# Computationally Diagnosing Stages of Alzheimer's Disease

Simon Lee
Baskin School of Engineering
siaulee@ucsc.edu

**Abstract**

Alzheimer's disease is a common neurodegenerative brain disorder that slowly begins to deteriorate our cognitive abilities (thinking, memory, carrying out simple tasks). Alzheimer's disease has been known to have an unpredictable disease onset and is therefore hard to detect in its early stages. Therefore, there is currently a lot of interest in applying deep learning models to learn from MRI images for early diagnosis. Using publicly available data from Alzheimer's Disease Neuroimaging Initiative (ADNI), we train an optimized Convolutional Neural Network (CNN) model to predict whether a patient has Alzheimer's through the features in an image. With an accuracy of 86.67 %, we believe our optimized model can have an impact in the medical field and assist medical professionals and researchers in advancing pathology-related tasks. Building these prediction based models can improve diagnostic accuracy, optimize patient care, and reduce costs by bringing global collaborations.

**Keywords:** Alzheimer's Disease (AD), Convolutional Neural Networks, Computational Pathology, Deep Learning

# 1 Introduction

Alzheimer's Disease (AD) is a progressive neurodegenerative condition that leads to short-term memory loss, dementia, and many other cognitive-related issues. These effects are commonly mistaken for the effects of stress or aging and often go unnoticed. However by the time health professionals do detect Alzheimer's it is often too late. For a disease that affects every 1 in 9 (11.3%) people ages 65 and older in the United States, this has become a rather large scale issue. Traditional practices of diagnosing Alzheimer's including a mixture of cognitive tests and brain scans have been difficult markers to track this progression which leaves health professionals stuck. However in today's age of data, innovative approaches like machine learning have helped propel not only the field of pathology but the world. Therefore this research aims to build a image analysis computational framework that will allow for the early detection of AD.

## 1.1 Motivation

In order to solve this problem, we need to leverage innovative approaches such as machine learning, which are computationally intensive and non-traditional in the medical field. However, machine learning techniques have been increasingly used in disease prediction and visualization to offer prescient and customized prescriptions. In some cases, physicians and medical specialists are left with tough decision making and fear the extreme penalty of misdiagnosing a patient. So by taking these computational approaches we not only improve the patients' quality of life, but this also aids physicians in making decisions. With computers showing promise with its high level of precision, this is only the beginning for this partnership between computation and healthcare. The goal through this approach is to identify the knowledge gaps, avoid human error and explore potential opportunities associated with machine learning frameworks.

### 1.2 Deep Learning

Deep learning is a subsection of machine learning methods, based around neural networks which have the power to learn based on representation (Schulz et al. 2012). These models learn very similarly to humans through mimicking the way that neurons signal and communicate from one to the next. Deep learning is especially important in the field of image analysis because images contain a lot of features (shape, color histogram, color contrast, texture, etc.). When working with such high dimensional data, a simple machine learning model will struggle to interpret and return useful data for us to analyze. In contrast, deep learning models have the ability to process a large number of features which make it very powerful in dealing with image data. So through the deep learning algorithm called the Convolutional Neural Network, we intend to be able to accurately predict whether a patient has Alzheimer's disease from a set of images. In the preceding sections, we will lay out the foundation, and the underlying background knowledge required to understand how we obtained our results.

## 2 Models & Methods

For pedagogical purposes, we present in this section the data availability as well as the definitions and tools needed to understand the Convolutional Neural Networks (CNN) that we use for this prediction model.

### 2.1 Alzheimer's Disease Neuroimaging Initiative (ADNI) Data

Data used in this work were downloaded from the ADNI database run by the Laboratory of NeuroImaging at the University of Southern California. The ADNI is an ongoing, longitudinal study designed to develop clinical, imaging, genetic, and biochemical biomarkers for the early detection and tracking of AD through Magnetic Resonance Imaging (MRI) scans. We therefore use the complete ADNI1 1yr 3T dataset, which contains 58 AD MRI scans, 113 controlled (CN) MRI scans, and 133 Mild Cognitive Impairment (MCI) MRI scans for a total of 304 complete 3D brain images. The 3T MRI scan is twice as strong as a standard 1.5T MRI. It operates at a strength of 3T; Tesla (T) is the unit of measurement indicating the strength of a magnetic field. A stronger magnetic field provides more detailed images, helping doctors and radiologists see more structures inside the body (Duggan-Jahns et al. 2013). For this reason, we also use this magnetic strength to get the most clear images for our image analysis.

