Bachelor thesis
Computing Science

Radboud University

Comparison of temporal pooling methods and fusion strategies
for Alzheimer's Disease classification using convolutional neural networks with 3D MRI data

Author:
Polina Moroza
s1016615

First supervisor/assessor:
Prof. Elena Marchiori
`Elena.Marchiori@ru.nl`

Second supervisor:
Maryam Khojaste
`M.Khojaste@cs.ru.nl`

Second assessor:
Dr. Johannes Textor
`johannes.textor@ru.nl`

June 23, 2022

Abstract

Alzheimer's disease (AD) is the most common type of dementia. Early diagnosis of AD is crucial in the management of the disease. Manually analyzing brain scans can be time-consuming and challenging. Researchers and doctors can benefit from computer-assisted interventions; in particular, convolutional neural networks (CNNs) can help in the automation of this process.

However, automated AD diagnosis with 3D MRI images is also memory intensive and time consuming, due to need to train 3D convolutional neural networks for this task. An approach to overcome these issues is to transform 3D MRI images into 2D. This can be done by using temporal pooling at the input (pixel) level (early fusion) or by applying the transformation at the feature map level (late fusion). Various temporal pooling methods have been introduced, but it is not clear which one should be preferred.

This study compares different temporal pooling methods, fusion strategies, data splitting techniques and pooling dimensions for the conversion of 3-D MRI images to 2-D for Alzheimer's Disease Classification using 2-D Convolutional Neural Networks. The Alzheimer's Disease Neuroimaging Initiative (ADNI) has gathered data that can be utilized to further research into the diagnosis of AD. In this study ADNI MRI images are used to train the 2D CNN models.

This research evaluates the performance of CNN classifiers for AD diagnosis with combinations of different temporal pooling methods, fusion strategies, data splitting techniques and pooling dimensions. We find the best results for Segmented Max pooling, a temporal pooling method based on a recent research on the classification of digital breast tomosynthesis (DBT), which is an emerging imaging technique for breast cancer screening. [36]

Acknowledgement

Data used in preparation of this article were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database (adni.loni.usc.edu). As such, the investigators within the ADNI contributed to the design and implementation of ADNI and/or provided data but did not participate in analysis or writing of this report. A complete listing of ADNI investigators can be found at: `http://adni.loni.usc.edu/wp-content/uploads/how_to_apply/ADNI_Acknowledgement_List.pdf`

Contents

Chapter 1

Introduction

1.1 Alzheimer's disease

The Alzheimer's disease (AD) is the most common type of dementia. The cumulative incidence of Alzheimer's dementia is expected to climb from 5% by age 70 to 50% by age 90, making it a fairly prevalent disease.[16]

Early diagnosis of AD is crucial in the management of the disease. Patients with early diagnosis can work with their family, caregivers and general practitioner to construct advanced care plans. [29] It also allows patients to get symptomatic relief and lifestyle adjustments to maintain quality of life. Unfortunately, due to a number of factors such as time constraints for clinicians, the difficulty of accurately diagnosing Alzheimer's pathology, and the fact that patients and healthcare providers frequently dismiss symptoms as part of the normal ageing process, it is very hard to detect the disease at an early stage. [29] Therefore it is important to find ways to speed up the detection of Alzheimer's disease.

1.2 Deep learning in medical image analysis

Medical image interpretation has traditionally been done in clinics by human experts such as radiologists and physicians. Researchers and doctors have begun to benefit from computer-assisted interventions due to wide variations in pathology and the potential fatigue of human experts.[33] Machine learning algorithms can find and learn informative features in data. These algorithms are widely used in medical image analysis to find patterns in data and be able to detect pathological images that can be further manually analysed by medical specialists. Deep learning requires a set of labelled images, after which it self-teaches itself to discover informative representations.[33]

Convolutional neural networks (CNN) is a class of deep learning that is often used to classify medical images. By taking 2D or 3D images as input, CNNs are developed to use spatial and configural information. Convolutional layers are interleaved with pooling layers in a CNNs structure, followed by fully linked layers, as in a normal multilayer neural network. [33]

In this study 3-dimensional (3D) magnetic resonance images (MRI) are analysed. The classification task is to distinguish between subjects with Alzheimer's disease (AD) and cognitively normal subjects (CN). Data for this study was taken from Alzheimer's Disease Neuroimaging Initiative (ADNI) database[1]. The Alzheimer's Disease Neuroimaging Initiative (ADNI) is a long-term natural history study whose main goal is to inform the design of Alzheimer's disease therapy trials (AD). [7]

3D information is often a necessity for good performance in object classification tasks in medical imaging, however 3D CNNs are computationally costly and hard to be optimized with small datasets. [23] A paper by Gongbo Liang et. al. [22] proposes to use 2D CNN models as alternative approaches for MRI classification. The key idea is to convert 3D MRI scans to a 2D format using temporal pooling and fusion strategies.

This study compares three temporal pooling methods, max pooling, segmented max pooling and dynamic pooling; two fusion strategies, early fusion and late fusion; two data splitting techniques, 4:1 ratio and cross-validation; and all pooling dimensions, X, Y and Z. For the used images, the X-dimension is the front-back view of the brain (coronal), the Y dimension is the side-view of the brain (saggital) and the Z dimension the top-down view of the brain (axial). The following research questions are addressed:

1. Are the results found in the research of Gongbo Liang et al. [22] reproducible?
2. How do the evaluation results vary when using different data splitting strategies?
3. How does the pooling dimension affect the performance of the temporal pooling methods?
4. How do the temporal pooling methods compare in terms of predictive performance and training time?

[1]adni.loni.usc.edu

Chapter 2

Background

2.1 Image classification tasks

Image classification is a critical task in a variety of medical imaging applications. To achieve good classification performance in image classification tasks, the descriptiveness and the discriminative capability of the extracted features are important.[21] Convolutional Neural Networks are a powerful classification tool that have been proven very successful in solving image classification problems.[21]

2.2 Convolutional neural networks

A Convolutional Neural Network (CNN) is a deep neural network. It takes this name from a mathematical operation between matrices called convolution. Convolution is achieved by multiplying the pixel's intensity value with the intensity values of its neighbors using a matrix. CNN's are mostly used for image-driven pattern recognition tasks. [25]

A CNN has multiple types of layers:

1. A convolutional layer consists of a set of small filters or kernels that is moved over the input. [2] The kernel is a filter that is used to extract features from the images. The calculations and the resulting feature is shown in Figure 2.1. The weighted sum of the values in the kernel is used to determine the output value for each patch of the input.

2. A non-linearity layer is used to adjust the generated output. This layer is applied in order to saturate the output or limit the generated output. It yields a reduced form without sacrificing crucial elements for accurate prediction, making the input easier to handle. [2]

3. A pooling layer is used to reduce the complexity for further layers

by down-sampling. In image analysis, pooling can be considered a resolution reduction. [2]

4. A fully-connected layer is a feed forward neural network. Each node of the fully-connected layer is connected to every node in the previous and in the next layer. [2]

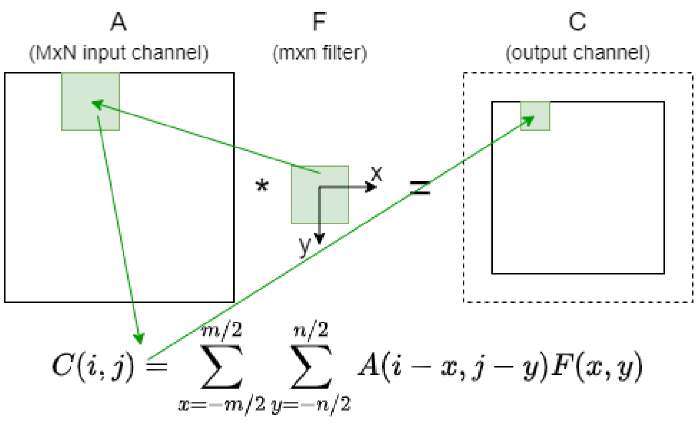
A
(MxN input channel)
F
(mxn filter)
C
(output channel)
*
x
y
=
$C(i,j) = \sum_{x=-m/2}^{m/2} \sum_{y=-n/2}^{n/2} A(i-x, j-y)F(x,y)$

Figure 2.1: Convolutional Layer Example

The first three layers form a convolutional block and a CNN usually contains several of these blocks. More such blocks increase the complexity of extracted features. The output of the last convolutional block is flattened or averaged, and connected to the output layer through fully connected layers.

2.3 Transfer learning

Transfer learning is the improvement of learning in a new task (called target) through the transfer of knowledge from a related task (called source) that has already been learned. [37] The purpose of transfer learning is to use information from the source task to increase learning in the target task. Transfer learning becomes necessary when there is a limited supply of target training data. [40]

Usually, the convolutional layers of an existing pre-trained model are used, such as AlexNet trained on a large dataset like Imagenet [8]. AlexNet is part of a deep CNN framework created by Krizhevsky, which debuted in the ImageNet LSVRC-2010 contest and got incredible results. [19] In medical analysis, more recent models such as Inception V3 [35], MobileNet V2 [32] and ResNet V2[13] are often used.[9] However, this study focuses on AlexNet, because of its usage by Gongbo Liang et al. [22] research, which we aim to reproduce and expand on.

When transfer learning, the weights of convolutional layers are typically frozen, preventing them from updating. Because AlexNet has 60 million parameters, retraining the entire structure takes a long time and effort. Furthermore, previous research showed that it is not necessary to fine-tune every parameter in AlexNet in order to detect pathological brain.[24] The next stage is to add a classifier with one or more fully connected layers, which is the part of the model that is trained.

2.4 Magnetic resonance imaging

Magnetic resonance imaging (MRI) of the brain produces a high-quality three-dimensional image for visualization and neuroanatomical classification of the brain structure. It is composed of 2D imaging slices. The voxels in MRIs correlate to physical locations in the brain. In Figure 2.2, the slices of an MRI from the Z dimension perspective are shown.

