Master's Thesis

Predicting Personalities through Multimodal Signals

Omar Kassem

Examiner: Prof. Dr. Hannah Bast &
Prof. Dr. Frank Hutter
Advisers: Prof. Dr. Hannah Bast

Albert-Ludwigs-University Freiburg
Faculty of Engineering
Department of Computer Science
Chair of Algorithms and Data Structures

September 02nd, 2019

Writing Period

02. 03. 2019 – 02. 09. 2019

Examiner

Prof. Dr. Hannah Bast

Second Examiner

Prof. Dr. Frank Hutter

Advisers

Prof. Dr. Hannah Bast

Declaration

I hereby declare, that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.
I hereby also declare, that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

_____ _____

Place, Date Signature

Acknowledgments

First and foremost, I would like to thank...

- Professor Bast for agreeing to be the supervisor of this thesis, even though it is an external project and giving me insightful guidance.
- Professor Hutter for agreeing to be an examiner.
- My family for their continuous support and encouragement.
- My friends and colleagues, Mostafa M. Mohamed and Mina Nessiem for the discussions and ideas that brought out this work and several proofreadings.
- MSc. Markus Näther for proofreading my thesis.
- SyncPilot GmbH for providing me with the opportunity to conduct this research.

Abstract

Automatic personality perception is a research area that has recently gained attention from the scientific community due to the wide range of fields that could benefit from personality predictions. The rise of social media platforms is providing access to huge amounts of labelled personality data. The dataset used for this research has been collected from the social media website Youtube. In this thesis, deep neural networks have been used to predict personality from one of three communication modalities: text, audio and images, or combinations thereof. The performance of the different trained models was used to compare the information embedded in each modality with respect to personality. This work's analysis points to visual information has the most personality content.

Zusammenfassung

Automatiserte Persönlichkeitserkennung (AP) ist ein Forschungsgebiet, das zunehmend in den Fokus der Forschungsgemeinschaft gerät. Das Spektrum an Gebieten, die von AP profitieren könnten, ist sehr gross. Der steigende Zuwachs von sozialen Netzwerken erlaubt den Zugriff auf große Mengen von persönlichkeitsannotierten Daten. Der Datensatz, der die Grundlage dieser Arbeit bildet,wurde von der Webseite des sozialen Netzwerks "Youtube" gesammelt. In dieser Arbeit wurden tiefe neuronale Netze angewendet, um Persönlichkeiten vorherzusagen. Dies erfolgte auf Basis von einem der drei Kommunikatsionsmodalitäten: Text, Audio und Bilder oder deren Kombinationen. Die Leistung von verschiedenen trainierten Modellen wurde angewendet, um die in den jeweiligen Modalitäten verankerten persönlichkeitsbezogenen Informationen zu vergleichen. Diese Arbeit zeigt, dass visuelle Informationen am meisten persönlichkeitsbezogene Informationen enthalten.

Contents

List of Figures

List of Tables

1 Introduction

Personality is a psychological model that identifies traits and characteristics that describe or explain a person's behaviors [2]. Successfully assessing personality could be beneficial in several contexts, for example:

- Diagnosing patients with psychiatric disorders [2].
- Social counselling or assisting hiring decisions of job applicants [2].
- Often people prefer to have conversations with people who have the same personality traits [3].

In other words, personality assessment might improve human to human interactions [3]. Additionally, algorithms that can assess peoples' personalities might also improve human-computer interactions [3]. For example, conversational chatbots could have customized responses for different personalities [3].

Automatically mapping information about some person to their personality is known as Automatic Personality Perception (APP) [4], and is the problem with which this thesis concerns itself. There are various types of data that could be utilized to solve this problem; however, in this work only the following data types were used, on the basis that these are the communication modalities that coexist the most together:

- Text written by the subject.

- Audio recordings of the subject.

- Spontaneous pictures of the subject.

The personality model used in this research is the Big Five (BF) [5] model, due to it's dominance and acceptance in the personality computing field [6]. The model defines personality as five traits:

- **Openness**: Shows the tendency of a person to be creative or curious to try new experiences. High openness is often correlated with a person being unpredictable, while with low openness are thought to be straight forward and having an easily predicted behaviour [3].