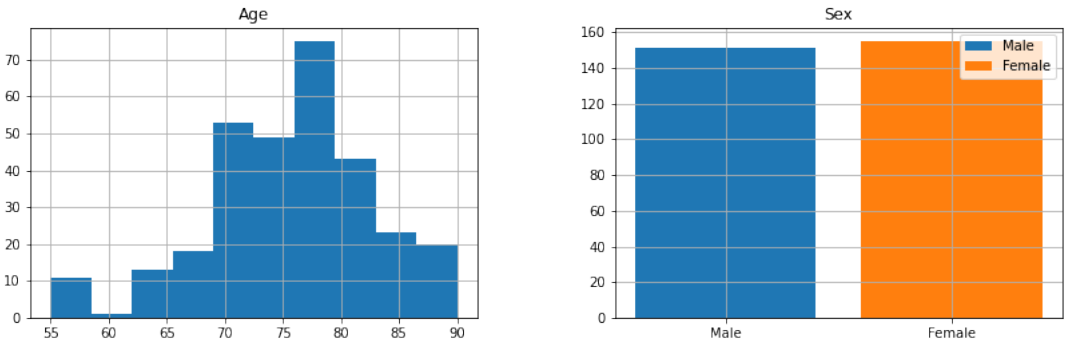


Figure 1: the demographic and the sex distribution of our samples

In addition to our sample count and our type of samples, we also want to briefly describe our demographic coming from our samples. In Sec. (1), we briefly described that 1 in 9 people over the age of 65 develop AD. Therefore we can see from Fig. (1) that the majority of our scans lie within the $< 65$ range. The mean age of our samples lies at 75.47, with a standard deviation of 7.06. On the right side of the figure we can also

see the distribution between the sexes. Although AD is not a genetic related disease (X or Y chromosome related disease), we thought it would be informative to get some insight into our patient samples.