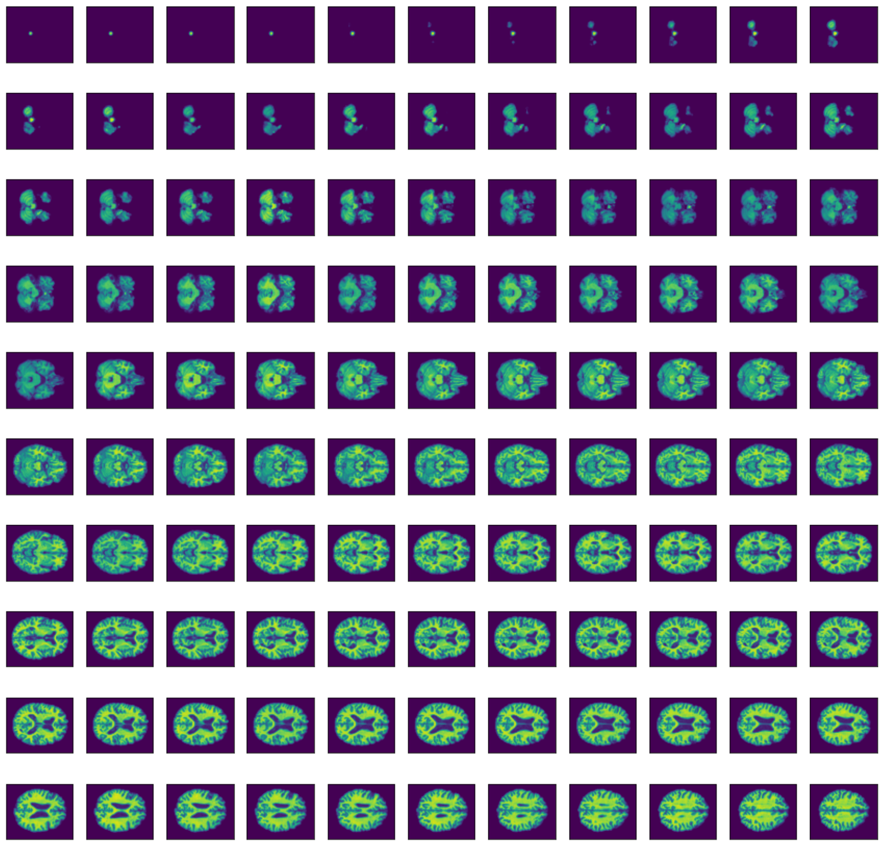

Figure 2.2: The individual slices of a 3-D MRI image

Chapter 3

Methodology

This chapter contains an overview of the methods that are used in this study. The code is accessible on GitLab[1].

3.1 Subjects

Data for this study was taken from Alzheimer's Disease Neuroimaging Initiative (ADNI) database[2]. The Alzheimer's Disease Neuroimaging Initiative (ADNI) is a long-term natural history study whose main goal is to inform the design of Alzheimer's disease therapy trials (AD). [7] ADNI was established in 2004 as a public-private collaboration under the direction of Dr. Michael W. Weiner.

A total of 223 images were downloaded from ADNI database for this research. After skull stripping, the resulting MRIs were visually verified and the ones that also had brain tissue removed were excluded. This resulted in a total of 198 images. The summary of the subjects is shown in Table 3.1. The subjects are divided in cognitive normal (CN) and Alzheimer's disease (AD) group. The age is represented as a mean ± standard deviation. Subjects are also divided in males (M) and females (F) by gender.

Group	Subjects	Age	Gender
CN	108	76.1 ± 5.2	52 M / 56 F
AD	90	76.6 ± 7.2	46 M / 44 F

Table 3.1: Subject data distribution summary.

[1]https://gitlab.science.ru.nl/pmoroza/bachelorthesis
[2]https://adni.loni.usc.edu/

3.2 Image pre-procesing

The MRIs taken from ADNI database have already undergone several pre-processing correction steps:

- T1 - all T1 images include an on-scanner non-uniformity correction.
- GradWarp - system-specific correction of image geometry distortion due to gradient non-linearity.
- B1 corrected - corrects the image intensity non-uniformity that results when radiofrequency transmission is performed with a more uniform body coil while reception is performed with a less uniform head coil.
- N3 - histogram peak sharpening algorithm. It is used to reduce residual intensity non-uniformity.

All scans were taken with 1.5T scanner.[3] An example of an original image is shown in Figure 3.1.

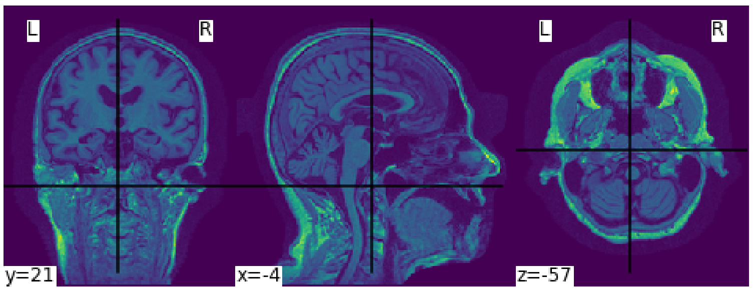

Figure 3.1: Example of an MRI before skullstripping

3.2.1 Skull stripping

A high resolution MRI contains some non-brain tissues such as skin, fat, muscle, neck, and eye balls. The presence of these tissues is considered a major obstacle for automatic brain image analysis. Skull-stripping is designed to eliminate non-brain tissues from an MRI.[26]

In this study structural MRI analysis software package FreeSurfer is used. It was developed by Laboratory for Computational Neuroimaging at the Athinoula A. Martinos Center for Biomedical Imaging for the analysis and visualization of structural and functional neuroimaging data from cross-sectional or longitudinal studies. [11] An example of a skull stripped image is shown in Figure 3.2.

[3]https://adni.loni.usc.edu/methods/mri-tool/mri-analysis/#mri-pre-processing-container

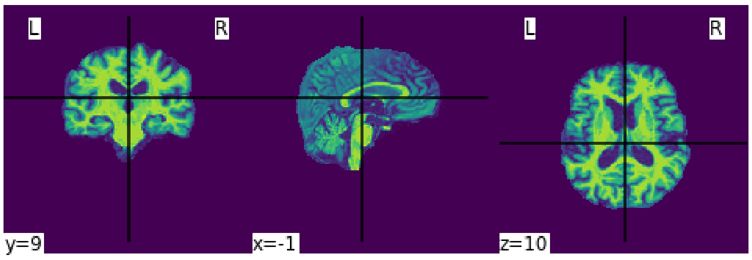

Figure 3.2: Skull-stripped image

3.2.2 Registering to a brain template

Registration to a brain template is the next step following skull-stripping. The most well-known and widely used template is provided by Montreal Neurological Institute (MNI).[38] By registering with MNI, all patients' brains are at the same place in the image box, and the network can learn the proper position of each image patch.

In this study registering a brain template to MNI is done using FSL FLIRT tool.[17] An example of skull-stripped and registered to a brain template image is shown in Figure 3.3.

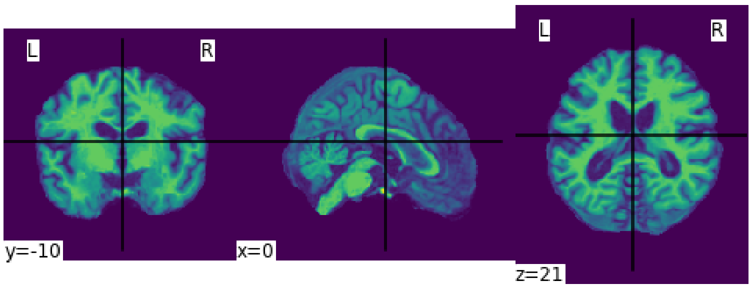

Figure 3.3: Skull-stripped and registered to a brain template image

3.2.3 Rescaling

The last pre-processing step is rescaling. The original images were of shape 182x218x182. Rescaling the images makes the data augmentation process easier. First, the images are rescaled with the scikit-image library by a factor of 0.504, which results in a shape of 92x110x92. Next, the images are padded and this gives a new image dimension of 110x110x110.

3.3 CNN Models

3.3.1 Weighted binary cross-entropy

Weighted binary cross-entropy is used to deal with imbalanced data set problem in neural networks. [4] It results into additional accuracy improvements compared to not modified loss function.[27]

3.3.2 Dropout

Dropout is a method for dealing with model over-fitting. For each training batch in every epoch, a dropout layer randomly omits each neuron with a given probability, so that a neuron cannot rely on the presence of other neurons. [14] In this study, the dropout probability is set to 50% and it is added after fully connected layers as it is known to perform well in prior research. [39, 42]

3.3.3 Feature extractor

All the convolutional layers of AlexNet are used as feature extractors in the models. [19] The approach is described in Section 2.3 .

3.3.4 Fusion strategies

The core idea of the fusion strategies is to down-sample the MRIs. It can be done in two ways:

Early Fusion

The feature extraction is done on a pre-pooled MRI. First, a 3D image is converted to 2D by applying one of the pooling strategies across one of the dimensions. After that, the features are extracted by giving the 2D image to the feature extractor.

Late Fusion

The feature extraction happens before applying the pooling operation. First, features are extracted from each slice individually and then the resulting feature maps are combined by applying a pooling operation.

The next section discusses the pooling methods and in Figures 3.5 and 3.6 application of max pooling with different fusion strategies is shown.

3.3.5 Pooling strategies

Pooling is a crucial step in further decreasing the dimensions of the activation map, preserving just the most relevant properties while diminishing spatial invariance. As a result, the number of learnable characteristics for the model is reduced. This aids in resolving the issue of overfitting. CNN uses pooling to absorb all of an image's dimensions, allowing it to recognize a particular item even if its form is warped or present at a different angle.[1]

A temporal pooling layer is placed either before the first fully connected layer (late fusion) or the feature extractor (early fusion) and it is used to convert 3D MRI's to 2D images by replacing temporal dimension values with a single value. Temporal pooling can be applied on the image-level and the feature-level.[22] In this study 3 different pooling strategies are used:

Max pooling

One of the most popular types of pooling is max pooling. [1] It extracts the greatest value from each sub matrix of the activation map and creates a new matrix. This guarantees that the number of learnable aspects remains limited while the important elements of any image are preserved. In this study, it is used to flatten one of the three dimensions of an MRI by extracting the maximum values. A simple example on a 2D matrix is shown in Figure 3.4.

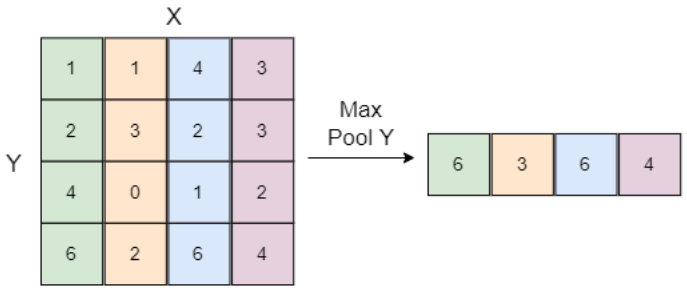

Figure 3.4: Max pooling on Y dimension.

In Figures 3.6 and 3.5 two different fusion strategies in combination with max pooling are shown. Early fusion selects the highest value of each pixel across all slices and passes it to the feature extractor. In the case of late fusion, all slices are fed into the feature extractor first, and then the largest values are extracted across all feature maps.

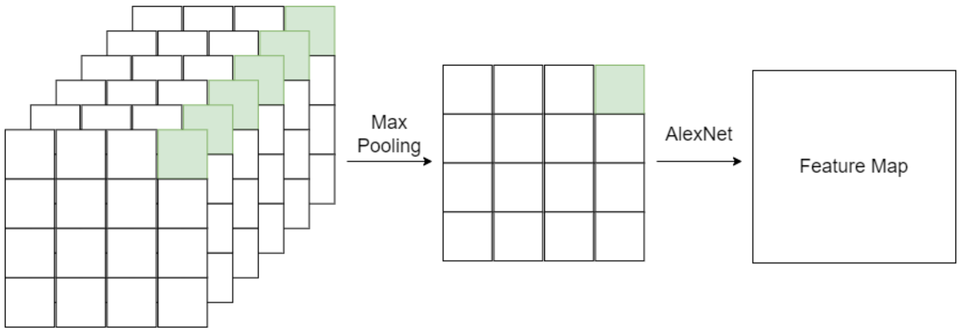

Figure 3.5: An example of Max Pooling Early Fusion. A 3D MRI is reduced to 2D by taking the maximum values across all slices. The result is then given to the feature extractor.