- **Conscientiousness**: Shows how organized and self-disciplined a person is. High conscientiousness values imply that a person is goal-oriented and handles their work with respect. On the other hand, people with low conscientiousness are often associated with being easy going and flexible [3].

- **Extraversion**: Shows how enjoyable is a person and active in other peoples' company. People with high extraversion are seen as social and friendly people, while people with low extraversion are often associated with being reluctant to express emotions [3].

- **Agreeableness**: Describes how likely is that a person is helpful or understanding versus being selfish or being cautious with other people. People who have low agreeableness are often associated with being competitive or challenging, while people with high agreeableness are often thought of as compliant or manageable [3].

- **Neuroticism**: Describes how easy it is for a person to have displeasing emotions. Some literature discuss neuroticism using it's negation, emotion stability [3].

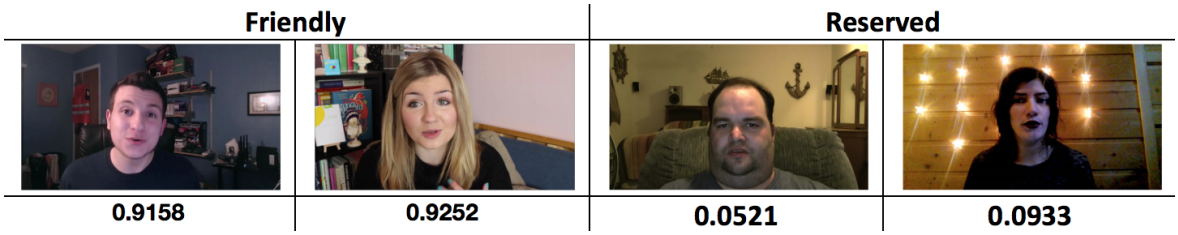
Figure 1 shows for each trait, photos of people who have high presence and absence of the trait to give an understanding of how each one of these traits could be interpreted. The BF model is also be referred to as OCEAN.

In this thesis, deep neural networks architectures that take as input each one of the modalities previously discussed will be introduced and implemented, as well as ensembles and multimodal neural networks. The goal of this thesis is to explore the capabilities of deep neural networks to apply them to these modalities and predict personality because of the advancments that deep learning achieved in other fields [8]. All of these models are trained and tested using the same dataset to have reliable results that can be used to understand the relevance and amount of information related to personality that is embedded in each modality.

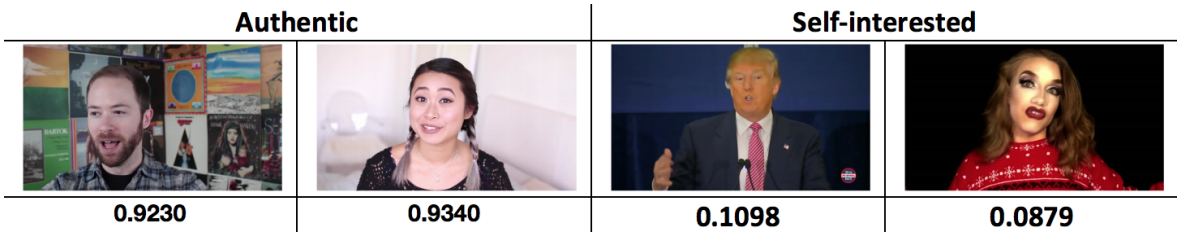
Chapter 2 goes over related works in the literature that have attempted to solve the APP problem using either one modality as input or several. The related work includes works that have used classical machine learning algorithms or deep learning algorithms. Chapter 3 is where the technical background of the algorithms and models used in this work is discussed. The models implemented in this thesis are explained in chapter 4. In chapter 5, the datasets and evaluation metrics are shown. The results of this thesis are presented in chapter 6, also a discussion of the results and interpretations of the used models are shown. This work concludes with chapter 7, wherein a summary of the methodologies and the results will be discussed.

