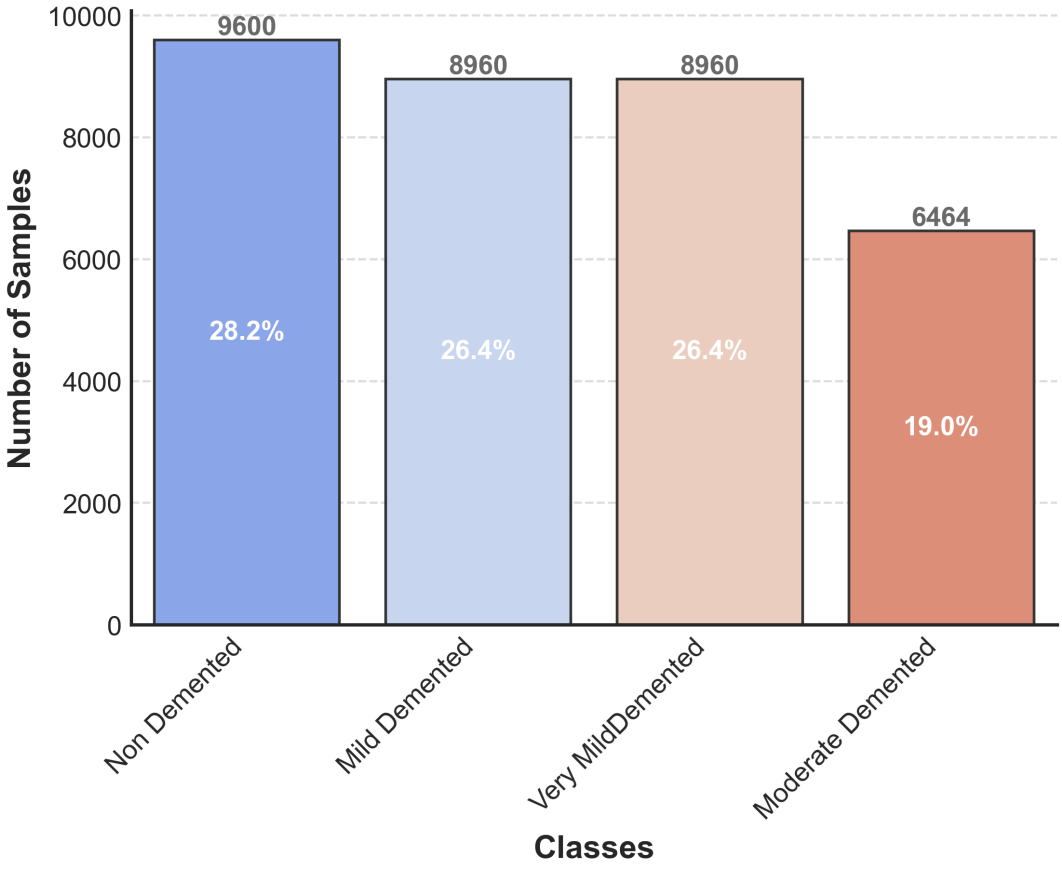


Figure 5.2: Dataset Class distribution

there 9600 images for Non Demented Class, and 8960 images for Mild Demented and Very Mild Demented Class and 6464 images for Moderately Demented Class. It is a moderately imbalanced dataset. But since we are going to check for accuracy scores for each classes we are not going to balance this dataset.

4. **Dataset Format** – Originally the ADNI datasets are stored in NIfTI format. But the Kaggle dataset that we are using is being preprocessed and saved in 2D format, all the images are stored in JPG format.

5.1.3 Ethical and Legal Considerations

This research used a publicly available Kaggle dataset for Alzheimer's disease detection, which strictly adheres to its licensing terms and conditions. The dataset used in this thesis is already anonymized by its authors which complies with data privacy and confidentiality standards by excluding personally identifiable information. Ethical considerations, including fairness, transparency, and interoperability, were of highest priority when designing the model. The use of Kaggle data aligns with non-commercial academic purposes, and proper attribution to the dataset's creators is provided. No new data was collected for this study.

5.2 Pre-Processing the Dataset

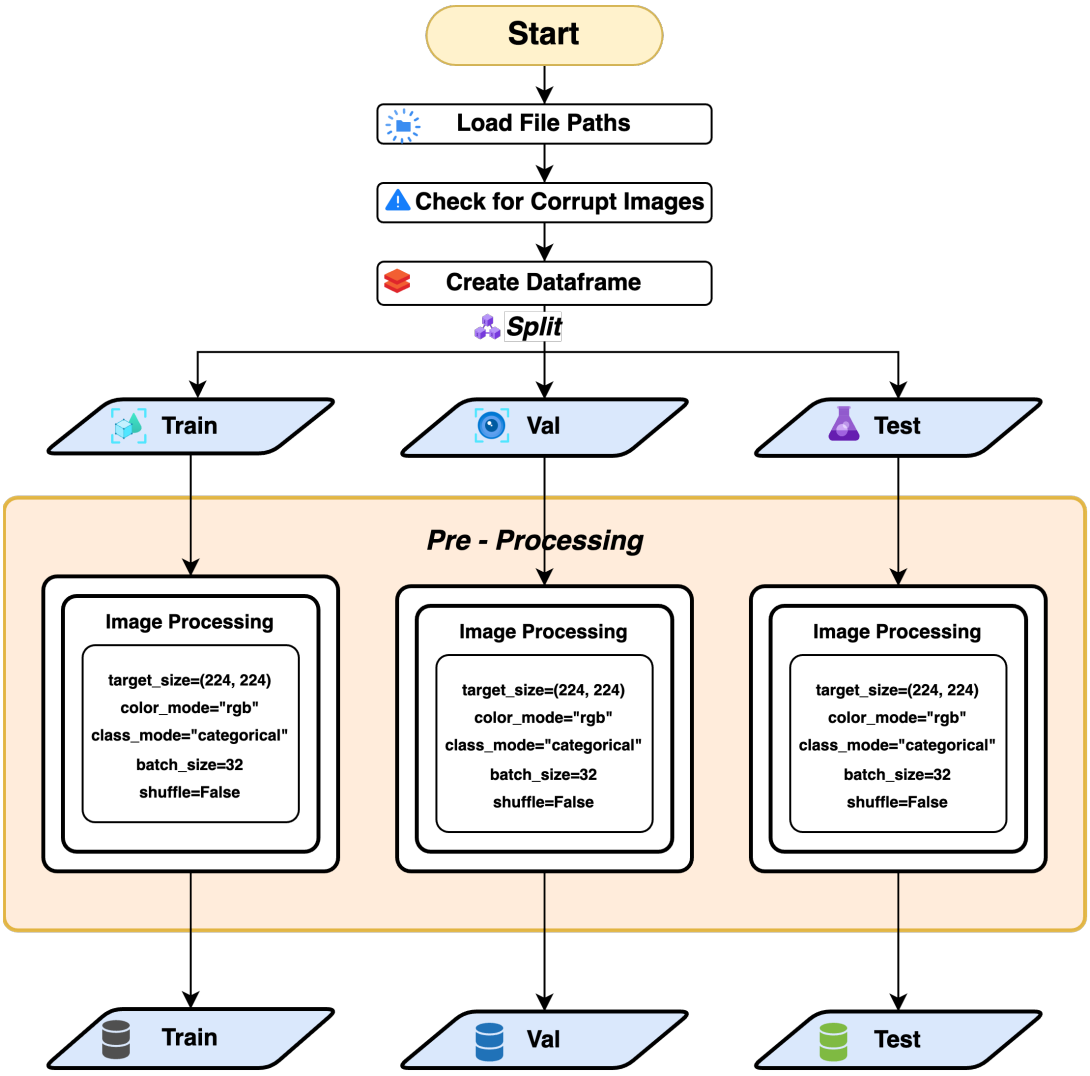


Figure 5.3: pre-processing flow

In order for the model to learn from the patterns in the images it is very import that the images are being processed properly. Data is processed so that the model can learn from it in a efficient and effective way. Figure **??** shows the steps applied on the images so that it can compatible with the model, and read as much details as possible but at the same time be efficient in learning from it.

5.2.1 Data Cleaning

In this step the corrupt image will be removed if incase any are found. But in this particular dataset we were not getting error. Hence none of the image were deleted from the dataframe.

5.2.2 Data Resizing

The images are resized into the size of to 224 × 224 pixels. This is because the model that we are using i.e. EfficientNet-Bo are designed for a specific image size that is 224 × 224 pixels. That is because EfficientNet-Bo

```
valid_filepaths, valid_labels = [], []
for filepath, label in zip(filepaths, labels):
    try:
        with Image.open(filepath) as img:
            img.verify()
            valid_filepaths.append(filepath)
            valid_labels.append(label)
    except (IOError, SyntaxError):
        print(f"Corrupted image file: {filepath}")
```

Figure 5.4: Dataset cleaning

uses compound scaling method where the width of the network, depth and the resolution are uniformly scaled. If the input sizes are changed the performance and the efficiency of model will be hampered. And accuracy of the model will be lowered. And if higher pixels are used then there will not be any performance gains but just extra effort while training the data . Hence we have to scale it down to that size [30] [6] [31]. While it is possible to input the sizes like 244×244 the performance of the model will be degrading hence it is better to get the input size as the one that is well optimized for for the best results.