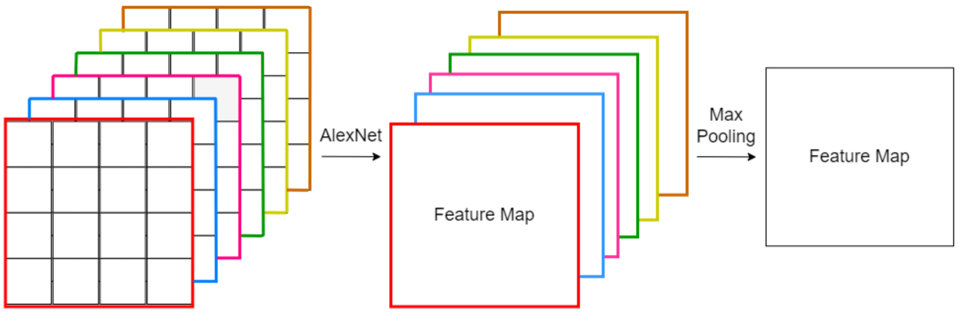

Figure 3.6: An example of Max Pooling Late Fusion. The slices are given to the feature extractor one by one. The resulting feature maps are then combined to one feature map by taking the maximum values across them.

Segmented max pooling

Segmented max pooling is inspired by a recent research on the classification of digital breast tomosynthesis (DBT), which is an emerging imaging technique for breast cancer screening. [36] These images also consist of slices and, Mickael Tardy and Diana Mateus [36] propose to summarise slices to slabs with an operation, for example, max pooling.

In this study, segmented max pooling works in a similar way as max pooling, but it takes a few slices at a time and summarises them into slabs by extracting the maximum values across them. The size T of the slabs is a hyper-parameter. For example, segmented max pooling on the Z dimension reduces an MRI $V \in R^{Y \times X \times Z}$ to $V' \in R^{Y \times X \times S}$, where $S = \lceil \frac{Z}{T} \rceil$. [36] This pooling method is only used with a late fusion strategy, because early fusion segmented max pooling is just early fusion max-pooling with an extra step (i.e. $T = Z$).

In Figure 3.7 segmented max pooling with T=2 is shown, where T is the amount of slices grouped together.

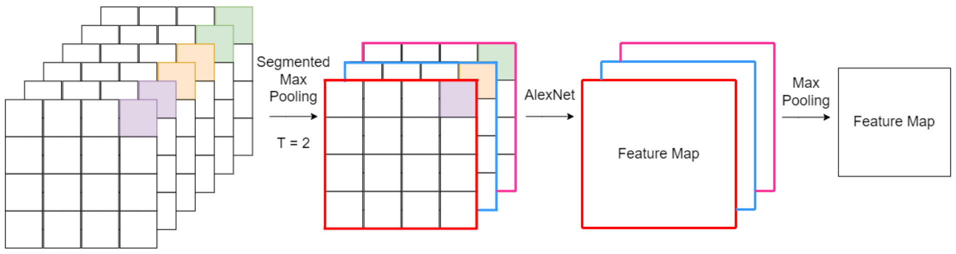

Figure 3.7: An example of Segmented Max Pooling Late Fusion. Every two slices are combined to a slab by applying max pooling. The three slabs are then given to AlexNet. The resulting feature maps are reduced to one feature map by applying max pooling.

Dynamic pooling

Dynamic image pooling is a temporal pooling method that was originally proposed for video clips summarization and is also referred to as rank pooling. [6] The paper by Basure Fernando et al. [10] proposes to capture the temporal ordering of a particular video by training a linear ranking machine on the frames of that video. The procedure entails arranging all of the video frames in chronological order based on their content. The parameters of the linear ranking function indicate the video's temporal appearance evolution in a systematic way.

The paper by Gongbo Liang et. al. [22] proposes that 3D MRIs can be treated like video clips, with each slice of the MRI acting as a video frame. After that, dynamic image pooling learns a dynamic image that can rank all of the MRI slices. This is done with rankSVM, one of the widely used methods for learning to rank.[20]

The MRI is represented as a sequence of vectors V, with each slice represented as a single vector. In this study, these vectors are obtained by applying max pooling on the individual slices. The paper by Basure Fernando et al. [10] applies a smoothing and non-linearity operation on V to account for the effect of noise, violent abrupt variation and the sometimes non-linear relation of these between frames. Since MRIs are not affected by speed of the action, these steps are skipped in this study. A rankSVM model[4] is then fit on the transpose of V and used to learn the dynamic image of the MRI. Formally, given an MRI as a sequence of slices, $V = [x_1, x_2, x_3, \ldots, x_n]$, with a shape of $w \times h \times n$, dynamic image μ is able to rank all the slices of the MRI, such that [22]:

$$\forall i, j \in [1..n] : i < j \iff \mu^T \cdot x_i < \mu^T \cdot x_j$$

In the case of late fusion, the feature maps of each slice are seen as the video frames instead. This study uses the implementation of Basure Fernando et al. for dynamic pooling[5].

Figure 3.8 shows an MRI after applying max pooling early fusion on the left and dynamic pooling early fusion on the right. The latter contains more details compared to the former.

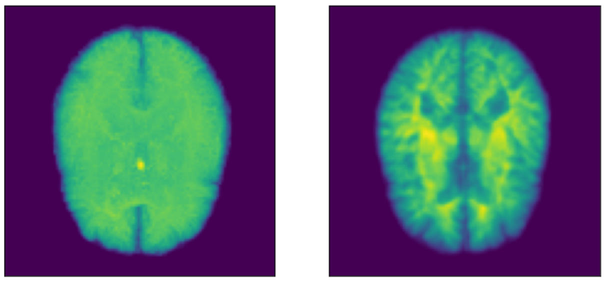

Figure 3.8: MRI after having undergone max pooling early fusion (on the left) and dynamic pooling early fusion (on the right).

3.3.6 Pooling dimension

In different sources authors give preference to different dimensions for the pooling direction. In this study all experiments are run for all dimensions X, Y and Z. The directions of all pooling dimensions is shown in Figure 3.9.

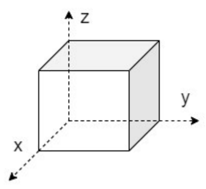

Figure 3.9: Pooling dimensions.

[4]https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVR.html
[5]https://github.com/MRzzm/rank-pooling-python/blob/master/rank_pooling.py

3.4 Data augmentation

CNNs are reliant on big data to avoid overfitting, which is a phenomenon when a network perfectly models the training data. [34] This often results in poor classification of unseen data. Data augmentation refers to a set of techniques that are used to enhance the size and quality of training data sets. [34]

In this study, random affine transformations that consist of a rotation, translation and scaling in all three dimension of the MRI are used to augment the training data. In order to implement these random affine transformations, the source code for data augmentation from a Master thesis "Computational diagnosis of Alzheimer's Disease" of Wieske de Swart[6].

The augmentation is done at the start of the training phase of each model. All MRIs in the training set are transformed and the results are then appended after which the new data is randomly shuffled. The data augmentation is exclusively done on the training data and increases its size by a factor of two. To not affect the class of the images, the random affine transformations are kept small at a maximum of 5% rotation, scaling or translation.

[6]https://github.com/Wieske/AD_thesis

Chapter 4

Experimental Setup

The models, data loading (with four CPU cores), training loop and testing loop are implemented with PyTorch, NumPy, scikit-learn (resizing and evaluation metrics) and nilearn (loading NIfTI format). All models are trained on a NVIDIA GeForce RTX 2080 Ti (11 GB) GPU using Python 3.9 and PyTorch 1.11.0.

4.1 PyTorch

PyTorch is a useful machine learning library for deep neural network development. It delivers a high-performance environment with simple access to automatic model differentiation across CPU and GPU.

4.2 NIfTI

NIfTI is a commonly used data format for neuroimaging data. Using NIfTI reduces the neccessary disk space by 70% due to used compression. [30]

4.3 CNN architecture

In this section the architecture of the model and the parameters are described.

4.3.1 Reproduced model

The code from the research of Gongbo Liang et. al. [22] was not ready at the time of writing this thesis, so the $\mathbf{Alex}_{Late-Max}$ model is recreated as precisely as possible based on what is written in the research article. Table 4.1 shows the detailed architecture of the reproduced model.

CCN layer	Layer type	Layer activations
110x110x110 input layer	Input	110x110x110
AlexNet convolutional layers (frozen)	Feature extractor	110x256x2x2
Temporal pooling layer (late fusion)	Pooling operation	256x2x2
256 1x1 convolutional layer	Convolutional	256x2x2
ReLU	ReLU	256x2x2
Flatten layer	Flatten	1x1x1024
512 fully connected layer	FC	1x1x512
2 fully connected layer	FC	1x1x2

Table 4.1: Generic reproduced model architecture. [22]

4.3.2 Early fusion strategy

Table 4.2 shows the detailed architecture of models with early fusion strategy.

CCN layer	Layer type	Layer activations
110x110x110 input layer	Input	110x110x110
Temporal pooling layer (early fusion)	Pooling operation	3x110x110
AlexNet convolutional layers (frozen)	Feature extractor	256x2x2
256 1x1 convolutional layer	Convolutional	256x2x2
ReLU	ReLU	256x2x2
Flatten layer	Flatten	1x1x1024
Dropout layer (p = 0.5)[39]	Dropout	1x1x1024
512 fully connected layer	FC	1x1x512
Dropout layer (p = 0.5)[39]	Dropout	1x1x512
1 fully connected layer	FC	1x1x1

Table 4.2: Generic early fusion model architecture.

4.3.3 Late fusion strategy

Table 4.3 shows the detailed architecture of models with late fusion strategy.

CCN layer	Layer type	Layer activations
110x110x110 input layer	Input	110x110x110
AlexNet convolutional layers (frozen)	Feature extractor	110x256x2x2
Temporal pooling layer (late fusion)	Pooling operation	256x2x2
256 1x1 convolutional layer	Convolutional	256x2x2
ReLU	ReLU	256x2x2
Flatten layer	Flatten	1x1x1024
Dropout layer (p = 0.5)[39]	Dropout	1x1x1024
512 fully connected layer	FC	1x1x512
Dropout layer (p = 0.5)[39]	Dropout	1x1x512
1 fully connected layer	FC	1x1x1

Table 4.3: Generic late fusion model architecture.