(a) Openness

(b) Conscientiousness



(c) Extraversion



(d) Agreeableness

(e) Neuroticism

Figure 1: Figures showing examples of the extreme values for each personality trait. Reprinted from [7]

2 Related Work

Tal Yarkoni has shown in his work [9], that there exists a correlation between the choice of certain words and the personalities of people that chose them, Table 1 shows samples of words that correlate with each trait positively or negatively. Studies such as [9] that suggests there is a relation between words used by people and their personalities, gave motivation for using machine learning to predict peoples' personalities from text using these kinds of relations. Data from social media can be used to solve this problem [10]; due to the availability of huge amounts of people and text that they have posted on these kinds of platforms.

Mypersonality.org was an application that was used to collect data from Facebook users by pulling the users' posts and asking them to fill a survey that will give these posts the personality trait labels of their authors. However, in 2018 the owners of the application stopped maintaining it and the data is not available anymore. Few works have used the mypersonality dataset[1] to predict the OCEAN traits either using classical machine learning algorithms such as Support Vector Machines (SVMs), Bayesian Logistic Regression, and Naive Bayes [11] or deep learning methods [12]. The best accuracy score achieved by [11] was the naive bayes model with an accuracy of 61.70% and 70.78% was the best accuracy achieved by [12] using a multilayer perceptron (MLP). Another dataset was collected from the social media website Twitter, is the Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection (PAN) dataset [13]. The dataset was made public as part of a competition

[1] `https://sites.google.com/michalkosinski.com/mypersonality`Mypersonality.org

Trait	Correl.	Related words
O	High	Culture, films, folk, humans, literature, moon, narrative, novel, poet, poetry, sky.
	Low	Anniversary, detest, diaper, hate, hatred, hubby, implore, loves, prayers, thankful, thanks.
C	High	Achieved, adventure, challenging, determined, discipline, persistence, recovery, routine, snack, vegetables, visit.
	Low	Bang, bloody, boring, deny, drunk, fool, protest, soldier, stupid, swear, vain.
E	High	Bar, concert, crowd, dancing, drinking, friends, girls, grandfather, party, pool, restaurant.
	Low	Blankets, books, cats, computer, enough, interest, knitting, lazy, minor, pages, winter.
A	High	Afternoon, beautiful, feelings, gifts, hug, joy, spring, summer, together, walked, wonderful.
	Low	Asshole, bin, cost, drugs, excuse, harm, idiot, porn, sexual, stupid, violence.
N	High	Annoying, ashamed, awful, horrible, lazy, sick, stress, stressful, terrible, upset, worse.
	Low	Completed, county, ground, later, mountain, oldest, poem, road, southern, sunset, thirty.

Table 1: A list of words that have positive or negative correlations with the personality traits of their authors. Reprinted from [18].

where participants had to use machine learning algorithms that given a tweet would predict it's author age, gender, and personality. The participants with the best models used classical machine learning algorithms such as Latent Semantic Analysis [14], SVMs and Linear Discriminant Analysis [15] and Term Frequency-Inverse Document Frequency (TF-IDF) [16]. Farnadi et al. wrote a survey about research works related to the two former datasets and another one collected from Youtube [17].

Concerning the audio modality, the earliest work that the author knows of is by Polzehl et al. [19], where the authors collected audio data with the help of a professional voice actor who was asked to read a piece of text and then repeat it while imitating different personality traits, in other words for each trait a speaker would read the text in their normal voice and then read it twice, one as the high extreme value of the trait

and one as the low. Audio features like pitch, loudness and Mel Frequency Cepstral Coefficients (MFCC) were then extracted and fed to an SVM classifier to predict each trait of the speaker as positive or negative. In another work by Mohammadi et al, the SSPNet Speaker Personality Corpus was introduced [4]. The dataset was extracted from the French speaking Swiss national broadcast service and annotated by people who did not understand French. The goal behind the annotators being selected while lacking an understanding of French was so they would only consider the prosody, but not the semantics of the text when annotating the audio. The authors also extracted handcrafted audio features and fed them both to a logistic regression model and an SVM model with average classification accuracies over all OCEAN traits of 63% and 65%. Carbonneau et al. [20] have also used an SVM model to classify the SSPNet data, but the used features were found by transforming the audio wave into a spectrogram image and then applying dictionary learning to extract high-level features. The benefit of using such approach, is that the extraction of hand-crafted audio features that would require professional audio engineering knowledge, was replaced by dictionary learning that will automatically extract features related to solving the problem. The proposed model had an unweighted average recall of 67% with much less handcrafted features.