5.2.3 Color Mode

The image color is chosen to be in RGB more. Generally, the color in all the digital images and displays is usually RGB (Red, Green, Blue). By using these three primary colors in different intensities, the model will be able to find patterns in a wider range of hues [26]. RGB is considered to be a standard right now as input for many trained convolution neural networks (CNNs) [29]. Using RGB images guarantees compatibility with these models, allowing for transfer learning and the utilization of pre-existing architectures. In many image identification tasks, color is an essential element. Because RGB photos preserve all the color information, models may recognize and learn complex patterns from color fluctuations much better than using the black-and-white color modes. This is especially beneficial for image classification tasks.

5.2.4 Class Mode

So the generator is told to give one-hot encoded labels,' where each label will have its own unique pattern in vector form [17]. Only one indices in the vector will have value other indices will be zero. That way the model will be able to understand which image is of which class. Training models with categorical is a common loss function in the case of multi-class classification.[34] The categorical cross-entropy loss function, commonly used in multi-class classification, goes well with one-hot encoded labels. This loss function efficiently calculates the difference

between the actual distribution and the predicted probability distribution. Hence, by utilizing category labels, the model can generate a probability distribution for all categories [33].

5.2.5 Batch size

In the preprocessing step a batch size of 32 is chosen for training. This will ensure that there is a balance between computational efficiency and model performance. Smaller batch sizes like 32 offer a trade-off between efficiency and gradient stability, potentially helping escape local minima and generalize better. They fit within modern GPUs' memory constraints while maintaining stability [16]. Larger batch sizes will be even though smoother it will require require alot more more memory and there is a high risk of overfitting.Additionally, batch size affects convergence speed, allowing sufficient weight updates without excessive overhead, balancing training speed and learned representation quality [22].

5.2.6 Normalization

The images are preprocessed using`preprocess_input` normalization for the model, this is by using the function from the EfficientNet library itself. This was used so that input images are properly scaled as required by the model itself. This ensures that the images are standardized by transforming pixel values into a range that aligns with the data the model was trained on. This kind of normalization becomes very much important in deep learning models to that it can improve the performance and efficiency of the models[30].

The approach used in EfficientNet's `preprocess_input` function will rescale values of the pixelfrom their original range of $[0, 255]$ to $[-1, 1]$. In this process it subtracts the mean and divided it by the standard deviation of pixel values.The formula for normalization is:

$$\text{Normalized Pixel Value} = \frac{\text{Pixel Value} - \mu}{\sigma}$$

where:

- μ represents the mean of the pixel values across the dataset.
- σ stands for the standard deviation of the values of the pixel values across the dataset.

Normalizing the image will ensures that all input data will lie withing a similar scale this will prevent the issues issues like convergence caused due to having large variations in input magnitudes [11].

5.3 Proposed Model Architecture

This section outlines the about the model architecture that is used for this research. The model of choice for this task is EfficientNetBo, because it is a very powerful and efficient pertained model which as it's names suggest's is very efficient and requires less computing power without compromising the final results. It will extract that

features and then can be used to include uncertainty estimation into the pipeline. This will help in improving the reliability in these models for the clinical environment.

5.3.1 Base Model Architecture

```
base_model = EfficientNetB0(
    include_top=False, weights="imagenet", input_shape=(224, 224, 3), pooling="avg"
)
```

Figure 5.5: Model Base

EfficientNet is considered to be one among the best models in the world on Convolution Neural Network. It is known to be one of the best models made for these sort of tasks which is able to do it much more efficiently but not compromising the performance. EfficientNetBo known for its efficient scaling of depth, width, and resolution and it is used as backbone of the proposed model [12]. The network will be initialized with pre-trained ImageNet weights so as to make use of the transfer learning. The architecture includes global average pooling at the output of the backbone, followed by a fully connected layer with 128 neurons and a ReLU activation function. A Dropout layer with a rate of 0.3 is introduced to prevent overfitting. And the input shape of the images will be set to 224×224 pixels as expected by the EfficientNetBo model for best performance.

5.3.2 Adding Uncertainty Estimation

As seen in the paper that have been working on Alzheimer's detection problem, inclusion of uncertainty/confidence parameters for the output to show the confidence of the models is something that is used little to none. Inclusion of this parameter will give the clinicians an idea of how confidently the model is been able to classify the image into that class. In this study to quantify uncertainty, Monte Carlo (MC) Dropout is been used during inference step. Monete Carlo is choosen as the method to calculate uncertainity because this model does not require additional model training. This means a dropout layer will be included while conducting the prediction to perform multiple stochastic forward passes. To calculate the uncertainty, we use the standard deviation of the predictions across all these passes. Let's break down the formal definition of uncertainty for a set of n forward passes:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - \bar{p})^2}, \tag{5.1}$$

here p_i stands for the prediction for the i-th forward pass, and $\bar{p}$ is the mean prediction.

The uncertainty value is then converted to a confidence score deducting the uncertainty score by 1. This is because the uncertainty and confidence are inversely promotional. Hence by deducting the value we get a confidence score of the model. Confidence score is calculated using the below formula:

$$C = 1 - \sigma, \tag{5.2}$$

where C represents the confidence score and σ is the uncertainty score that is calculated using the previous formula. To determine whether a prediction is reliable or not, a confidence score threshold of 0.2 is selected. The model's prediction is considered as confident if the score is greater than 0.8 and anything below will be considered.

5.3.3 Output Layer

```
model = Sequential(
    [
        base_model,
        Flatten(),
        Dense(128, activation="relu"),
        Dropout(0.3),
        Dense(4, activation="softmax"),
    ]
)
```

Figure 5.6: Model sequential

The output layer has 4 neurons. This is because there are four different classes that the model will be predicting. Hence there are four neurons. The activation layer user for the final output layer is 'softmax'. The output layer will be creating probability scores for each class: NonDemented, MildDemented, VeryMildDemented and ModerateDemented.

5.3.4 Training and Evaluation

The model is compiled using the Adamax optimizer with a learning rate of 0.001. The categorical crossentropy loss function is employed to optimize multi-class classification performance. Early stopping and learning rate reduction strategies are implemented to ensure robust convergence.

The model is trained on the training dataset for N epochs with validation monitoring. Performance metrics, including accuracy, precision, recall, and F1 score, are computed on the test dataset. Monte Carlo predictions are utilized during evaluation to derive the mean prediction and uncertainty values.

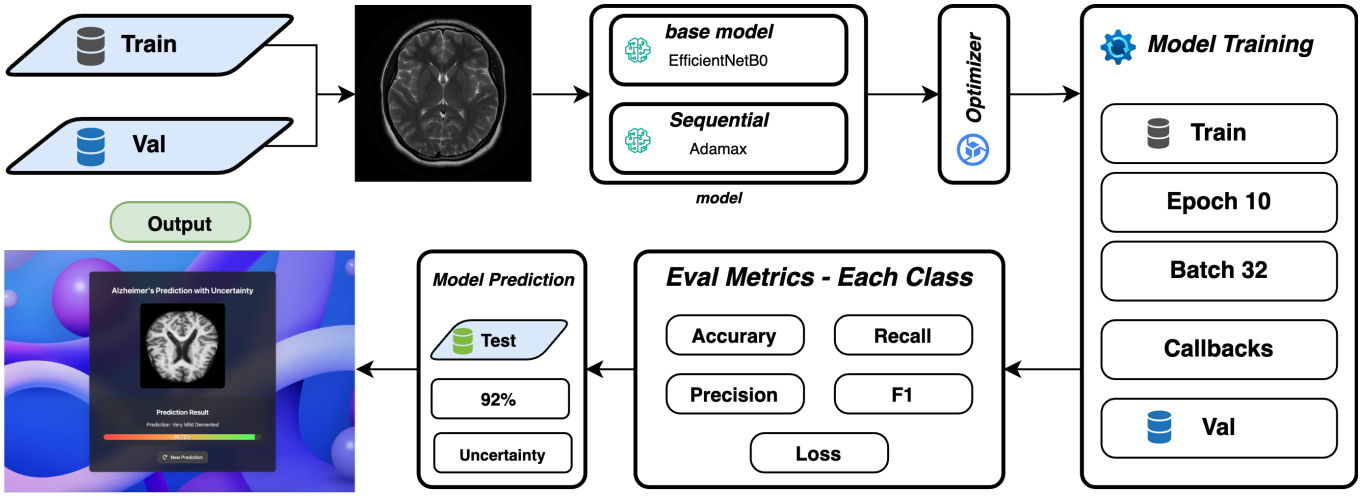


Figure 5.7: Flow of model training and output

5.4 Model Training and Optimization

5.4.1 Training Configuration

This step is consider to be the most critical steps in the machine learning process. How model is configured has a huge impact on the models perfomance on the final task as well as the efficiency of the model while training as well while predicting the results. The model used here make use of the Adamax optimizer. It is a variant of the Adam optimizer. But we make use of Adamax optimizer particularly because Adamax is suits extremely well for the sparse gradients and high dimensional data.So it fit's well for the image dataset that we are using. The learning rate is set to 0.001. In several studies it has been showed that learning rate of 0.001 balances convergence speed and model stability as it has been demonstrated in prior studies to balance convergence speed and model stability [18]. The categorical crossentropy loss function is utilized as this is a multi class classification. It has proved to be very effective for the multi-class classification.The Formula for categorical crossentropy is as follows 5.3:

$$L = -\sum_{i=1}^{N} y_i \log(\hat{y}_i), \tag{5.3}$$

here y_i stands for the true label, and $\hat{y}_i$ stands for the predicted probability for the i-th class.