4.4 Parameters

4.4.1 Reproduced model

The reproduced model is trained with the settings shown below. The loss function is given slightly different weights for the output neurons, because the data set contains 18 more CN images than AD images. The normalization method is taken from another paper, because Gongbo Liang et al. [22] do not mention how their data is normalized.

- Pooled on Z dimension
- Loss function: weighted cross entropy, weights $[1-108/198, 1-90/198]$
- Learning rate: 0.0001
- Optimizer: Adam
- On-the-fly data augmentation of random horizontal flips and random rotations of 90, 180 or 270 degrees
- Min-Max normalization to [0,1] range [41]
- 100 epochs
- 4:1 data split
- Batch size 16
- Data is skull-stripped, registered to a brain template and resized to a shape of 110x110x110.

4.4.2 Custom model

The custom model is trained with the settings shown below. This model uses weighted binary-cross entropy, because it has only one output neuron. The loss function is given a positive weight, because the data set contains 18 more CN images than AD images.

- Pooled on all dimensions
- Loss function: weighted binary-cross entropy, positive weight 108/90
- Learning rate: 0.0001 [22]
- Optimizer: Adam, weight decay of 0.001 [41]
- Data augmentation mentioned in section 3.4
- Min-Max normalization to [0,1] range [41]
- 100 epochs and, for cross validation, early stopping with patience of 15 [41]
- 4:1 and cross-validation data split
- Batch size 16
- Data is skull-stripped, registered to a brain template and resized to a shape of 110x110x110.

4.5 Evaluation

Unbiased evaluation is essential to examine the potential clinical value of classification models. [41] Using test data in any part of the training process is one of the main sources of bias. [41] Therefore, any data set that is used to train a machine learning model must be divided into training data and independent, unseen testing data.

In this study, two different data splitting approaches are used and compared for the evaluation. No samples for the same patient can appear in both training and testing sets, because this would introduce bias. According to Junhao Wen et al. [41] bias from subjects appearing in both training and test data can lead to up to 8% increase of accuracy.

4.5.1 4:1 ratio

One of the most common data splits is the 4:1 ratio, which means that 80% of the data will be used to train the model and 20% will be used to test the model after it is trained on the training data. This is the evaluation approach used by Gongbo Liang et al. [22].

All splitting and shuffling is seeded both for reproducibility and to make sure that all models see the same splits. The models are trained for 100 epochs on the training data and then evaluated on the test set. For each model, this is repeated three times to get a better estimate of the average performance. The 4:1 accuracy is the average of the three accuracy measurements across the test set. The 4:1 auROC score is the average of the three auROC measurements across the test set.

4.5.2 Cross-validation

Another extensively used data resampling approach for estimating the true prediction error of models and tuning model parameters is cross-validation [5]. The data set is assumed to be a sample from a population of interest. In this paper, *stratified* k-fold cross-validation sub-sampling method is used to generate a training set and validation set with k equal to five. Stratified ensures that each fold reflects the proportion of AD and CN subjects of the entire training data. [5]

The available training set (80% of the original data) is divided into k disjoint subsets (folds) of approximately equal size. This division is done by selecting cases from the learning set at random and not replacing them. All random selection and/or shuffling is seeded for both reproducibility and to make sure that all models see the exact same splits.

The model is trained with k-1 subsets and the model is then evaluated on the remaining subset, the validation set, each epoch to monitor the training progress. This process is repeated until all k subsets have served as validation sets [41]. A test set (20% of the original data) is kept separate for the final evaluation of each model and this is also the same test set used in the 4:1 ratio evaluation.

The reported cross-validated accuracy in this study is the average of the k accuracy measurements across the test set. The reported cross-validated auROC score is the average of the k auROCs achieved on the test set. This cross validation process is repeated three times to get a more reliable estimate of the performance.

Early stopping

Early stopping is a technique which stops the learning process at an earlier point than the predefined number of epochs. Its goal is to figure out the epoch at which to stop training the model in order to prevent severe overfitting [41]. Furthermore, it is also a solution for reducing training time [15].

In this study, early stopping is used in combination with cross-validation. The validation loss is monitored and, when it does not improve for fifteen epochs, the training is interrupted. Otherwise, the model is trained for 100 epochs. When the training of a fold is finished, the model's weights are restored to those of the epoch with the highest auROC score. This approach is the same as Junhao Wen et al. [41] have for a similar 2D slice-based CNN model.

4.6 Statistical significance

4.6.1 McNemar's test

In this study the non-parametric McNemar's statistical test is used to compare the performance of the best trained classifiers with 4:1 ratio dataset split.[31] The first step to apply McNemar's test is to summarize the number of agreements and disagreements between the trained classifiers as shown in Table 4.4.

	Classifier2 Incorrect	Classifier2 Correct
Classifier1 Incorrect	No\No	No \Yes
Classifier1 Correct	Yes \No	Yes \Yes

Table 4.4: First step of McNemar statistical analysis.

The second step is to test whether two classifiers have the same error rate. The chi-square value is computed and tested against the theoretical chi-square with one degree of freedom with the following formula [18]:

$$\tilde{\chi}^2 = \frac{(|n_{No\backslash Yes} - n_{Yes\backslash No}| - 1)^2}{n_{No\backslash Yes} + n_{Yes\backslash No}}$$

4.6.2 Bonferroni correction

The Bonferroni correction is applied to P values when numerous dependent or independent statistical tests are run simultaneously on a single data set. It was created to address the problem that as the number of tests increases, the risk of concluding that a significant difference exists, when it does not, increases.[3]

In this study, Bonferroni correction is used to adjust the critical P value when comparing different classifiers. To perform a Bonferroni correction, divide the critical P value by the number of comparisons being made. For example, when we compare three models we get three comparisons (A with B, A with C and B with C), so the critical value becomes 0.05/3. When comparing five models we get ten comparisons, so the critical value becomes 0.05/10.

Chapter 5

Related Work

5.1 Max pooling late fusion and dynamic pooling

The research of Gongbo Liang et al. [22] uses two types of temporal pooling methods; max pooling and dynamic image pooling, two types of fusion strategies; early fusion and late fusion, and two types of feature extractors; AlexNet and ResNet-18.[12, 19, 28]. All models are trained for 100 epochs and the models are evaluated with the 4:1 data split earlier explained in this paper. The authors of [22] find that their proposed models are able to improve the classification performance by 10.11% in terms of auROC score compared to a conventional 3D CNN. In the upcoming Section (6), their results are discussed in more detail, as well as, our findings regarding our attempt to reproduce their models.

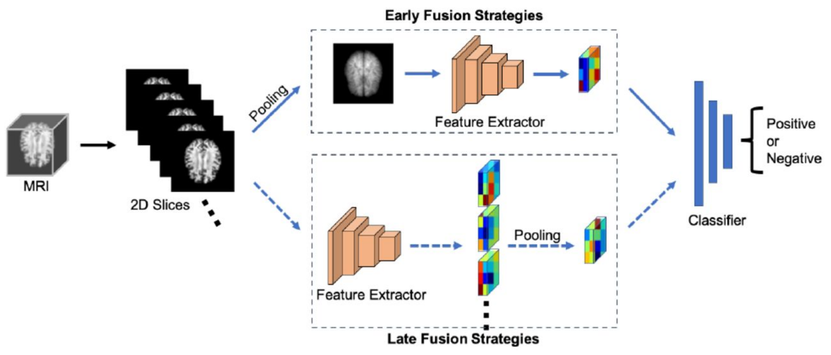

Figure 5.1: An illustration of different fusion strategies. Early fusion (Top) converts 3D images to 2D before feeding the data into a feature extractor. Late fusion (Bottom) converts 3D images to 2D after feeding the data into a feature extractor by Gongbo Liang et al.[22]

In Figure 5.1, an illustration is shown of the different fusion strategies proposed by Gongbo Liang et al. [22], which are coherent with the descriptions in Chapter 2. An illustration of dynamic image pooling proposed by Gongbo Liang et al. [22] is shown in Figure 5.2. It shows how the learned dynamic image μ can rank the sequence of slices, $V = [x_1, x_x, x_3, \ldots, x_n]$, of the original MRI, such that [22]:

$$\forall i, j \in [1..n] : i < j \iff \mu^T \cdot x_i < \mu^T \cdot x_j$$

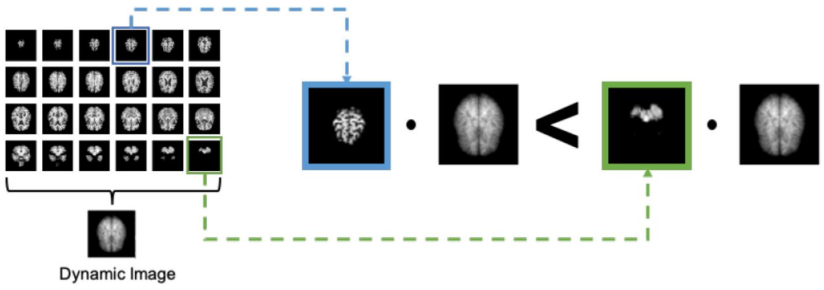

Figure 5.2: An illustration of dynamic image pooling. If the index of the blue slice (Slice$_B$) smaller than the index of the green slice (Slice$_G$), DynamicImage $\times$ Slice$_B$ $<$ DynamicImage $\times$ Slice$_G$ by Gongbo Liang et al.[22]

5.2 CNN comparison studies for AD classification

Junhao Wen et al. [41] conduct a large comparison study on CNNs for the classification of Alzheimer's disease. Their work includes a systemic literature review and rigorous CNN comparison. A strong motivation for their study is numerous recent proposals of machine learning approaches for the automation of Alzheimer's disease classification. However, those proposals are often difficult to reproduce due to unavailable frameworks or lack of implementation details.

Their study [41] compares five different CNN models and evaluates them with cross validation similar to this study. However, only one of their models is a 2D slice-level model. It uses transfer learning with all layers of ResNet pretrained on the ImageNet dataset with an additional fully connected layer at the end. The slices are taken from the Y dimension and repeated to RGB. All slices go through the entire model and a final prediction is made based on a majority voting system considering the predictions on all individual slices. Junhao Wen et al. [41] report an auROC of 76% $\pm$ 1.3% on their ADNI test set.

In this study, only 2D slice level models are compared and the fusion always happens within the model. All models are also tested with all possible pooling dimensions.

5.3 Segmented max pooling

Research on slice level models is not exclusive to Alzheimer's Disease classification. Mickael Tardy and Diana Mateus [36] propose a method for summarising slices for the classification of digital breast tomosynthesis (DBT). Instead of aggregating predictions on all slices [41], the authors of [36] first partition all slices into N groups and then summarise them to slabs with a slab-generation function b(·). Predictions are made slab-wise and all predictions are combined with a function a(·), e.g. a max operation. In their work [36], a trainable implementation of b(·) is proposed for slab generation.

In this study, inspiration from the slabbing proposal for DBT classification is taken for the Segmented Max Pooling model in Section 3.3.5. Instead of a trainable implementation for b(·), this research uses a max pooling operation to create slabs out of groups of slices. Each slab is then given to the feature extractor and the resulting feature maps are fused with max pooling.