Furthermore, there exist few research works that predict a person's personality using extracted facial features from the person's images either using SVMs [21], neural networks [22], or various classic machine learning algorithms [23]. In Cucurull et al. [18], a deep learning model was used to infer if a personality trait has a positive or negative presence in an image. The image dataset was collected from the social media website Instagram and the images were chosen based on the textual tags that were mostly correlated with personality traits based on Yarkoni's work [9]. The model had an average accuracy over the five personality traits of 72%.

Concerning related works that used several modalities as input, there are few models that predict personality from audio-visual features using Support Vector Regression

for getting regression scores [24], SVMs for classification [25] and deep neural networks [26]. Some experiments of early and late fusion of modalities were done in [26].

Additionally, there is the First Impressions competition, where a video dataset was collected from the social media website Youtube and participants could use text, audio, and visual content to predict the personalities of the video subjects [7]. The deep learning approaches introduced by the top three participants of the competition have heavily influenced this thesis [27][28][29], due to their high regression accuracies (91.1%-91.3%). In the work of Kampman et al.[1], they used ensembles and multimodal neural networks to predict personalities on the First Impressions dataset, but the accuracies achieved were not that high, because the authors were more focused on investigating the effect of predicting personality from each modality not getting the best accuracies.

3 Background

In this chapter, the basic building blocks of the models that are used in this thesis will be explained. In section 3.1, an introduction to the pipeline used to solve the problem will be explained, next section 3.2 explains what are the artificial neural networks and how they work. Sections 3.3 and 3.4 will showcase convolutional neural networks and recurrent neural networks which are each used to handle certain types of problems. Finally, section 3.5 introduces embedding vectors which are the method used to represent text in neural networks.

3.1 Machine learning pipeline

Machine learning algorithms solve problems by estimating a function f, which could map an input value X to the desired output y [30]. The function f has the ability to extract information from data that are relevant to solving a given problem. There are three types of machine learning problems:

- **Supervised learning**: Given a list of pairs of inputs and their corresponding desired outputs (often referred to as labels), f should map X to y. This list of pairs is called a dataset [30].

- **Unsupervised learning**: Given a list of input features X only, an unsupervised learning algorithm will predict information based on the structure of X.

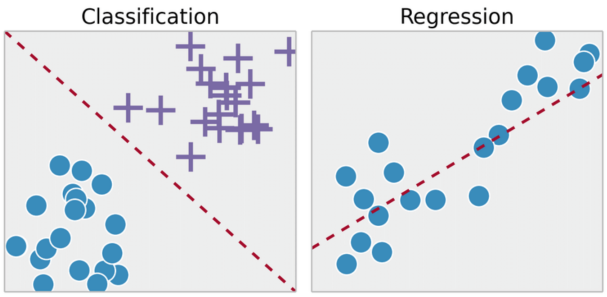


Figure 2: Two plots showing the difference between regression and classification problems. For classification problems, a model should find the best separator of the data, while in regression problems a model should find the best fit of the data. Reprinted from [32]

An example of unsupervised learning would be clustering, where the algorithm groups different subsets of the data based on their similarity [31].

- **Reinforcement learning**: Reinforcement learning models different kind of problems than supervised and unsupervised learning. Where there is an agent, a state space and actions that have consequences either good or bad. The goal of a reinforcement learning algorithm is to predict the best set of actions that will bring an agent the highest reward. A reinforcement algorithm learns using repetitive trial and error [31].

The problem being dealt with in this thesis is a supervised learning problem. The datasets used in this thesis are pairs of pieces of information in the form of modalities (text, audio clip, or image) as input and the OCEAN traits of the person this data is gathered from is the output. There are two types of supervised learning problems:

- **Regression**: Problems where the outputs are scalars. For example, predicting the price of a house based on it's specifications.

- **Classification**: Problems where the outputs have categorical values. For example, predicting the kind of animal in an image.