5.4.2 Optimization Strategies

To improve the model's generalization, the proposed model introduces two optimization strategies: Early Stopping and Learning Rate reduction

Early Stopping

Introduction of the Early stopping while training model will make sure the the validation loss in always monitored. And when performance of the validation set does not improve it will stop the training, once the patience period

is reached. This will ensure that the model is not overfitted for the test model. This is improve the generalizability of the model. And also this will prevent the unnecessary computation power that is required [25]. The patience parameter is set to five epochs. The parameter `restore_best_weights` will make sure thaat model always comes back to it's best-performing weights. This is an example algorithm 1 showing the same.

Algorithm 1 Early Stopping

```
Set's patience p to 0
for every epoch do
    if Validation loss improves then
        Saves the model weights
        Resets patience p equals 0
    else
        Increments patience p by 1
    end if
    if p is >= to patience then
        Training will be Stopped
    end if
end for
```

Learning Rate Reduction on Plateau

If the learning rate doesn't improve for three epochs in a row, we decrease it by half. This way, the model finds a better local minimum and keeps training stable. [21] says this method works really well. After the reduction, we calculate the new learning rate lr_t as follows:

$$lr_t = \max(f \cdot lr_{t-1}, lr_{\min}), \tag{5.4}$$

where f is the reduction factor, lr_{t-1} is the learning rate in the previous epoch, and $lr_{\min}$ is the minimum allowable learning rate.

5.4.3 Training Process

The model training process was done using 10 epochs, on the training set and then it is validated on the validation set. So that model can be evaluated on each epoch. The history which training will be saved which included parameters such as accuracy, loss information. These evaluation are being saved for both the sets i.e. training set as well as the validation set.Table 5.1 give a overall summary of the full training process.

5.5 Evaluation Metrics

To evaluate the performance of the base as well as the proposed model, we will be taking into account several metrics. They are accuracy, precision, recall, F1-score, macro average and weighted average. This will allows to get a very detailed understand of the models performance. Additionally because this is a multi class classification

Table 5.1: Training Parameters and Configuration

Parameter	Value
Optimizer	Adamax
Learning Rate	0.001
Loss Function	Categorical Crossentropy
Early Stopping Patience	5 epochs
Learning Rate Reduction Factor	0.5
Minimum Learning Rate	1×10^{-6}
Epochs	Set to 10 but it is Dynamic (Based on Early Stopping)

this thesis put light to the evaluation metrics on each classes to that it records how well it performs for each class. These metrics will provide a detailed view of how well the model is being able to make predictions for each class. Furthermore, uncertainty evaluation is being implemented in this thesis. This is something that was not included in the base model. Tell me give us idea about how confident the model is when it comes to predicting the values.

5.5.1 Accuracy

Accuracy is a very important metric while evaluating classification models, in this case a multi class classification model. Accuracy shows the ratio of correct predictions to the total number of predictions made. The formula used to calculate accuracy is as follows:

$$\text{Accuracy} = \frac{\text{No. of correct predictions}}{\text{Total no. of predictions}} = \frac{\sum_i I(y_{\text{pred},i} = y_{\text{true},i})}{N}, \tag{5.5}$$

here $y_{\text{pred},i}$ stands for the predicted label i, $y_{\text{true},i}$ stands for the true label, and N is the total number of samples.

5.5.2 Precision

Precision metrics will find a proportion of positive predictions that are that are actually positive. It is important for this thesis because the cost of false positive can be very risky. When it comes to medical diagnosis a false positive prediction can lead to unnecessary medications and treatments. And since the dataset is moderately imbalanced this evaluation metrics becomes very important. The formula for calculating Precision is as follows:

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{5.6}$$

where TP represents true positives and FP represents false positives. In our case of Alzheimer's diagnose it becomes important because minimizing false positive is very important to avoid over diagnosis.

5.5.3 Recall

Recall is also called sensitivity or true positive rate. Recall will be measuring the proportion of actual positive class that are correctly identified by the model. In the context of Alzheimer's disease detection, recall indicates the model's ability to classify all potential Alzheimer's patients. High recall ensures that fewer true positives are

missed, which is crucial for early diagnosis. The formula to calculate Recall is:

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{5.7}$$

where TP represents true positives and FN represents false negatives.

In our case of Alzheimer's classification it becomes where important to check if the model is failing to detect positive case of Alzheimer's or in other words false positives. Because if the model is failing to tell the class which the patient is in then the disease might go unnoticed and it might worsen at the later stages of his/her life.

5.5.4 F1-Score

F1-score is calculated using all of the above scores. It is a 2 times the multiplication of Precision and Recall and devided by addition of Precision and Recall as shown in the equation. The Formula for F1 score is as follow: The F1-score is the harmonic mean of precision and recall, providing a balanced measure of the model's performance. It is particularly useful when dealing with class imbalance, as it considers both false positives and false negatives. The F1-score is defined as:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{5.8}$$

5.5.5 Confusion Matrix

By using the confusion matrix we are able to visualize the performance of the model. Confusion matrix visualizes the number of correctly predicted items and incorrectly predicted items across all classes. This will enable to calculate the evaluation metrics like accuracy, recall, precision and F1-score for each class as shown the table 5.2.

	Predicted			
True	**NonDemented**	**VeryMildDemented**	**MildDemented**	**ModerateDemented**
NonDemented	TP_1	FN_1	FN_2	FN_3
VeryMildDemented	FP_1	TP_2	FN_3	FN_4
MildDemented	FP_2	FP_3	TP_3	FN_5
ModerateDemented	FP_4	FP_5	FP_6	TP_4

Table 5.2: Confusion Matrix for Model Evaluation

5.5.6 Uncertainty Estimation

Along with all the standard classification metrics that are used in previous researches, in this research models uncertainty while predicting the class in being calculated. For this task the chosen method is Monte Carlo Sampling. For each prediction the model will evaluate multiple times, and the variance which predicting among all samples is used to calculate uncertainty. The predicted mean and the uncertainty is calculated using the below formula:

$$\text{mean predictions} = \frac{1}{N_{\text{samples}}} \sum_{i=1}^{N_{\text{samples}}} \hat{y}_i, \tag{5.9}$$

$$\text{uncertainty} = \sqrt{\frac{1}{N_{\text{samples}}} \sum_{i=1}^{N_{\text{samples}}} (\hat{y}_i - \text{mean predictions})^2}, \tag{5.10}$$

where $\hat{y}_i$ represents the i-th prediction, and N_{samples} is the number of Monte Carlo samples [9].

In this study, we used 5 samples for each test input, and the predictions with higher uncertainty (above a threshold of 0.2) were identified to assess where the model's confidence is low.

5.5.7 Performance Evaluation on Test Set

The model's performance was evaluated on the test set using the metrics discussed above. The test accuracy was computed and reported, along with the confusion matrix, as shown in Table 5.2. Additionally, class-specific metrics such as precision, recall, and F1-score were calculated and displayed in a detailed classification report.

5.6 Pipeline Integration

The final model after all the testing and comparisons in integrated with a Web Application which based on Flask API. This will allow the clinicians to upload the MRI images and get the predictions along with the uncertainty scores. The web application is ready to be deployed and will be accessible through a web browser. You can install the model locally and then move it to a cloud server later. The interface is super easy to use, with a simple way to upload photos and see the predictions. Plus, the uncertainty ratings are shown alongside the predictions, so you can see how reliable the model is. That's will be a big help for clinicians!

5.7 Training environment for Model Comparison

To make sure that there is farness in comparing the base model and the and the proposed model. Both were run in the Kaggle notebook. Below table shows the system configuration sleeted for training the model.

	Configuration
Platform	Kaggle Notebooks
language	Python
GPU	GPU T4 x2
CPU	Intel Xenon
RAM	Max 29 GB

Table 5.3: System configuration

5.8 Implementation

A thesis implementation is what bridges the gap between the theory what is taught about to implement and the practical application of the ideas. It will show how we have turned everything that we have discussed in the Methodology section into code , and it will also explain what steps are taken and the reason behind the choices that are make. It will also include the challenges faced during the implementation phase of the thesis.

Here we will also include the detailed explanation of the environment used for the development, including programming languages, hardware and software specs, and the additional tools used. It explains the steps taken, the choices made, and the reasons behind the approaches chosen. This is to make sure that the others who will be continuing this study can to follow the research and even build on it later on.

Any changes or differences from the original plan will also be covered in the this chapter, along with the reasons for them. So that it gives a realistic view of how practical the suggested solution is by showing how real-world limitations and constraints affect it. This important study shows possible directions for further research and helps us understand the topic better.

In summary, a thesis's implementation chapter is a key part that turns ideas into real-world applications. It shows how the research was done, makes it easy to repeat the research, and gives a clear understanding of the results.

5.8.1 Data Preparation

The performance and dependability of machine learning models are greatly impacted by the data preparation stage of implementation. This process includes operations like data cleansing, normalization, transformation, and feature engineering, entails converting raw data into a clear and organized format which is more optimized for the model that chosen. The efficiency of machine learning models is directly impacted by the quality of the input data. Hence it is very important spend enough time and effort on data preparation procedures in order to make sure precise and significant improvement of the models performance.

To reduce problems like data noise, input size error, data preparation methods are very important. This includes deleting corrupt images, encoding class values using one hot encoding, and normalizing images data according to the requirement of the model.By taking these actions, the model's capacity to efficiently learn from the data is increased, improving its prediction performance.