Chapter 6

Results

This section will describe the comparison of the results. Detailed results of all models can be found in Section A.2 .

6.1 Reproducibility of previous research

In Table 6.1 the classification performance of all models from the research of Gongbo Liang et al. [22] is shown. The **Alex**$_{Late-Max}$ model, which uses AlexNet feature extractor, late fusion strategy, max-pooling and pooling dimension Z, achieves the best results.

Model	ACC	auROC	F1	Prec	Recall	AP
3D-ResNet	0.84	0.82	0.82	0.86	0.79	0.78
Alex$_{Early-Dyn}$	0.90	0.89	0.89	0.93	0.86	0.86
Alex$_{Late-Max}$	**0.91**	**0.91**	**0.91**	**0.97**	**0.85**	**0.90**
Alex$_{Late-Dyn}$	0.90	0.88	0.90	0.94	0.88	0.88
Res$_{Early-Dyn}$	0.83	0.81	0.82	0.85	0.80	0.78
Res$_{Late-Max}$	0.84	0.76	0.84	0.85	0.80	0.78
Res$_{Late-Dyn}$	0.88	0.86	0.86	0.91	0.85	0.84

Table 6.1: The evaluation result for all the compared models from research of Gongbo Liang et al. [22].

In this study, the Alex$_{Late-Max}$ model is reproduced as close as possible based on what is described in the research of Gongbo Liang et al. [22] The only difference from the mentioned paper is the dataset:

- Study of Gongbo Liang et. al. [22] uses 100 cases of spatially normalized, masked, and N3-corrected T1 MRI.

- This study uses 223 cases of spatially normalized, GradWarp-corrected, B1-corrected and N3-corrected T1 MRI.

Both datasets are taken from ADNI.

The results of the reproduced model are shown in Table 6.2.

Alex $_{Late-Max}$	ACC (test)	auROC (test)	ACC (train)	auROC (train)	Train Time (100 epochs)
Run 1	0.68	0.65	0.88	0.88	430 sec
Run 2	0.63	0.63	0.86	0.86	445 sec
Run 3	0.66	0.65	0.88	0.88	443 sec
Average	0.66 ± 0.03	0.64 ± 0.01	0.87 ± 0.01	0.87 ± 0.01	439 sec

Table 6.2: The evaluation result of recreated $\mathbf{Alex}_{Late-Max}$ model.

The best auROC score reported by Gongbo Liang et al. [22] is 91%, achieved with their AlexNet max pooling late fusion model. In this study, the auROC score for the reproduced AlexNet max pooling late fusion with a 4:1 data split is $66\% \pm 3\%$.

It is hard to tell what causes this difference. The code used by Gongbo Liang et al. [22] is not available at the time of writing, so this study might miss several important implementation details. The performance could also be different because of differences in the used data sets.

The custom AlexNet max pooling late fusion model achieves an auROC of $72.93\% \pm 1.52\%$ A.2 with 4:1 data split. The key differences are one output neuron instead of two, different data augmentation, two dropout layers and weight decay. This further indicates that implementation details can play an important role.

6.2 Influence of experimental setup

In this section, the best performing models for each data split are compared to answer the research question *"How do the evaluation results vary when using different data splitting strategies?"*.

The best performing models of each data split are shown in Table 6.3.

Model	Data Split	ACC	auROC	Time
Segmented max pooling	4:1	$74.56\% \pm 5.48\%$	$73.58\% \pm 4.72\%$	$555\ s \pm 10\ s$
Segmented max pooling	Cross-validation	$71.40\% \pm 4.36\%$	$72.13\% \pm 4.04$	$1101\ s \pm 57\ s$

Table 6.3: Best performing CNN custom models for each data split.

It is not possible to statistically compare the results of these two data splits. However, both splits find the segmented max pooling model to have the best performance in terms accuracy and auROC score. The performance seems to be similar, because both accuracy and auROC differ only between 1 to 3 percentage points. The total training time of 4:1 is approximately twice as fast, but cross validation does consist of five splits training five separate models.

6.3 Influence of pooling dimensions

In this section, the best performing models for each dimension for both data splits are compared to answer the research question *"How does the pooling dimension affect the performance of the temporal pooling methods?"*. The best models are chosen by considering all models for a dimension and then taking the one with the highest average auROC.

6.3.1 4:1 ratio

The best performing models of each pooling dimension for 4:1 data split are shown in Table 6.4.

Model	ACC	auROC
Segmented max pooling (T=2) X	$64.04\% \pm 4.02\%$	$65.26\% \pm 3.91\%$
Segmented max pooling (T=2) Y	$73.68\% \pm 2.64\%$	$71.69\% \pm 4.26\%$
Segmented max pooling (T=5) Z	$74.56\% \pm 5.48\%$	$73.58\% \pm 4.27\%$

Table 6.4: Best performing CNN custom models with data split 4:1 ratio for each pooling dimension.

McNemar's test is used to compare the proportion of errors between the models in table 6.4. The result is statistically significant if the p-value is below the bonferonni corrected critical value of 0.01667. This value is calculated by dividing 0.05 by 3, because three models need three comparisons. Table 6.5 shows the results of run one.

Run 1			
Dimension	X	Y	Z
X		0.71613	0.07544
Y	0.71613		0.90052
Z	0.07544	0.90052	

Table 6.5: Results of McNemar's test on run 1.

Table 6.6 shows the results of run two.

Run 2			
Dimension	X	Y	Z
X		0.21841	0.01640
Y	0.21841		0.90052
Z	0.01640	0.90052	

Table 6.6: Results of McNemar's test on run 2.

Table 6.7 shows the results of run three.

Run 3			
Dimension	X	Y	Z
X		0.71613	0.93359
Y	0.71613		0.75832
Z	0.93359	0.75832	

Table 6.7: Results of McNemar's test on run 3.

This study finds a significant result for the proportion of errors between the best model for pooling dimension X and pooling dimension Z in run 2. All other combinations of pooling dimensions in all other runs show no significant result.

6.3.2 Cross-validation

The best performing models of each pooling dimension for cross validation are shown in Table 6.8.

Model	ACC	auROC
Segmented max pooling (T=2) X	71.40% ± 4.36%	72.13% ± 4.04%
Segmented max pooling (T=2) Y	68.42% ± 3.16%	67.57% ± 4.11%
Dynamic pooling early fusion Z	70.70% ± 4.38%	71.39% ± 3.75%

Table 6.8: Best performing CNN custom models with data split cross-validation for each pooling dimension.

McNemar's test is used to compare the proportion of errors between the models in table 6.8. The result is statistically significant if the p-value is below the bonferonni corrected critical value of 0.01667. This value is calculated by dividing 0.05 by 3, because three models need three comparisons. Table 6.9 shows the results of run one.

Run 1			
Dimension	X	Y	Z
X		1.00000	0.80259
Y	1.00000		1.00000
Z	0.80259	1.00000	

Table 6.9: Results of McNemar's test on run 1.

Table 6.10 shows the results of run two.

Run 2			
Dimension	X	Y	Z
X		0.00137	0.86763
Y	0.00137		0.65672
Z	0.86763	0.65672	

Table 6.10: Results of McNemar's test on run 2.

Table 6.11 shows the results of run three.

Run 3			
Dimension	X	Y	Z
X		1.00000	0.86763
Y	1.00000		1.00000
Z	0.86763	1.00000	

Table 6.11: Results of McNemar's test on run 3.

This study finds a significant result for the proportion of errors between the best model for pooling dimension X and pooling dimension Y in run 2. All other combinations of pooling dimensions in all other runs show no significant result.

6.4 Comparison of temporal pooling methods

In this section, the best performing models for each pooling method for both data splits are compared to answer the research question *"How do the temporal pooling methods compare in terms of performance and training time?"*. The best model for each temporal pooling method is chosen by looking at all pooling dimensions of a method and taking the one with the highest average auROC.

6.4.1 4:1 ratio

The best performing models of each temporal pooling method for 4:1 data split are shown in Table 6.12.

Model	ACC	auROC	Dim	Time
Max pooling early fusion	56.14% ± 1.52%	53.33% ± 1.25%	X	560 s ± 15 s
Max pooling late fusion	71.93% ± 1.52%	72.17% ± 2.33%	Z	729 s ± 17 s
Segmented max pooling (T=5)	74.56% ± 5.48%	73.58% ± 4.72%	Z	555 s ± 10 s
Dynamic pooling early fusion	70.17% ± 3.04%	70.72% ± 3.26%	Z	1986 s ± 26 s
Dynamic pooling late fusion	69.3% ± 6.62%	68.07% ± 6.74%	Z	1594 s ± 25 s

Table 6.12: Best performing CNN custom models with data split 4:1 ratio for each temporal pooling method.

McNemar's test is used to compare the proportion of errors between the models in Table 6.12. The result is statistically significant if the p-value is below the bonferroni corrected value of 0.005. This value is calculated by dividing 0.05 by 10, because five models need ten comparisons. Table 6.13 shows the result of run one.

Run 1					
Model	MPEF	MPLF	SMPLF	DPEF	DPLF
MPEF		0.01640	0.00004	0.39973	0.21841
MPLF	0.01640		1.00000	0.93359	0.92034
SMPLF	0.00004	1.00000		0.71613	0.65672
DPEF	0.39973	0.93359	0.71613		0.94306
DPLF	0.21841	0.92034	0.65672	0.94306	

Table 6.13: Comparison of best performing CNN custom models with data split 4:1 ratio for run 1.

Table 6.14 shows the result of run two.

Run 2					
Model	MPEF	MPLF	SMPLF	DPEF	DPLF
MPEF		0.01242	< 0.00001	0.00562	0.52032
MPLF	0.01242		0.65672	1.00000	0.92034
SMPLF	< 0.00001	0.65672		0.92034	0.21841
DPEF	0.00562	1.00000	0.92034		0.65672
DPLF	0.52032	0.92034	0.21841	0.65672	

Table 6.14: Comparison of best performing CNN custom models with data split 4:1 ratio for run 2.

Table 6.15 shows the result of run three.

Run 3					
Model	MPEF	MPLF	SMPLF	DPEF	DPLF
MPEF		0.21841	0.36812	0.36812	0.65672
MPLF	0.21841		1.00000	1.00000	0.92034
SMPLF	0.36812	1.00000		0.90052	1.00000
DPEF	0.36812	1.00000	0.90052		1.00000
DPLF	0.65672	0.92034	1.00000	1.00000	

Table 6.15: Comparison of best performing CNN custom models with data split 4:1 ratio for run 3.

This study finds a significant result for the proportion of errors between the best model for max pooling early fusion and segmented max pooling late fusion in run 1 and run 2. All other combinations of temporal pooling methods in all other runs show no significant result.

In terms of accuracy and auROC, max pooling early fusion stands out by scoring 13 to 18 percentage points less. All other temporal pooling methods only differ 1 to 5 percentage points. In terms of training time, segmented max pooling trained the fastest on average of all models. Dynamic pooling early fusion and dynamic pooling late fusion train significantly slower than all other models.