Figure 2 illustrates the difference between classification and regression models. The goal of classification models is to find a separator between data points belonging to different categories. The goal of regression models is to predict values with minimum distance from the output labels. For classification problems, if the data could be separated by a straight line, then it is a linearly separable problem. This thesis tackles the problem from a regression perspective, due to the author's interest in representing personalities on a scale as opposed to binary options.

To get the desired approximation function, a machine learning model is utilized which will provide the approximation function as an output. In literature, there exists a various number of machine learning models, such as SVMs [33], linear regression [30], decision trees [34], and random forests [35]. The main difference between them is how the model utilizes the inputs to map the outputs, for example, decision trees use a list of binary rules to classify the input [34], while SVMs fit hyperplanes to separate data that belong to different classes [33]. In this thesis, only one type of the machine learning model known as Neural Networks was utilized to solve the APP problem and in the next sections, the way it interacts with it's inputs will be discussed. The final essential component for the machine learning pipeline is the loss function, which is a function, that given the ground truth labels of the dataset and the approximated labels output of the model, would give a metric describing how far the approximation made by the model is from the desired value. The machine learning pipeline could be divided into the following steps:

1. **Dataset splitting**: The dataset is split into two parts: a training dataset and a testing dataset. The idea behind this technique is generalizing how good the model is in solving the problem when it gets as input data that it has not seen before. When splitting a dataset the data used for learning is called training data and test data is the data used for evaluating the model performance. One input-output pair from the training dataset is called a training example.

2. **Data pre-processing**: In some cases, the format of the dataset can not be directly used by the model and would need to undergo transformations before being used. For example, a machine learning model that only accepts numbers as input will transform the input text into a vector of numbers.

3. **Model initialization**: Initializing the parameters of the machine learning model used.

4. **Computing predictions**: Training data is fed to the model to get predictions. Then a loss value is computed using the predictions and the output labels. Based on the loss value the model parameters will be updated. Each model has a set of parameters that do not change during training, however they still affect how the model does it's predictions, these parameters are called hyperparameters.

5. **Model training**: What remains of the pipeline is a repetitive process of feeding data to the model and then training the model to decrease the loss, until the loss value is as small as possible.

6. **Hyperparameter tuning**: Based on the performance of a model after training, an inference about how could a hyperparameter of the model be changed to yield better results. So the pipeline could be repeated using the same model, but with different hyperparameters.

A good strategy to understand a model's performance is to evaluate the performance of the model's predictions using evaluation metrics on both training and testing datasets. If the model's metric scores are high when evaluating the training dataset, but low for the testing dataset, the means that the model is only learning to predict the training dataset and not the underlying problem represented in the dataset. This is referred to as overfitting [30]. Conversely, if a model accuracy is low in predicting both training and testing datasets, this means that the model is not complex enough to learn the relationship between the datasets' inputs and labels, this is referred to

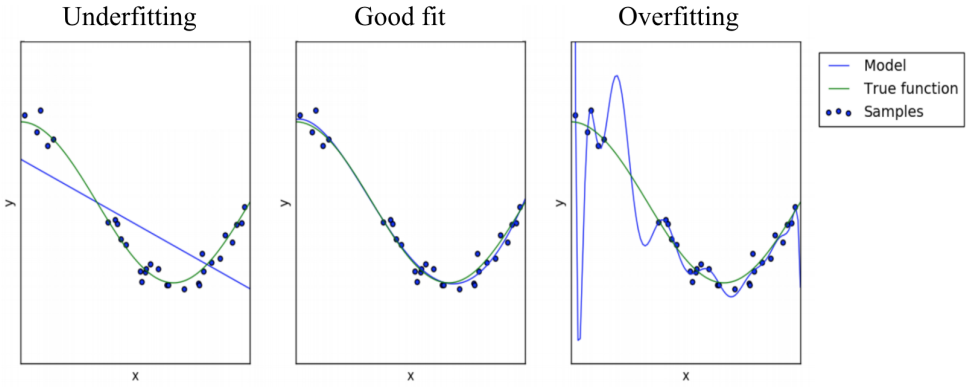


Figure 3: Plots illustrating the difference between the predictions of underfitted, overfitted and good fitted models. An underfitting model does not have good predictions of the data. An overfitting model has very accurate predictions, but the approximated function of the model does not generalize to the data distribution. A good fit model does both, has accurate prediction and generalize the function to predict the distibution of the data. Reprinted from [36]

as underfitting [30]. Figure 3 shows an example of the predictions of an underfitting, a good fit and overfitting models.