In conclusion, careful data preparation is essential to machine learning model deployment success. Practitioners can greatly improve model performance and obtain more precise and useful insights from their studies by guaranteeing data quality and relevance through thorough preprocessing. The data preparation will have these 5 steps:

- **Loading the Images Data**: In order to organize and label MRI image data associated with Alzheimer's disease, the given code segment is an essential component of data preparation for a machine learning assignment. It starts by specifying the directory paths for four different types of images: Very Mild Demented, Moderate Demented, Non-Demented, and Mild Demented. Unambiguous access to the data stored in the filesystem is ensured by the absolute paths used to specify these directories.

```
MildDemented_dir = "/Volumes/JasonT7/2.Education/Research/Thesis/Paper/0017.
alzheimerPrediction/data2/external/MildDemented"
ModerateDemented_dir = "/Volumes/JasonT7/2.Education/Research/Thesis/Paper/0017.
alzheimerPrediction/data2/external/ModerateDemented"
NonDemented_dir = "/Volumes/JasonT7/2.Education/Research/Thesis/Paper/0017.
alzheimerPrediction/data2/external/NonDemented"
VeryMildDemented_dir = "/Volumes/JasonT7/2.Education/Research/Thesis/Paper/0017.
alzheimerPrediction/data2/external/VeryMildDemented"

filepaths, labels = [], []
class_labels = [
    "Mild Demented",
    "Moderate Demented",
    "Non Demented",
    "Very MildDemented",
]
dict_list = [
    MildDemented_dir,
    ModerateDemented_dir,
    NonDemented_dir,
    VeryMildDemented_dir,
]

for i, j in enumerate(dict_list):
    flist = os.listdir(j)
    for f in flist:
        fpath = os.path.join(j, f)
        filepaths.append(fpath)
        labels.append(class_labels[i])
```

Figure 5.8: Loading the image paths into a dataset

The MRI scans in each directory correspond to the associated categorization, which makes supervised learning easier by matching each image to the appropriate label. Next, two empty lists, `filepaths` and `labels`, are created. These lists were created to hold the full paths of all image files and the class labels that the images belong to. Additionally, it creates a list called `class_labels` that will store the class names for each Alzheimer's disease category that are present in our dataset. This list keeps a relationship in the code by acting as a mapping to the class labels and the path. In order for the script to cycle over the directories in an organized way, a list called `dict_list` is simultaneously formed that will store the directory paths.

A nested loop structure is used for achieving the main functionality. The outer loop uses an enumeration to loop through the `dict_list`, where `i` refers to the image paths's index and `j` is the image path. This code stores the names of all the files in the current folder after retrieving the list of filenames for each directory using the `os.listdir()` method. Each file in this list is processed by an inner loop, which uses `os.path.join()` to add the directory path that is `j` and the filename that is `f` to create the file's complete path.

The proper class labels, obtained from the `class_labels` list using the index `i`, is added to the `labels` list, and these complete file paths are appended to the `filepaths` list. The organized dataset that this code creates, where each image corresponds to its proper class name, allows the machine learning model to differentiate between the different stages of Alzheimer's disease. By keeping separate lists of file locations and labels, the code makes sure that the data is ready for further procedures like data splitting, preprocessing, and input into the learning method. Additionally, because it uses directory-specific processing and relative indexing, this method is adaptable and scalable for datasets with similar structures. Further, this integrated method for dataset preparation enables efficient handling of huge, complex data and improves reproducibility.

- **Validation the images using File Path**: The 5.9 shows the code for the preprocessing step meant to make

```
valid_filepaths, valid_labels = [], []
for filepath, label in zip(filepaths, labels):
    try:
        with Image.open(filepath) as img:
            img.verify()
            valid_filepaths.append(filepath)
            valid_labels.append(label)
    except (IOError, SyntaxError):
        print(f"Corrupted image file: {filepath}")
```

Figure 5.9: code for validate the images

sure that only valid, uncorrupted images files are included in an array of image files and the labels which belong to them. It works by continuously iterating through the lists of `filepaths` and `labels`, where each `filepath` is an image file's disk location and the `label` is the class or category that goes with it. The `Image.open(filepath)` function will attempt to try to open each image file that is given by filepath within the loop. The `img.verify()` method, which checks the image file's validity without fully decoding it. The code adds the image file's path into the `valid_filepaths` list and the correct label to the `valid_labels` list if the image file is legitimate. The final list will have `filepaths` and `labels` for the images that are valid only.

The code will use the except block to catch any errors that arise in this procedure, whether they are caused by damaged, unreadable, or invalid file formats. Particularly, the code deals with two exceptions: `SyntaxError`, which may be raised if the file is incorrectly written as an image, and `IOError`, which usually happens when a file cannot be read. By using this method, it is ensured that corrupted files won't interfere with further processing steps or model training, which might otherwise end up in mistakes or lower dataset quality.

- **Dividing the data**: Figure 5.10 shows the code snippet for having the dataset ready for training, validation, and testing. It makes use of the `sklearn.model_selection` module's `train_test_split` function to

```
train_set, test_images = train_test_split(data_df, test_size=0.3, random_state=42)
val_set, test_images = train_test_split(test_images, test_size=0.5, random_state=42)
```

Figure 5.10: code for dividing the dataset

divide datasets into three dataframes that is test, train and validation. The dataset, denoted by `data_df`, is divided into two halves in the first line, a temporary test set (`test_images`) and a training set (`train_set`). 30% of the dataset is allotted to the test set, with the remaining 70% left aside for training, according to the `test_size=0.3` option. The reproducibility of the splitting operation is guaranteed by the setting `random_state=42`.The second line splits the temporary test set (`test_images`) into two equal parts, a final test set and a validation set (`val_set`). The code makes sure that the temporary test set is divided equally by assigning 50% of its data to each subgroup by setting `test_size=0.5`. To keep the splitting procedure consistent, the `random_state=42` argument is again used.

- **Preprocessing the Images**: This is an important step for getting the picture datasets set up for training, validation, and testing with `Keras' ImageDataGenerator` class. The figure 5.11 shows the code for preprocessing. Image Data Generator is a component of the `TensorFlow` library. It guarantees that the information is correctly organized and preprocessed before being fed into a convolutional neural network. For the training data (`image_gen_train`) and the validation and test datasets (`image_gen_val_test`), `ImageDataGenerator` is generated. The preprocessing function `preprocess_input`, which is utilized by the generators, standardizes the input photos in accordance with the expected specifications of the `EfficientNetB` model.

  The training, validation, and testing datasets are then loaded and preprocessed using the `flow_from_dataframe` function. It resizes photos to 224x224 pixels, formats them as RGB, applies categorical labels for a multi-class classification task, and extracts image paths and labels from the corresponding dataframes (`train_set`, `val_set`, `test_images`). The number of photos processed in each batch is determined by the `batch_size`, and the data order is maintained by using `shuffle=False`.

  For consistency, the same generator is used to process the training and validation datasets, while a different generator (`image_gen_val_test`) handles the test set, providing flexibility for different preprocessing if necessary. Every dataset preserves consistent transformations, guaranteeing that the model assesses performance in a consistent manner.

```
image_gen = ImageDataGenerator(preprocessing_function=preprocess_input)

    train = image_gen.flow_from_dataframe(
        train_set,
        x_col="filepaths",
        y_col="labels",
        target_size=(224, 224),
        color_mode="rgb",
        class_mode="categorical",
        batch_size=bs,
        shuffle=False,
    )

    val = image_gen.flow_from_dataframe(
        val_set,
        x_col="filepaths",
        y_col="labels",
        target_size=(224, 224),
        color_mode="rgb",
        class_mode="categorical",
        batch_size=bs,
        shuffle=False,
    )

    test = image_gen_val_test.flow_from_dataframe(
        test_images,
        x_col="filepaths",
        y_col="labels",
        target_size=(224, 224),
        color_mode="rgb",
        class_mode="categorical",
        batch_size=bs,
        shuffle=False,
    )

    print(f"Train images:{len(train_set)}")
    print(f"Val images:{len(val_set)}")
    print(f"Test images:{len(test_images)}")
```

Figure 5.11: code for image data pre-processing

5.8.2 Model Development

```
base_model = EfficientNetB0(
    include_top=False, weights="imagenet", input_shape=(224, 224, 3), pooling="avg"
)

model = Sequential(
    [
        base_model,
        Flatten(),
        Dense(128, activation="relu"),
        Dropout(0.3),
        Dense(4, activation="softmax"),
    ]
)

model.compile(
    optimizer=Adamax(learning_rate=0.001),
    loss="categorical_crossentropy",
    metrics=["accuracy"],
)

early_stopping = EarlyStopping(
    monitor="val_loss", patience=5, restore_best_weights=True
)

reduce_lr = ReduceLROnPlateau(monitor="val_loss", factor=0.5, patience=3, min_lr=1e-6)

model.summary()

start_time = time.time()
history = model.fit(
    train, epochs=ep, validation_data=val, callbacks=[early_stopping, reduce_lr]
)
```

Figure 5.12: code for model training

Base model

`EfficientNetB0`, a pretrained model, is selected as the base model to extract the basic features at the beginning of the process. `EfficientNetB0` is an advanced convolutional neural network that was trained on the dataset from `ImageNet`. It is known for its effectiveness and strong performance on a range of uses. The pretrained model's top classification layer is excluded by setting `include_top=False`, enabling customization for the task at hand. `Pooling="avg"` uses global average pooling to lower the dimensionality of the output from the convolutional layers, while the `input_shape` parameter sets the size of the input images to ensure network compatibility. Pretrained weights from `ImageNet` are included to guarantee that the network makes use of a wealth of characteristics acquired during extensive training, greatly accelerating convergence and enhancing accuracy for the specified dataset.