6.4.2 Cross-validation

The best performing models of each temporal pooling method for cross validation data split are shown in Table 6.16.

Model	ACC	auROC	Dim	Time
Max pooling early fusion	53.69% ± 4.94%	53.16% ± 3.64%	Y	967 s ± 140 s
Max pooling late fusion	68.77% ± 2.92%	68.79% ± 3.42%	X	1118 s ± 135 s
Segmented max pooling (T=2)	71.40% ± 4.36%	72.13% ± 4.04%	X	1101 s ± 57 s
Dynamic pooling early fusion	70.70% ± 4.38%	71.39% ± 3.75%	Z	2676 s ± 277 s
Dynamic pooling late fusion	67.72% ± 1.97%	68.46% ± 2.23%	Z	1442 s ± 201 s

Table 6.16: Best performing CNN custom models with data split cross-validation for each temporal pooling method.

McNemar's test is used to compare the proportion of errors between the models in Table 6.16. The result is statistically significant if the p-value is below the bonferroni corrected value of 0.005. This value is calculated by dividing 0.05 by 10, because five models need ten comparisons. Table 6.17 shows the result of run one.

Run 1					
Model	MPEF	MPLF	SMPLF	DPEF	DPLF
MPEF		0.44612	0.00648	0.21130	0.07415
MPLF	0.44612		0.42371	1.00000	1.00000
SMPLF	0.00648	0.42371		0.80259	0.86763
DPEF	0.21130	1.00000	0.80259		0.90052
DPLF	0.07415	1.00000	0.86763	0.90052	

Table 6.17: Comparison of best performing CNN custom models with cross validation for run 1.

Table 6.18 shows the result of run two.

Run 2					
Model	MPEF	MPLF	SMPLF	DPEF	DPLF
MPEF		0.81398	0.01640	0.14580	0.52032
MPLF	0.81398		0.13361	0.92034	1.00000
SMPLF	0.01640	0.13361		0.86763	0.56771
DPEF	0.14580	0.92034	0.86763		1.00000
DPLF	0.52032	1.00000	0.56771	1.00000	

Table 6.18: Comparison of best performing CNN custom models with cross validation for run 2.

Table 6.19 shows the result of run three.

Run 3					
Model	MPEF	MPLF	SMPLF	DPEF	DPLF
MPEF		0.03722	0.11817	0.00047	0.78973
MPLF	0.03722		0.92034	0.93359	0.56771
SMPLF	0.11817	0.92034		0.86763	0.42371
DPEF	0.00047	0.93359	0.86763		0.14580
DPLF	0.78973	0.56771	0.42371	0.14580	

Table 6.19: Comparison of best performing CNN custom models with cross validation for run 3.

This study finds a significant result for the proportion of errors between the best model for max pooling early fusion and dynamic pooling early fusion in run 3. All other combinations of temporal pooling methods in all other runs show no significant result.

In terms of accuracy and auROC, max pooling early fusion stands out by scoring 15 to 20 percentage points less. All other temporal pooling methods only differ 1 to 4 percentage points. In terms of training time, max pooling early fusion trained the fastest on average of all models. Dynamic pooling early fusion and dynamic pooling late fusion train significantly slower than all other models.

Chapter 7

Discussion

First, this study takes a look at the reproducibility of the research by Gongbo Liang et al. [22] For this purpose, the best performing CNN model of that research is recreated as close as possible. It uses AlexNet feature extractor, late fusion strategy, max-pooling and pooling dimension Z. The results of the evaluation of this reproduced model (accuracy: 66% $\pm$ 3%, auROC: 64% $\pm$ 1%) shows worse results than results obtained by Gongbo Liang et al. (accuracy: 91%, auROC: 91%) [22]. This indicates that it is essential to provide all details of implementation and evaluation.

Secondly, this study compares the 4:1 data split used in CNN models to a cross validation technique. For this purpose, the best performing models of each splitting techniques are compared: AlexNet feature extractor, late fusion strategy, segmented max pooling, in case of 4:1, dimension Z, T=5, in case of cross-validation, dimension X, T=2. The results of the comparison show that both models have similar performance.

Thirdly, this study takes a look at the effect of the used pooling dimension in a model. For this purpose, this study performs McNemar's test between the best performing models for each pooling dimension with 4:1 data split. The results shows a statistically significant difference for pooling dimensions X and Z in run 1. McNemar's test between the best performing models for each pooling dimension with cross validation shows one statistically significant difference. Based on these results, it seems that overall the pooling dimension has a negligible influence for the classification of Alzheimer's disease. However, when using dynamic pooling early fusion it does seem to be critical to use the Z dimension. In this study, the X and Y dimension models for dynamic pooling early fusion score between 12 to 24 percentage points less in accuracy and auROC for both data splitting techniques (see Table A.9).

Finally, this study compares the performance of different temporal pooling methods. For this purpose, this study performs McNemar's test between the best performing models for each temporal pooling method with both 4:1 data split and cross validation. The former found two statistically significant differences. These differences are between the results of max pooling early fusion and segmented max pooling late fusion for run 1 and run 2. The latter found one statistically significant difference, comparison of max pooling early fusion and dynamic pooling and early fusion for run 3. Based on these results, it seems that there is a negligible difference in the proportion of errors of Alzheimer's disease classification between the different temporal pooling methods. However, there is some statistical evidence that max pooling early fusion can be expected to have a different proportion of errors.

When considering the accuracy and auROC scores, this study finds that all models have comparable performance, except for max pooling early fusion, which performs worse. The tables in the appendix also show that for segmented max pooling, larger T values (10,11,22,55) lead to reduced performance. Differences in efficiency can also be compared, because the training time is measured. This research finds that the dynamic pooling models take a lot longer to train than all other models. However, this does not lead to any significant improvements, so the reduced efficiency could be a reason to use a different temporal pooling method.

Chapter 8

Conclusions and future work

The main contribution of this study is the comparison and evaluation of different temporal pooling methods, fusion strategies, data splitting techniques and pooling dimensions for the classification of Alzheimer's disease.

The main limitation was not having the code that was used in the research of Gongbo Liang et al. [22] and also limited implementation details. This likely played a large role in the reproduced model scoring almost 30 percentage points less for both accuracy and auROC. In that context, this study shows that medical image analysis can be complex and difficult to reproduce, which means that it is crucial to comprehensively provide implementation details of the used machine learning models.

Furthermore, this study finds that it does not matter which pooling dimension is used when considering the best performing temporal pooling methods for each dimension. Only for dynamic pooling early fusion specifically, this research would recommend to use the Z dimension.

Moreover, this study shows that segmented max pooling (T=2, T=5) and all late fusion models have similar performance in terms of accuracy and auROC. Only one early fusion model reaches similar performance, which is dynamic pooling early fusion of the Z dimension, but all other dynamic pooling early fusion models and all max pooling early fusion models perform worse. Based on these results, it seems to be the case that it is better to use a late fusion strategy.

When considering efficiency, the dynamic image pooling models take a significantly longer time to train. For example, in the case of a 4:1 data split, dynamic image pooling early fusion takes almost four times longer to train than segmented max pooling (T=5). Based on these findings, it might be better to use either a max pooling late fusion model or segmented max pooling with a small value for T.

8.1 Limitations and future work

Since this study showed good performance for segmented max pooling, it might be interesting to further look into this temporal pooling technique for Alzheimer's disease classification. For both data splitting approaches segmented max pooling shows the best performance in terms of accuracy and auROC. In the case of 4:1 ratio, it also shows the best efficiency in terms of training time. However, a limitation of our investigation is the very small number of repetitions (3) of the experiments. This was due to the lack of computational resources and time constraints. More runs are necessary in order to provide stronger evidence of the above mentioned results.

Future work could also investigate the use of batch normalization. It is not used in this research, because the research of Gongbo Liang et al. [22] does not mention it. However, it is a recommended approach for reducing overfitting [41].

Furthermore, this study only used AlexNet as feature extractor. It could be interesting to try and compare newer pre-trained models for transfer learning, of which examples are briefly mentioned in section 2.3. Additionally, the classification performance might also benefit from a larger data set. We did not analyze the effect of data augmentation due to time constraints. An in-depth investigation of this technique could have provided insights, in particular, about the need of more data.

Finally, this study used the same values for the hyper-parameters of all models such as dropout, weight decay and learning rate. It could be interesting to investigate if the different temporal pooling methods benefit from more targeted tuning.

Bibliography

[1] Arohan Ajit, Koustav Acharya, and Abhishek Samanta. A review of convolutional neural networks. In *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, pages 1–5, 2020.

[2] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[3] Richard A Armstrong. When to use the bonferroni correction. *Ophthalmic Physiol. Opt.*, 34(5):502–508, September 2014.

[4] Yuri Aurelio, Gustavo De Almeida, Cristiano Castro, and Antônio Braga. Learning from imbalanced data sets with weighted cross-entropy function. *Neural Processing Letters*, 50, 10 2019.

[5] Daniel Berrar. *Cross-Validation.* 01 2018.

[6] Hakan Bilen, Basura Fernando, Efstratios Gavves, and Andrea Vedaldi. Action recognition with dynamic image networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2799–2813, 2018.

[7] Mayo Clinic and The Alzheimer's Disease Neuroimaging Initiative (ADNI). *Alzheimer's Disease Neuro Imaging MRI Overview*, 2018. `https://adni.loni.usc.edu/wp-content/themes/freshnews-dev-v2/documents/mri/ADNI_MRI_overview_2.6.18.pdf`.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[9] Oussama El Gannour, Soufiane Hamida, Bouchaib Cherradi, Mohammed Al-Sarem, Abdelhadi Raihani, Faisal Saeed, and Mohammed Hadwan. Concatenation of pre-trained convolutional neural networks

for enhanced covid-19 screening using transfer learning technique. *Electronics*, 11(1), 2022.

[10] Basura Fernando, Efstratios Gavves, Jose Oramas M., Amir Ghodrati, and Tinne Tuytelaars. Rank pooling for action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):773–787, apr 2017.

[11] Bruce Fischl. Freesurfer. *NeuroImage*, 62(2):774–781, 2012. 20 YEARS OF fMRI.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016.

[14] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012.

[15] Kazunori Imoto, Tomohiro Nakai, Tsukasa Ike, Kosuke Haruki, and Yoshiyuki Sato. A cnn-based transfer learning method for defect classification in semiconductor manufacturing. In *2018 International Symposium on Semiconductor Manufacturing (ISSM)*, pages 1–3, 2018.

[16] Holger Jahn. Memory loss in alzheimer's disease. *Dialogues in Clinical Neuroscience*, 15(4):445–454, 2013. PMID: 24459411.

[17] Mark Jenkinson, Peter Bannister, Michael Brady, and Stephen Smith. Improved optimization for the robust and accurate linear registration and motion correction of brain images. *NeuroImage*, 17(2):825–841, 2002.