3.2 Neural networks

Neural networks are the class of machine learning models of interest in this thesis. The choice of the name is due to the ideas of building this model that were inspired by the biological neural networks [30]. The subsections 3.2.1 and 3.2.2 will show examples of how simple neural networks process input data and predict the desired output values. Subsection 3.2.3 will explain loss functions, while subsection 3.2.4 will show how do neural networks optimize their parameters. An overview of the activation functions is shown in subsection 3.2.5. An algorithm that is used to avoid overfitting of a neural network is explained in subsection 3.2.6 and examples of neural networks hyperparameters are illustrated in subsection 3.2.7.

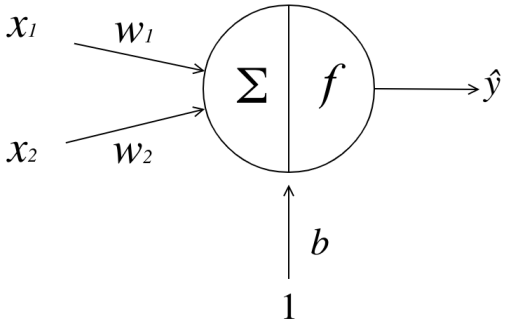


Figure 4: Single unit neural network

3.2.1 Single layer perceptron

This section will illustrate how neural networks work using an example of a single layer perceptron (SLP) [37]. Figure 4 shows a single unit neural network that has two input scalars (x_1 and x_2) and an output scalar $\hat{y}$. In neural networks methodology, a neuron (also could be referred to as a unit or a perceptron) is a representation of an approximation function that gets input values and applies a certain function to it and then passes it as an output. Each neuron has a set of parameters known as the bias value and a vector of weights that has a value corresponding to each input value connected to the neuron which are all initialized with random values. The neuron computes a linear combination of inputs and weights and adds the bias value to the output:

$$\hat{y} = f(b + w_1 \cdot x_1 + w_2 \cdot x_2)$$

The function f is a component in the neuron known as the activation function, which is a function applied to the multiplication and addition operations of the inputs and parameters of the neuron. There exist various activation functions that will be discussed in details in subsection 3.2.5

The predictions of a single neuron are based on a linear combination of it's inputs and weights. However, if this combination is not linearly separable in terms of the

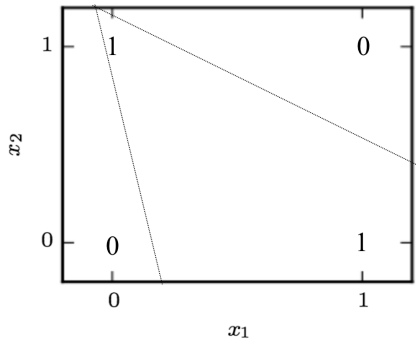


Figure 5: A plot showing the inputs x_1 and x_2 of an XOR function and the output values are written on the plot. The plot shows that the output values of the XOR function are not linearly separable in terms of x_1 and x_2. Reprinted from [30]

desired output, an SLP can not approximate a mapping between the input and output. Figure 5 shows a plot of the inputs (x_1, x_2) for the exclusive or (XOR) function and their corresponding outputs are on the plot. XOR is an example of a linearly non-separable problem that an SLP can not solve. To solve such problems the inputs need to be mapped to another space where a separator could be defined with just a line [30].

3.2.2 Multilayer perceptron

A multilayer perceptron (MLP) is a directed acyclic graph of neurons. In MLPs neurons are arranged vertically in layers and these layers are arranged horizontally, with each of the nodes in a layer connected to each of the nodes in the next layer [30]. The motivation behind the MLP structure is that each layer can transform it's inputs into a different space to solve more complex problems. For example, considering again the XOR example, an MLP layer could be used to transform the inputs $x1$, $x2$ into two other values $h1$, $h2$, such that the outputs of the XOR are linearly separable. Figure 6 shows a plot of the XOR function, in the transformed space.