Training the Model and Optimizing

The base model is then easily customized and made more adaptable by integrating it into a `Sequential` model, which is a linear stack of layers in `Keras`. The output is flattened using the `Flatten()` layer after the base model, which transforms the multidimensional tensor into a one-dimensional vector appropriate for the dense layers. The addition of a `Dense` layer with 128 units and `ReLU` activation creates a fully connected layer that can recognize intricate patterns in the data. A `Dropout` layer with a rate of 0.3 is incorporated to lessen overfitting. During training, 30% of the weights are randomly set to zero. The last layer, another `Dense` layer, uses a `softmax` activation function to output class probabilities and has four units, which correspond to the number of output classes. The `Adamax` optimizer is an adaptive learning rate optimization method that was inspired by `Adam`. `Adamax` is used to construct the model. It is set up with a learning rate of 0.001 and is especially good at managing sparse gradients. `Accuracy`, which measures the percentage of accurate predictions, is the evaluation metric. The loss function is set to `categorical_crossentropy`, a common option for multi-class classification tasks.

Two callbacks, `EarlyStopping` and `ReduceLROnPlateau`, are used to improve training efficiency and avoid overfitting. The `EarlyStopping` callback restores the optimal weights seen during training by tracking the validation loss (`val_loss`) and stopping training if there is no improvement for five consecutive epochs. With a minimum learning rate threshold of 1×10^{-6}, the `ReduceLROnPlateau` callback automatically lowers the learning rate by a factor of 0.5 if the validation loss does not improve after three epochs. By avoiding needless training epochs, these callbacks guarantee that the model converges efficiently.

And then a summary of the model is generated. The `fit` method, which starts training, requires three inputs: the validation dataset (`val`), the number of epochs (`ep`), and the training dataset (`train`). To ensure no dynamic changes during training, the callbacks are provided as a list. The training duration is measured using the `time.time()` method, which offers information about the model's computational effectiveness. To attain the best results in multi-class image classification jobs, this end-to-end configuration incorporates cutting-edge methods and resources.

5.8.3 Adding Uncertainity estimation to the mix

```python
def predict_with_uncertainty(model, dataset, n_samples=5):
    predictions = []
    total_batches = len(dataset)
    pbar_outer = tqdm(total=n_samples, desc="Monte Carlo Sampling", dynamic_ncols=True)

    for sample_idx in range(n_samples):
        batch_preds = []
        dataset.reset()
        pbar_inner = tqdm(
            total=total_batches,
            position=1,
            leave=False,
            desc=f"Processing Sample {sample_idx + 1}/{n_samples}",
            ncols=80,
        )

        for batch_idx, (images, _) in enumerate(dataset):
            if batch_idx >= total_batches:
                break
            preds = model(images, training=True)
            batch_preds.append(preds.numpy())
            pbar_inner.update(1)
        pbar_inner.close()
        predictions.append(np.vstack(batch_preds))
        pbar_outer.update(1)
    pbar_outer.close()
    predictions = np.stack(predictions, axis=0)
    mean_preds = np.mean(predictions, axis=0)
    uncertainty = np.std(predictions, axis=0)

    return mean_preds, uncertainty

mean_predictions, uncertainty = predict_with_uncertainty(model, test, n_samples=5)
```

Figure 5.13: code for uncertainity estimation

This code uses `Monte Carlo dropout` sampling for calculating uncertainty. It is a `Bayesian` technique that takes into account model prediction variability based on the random nature of `dropout` layers, to evaluate predictive uncertainty. To measure uncertainty, the `predict_with_uncertainty` function does multiple forward tests with the model in training mode even during inference. This function needs three inputs: the number of MC samples (`n_samples`), a dataset (usually for testing), and the trained model. It determines the total number of batches in the dataset and initializes an empty list called `predictions` to hold the outcomes of several forward runs. Throughout the designated number of iterations, the sampling process is tracked via a progress bar (`pbar_outer`).

To make sure of consistent processing, the dataset is reset after every MC sample, and batch-wise progress is monitored by a separate progress bar (`pbar_inner`). The model adds predictions to `batch_preds` as `NumPy` arrays while processing batches of photos while maintaining `dropout` layers active, that is, with `training=True`. Pre-

dictions for the sample are stacked and added to the `predictions` list once all batches have been processed. For the designated number of samples, this procedure is repeated. `Predictions` are merged into a three-dimensional array after every sample is finished. Along the sample axis, the function computes the `standard deviation` (that is, `uncertainty`) and `mean predictions` (that is, `mean_preds`). The standard deviation measures the degree of uncertainty surrounding the model's final predictions, whereas the mean represents those forecasts.

5.8.4 Performance Evaluation

```python
def predict_with_uncertainty(model, dataset, n_samples=5):
    predictions = []
    total_batches = len(dataset)
    pbar_outer = tqdm(total=n_samples, desc="Monte Carlo Sampling", dynamic_ncols=True)

    for sample_idx in range(n_samples):
        batch_preds = []
        dataset.reset()
        pbar_inner = tqdm(
            total=total_batches,
            position=1,
            leave=False,
            desc=f"Processing Sample {sample_idx + 1}/{n_samples}",
            ncols=80,
        )

        for batch_idx, (images, _) in enumerate(dataset):
            if batch_idx >= total_batches:
                break
            preds = model(images, training=True)
            batch_preds.append(preds.numpy())
            pbar_inner.update(1)
        pbar_inner.close()
        predictions.append(np.vstack(batch_preds))
        pbar_outer.update(1)
    pbar_outer.close()
    predictions = np.stack(predictions, axis=0)
    mean_preds = np.mean(predictions, axis=0)
    uncertainty = np.std(predictions, axis=0)

    return mean_preds, uncertainty

mean_predictions, uncertainty = predict_with_uncertainty(model, test, n_samples=5)
```

Figure 5.14: code for uncertainity estimation

This code evaluates a multi-class classification model by calculating metrics and organizing them. It extracts ground truth labels `y_true` from the test dataset and converts them into a `NumPy` array. Predicted labels `y_pred` are generated using `np.argmax` on `mean_predictions`. The `sklearn.metrics` `classification_report` function computes precision, recall, and F1-score for each class, linking them to class names with `target_names=class_la` The `output_dict=True` option returns the report as a dictionary, converted to a `Pandas DataFrame` `class_metrics` for analysis. Class-level accuracies are calculated as correct predictions per class relative to total samples, stored in

`class_accuracy` and added to `class_metrics`. Aggregate accuracy metrics—macro, micro, and weighted averages—are computed for broader evaluation. Precision, recall, and F1-score are also computed for micro-average, stored in the `"micro avg"` row of `class_metrics`. Finally, `class_metrics` `DataFrame` is displayed, offering a comprehensive breakdown of metrics for each class and overall performance.

5.8.5 Integrating the Model with a Web Application

The model that is trained on these images of MRI multiples classes images will be saved to the directory, and then this model will be used to connect to an web application which makes use of Flask API to upload the image and the predict the images into the class which it belongs to. Python code used to write the backend of this webapplication and a basic html to design the page and the a css is applied on top of this html to style the web application. The Flask framework will make use fo RESTfull API. The Webapplication has two buttons:

- **Upload Button**: The upload button is used to select the images from the local storage. Once it is upload there will be a preview of the image that is being uploaded.
- **Predict Button**: The Predict button will take the user to the prediction screen where the predicted class and the confidence score is calculated and displayed in the progress bar.

Connect the backend with the Flask API Integration

```python
@app.route("/predict", methods=["POST"])
def predict():
    if "file" not in request.files:
        return jsonify({"error": "No file part"}), 400

    file = request.files["file"]

    if file.filename == "":
        return jsonify({"error": "No selected file"}), 400

    if file:
        try:
            with tempfile.NamedTemporaryFile(delete=False, suffix=".png") as temp_file:
                file.save(temp_file.name)
                filepath = temp_file.name
            img_array = preprocess_image(filepath)
            predictions, confidence, predicted_class = predict_with_uncertainty(
                model, img_array, n_samples=10
            )
            predicted_label = class_labels[predicted_class]
            with open(filepath, "rb") as image_file:
                encoded_image = base64.b64encode(image_file.read()).decode("utf-8")
            os.unlink(filepath)
            return jsonify(
                {
                    "predicted_label": predicted_label,
                    "confidence": float(confidence),
                    "predictions": predictions.tolist(),
                    "image": encoded_image,
                }
            )
        except Exception as e:
            return jsonify({"error": str(e)}), 500
    return jsonify({"error": "Invalid file"}), 400
```

Figure 5.15: code for flask api to predict the image

To connect the Front end with the back end prediction code we have made use to Flask API. It will be using RESTfull API. There is only one POST method that is used which will be sending the image, which once recieved in the backend will run the prediction model with uncertainity score and confidence score. The figure 5.15 shows you the code for predict method that is called to predict the image and also send you the confidence score. The confidence score and the predicted class will be then sent to the front end. The results will be then with a progresss bar.

1. Preprocesses the image to match the model's input requirements.
2. Performs inference using the trained model.
3. Computes uncertainty metrics using the Monte Carlo dropout technique.
4. Returns the prediction results and uncertainty measures to the frontend.