## 2.2 Model Architecture

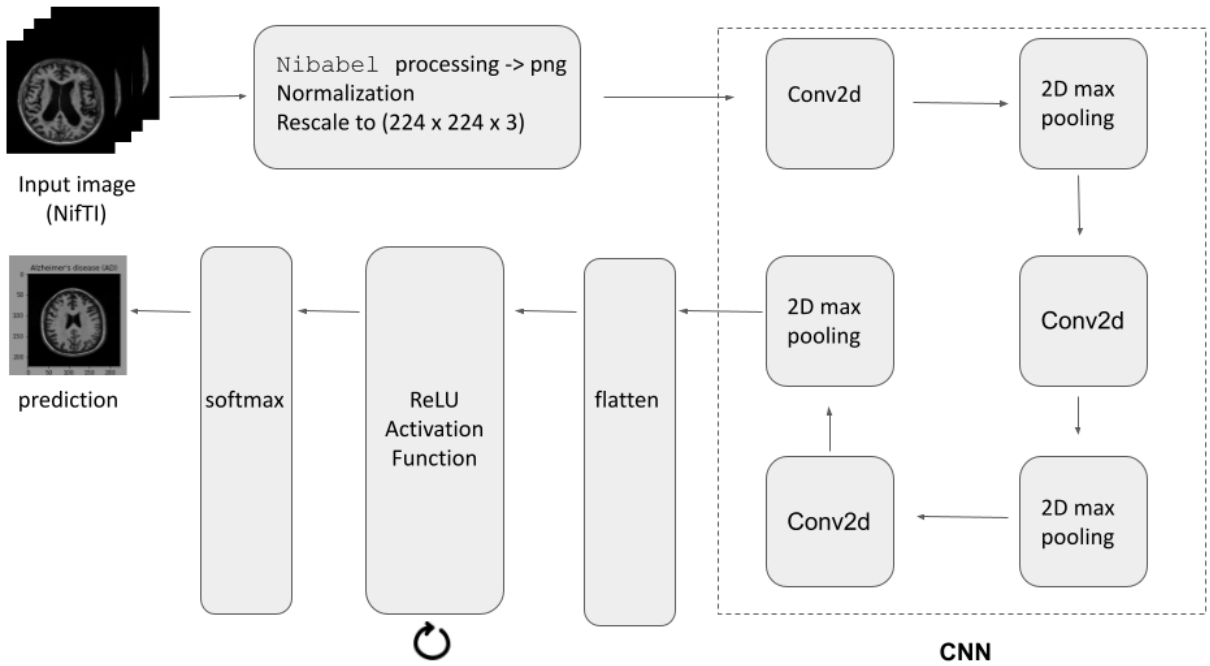


Figure 2: Model Architecture

Now we begin explaining our model that we will use to train for accurate prediction. As seen in Fig (2), we can see that a lot is going on before we reach our prediction stage. Therefore in the preceding subsubsections we will break down the processes that goes on within our optimized CNN model.

### 2.2.1 Preprocessing

Pre-processing is one of the most important steps in machine learning especially in the field of computational biology (Chicco et al. 2017). Data pre-processing is the manipulation and dropping of data to ensure/enhance the performance of the model. Often times if pre-processing is not involved it can lead to misleading results. Especially in medical images, pre-processing is an important step because we often get varying levels of quality, and it is important to run some cleanup on our data before feeding it into our model.

One of the first major steps during the pre-processing was to convert our MRI scans from the NifTI (.nii) format and into the png lossless image format. We therefore used a python script from the (OfStack et al. 2021) which performs this conversion for us. By performing this important step we greatly reduce our dataset from 11.94 Gb to 52.10 mb which is a 99.60 % compression rate. We are hoping that through this size reduction, our model can be optimally trained by not having to feed such a large amount of data during our training process.

In addition to our file format conversion, we also normalized our image data because some of the MRI scans come at varying qualities. Neural networks process inputs using small weight values, and inputs with large integer values can disrupt or slow down the learning process (Brownlee et al. 2019). As such it is good

practice to normalize the pixel values so that each pixel value has a value between 0 and 1. Additionally we also converted the shape of the image to 224 x 224 x 3 which will make it really easy to feed into our Convolutional Neural Network. Our data can be visualized as seen in Fig (3).

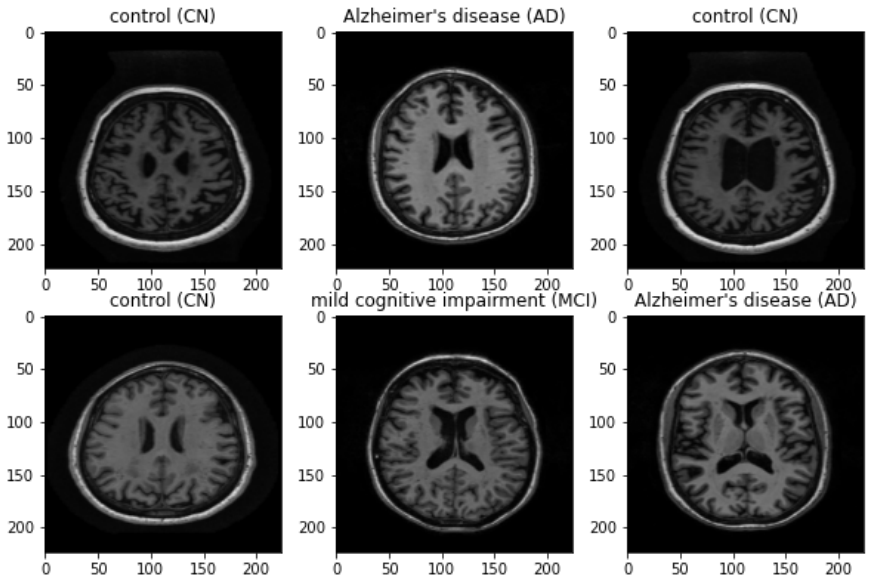


Figure 3: six brain samples labeled and visualized

Lastly as part of the preprocessing step we split our images into testing, validation, and training sets. We manually performed this split and came to the following breakdown: 1568 images for the training set, 391 images for the validation set, & 419 images for the testing set.