[18] Jaber Juntu, Jan Sijbers, Steve De Backer, Jeny Rajan, and Dirk Van Dyck. Machine learning study of several classifiers trained with texture analysis features to differentiate benign from malignant soft-tissue tumors in t1-mri images. *Journal of Magnetic Resonance Imaging*, 31(3):680–689, 2010.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[20] Ching-Pei Lee and Chih-Jen Lin. Large-scale linear ranksvm. *Neural Computation*, 26(4):781–817, 2014.

[21] Qing Li, Weidong Cai, Xiaogang Wang, Yun Zhou, David Dagan Feng, and Mei Chen. Medical image classification with convolutional neural network. In *2014 13th International Conference on Control Automation Robotics Vision (ICARCV)*, pages 844–848, 2014.

[22] Gongbo Liang, Xin Xing, Liangliang Liu, Qi Ying, Ai-Ling Lin, and Nathan Jacobs. Alzheimer's disease classification using 2d convolutional neural networks. *medRxiv*, 2021.

[23] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017.

[24] Siyuan Lu, Zhihai Lu, and Yu-Dong Zhang. Pathological brain detection based on alexnet and transfer learning. *Journal of Computational Science*, 30:41–47, 2019.

[25] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015.

[26] Kalavathi Palanisamy and Surya Prasath. Methods on skull stripping of mri head scan images—a review. *Journal of Digital Imaging*, 29, 12 2015.

[27] Sankaran Panchapagesan, Ming Sun, Aparna Khare, Spyros Matsoukas, Arindam Mandal, Björn Hoffmeister, and Shiv Vitaladevuni. Multi-task learning and weighted cross-entropy for dnn-based keyword spotting. In *Interspeech*, volume 9, pages 760–764, 2016.

[28] Lionel Pigou, Aäron Oord, Sander Dieleman, Mieke Van Herreweghe, and J. Dambre. Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video. *arXiv Preprint*, 126, 04 2018.

[29] Anton P. Porsteinsson, Richard Isaacson, Sean A. Knox, Marwan N Sabbagh, and Ivana Rubino. Diagnosis of early alzheimer's disease: Clinical practice in 2021. *The Journal of Prevention of Alzheimer's Disease*, 8:371–386, 2021.

[30] Zalán Rajna, Anja Keskinarkaus, Vesa Kiviniemi, and Tapio Seppänen. Speeding up the file access of large compressed nifti neuroimaging data. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 654–657, 2015.

[31] Steven L. Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1:317–328, 2004.

[32] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018.

[33] Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annual Review of Biomedical Engineering*, 19(1):221–248, 2017. PMID: 28301734.

[34] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6:1–48, 2019.

[35] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[36] Mickael Tardy and Diana Mateus. Trainable summarization to improve breast tomosynthesis classification. In Marleen de Bruijne, Philippe C. Cattin, Stéphane Cotin, Nicolas Padoy, Stefanie Speidel, Yefeng Zheng, and Caroline Essert, editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2021*, pages 140–149, Cham, 2021. Springer International Publishing.

[37] Lisa Torrey and Jude Shavlik. *Transfer Learning.* Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques. IGI Global, 2010.

[38] N. Tzourio-Mazoyer, B. Landeau, D. Papathanassiou, F. Crivello, O. Etard, N. Delcroix, B. Mazoyer, and M. Joliot. Automated anatomical labeling of activations in spm using a macroscopic anatomical parcellation of the mni mri single-subject brain. *NeuroImage*, 15(1):273–289, 2002.

[39] Shui-Hua Wang, Chaosheng Tang, Junding Sun, Jingyuan Yang, Chenxi Huang, Preetha Phillips, and Yu-Dong Zhang. Multiple sclerosis identification by 14-layer convolutional neural network with batch normalization, dropout, and stochastic pooling. *Frontiers in Neuroscience*, 12, 2018.

[40] Karl Weiss, Taghi Khoshgoftaar, and Ding Wang. A survey of transfer learning. *J Big Data 3*, 9, 2016.

[41] Junhao Wen, Elina Thibeau-Sutre, Mauricio Diaz-Melo, Jorge Samper-González, Alexandre Routier, Simona Bottani, Didier Dormont, Stanley Durrleman, Ninon Burgos, and Olivier Colliot. Convolutional neural networks for classification of alzheimer's disease: Overview and reproducible evaluation. *Medical Image Analysis*, 63:101694, 2020.

[42] Haibing Wu and Xiaodong Gu. Max-pooling dropout for regularization of convolutional neural networks. In Sabri Arik, Tingwen Huang, Weng Kin Lai, and Qingshan Liu, editors, *Neural Information Processing*, pages 46–54, Cham, 2015. Springer International Publishing.

Appendix A

Appendix

A.1 ADNI

Data collection and sharing for this project was funded by the Alzheimer's Disease Neuroimaging Initiative (ADNI) (National Institutes of Health Grant U01 AG024904) and DOD ADNI (Department of Defense award number W81XWH-12-2-0012). ADNI is funded by the National Institute on Aging, the National Institute of Biomedical Imaging and Bioengineering, and through generous contributions from the following: AbbVie, Alzheimer's Association; Alzheimer's Drug Discovery Foundation; Araclon Biotech; BioClinica, Inc.; Biogen; Bristol-Myers Squibb Company; CereSpir, Inc.; Cogstate; Eisai Inc.; Elan Pharmaceuticals, Inc.; Eli Lilly and Company; EuroImmun; F. Hoffmann-La Roche Ltd and its affiliated company Genentech, Inc.; Fujirebio; GE Healthcare; IXICO Ltd.; Janssen Alzheimer Immunotherapy Research Development, LLC.; Johnson Johnson Pharmaceutical Research Development LLC.; Lumosity; Lundbeck; Merck Co., Inc.; Meso Scale Diagnostics, LLC.; NeuroRx Research; Neurotrack Technologies; Novartis Pharmaceuticals Corporation; Pfizer Inc.; Piramal Imaging; Servier; Takeda Pharmaceutical Company; and Transition Therapeutics. The Canadian Institutes of Health Research is providing funds to support ADNI clinical sites in Canada. Private sector contributions are facilitated by the Foundation for the National Institutes of Health (www.fnih.org). The grantee organization is the Northern California Institute for Research and Education, and the study is coordinated by the Alzheimer's Therapeutic Research Institute at the University of Southern California. ADNI data are disseminated by the Laboratory for Neuro Imaging at the University of Southern California.

A.2 CNN models evaluation

MPEF	4:1					
	X		Y		Z	
	ACC	auROC	ACC	auROC	ACC	auROC
Run 1	55.26%	52.61%	52.63%	53.91%	52.63%	53.91%
Run 2	55.26%	52.61%	47.37%	47.25%	52.63%	52.75%
Run 3	57.89%	54.78%	57.89%	58.26%	42.10%	49.85%
Average	56.14% ± 1.52%	53.33% ± 1.25%	52.63% ± 5.26%	53.14% ± 5.55%	49.12% ± 6.08%	52.17% ± 2.09%

	Cross-validation					
	X		Y		Z	
	ACC	auROC	ACC	auROC	ACC	auROC
Run 1	42.63% ± 3.87%	42.87% ± 3.23%	52.11% ± 5.86%	51.86% ± 3.73%	42.63% ± 6.09%	47.04% ± 4.64%
Run 2	34.21% ± 4.43%	37.54% ± 4.43%	56.84% ± 2.68%	55.30% ± 2.50%	51.58% ± 8.25%	53.28% ± 4.77%
Run 3	43.68% ± 10.07%	45.36% ± 6.50%	52.11% ± 5.62%	52.32% ± 4.43%	44.74% ± 7.98%	48.09% ± 4.70%
Average	40.17% ± 6.73%	41.92% ± 4.91%	53.69% ± 4.94%	53.16% ± 3.64%	46.32% ± 7.50%	49.47% ± 4.70%

Table A.1: The evaluation results for Max Pooling Early Fusion for test data.

MPLF	4:1					
	X		Y		Z	
	ACC	auROC	ACC	auROC	ACC	auROC
Run 1	63.16%	60.29%	65.79%	62.46%	73.68%	74.78%
Run 2	60.53%	55.80%	65.79%	67.10%	71.05%	70.29%
Run 3	68.42%	68.12%	65.79%	62.46%	71.05%	71.45%
Average	$64.04\% \pm 4.02\%$	$61.4\% \pm 6.24\%$	$65.79\% \pm 0\%$	$64.01\% \pm 2.68\%$	$71.93\% \pm 1.52\%$	$72.17\% \pm 2.33\%$
	Cross-validation					
	X		Y		Z	
	ACC	auROC	ACC	auROC	ACC	auROC
Run 1	$70.53\% \pm 3.07\%$	$70.32\% \pm 2.80\%$	$65.79\% \pm 3.33\%$	$63.62\% \pm 4.04\%$	$66.84\% \pm 4.59\%$	$66.35\% \pm 2.93\%$
Run 2	$65.26\% \pm 3.07\%$	$67.13\% \pm 4.18\%$	$64.74\% \pm 2.68\%$	$63.22\% \pm 4.35\%$	$66.84\% \pm 2.68\%$	$66.58\% \pm 2.96\%$
Run 3	$70.53\% \pm 2.58\%$	$68.93\% \pm 3.12\%$	$64.21\% \pm 2.68\%$	$63.48\% \pm 4.95\%$	$68.42\% \pm 4.08\%$	$67.42\% \pm 3.84\%$
Average	$68.77\% \pm 2.92\%$	$68.79\% \pm 3.42\%$	$64.91\% \pm 2.91\%$	$63.44\% \pm 4.46\%$	$67.37\% \pm 3.87\%$	$66.78\% \pm 3.27\%$

Table A.2: The evaluation results for Max Pooling Late Fusion for test data.

SMPLF	4:1					
T=2	X		Y		Z	
	ACC	auROC	ACC	auROC	ACC	auROC
Run 1	63.16%	61.45%	71.05%	66.81%	73.68%	73.62%
Run 2	60.53%	65.07%	73.68%	73.62%	71.05%	71.45%
Run 3	68.42%	69.27%	76.32%	74.64%	73.68%	74.78%
Average	64.04% ± 4.02%	65.26% ± 3.91%	73.68% ± 2.64%	71.69% ± 4.26%	72.8% ± 1.52%	73.28% ± 1.69%
	Cross-validation					
	X		Y		Z	
	ACC	auROC	ACC	auROC	ACC	auROC
Run 1	74.21% ± 3.07%	75.45% ± 2.92%	71.58% ± 2.58%	71.65% ± 3.27%	68.42% ± 2.88%	68.58% ± 2.01%
Run 2	72.63% ± 4.28%	72.99% ± 3.70%	65.79% ± 1.66%	65.01% ± 2.50%	64.21% ± 10.60%	65.57% ± 7.91%
Run 3	67.37% ± 5.42%	67.94% ± 5.18%	67.89% ± 4.53%	66.06% ± 5.80%	71.05% ± 4.70%	70.06% ± 4.12%
Average	71.40% ± 4.36%	72.13% ± 4.04%	68.42% ± 3.16%	67.57% ± 4.11%	67.89% ± 6.90%	68.07% ± 5.28%

Table A.3: The evaluation results for Segmented Max Pooling Late Fusion for test data with T=2.