The structure of an MLP is shown in Figure 7. The MLP in Figure 7 has four layers:

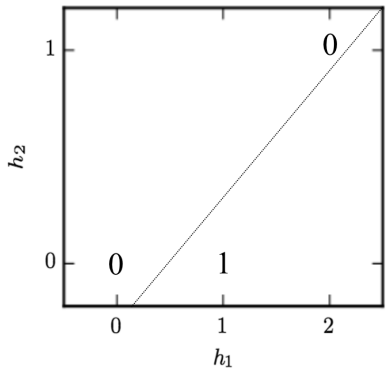


Figure 6: A plot of the XOR function in a transformed space h. This transformed space is the representation of the inputs of the XOR function using an MLP layer. The output of the XOR function is linearly separable in terms of h_1 and h_2, making the MLP able to approximate the XOR function. Reprinted from [30]

an input layer containing nodes that hold the values of the input values, an output layer where the predictions of the network are computed and two other layers; the layers between the input and output layers are called hidden layers. The depth of a network is determined by the number of layers. An MLP is also known as a fully-connected network because each node in every layer is connected to every node from it's previous layer.

The predictions $\hat{y}$ of an MLP are computed using: the vector X of the input values $\{x_1, x_2, x_3, x_4\}$ which is fetched from the dataset, the weight matrices and bias of the hidden layers and the output layer are $W_1, b_1, W_2, b_2, W_3, b_3$ and the activation functions of the hidden layers and the output layer f_1, f_2, f_3. The size of a weight matrix is equal to [the number of units in the previous layer, the number of units in the current layer], in this way the weight matrix of a layer has a weight value describing the relation between each node in it and each node from the previous layer. The output of the network can now be computed by computing the following equations:

$$z_1 = b_1 + X \cdot W_1 \ , \ h_1 = f_1(z_1)$$

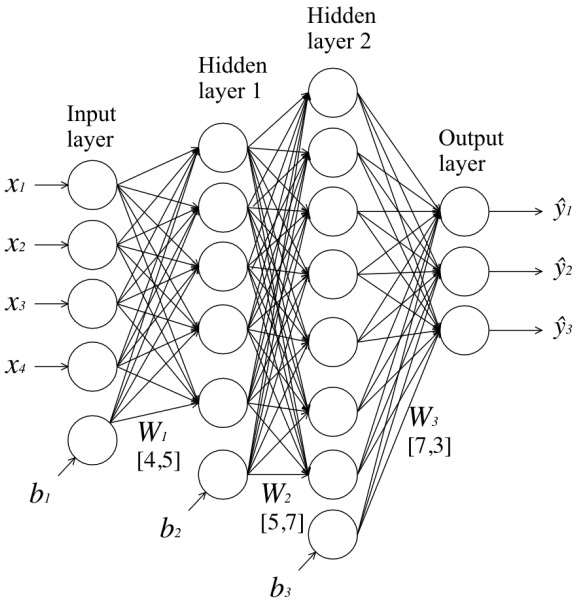


Figure 7: An example of a Multilayer perceptron with two hidden layers.

$$z_2 = b_2 + h_1 \cdot W_2 \ , \ h_2 = f_2(z_2)$$

$$z_3 = b_3 + h_2 \cdot W_0 \ , \ \hat{y} = f_3(z_3)$$

After the predictions $\hat{y}$ are calculated the loss value could be evaluated using the loss function l and the output label y:

$$\mathcal{L} = l(y, \hat{y})$$

After the loss value is evaluated for the whole dataset, this loss value is used to update the MLP's parameters by means of backpropagation, which will be discussed in details in subsection 3.2.4.

3.2.3 Loss functions

As briefly mentioned in section 3.1, loss functions are evaluation functions that output a value that describes how far the model predictions are from the desired outputs.