### 2.2.2 Convolutional Neural Network

CNN's are currently the standard to any image analysis problems and similar works have been done by the following groups (Oh et al. 2019, Folego et al. 2019). A CNN is deemed the perfect type of neural network for these problems because they are able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and the re-usability of weights. In other words, the network can be trained to understand the sophistication of the image better. The role of the CNN is to greatly reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

**I. Convolution 2D layers (Conv2d)**

The 2D convolution layer is a main building block of the CNN (Habibi et al. 2017). It performs a simple operation: we begin with a filter, which is simply a smaller matrix of weight's and begin sliding it over the 2d image matrix, and perform an element-wise multiplication on it. This element-wise multiplication then gets summed up and is stored into a single output pixel. A visualization of this process is shown in Fig (4). below, where an image containing 25 features, gets reduced down to an image with 9 features.

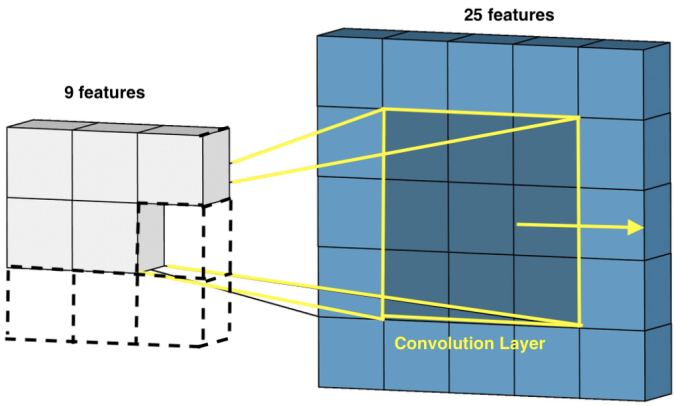


Figure 4: Convolution layer visualization

The filter repeats itself until every location of the 2d matrix is filled. From this convolution filter we converted a 2D matrix of features into a smaller reduced 2d matrix of features. The reduced feature matrix is determined based on the size and shift of the filter. For example in our Fig (4). we can see that our filter of size 3 x 3, and a shift of 1, produces an output of a 3 x 3 features matrix.

Let us also put into perspective why this convolution is powerful compared to a standard fully connected layer. In Fig (4)., we have a $5 \times 5 = 25$ input features, and $3 \times 3 = 9$ output features. If we were to use a standard fully connected layer, we would have a weight matrix of 25 x 9 = 225 parameters which is too large to consider. Especially with the intensive sizes of the image files, this would not perform optimally. Therefore, convolution allows us to do this transformation with only 9 parameters. The big idea of convolution is that we don't have to look through all the input features but rather get a general sense of an input feature that is coming from the same location. In hindsight, this process is a dimensionality reduction occurring.

**II. 2D Max pooling (2d max pooling)**

The other big component of the CNN is the 2d max pooling. A pooling layer is a new layer added after the convolutional layer. It is very common that this pooling happens after the convolutional layer because it helps order layers within a CNN that may be repeated multiple times in a model. It operates separately from our output feature matrix produced by the convolutional layer and creates a new set of the same number of pooled feature maps.

Pooling involves selecting a pooling operation (average pooling/ maximum pooling), much like a filter to be applied to our feature matrices. Typically when implementing CNN's the size of the pooling filter is typically a 2 x 2 matrix. What this does is reduce the size of features by at least a factor of 2. In the context of our problem, we have included three maximum 2d poolings that simply take the maximum value for each patch (pooling filter) of our features matrix. A visual can be seen as shown in Fig (5).