Chapter 6

Results and Discussion

The results of the implementation of the code developed for creating a deep learning model for Alzheimer's disease using MRI image will be shows here in this section. The results consists of multi-class classification model's performance metrics, an analysis of uncertainty estimations, performance analysis and a study of the insights gained from these. The results are analyzed in terms of overall accuracy of the model, class-wise performance of the model, and the advantages of incorporating uncertainty estimates to improve the model's reliability and interpretability. To further underscore the benefits of the suggested approach, comparisons with base models are provided.

6.1 Overall performance of the model

The performance of the model was excellent. Once the model was created it was run on the test dataset. It was able to predictes the four classes of Alzheimer's stages in our dataset exceptionally well. The model was consistent and reliability in correctly determining MRI images among the four target classes Mild Demented, Moderate Demented, Non-Demented, and Very Mild Demented were shown by its 99.55% overall accuracy on the test dataset. Let's now compare that to the base model.

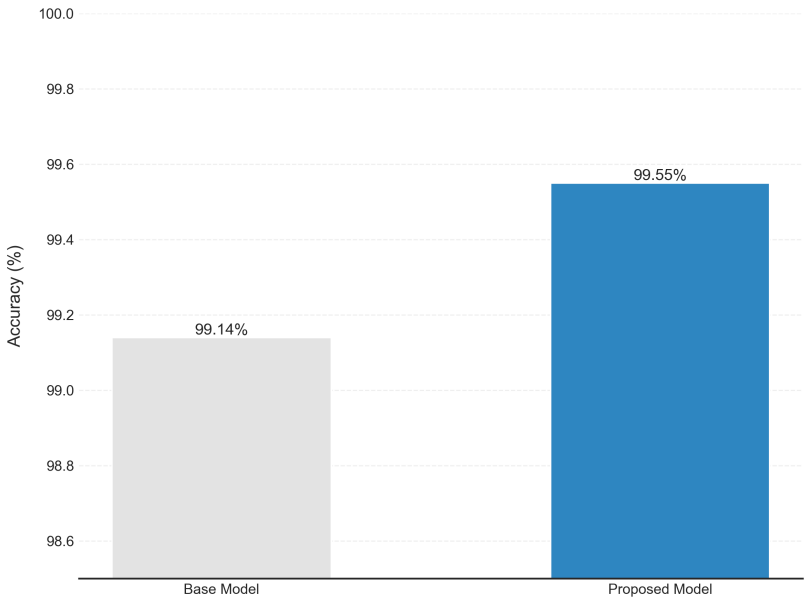


Figure 6.1: figure showing the performance improvement in the proposed model (Higher the Better)

Note: the figure 6.1 is zoomed in between 99% to 99.1% to view the performance impovement in the proposed model. And these numbers are based on my tests conducted on these same exacpt system specifications. This

figure figure 6.1 shows the slight improvement in the models accuracy percentage. This improve might look not very significant, but when it comes to a medical world even a sligtest of mistakes can major impact on a persons life. Hence this a good performance gain achieved.

6.2 Evaluating Classwise the model using the Metrics Chosen

Class	Precision	Recall	F1-Score	Accuracy
Mild Demented	0.99	1.00	1.00	1.00
Moderate Demented	1.00	1.00	1.00	1.00
Non Demented	0.99	0.99	0.99	0.99
Very Mild Demented	0.99	0.99	0.99	0.99
Macro Avg	0.99	0.99	0.99	0.99
Weighted Avg	0.99	0.99	0.99	0.99

Table 6.1: Table showing classwise evaluation metrics

This class wise evaluation metrics in the figure 6.1 provides a better idea of model's performance for each stages of the Alzheimers disease classes that are present in our dataset provided by the class-wise summary. In medical applications, where particular classes may need higher sensitivity because of their clinical implications, this precision is essential.For the Moderate Demented and Mild Demented classes, the model shows perfect scores that is Precision, Recall, F1-Score, and Accuracy, showing its great reliability in identifying these states. Strong generalization across all diagnostic classes is shown by the metrics for the Non-Demented and Very Mild Demented classes, which are a little lower but yet so close to ideal score that is 0.99 across all categories. So we can see that accross all the metrics the results are very balanced. Even the the dataset being moderately imbalanced did not have any effect on the final results as seen the weighted average row.

6.3 visualizing the confidence score to the plot

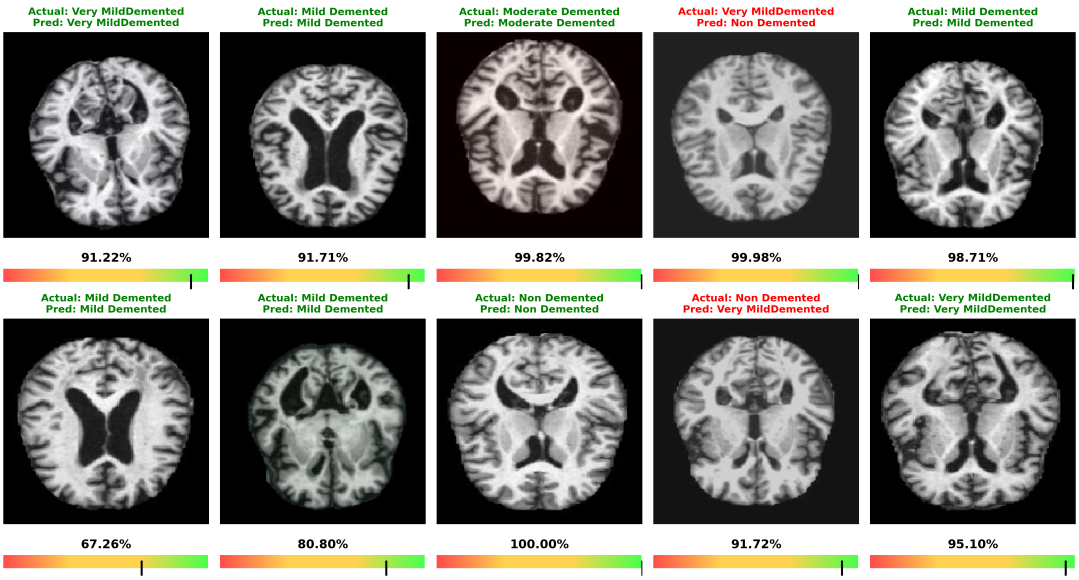
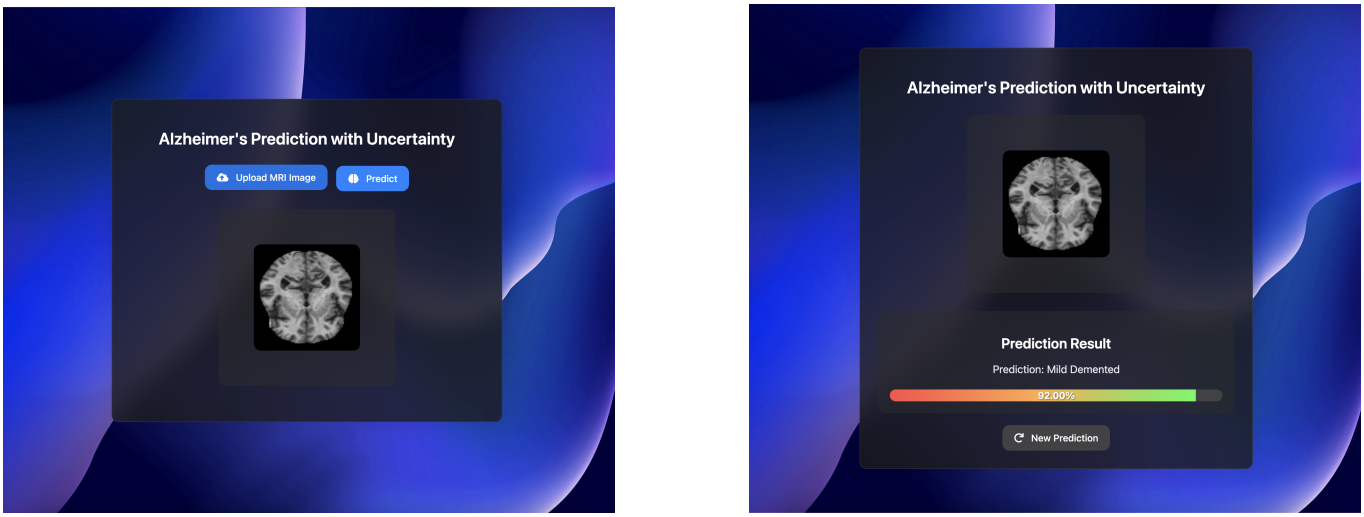


Figure 6.2: image plotting the predicted class, actual class as well as the condifence percentage in progress bar.

A sample code was writen to predict a sample of 10 image and plot them in a image and show the confidence(1 - uncertainity) indicator as a progress bar. The figure 6.2 show us as plot of these sample 10 images. As you can see in this figure out of the 10 images except the 2 images the rest of the images were predicted correctly. The images which are predicted wrong , the text is been highlighted in red. And this image is a proof why including confidence score is such a gamechanger. Because if you see the first image in the second row even though the image class is predicted correctly, the model does not look confident of this class. When there are scenarios where a models get's a input which is different from the one's it has been trained on it's tends to be uncertain. This is where having this as a output will make sure that the clinicians take further action and analysis thes MRI's further and take a informed decision.