SMPLF	4:1					
T=5	X		Y		Z	
	ACC	auROC	ACC	auROC	ACC	auROC
Run 1	55.26%	53.77%	60.53%	60.43%	76.32%	75.80%
Run 2	63.16%	61.45%	68.42%	66.96%	78.95%	76.81%
Run 3	63.16%	59.13%	71.05%	71.45%	68.42%	68.12%
Average	60.53% ± 4.56%	58.12% ± 3.94%	66.67% ± 5.47%	66.28% ± 5.54%	74.56% ± 5.48%	73.58% ± 4.72%
	Cross-validation					
	X		Y		Z	
	ACC	auROC	ACC	auROC	ACC	auROC
Run 1	70.00% ± 5.91%	69.88% ± 3.44%	67.37% ± 3.94%	67.71% ± 4.23%	73.68% ± 2.35%	74.32% ± 2.50%
Run 2	70.00% ± 3.57%	68.72% ± 4.95%	67.89% ± 1.97%	67.68% ± 2.61%	67.37% ± 7.55%	68.64% ± 4.96%
Run 3	63.68% ± 5.86%	63.97% ± 3.48%	63.16% ± 3.72%	63.30% ± 2.74%	68.42% ± 8.65%	68.58% ± 6.94%
Average	67.89% ± 5.23%	67.52% ± 4.02%	66.14% ± 3.33%	66.23% ± 3.28%	69.82% ± 6.77%	70.51% ± 5.13%

Table A.4: The evaluation results for Segmented Max Pooling Late Fusion for test data with T=5.

SMPLF	4:1					
T=10	X		Y		Z	
	ACC	auROC	ACC	auROC	ACC	auROC
Run 1	55.26%	53.77%	71.05%	71.45%	71.05%	73.77%
Run 2	57.89%	59.42%	63.16%	63.77%	65.79%	67.10%
Run 3	57.89%	58.26%	55.26%	54.92%	57.89%	58.26%
Average	57.01% ± 1.52%	57.15% ± 2.98%	63.16% ± 7.9%	63.38% ± 8.27%	64.91% ± 6.62%	66.38% ± 7.78%
	Cross-validation					
	X		Y		Z	
	ACC	auROC	ACC	auROC	ACC	auROC
Run 1	63.16% ± 5.77%	61.91% ± 6.70%	63.68% ± 4.21%	62.12% ± 3.21%	67.89% ± 6.09%	67.68% ± 3.79%
Run 2	65.26% ± 1.97%	61.10% ± 3.24%	67.89% ± 1.05%	67.22% ± 1.32%	66.32% ± 6.53%	66.14% ± 5.87%
Run 3	67.37% ± 5.42%	64.00% ± 3.42%	61.58% ± 4.28%	59.91% ± 4.34%	67.89% ± 7.14%	66.75% ± 7.14%
Average	65.26% ± 4.71%	62.34% ± 4.73%	64.38% ± 3.52%	63.08% ± 3.21%	67.37% ± 6.60%	66.86% ± 5.77%

Table A.5: The evaluation results for Segmented Max Pooling Late Fusion for test data with T=10.

SMPLF	4:1					
T=11	X		Y		Z	
	ACC	auROC	ACC	auROC	ACC	auROC
Run 1	50.00%	50.58%	60.53%	58.12%	65.79%	67.10%
Run 2	60.53%	58.12%	68.42%	65.80%	68.42%	69.27%
Run 3	57.89%	55.94%	68.42%	65.80%	65.79%	65.94%
Average	56.14% ± 5.48%	54.88% ± 3.88%	65.79% ± 4.56%	63.24% ± 4.43%	66.67% ± 1.52%	67.44% ± 1.69%
	Cross-validation					
	X		Y		Z	
	ACC	auROC	ACC	auROC	ACC	auROC
Run 1	59.47% ± 5.42%	61.88% ± 5.01%	61.05% ± 3.49%	60.64% ± 3.97%	61.58% ± 3.94%	61.77% ± 3.20%
Run 2	61.58% ± 3.94%	61.54% ± 3.82%	60.53% ± 1.66%	58.12% ± 1.91%	62.63% ± 4.53%	63.10% ± 5.29%
Run 3	61.58% ± 7.74%	63.62% ± 4.63%	62.11% ± 3.16%	61.97% ± 4.05%	61.58% ± 2.11%	59.45% ± 2.84%
Average	60.88% ± 5.91%	62.35% ± 4.51%	61.23% ± 2.88%	60.24% ± 3.46%	61.93% ± 3.67%	61.44% ± 3.93%

Table A.6: The evaluation results for Segmented Max Pooling Late Fusion for test data with T=11.

SMPLF	4:1					
T=22	X		Y		Z	
	ACC	auROC	ACC	auROC	ACC	auROC
Run 1	47.37%	48.41%	57.89%	52.46%	65.79%	64.78%
Run 2	63.16%	57.97%	65.79%	62.46%	76.31%	74.64%
Run 3	60.53%	54.64%	57.89%	52.46%	63.15%	56.81%
Average	$57.02\% \pm 8.46\%$	$53.67\% \pm 4.85\%$	$60.52\% \pm 4.56\%$	$55.79\% \pm 5.77\%$	$68.42\% \pm 6.96\%$	$65.41\% \pm 8.93\%$
	Cross-validation					
	X		Y		Z	
	ACC	auROC	ACC	auROC	ACC	auROC
Run 1	$55.79\% \pm 4.53\%$	$57.91\% \pm 5.46\%$	$60.00\% \pm 5.10\%$	$58.84\% \pm 3.57\%$	$57.89\% \pm 8.49\%$	$58.96\% \pm 7.07\%$
Run 2	$55.26\% \pm 4.71\%$	$57.25\% \pm 6.97\%$	$61.05\% \pm 4.53\%$	$61.80\% \pm 3.18\%$	$55.26\% \pm 3.33\%$	$54.00\% \pm 4.61\%$
Run 3	$61.58\% \pm 2.11\%$	$59.68\% \pm 2.98\%$	$63.68\% \pm 5.86\%$	$63.51\% \pm 5.68\%$	$50.53\% \pm 7.7\%$	$54.26\% \pm 5.24\%$
Average	$57.54\% \pm 3.96\%$	$58.28\% \pm 5.39\%$	$61.58\% \pm 5.19\%$	$61.38\% \pm 4.29\%$	$54.56\% \pm 6.89\%$	$55.74\% \pm 5.74\%$

Table A.7: The evaluation results for Segmented Max Pooling Late Fusion for test data with T=22.

SMPLF	4:1					
T=55	X		Y		Z	
	ACC	auROC	ACC	auROC	ACC	auROC
Run 1	55.26%	57.25%	57.89%	55.94%	60.53%	61.59%
Run 2	60.53%	58.12%	55.26%	56.09%	44.74%	52.03%
Run 3	52.63%	52.75%	52.63%	52.75%	55.26%	58.41%
Average	56.14% ± 4.02%	56.04% ± 2.88%	55.26% ± 2.63%	54.93% ± 1.89%	53.51% ± 8.04%	57.34% ± 4.87%
	Cross-validation					
	X		Y		Z	
	ACC	auROC	ACC	auROC	ACC	auROC
Run 1	56.32% ± 2.11%	56.49% ± 3.66%	58.42% ± 7.14%	57.54% ± 5.04%	50.00% ± 8.15%	50.58% ± 6.44%
Run 2	53.68% ± 7.91%	55.25% ± 3.90%	60.0% ± 5.86%	58.84% ± 5.31%	57.89% ± 7.98%	58.26% ± 4.66%
Run 3	55.26% ± 5.52%	53.77% ± 4.16%	57.89% ± 5.52%	56.17% ± 3.10%	51.58% ± 6.98%	54.90% ± 3.98%
Average	55.09% ± 5.70%	55.17% ± 3.91%	58.77% ± 6.21%	57.52% ± 4.59%	53.16% ± 7.72%	54.58% ± 5.13%

Table A.8: The evaluation results for Segmented Max Pooling Late Fusion for test data with T=55.

DPEF	4:1					
	X		Y		Z	
	ACC	auROC	ACC	auROC	ACC	auROC
Run 1	57.89%	57.1%	63.16%	64.93%	68.42%	72.75%
Run 2	44.74%	45.07%	55.26%	57.25%	73.68%	72.46%
Run 3	36.84%	40.87%	63.16%	62.61%	68.42%	66.96%
Average	46.49% ± 10.63%	47.68% ± 8.42%	60.53% ± 4.56%	61.59% ± 3.94%	70.17% ± 3.04%	70.72% ± 3.26%
	Cross-validation					
	X		Y		Z	
	ACC	auROC	ACC	auROC	ACC	auROC
Run 1	51.05% ± 8.09%	53.54% ± 5.76%	55.79% ± 3.07%	57.91% ± 5.39%	71.05% ± 5.52%	72.61% ± 4.10%
Run 2	51.58% ± 6.78%	53.51% ± 6.84%	57.89% ± 6.23%	61.04% ± 8.08%	67.89% ± 3.49%	68.84% ± 3.48%
Run 3	53.16% ± 6.09%	54.81% ± 5.52%	57.89% ± 1.66%	57.57% ± 3.27%	73.16% ± 3.87%	72.72% ± 3.65%
Average	51.93% ± 7.04%	53.95% ± 6.07%	57.19% ± 4.12%	58.84% ± 5.92%	70.70% ± 4.38%	71.39% ± 3.75%

Table A.9: The evaluation results for Dynamic Pooling Early Fusion.

DPLF	4:1					
	X		Y		Z	
	ACC	auROC	ACC	auROC	ACC	auROC
Run 1	63.16%	66.08%	68.42%	66.96%	76.32%	72.32%
Run 2	63.16%	62.61%	65.79%	67.1%	68.42%	71.59%
Run 3	65.79%	64.78%	65.79%	62.46%	63.16%	60.29%
Average	64.04% ± 1.52%	64.49% ± 1.75%	66.67% ± 1.52%	65.61% ± 2.64%	69.3% ± 6.62%	68.07% ± 6.74%
	Cross-validation					
	X		Y		Z	
	ACC	auROC	ACC	auROC	ACC	auROC
Run 1	66.32% ± 1.97%	65.68% ± 3.30%	66.32% ± 3.07%	64.75% ± 3.66%	70.00% ± 2.11%	69.88% ± 2.32%
Run 2	66.32% ± 5.37%	65.91% ± 6.27%	66.32% ± 3.49%	68.23% ± 3.10%	67.37% ± 2.11%	68.41% ± 2.86%
Run 3	65.79% ± 3.33%	65.94% ± 3.92%	65.79% ± 3.33%	65.71% ± 4.44%	65.79% ± 1.66%	67.10% ± 1.19%
Average	66.14% ± 3.82%	65.84% ± 4.68%	66.14% ± 3.30%	66.23% ± 3.77%	67.72% ± 1.97%	68.46% ± 2.23%

Table A.10: The evaluation results for Dynamic Pooling Late Fusion.