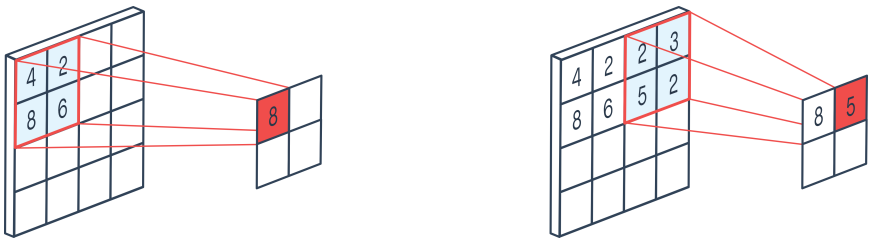


Figure 5: Maximum Pooling visualization

The main result we get from applying a pooling layer is that the newly produced pooled feature matrix summarizes the features detected within our input. Maximum pooling is done in part to help over-fitting by providing an abstracted form of our image. And similarly to the convolution layer, it reduces the computational cost by reducing the number of features to learn from as a form of dimensionality reduction. The outcome from pooling is what we call the model's invariance to local translation (Biscione et al. 2021). So the CNN's primary purpose after applying a series of convolutional layers and Maximum pooling layers it to get a widely reduced representation of our original image.

**III. Output Layer**

Lastly we wanted to discuss our output layer which consists of a flatten layer, seven Rectified Linear Unit (ReLU) activation functions, as well as a softmax before making an informed prediction. Flattening is the conversion of our data into a column vector, which will then be inputted into our ReLU activation functions. As part of the output layer, one of the most important aspects is to clean up our reduced data before making a prediction. Therefore the ReLU activation function has become notorious for this post-processing step. It's primary goal is to drown out the negative numbers to zero, while not changing the positive input. However in our case, we take the activation function of different size units meaning that everytime we call the ReLU, a new positive integer defining the dimensionality of the output space is produced. After we have iterated through the 7 different ReLU functions we finish it off with a Softmax function which is another type of activation function. At a high level it predicting a multinomial probability distribution, and what this means is that it calculates a probability between 0 to 1 for that feature. The softmax is used especially in these multi-class classification problems where class membership is required on more than two class labels. So in the context of our problem, these probabilities will allow us to classify our image into one of three labels: AD, MCI, CN. Our model is now complete to make informed decisions.

### 2.2.3 Training: The Adam Optimization function and Sparse Categorical Crossentropy Loss Function

Lastly in this section, we wanted to describe the optimization function & loss functions we chose to train our model. Machine learning models learn from gradient descent, which is a process by which we find the local minimums of a differentiable function. It is hard to think that images are differentiable functions so we have can view it more as a method in machine learning that helps us find the values of a function's parameters (coefficients) that minimize a loss function as far as possible. Therefore we used the Adam Optimization function (Kingma et al. 2014), which is a stochastic gradient descent algorithm which is performs exceptionally with images. The method is really efficient when working with large problem involving a lot of data or parameters. It requires less memory and is highly efficient. Intuitively, it is a combination of the 'gradient descent with momentum' algorithm and the 'RMSP' algorithm. A combination of these two algorithms allow for minimal training cost which is exactly what is seeked in an optimization algorithm. A performance graph was created in Fig (6). to describe the efficiency compared to other popular optimization algorithms.