6.4 Model implementation on the WebPage



(a) step 1: Upload and preview image

(b) step 2: Predict the image class with confidence score

Figure 6.3: Image Classification Web Application Workflow

The figure 6.3 shows the final images of a simple webpage application where in the initial screen there are two buttons. Upload and predict. The user needs to click on the upload button first which will take him to the image selection dialogue box. Once selected it will give a preview of the image. Next, predict botton needs to be clicked. Which make a predict API call. The model is connected to the webpage using Flask architecture. Using RESTfull API's a method called 'predict' is created. The API will then send the predicted class and the confidence score in return. The final screen has progress bar which has a gradient of red to green. That is where the uncertainity score is shown to the user. Anything below 80% is considered as bad confidence score. If in that case further study needs needs to be done on that case.

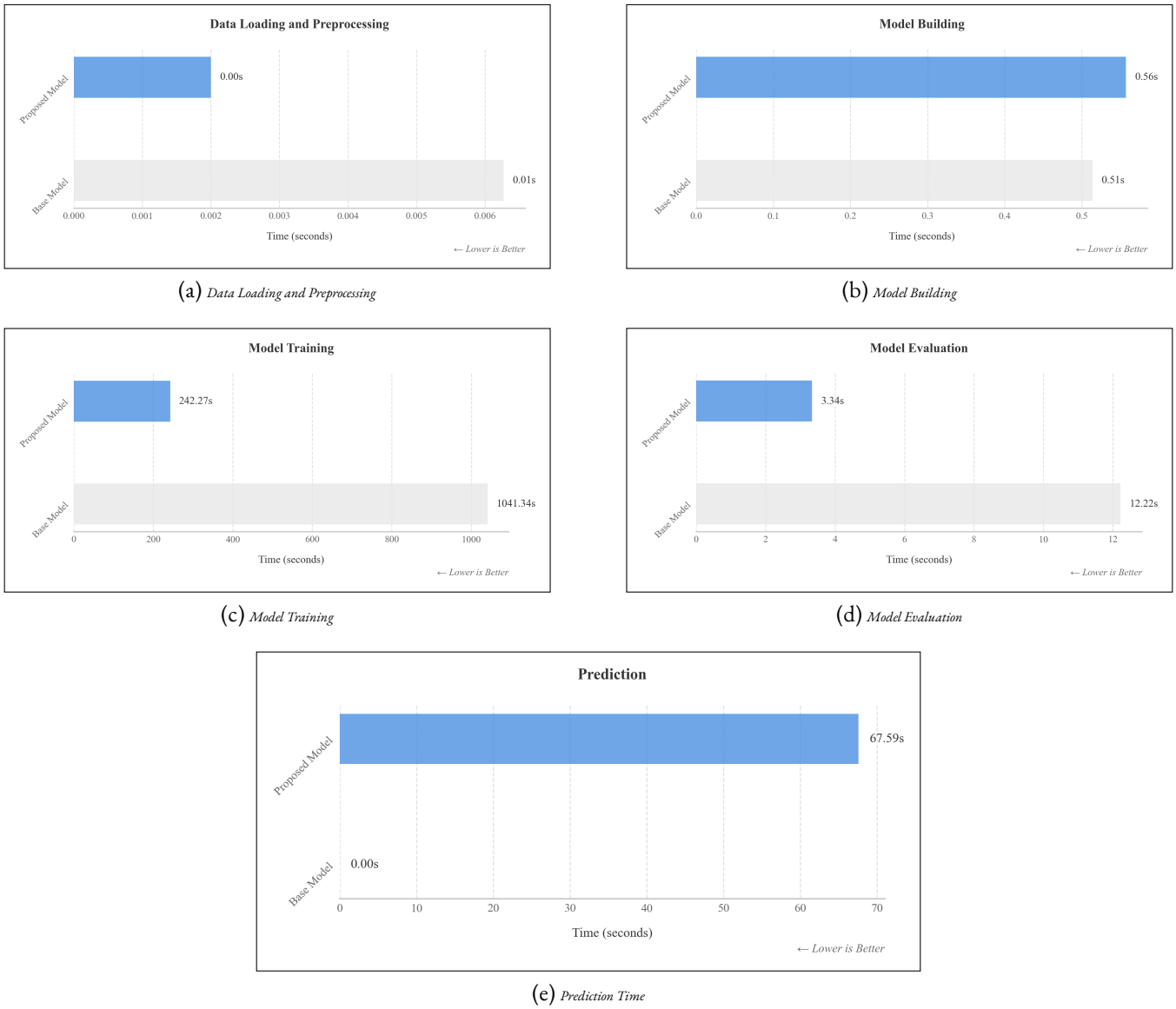
6.5 Model Training performance and Comparision with base model

Table 6.2: Computing time comparison between the base model and the proposed model (in seconds).

Stage	Base Model	Proposed Model
Data Loading and Preprocessing	0.01	0.00
Model Building	0.51	0.56
Model Training	1041.34	242.27
Model Evaluation	12.22	3.34
Prediction	0.00	67.59

6.5.1 Data Loading and Preprocessing

The proposed model demonstrated a marginally faster time of 0.00 seconds compared to the base model of 0.01 seconds. This is very negligible and can be ignored due to the fact that 0.01 secs does not effect the computational



(a) *Data Loading and Preprocessing*

(b) *Model Building*

(c) *Model Training*

(d) *Model Evaluation*

(e) *Prediction Time*

Figure 6.4: Computing time comparison between the base model and the proposed model for different stages of the pipeline. Lower values indicate better performance.

cost. You can see this in the Figure 6.4a.

6.5.2 Model Building

While comes to Model building there 4 secs more taken by the proposed mode. This maybe due to the fact the as base model (Xception) code was giving some errors we had to download and and then load it. Which reduced the time to download and load it in the process flow itself. You can refer this in the Figure 6.4b, it provides a bar chart comparison of the model-building times.

6.5.3 Model Training

The most significant improvement notices was in the training stage. The proposed model reduced the training time to 242.27 seconds, compared to 1041.34 seconds for the base model which is approximately 4x times faster that the base mode. This reduction shows that the proposed model's design enables more efficient learning without compromise on the perfomance. As seen in Figure 6.4c, this stage is a critical performance improvement.

6.5.4 Model Evaluation

In the evaluation stage, the proposed model completed the process in just 3.34 seconds, significantly outperforming the base model, which required 12.22 seconds. This improvement is shown in the Figure 6.4d, where the proposed model again shows a 4x times better performance improvement.

6.5.5 Prediction

The prediction stage showed us a differnt story where base model clearly outperformed the proposed model by 6x times better perfomance. The base model's prediction time was very negligible that is 0.00 seconds and the proposed model took 67.59 seconds. The reason behind this might be the inclusion of uncertainity calculation layer and calation of confidence percentage. And the prediction runs for number of time for each image this add for extra computation time and overhead. Figure 6.4e shows these prediction times.

6.5.6 Summary of Observations

The proposed model consistently outperformed the base model in most stages, particularly in training and evaluation. These stages contribute significantly to the overall pipeline, making the proposed model a better candidate for scenarios emphasizing training and evaluation efficiency. However, the longer prediction time is a bit of a problem but having said the what we are getting after this little extra time is a more robust model that is better performing that the base model and the confidence score which it a significant improvement over the base model as the user will know how confidently the model is abount Alzheimer's class. So that they can take a informed decision.

Table 6.3: Computing time comparison between the base model and the proposed model (in seconds).

Stage	Base Model	Proposed Model
Data Loading and Preprocessing	0.01	0.00
Model Building	0.51	0.56
Model Training	1041.34	242.27
Model Evaluation	12.22	3.34
Prediction	0.00	67.59

Chapter 7

Limitation found in this study and what can be improved

Even if the proproposed model's findings showed great results in many number of areas, there are still various ways to increase the model's functionality and performance. The possible options for further research that might improve on the present findings will be covered in this section. A main area where this research can be improved in the furture is the improving the time taken to prediction the images along with Uncertainity. As demonstrated in Section 6.5.5, the proposed model's prediction time is significantly longer compared to the base model, mainly due to the addition of the uncertainty estimation layer and the calcution of confidence scores. One possible way for improvement is to explore more efficient methods for calculating uncertainty like approximate inference techniques or employing model compression methods to reduce the computational load during prediction. The model is currently only trained on MRI scans in order to detect the stage of Alzheimer's disease. However, adding other data modalities like genetic data, PET scans, or cognitive test results could improve the model's performance and robustness. In addition to maybe increasing the model's overall accuracy, multimodal learning techniques, in which the model may process various forms of input at once, may offer more thorough insights into the patient's state. This would be especially helpful in situations where MRI data might not be enough on its own to make precise predictions. Despite its strong performance, the suggested model could be improved by experimenting with various deep learning architectures and carrying out more thorough hyperparameter tuning. Performance gains might be possible, for example, by investigating alternative neural network designs like Transformer-based models or hybrid models that combine convolutional and recurrent layers. More optimization and effective training may also result from adjusting the learning rate, batch size, and other hyperparameters. In the future we could involve collaborating with medical institutions to conduct clinical trials and gather feedback to improve the model's usability and reliability in practice. Although the proposed model's use of uncertainty estimations is a significant addition, this part still needs work. Alternative approaches to measuring uncertainty, like Bayesian Neural Networks, or other ensemble techniques may be explored in future research. These techniques may yield more accurate estimates of uncertainty, especially when the model comes into new or unclear data that it hasn't encountered during training.