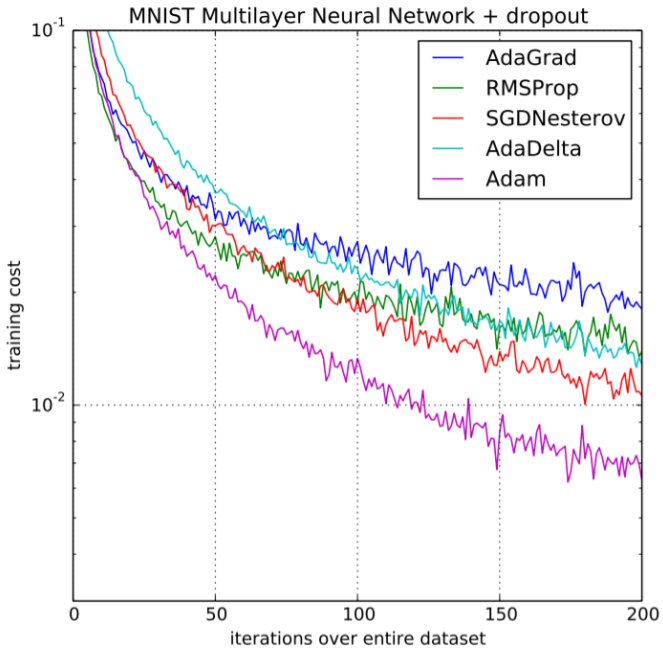


Figure 6: Optimization algorithm performance (Kingma et al. 2014)

With our optimization function being explained, we now guide our attention towards the loss function which we want to minimize. The Sparse Categorical Crossentropy Loss Function is the standard in image processing problems and the equation can be visualized by the following:

$$J(w) = -\frac{1}{N}\sum_{i=1}^{N}[y_i log(\hat{y}_i) + (1 - y_i) log(\hat{y}_i)] \tag{1}$$

Let us break down this Eq. (1). $w$ refer to the model parameters, e.g. weights of the neural network. $N$ is your number of parameters of your current model. $y_i$ is the true label, and $\hat{y}_i$ is your predicted label. Its main advantage is simply its computational cost again which helps us build our optimized CNN. The sparse categorical cross entropy saves time in memory as well as computation because it simply uses a single integer for a class, rather than a whole vector. With that we have everything we need to know in order for us to obtain our results.

### 2.3 Methods

All the content that was described in this section was translated into Python code, and our Jupyter Notebook is available via the Github on the title page. We trained our model using the training and validation sets and tracked their accuracy along with the loss that occurred at each epoch (iteration). The training itself was very computationally efficient and was able to execute our small data set of nearly 2000 images in a very reasonable time. Therefore we believe we have achieved an optimal model that generates successful predictions as we have proposed.

## 3 Results

In this section we analyze our results from our prediction model. Fig. (7), shows our accuracy and loss functions that were tracked in every epoch.

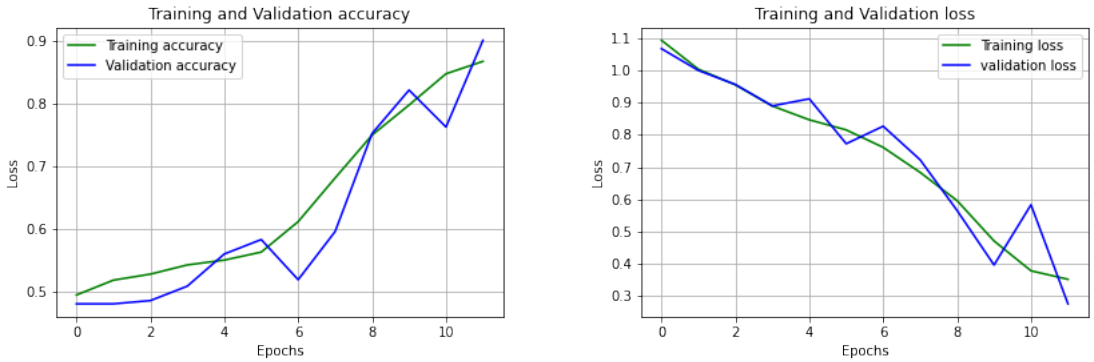


Figure 7: Model performance

Because the Adam optimization function is stochastic we can see that there is some fluctuation from epoch to epoch. However what I find more interesting is that the validation set seems to fluctuate more drastically than our training set, yet it has a higher level of accuracy in the prediction. We know this fluctuation behavior is due to the fact that there are less images that it is being trained on, but we are not totally sure why it is more accurate. However these graphs are a little misleading in terms of its accuracy. In our Jupyter notebook script, we have left our last cell to make these predictions. On the validation set, the model is able to predict all the six random images accurately, but when we run it through our unused testing set, it only usually identifies 4 out of the 6 images with the accurate diagnosis. So while our model claims to have a 86.67 % accuracy rate, we believe that this number is not necessarily true. Alzheimer's disease is very hard to predict after all and most machine learning models used in these type of studies have tended to have a little less than 50 % on accuracy (Marinescu et al. 2020).