Chapter 8

Conclusion

The deep learning model created to classify the stages Alzheimer's disease from MRI scans performed really well by classifying the four stages of the disease with 99.55% accuracy. Even with a somewhat unbalanced dataset, the model consistently performed well across all classes. When the model conveys uncertainty, a crucial aspect in medical applications, the inclusion of uncertainty estimation improved the model's understanding and enabled physicians to make well-informed decisions.

Although the prediction time was marginally higher because of the uncertainty computation, the suggested model performed better than the base model in training and evaluation times. More accurate predictions with confidence scores are produced by this additional complexity, which outweighs the computing expense.

Future research should concentrate on improving uncertainty estimation techniques, investigating multimodal learning with new kinds of meaningful data, and optimizing prediction time. Real-world applicability will also be ensured by working with medical institutions to conduct clinical studies. By incorporating uncertainty-aware deep learning, this study improves Alzheimer's detection and provides important assistance for clinical decision-making and early diagnosis.

Bibliography

[1] "2012 Alzheimer's disease facts and figures". In: Alzheimer's & Dementia 8.2 (2012), pp. 131–168. ISSN: 1552-5260. DOI: https://doi.org/10.1016/j.jalz.2012.02.001. URL: https://www.sciencedirect.com/science/article/pii/S1552526012000325.

[2] Gulnaz Ahmed et al. "DAD-Net: Classification of Alzheimers Disease Using ADASYN Oversampling Technique and Optimized Neural Network". In: Molecules 27.20 (2022), p. 7085. ISSN: 1420-3049. DOI: 10.3390/molecules27207085. URL: https://www.mdpi.com/1420-3049/27/20/7085.

[3] Alzheimer's Association. "2019 Alzheimer's disease facts and figures". In: Alzheimer's & Dementia 15.3 (), pp. 321–387. DOI: https://doi.org/10.1016/j.jalz.2019.01.010. eprint: https://alz-journals.onlinelibrary.wiley.com/doi/pdf/10.1016/j.jalz.2019.01.010. URL: https://alz-journals.onlinelibrary.wiley.com/doi/abs/10.1016/j.jalz.2019.01.010.

[4] Mayo Clinic. "Alzheimer's Disease: Symptoms and Causes". In: (2024). Accessed: November 19, 2024. URL: https://www.mayoclinic.org/diseases-conditions/alzheimers-disease/symptoms-causes/syc-20350447.

[5] Maitry Bimalbhai Desai, Yogesh Kumar, and Shilpa Pandey. "Efficient Approach for Diagnosis and Detection of Alzheimer Diseases Using Deep Learning". In: (2024), pp. 1–5. DOI: 10.1109/ACROSET62108.2024.10743886.

[6] DhanushKumar. EfficientNet — Scaling Depth, Width, Resolution. https://medium.com/@danushidk507/efficientnet-scaling-depth-width-resolution-11e2d4311357. 2023.

[7] Dao Duy Phuong, Hyung-Jeong Yang, and Jahae Kim. "Conditional Diffusion Model for Longitudinal Medical Image Generation". In: (Nov. 2024). DOI: 10.48550/arXiv.2411.05860.

[8] Dheiver Francisco Santos. "Advancing Automated Diagnosis: Convolutional Neural Networks for Alzheimer's Disease Classification through MRI Image Processing". In: (May 2023). DOI: 10.36227/techrxiv.23002007.v1. URL: http://dx.doi.org/10.36227/techrxiv.23002007.v1.

[9] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: (2016). arXiv: 1506.02142 [stat.ML]. URL: https://arxiv.org/abs/1506.02142.

[10] Yubraj Gupta et al. "Early diagnosis of Alzheimer's disease using combined features from voxel-based morphometry and cortical, subcortical, and hippocampus regions of MRI T1 brain images". In: PLOS ONE 14.10 (Oct. 2019), pp. 1–30. DOI: 10.1371/journal.pone.0222446. URL: https://doi.org/10.1371/journal.pone.0222446.

[11] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: Proceedings of the IEEE International Conference on Computer Vision. 2015, pp. 1026–1034.

[12] Van-Thanh Hoang and Kang-Hyun Jo. "Practical Analysis on Architecture of EfficientNet". In: (2021), pp. 1–4. DOI: 10.1109/HSI52170.2021.9538782.

[13] Farhana Islam et al. "A Novel Method for Diagnosing Alzheimer's Disease from MRI Scans using the ResNet50 Feature Extractor and the SVM Classifier". In: (2023). Accessed online from IJACSA website, pp. 1233–1240. URL: http://www.ijacsa.thesai.org.

[14] Bala Venkateswarlu Isunuri and Dr Jagadeesh Kakarla. "Alzheimer's severity classification using Transfer Learning and Residual Separable Convolution Network". In: ICVGIP '22. Gandhinagar, India: Association for Computing Machinery, 2023. ISBN: 9781450398220. DOI: 10.1145/3571600.3571610. URL: https://doi.org/10.1145/3571600.3571610.

[15] T Jo, K Nho, and AJ Saykin. "Deep Learning in Alzheimer's Disease: Diagnostic Classification and Prognostic Prediction Using Neuroimaging Data". In: Frontiers in Aging Neuroscience 11 (Aug. 2019), p. 220. DOI: 10.3389/fnagi.2019.00220.

[16] Nitish Shirish Keskar et al. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. 2017. arXiv: 1609.04836 [cs.LG]. URL: https://arxiv.org/abs/1609.04836.

[17] Asifullah Khan et al. "A Wide Analysis of Loss Functions for Image Classification Using Convolution Neural Network". In: arXiv preprint arXiv:2005.14647 (2020).

[18] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: (2017). arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[19] Ahmed A. Abd El-Latif et al. "Accurate Detection of Alzheimer's Disease Using Lightweight Deep Learning Model on MRI Data". In: Diagnostics 13.7 (2023). ISSN: 2075-4418. DOI: 10.3390/diagnostics13071216. URL: https://www.mdpi.com/2075-4418/13/7/1216.

[20] S. Liu, A.V. Masurkar, and H. et al. Rusinek. "Generalizable deep learning model for early Alzheimer's disease detection from structural MRIs". In: Scientific Reports 12.1 (2022). DOI: 10.1038/s41598-022-20674-x.

[21] Ilya Loshchilov and Frank Hutter. "SGDR: Stochastic Gradient Descent with Warm Restarts". In: (2017). arXiv: 1608.03983 [cs.LG]. URL: https://arxiv.org/abs/1608.03983.

[22] Dominic Masters and Carlo Luschi. Revisiting Small Batch Training for Deep Neural Networks. 2018. arXiv: 1804.07612 [cs.LG]. URL: https://arxiv.org/abs/1804.07612.

[23] Jie Mi et al. "Light on Alzheimer's disease: from basic insights to preclinical studies". In: Frontiers in Aging Neuroscience 16 (2024). ISSN: 1663-4365. DOI: 10.3389/fnagi.2024.1363458. URL: https://www.frontiersin.org/journals/aging-neuroscience/articles/10.3389/fnagi.2024.1363458.

[24] Suriya Murugan et al. "DEMNET: A Deep Learning Model for Early Diagnosis of Alzheimer Diseases and Dementia From MR Images". In: IEEE Access 9 (2021), pp. 90319–90329. DOI: 10.1109/ACCESS.2021.3090474.

[25] Lutz Prechelt. "Early Stopping-But When?" In: (1996). URL: https://api.semanticscholar.org/CorpusID:14049040.

[26] ePack Printing. Understanding RGB Color Model. https://www.epackprinting.com/support/what-is-rgb-color/. 2023.

[27] Saman Sarraf and Ghassem Tofighi. "Classification of Alzheimer's Disease using fMRI Data and Deep Learning Convolutional Neural Networks". In: (2016). arXiv: 1603.08631 [cs.CV]. URL: https://arxiv.org/abs/1603.08631.

[28] Sarang Sharma et al. "HTLML: Hybrid AI Based Model for Detection of Alzheimer's Disease". In: Diagnostics 12.8 (2022). ISSN: 2075-4418. DOI: 10.3390/diagnostics12081833. URL: https://www.mdpi.com/2075-4418/12/8/1833.

[29] DSP StackExchange. Why do we use the HSV colour space so often in vision and image processing? https://dsp.stackexchange.com/questions/2687/why-do-we-use-the-hsv-colour-space-so-often-in-vision-and-image-processing. 2012.

[30] Mingxing Tan and Quoc V Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: arXiv preprint arXiv:1905.11946 (2019).

[31] Keras Team. Image classification via fine-tuning with EfficientNet. https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/. 2020.

[32] Yanzheng Yu. "Deep Learning Approaches for Image Classification". In: EITCE '22 (2023), pp. 1494–1498. DOI: 10.1145/3573428.3573691. URL: https://doi.org/10.1145/3573428.3573691.

[33] Xiang Zhang, Liang Zhang, and Jian Yang. "Robust Multi-Class Classification Using Linearly Scored Categorical Cross-Entropy". In: IEEE Transactions on Neural Networks and Learning Systems (2021).

[34] Ying Zhang et al. "Generalization of Cross-Entropy Loss Function for Image Classification". In: IEEE Access 7 (2019), pp. 86441–86453.

[35] Ke Zou et al. "A Review of Uncertainty Estimation and its Application in Medical Imaging". In: (2023). arXiv: 2302.08119 [eess.IV]. URL: https://arxiv.org/abs/2302.08119.