We also wanted to show why model selection is very important. We trained a similar model using a different loss function and got drastically worse results from this loss function, optimization function pairing. We tried using the Kullback Leibler Divergence loss function and got the following results:

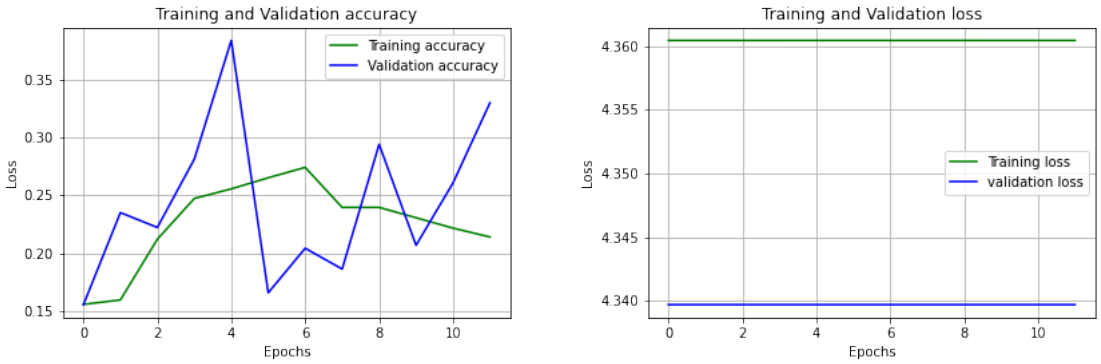


Figure 8: Model performance with the Kullback Leibler Divergence loss function

As seen in Fig. (8), the training and validations sets have somewhat of a steady incline before eventually declining at some point in the training process. Only around 1 out of 6 images were predicted accurately through every random set. We also identified that the loss function is constant throughout the training process and is therefore not really achieving anything in the stochastic gradient descent. Putting that into perspective, model selection was an emphasis in this work and we can see the impact of these two functions in our optimized CNN model.

# 4 Discussion & Future Works

In our discussion we wanted to highlight some caveats that we can hopefully continue as future works. One of the largest things we want to counter in these pathology problems is to avoid false negative. A case for a false negative diagnosis would be if a person with undetected Alzheimer's disease (AD) was predicted to be controlled (CN). This would obviously be detrimental as missing a AD diagnosis can lead to worsening effects as the time goes on. Even in our data set, patients were getting at most 3 MRI scans per year while the normal person gets one every 2-5 years. So while we cannot necessarily provide perfect predictions due to unclear disease onset, the least we could do when the model is unsure is to avoid a false negative diagnosis. In terms of doing that computationally, we currently have ideas of making another category in which images could classify as. Along with CN, MCI, and AD, we might try to design an architecture where NA could be some uncertain label for predictions. This way we can further examine these MRI scans using more traditional practices through medical professionals. We would also like to avoid false positives (if a CN patient is predicted to have AD), but it is not the worst thing to be cautious of these neurodegenerative diseases.

Another future works is the application of this model on other diseases. With this recent surge of data, one of the more accessible data modalities has been real patient data. With all types of researchers trying to understand these more complex diseases, patients have been more willing to give scientists this type of accessibility. Therefore we believe that we could use this same model, change up the labels that they could classify as and try running it on other diseases. Our hope is that once there is more understanding of what physical changes occur in some of these complex diseases, that we can highlight those features and use this software to assist medical facilities. Therefore there is a lot of excitement that lies ahead for the field of medical image analysis.

# 5 Conclusion

With that we have built our optimal CNN model that is operable to make well informed diagnosis predictions. By using various techniques that reduce the dimensions of an image significantly along with highlighting the general location of important features, our model was able to learn something that is not so easily seen with the naked eye. Looking at the big picture, we believe that our work is very practical and will have an impact on a global scale. With more uncertain events like global pandemics, and the lack of understanding of many complex diseases, we believe the works in image analysis are the future of the medical industry. Our big hope is to one day adapt our software to work alongside the doctors of the world.

# 6 Acknowledgement

This senior thesis was my last big obligation as an undergraduate student and I wanted to give a round of thank you's to some of the people that have supported me. I first wanted to thank Razvan Marinescu for helping me find this data set and pitched this idea of predicting whether Patients had Alzheimer's Disease or not. My collaborations with him all year have been very informative and I am excited to continue on in my education in this field of computational pathology. I also wanted to thank Professor Vanessa Jonsson, my research advisor for all the opportunities that came from this year. I met so many bright minded colleagues as a member of her lab, and I feel like I have grown and learned a tremendous amount in this year. Lastly I wanted to thank UC Santa Cruz for all the people I met, and the opportunities that came from coming to this school. I feel that this year I was able to advance my educational experience by not only conducting research, but also by being an educator as a tutor, and through working on some cool projects with some professors. I am very grateful to have gotten such an incredible education at UC Santa Cruz, and I can't wait for what is ahead at EPFL.

# References


[1] Schulz, Hannes; Behnke, Sven (1 November 2012). "Deep Learning". KI - Künstliche Intelligenz. 26 (4): 357–363. doi:10.1007/s13218-012-0198-z. ISSN 1610-1987. S2CID 220523562.

[2] Duggan-Jahns, Terry (24 June 2013). "The Evolution of Magnetic Resonance Imaging: 3T MRI in Clinical Applications". eRADIMAGING.com. eRADIMAGING.com.

[3] Chicco D (December 2017). "Ten quick tips for machine learning in computational biology". BioData Mining. 10 (35): 35. doi:10.1186/s13040-017-0155-3. PMC 5721660. PMID 29234465.

[4] OfStack (July 22, 2021) "python converts batch nii files into png images" https://ofstack.com/python/37028/python-converts-batch-nii-files-into-png-images.html

[5] Brownlee Jason (July 5, 2019) "How to Manually Scale Image Pixel Data for Deep Learning" https://machinelearningmastery.com/how-to-manually-scale-image-pixel-data-for-deep-learning/

[6] Oh, K., Chung, YC., Kim, K.W. (2019). "Classification and Visualization of Alzheimer's Disease using Volumetric Convolutional Neural Network and Transfer Learning . Sci Rep 9, 18150. https://doi.org/10.1038/s41598-019-54548-6

[7] Folego, Guilherme, Folego, G., Weiler, M., Casseb, R. F., Pires, R., Rocha, A. (30 Oct. 2020). "Alzheimer's Disease Detection Through Whole-Brain 3D-CNN MRI." Frontiers in bioengineering and biotechnology vol. 8 534592. doi:10.3389/fbioe.2020.534592

[8] Habibi, Aghdam, Hamed (2017-05-30). Guide to convolutional neural networks : a practical application to traffic-sign detection and classification. Heravi, Elnaz Jahani. Cham, Switzerland. ISBN 9783319575490. OCLC 987790957.

[9] Biscione, V. and Bowers, J (2021). Convolutional Neural Networks are not invariant to translation, but they can learn to be. https://openreview.net/forum?id=WUTkGqErZ9

[10] Kingma, Diederik P. and Ba, Jimmy (2014) Adam: A Method for Stochastic Optimization. https://arxiv.org/abs/1412.6980

[11] Razvan V. Marinescu, Neil P. Oxtoby, Alexandra L. Young, Esther E. Bron, Arthur W. Toga, Michael W. Weiner, Frederik Barkhof, Nick C. Fox, Stefan Klein, Daniel C. Alexander, (February 2020) TADPOLE Challenge: Results after 1 Year Follow-up. the EuroPOND Consortium. https://arxiv.org/pdf/2002.03419.pdf