The choice of a neural network's loss function depends on the problem the network is trying to solve and the format of the output labels:

- **Regression**: Since both output labels and predictions are scalar values, a loss function can compute the difference between the prediction and the label, so the bigger the difference the worse the model is [30]. An example of a loss function for regression problems is the mean squared error (MSE) function. MSE is the loss function used to optimize the models described in chapter 4. MSE can be computed using the following equation:

$$MSE(y, \hat{y}) = (y - \hat{y})^2$$

- **Classification**: In classification problems, the predicted output value of the network is a probability value that an input feature vector should be classified under one category. For example, in a binary classification problem there are two classes, so the value of the output label y is either 0 or 1. A machine learning model will output a probability p of the input belonging to class 1. A loss function that could be used for classification problems is the binary cross entropy function [38], the output of the function depends on the value of the class label:

$$l(y, p) = \begin{cases} -\log(p), & y = 1 \\ -\log(1 - p), & y = 0 \end{cases}$$

  Figure 8 shows a plot for the cross entropy function for both classes. The plot shows that for each class the farther the prediction is from the label, the higher the loss values get.

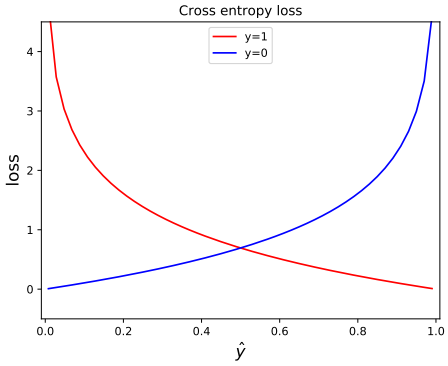


Figure 8: A plot of the binary cross entropy loss function for both cases of the output label y. The plot shows that for both classes of y, whenever the output probability p is far from the value of y the loss value is big and the closer p is to y, the loss value decreases.

3.2.4 Parameter optimization

Parameter optimization is an iterative process of updating the parameters of a neural network to improve it's predictions. When an input vector is fed to a neural network and information is propagated from the input layer to the output layer to generate predictions, this sequence of operations is called forward propagation. The loss value is then computed using the predictions $\hat{y}$ computed from the forward propagation and the labels y. Finally, the parameters are updated using the loss value.

Gradient descent

Gradient descent is an optimization algorithm used to solve problems that have no analytical solution using gradients [38]. Assuming f is a function that maps the variable x to the value y, to find the value of x that minimizes y, gradient descent will start by assigning x a random value and then update the value of x iteratively in the opposite direction of the gradient:

$$x = x - \frac{\partial f}{\partial x}$$

In that sense, x is taking iterative steps to the value that solves:

$$\frac{\partial f}{\partial x} = 0$$

Also, the size of the step is the magnitude of the gradient, so the closer the algorithm is to the goal the smaller the value. Using an iterative approach does not guarantee to find the global minimum, because the non-linearities lead to the presence of local minimal points [38]. There are several strategies used to terminate gradient descent:

- **Threshold**: The algorithm is terminated when the function is minimized beyond a desired threshold.

- **Convergence**: Sometimes the minimum desired threshold is not reachable by the algorithm. In that case, gradient descent terminates when the values converge, which implies that no more improvement could be done.

- **Early stopping**: When a neural network's evaluation metrics indicate high performance for the training dataset, but low for the testing dataset, that is a sign of overfitting. The early stopping technique stops the parameter optimization when detecting early signs of overfitting [39].

Types of gradient descent

When optimizing the parameters of a neural network, gradient descent can be used in several ways:

- **Batch gradient descent**: The entirety of the dataset is fed to the network in one batch and the average of losses from all of the training examples is used to update the network's parameters.

- **Stochastic gradient descent**: The parameters update is done based on the loss computed from one training example. Stochastic gradient descent (SGD) is more computationally expensive compared to batch gradient descent when using huge datasets because the parameter update operations are done for each training example in the dataset [30]. However, SGD could achieve better results because it does not approximate the loss value, but rather update the parameters using them all.

- **Mini-batch gradient descent**: The dataset is divided into equal batches of size m. Forward propagation is applied to each mini-batch and the average loss of the training examples in the mini-batch is computed and used to update the network's parameters. The size of a mini-batch m during training affects the computed loss value and the parameter update values, which is why m is a hyperparameter [30].

When gradient descent updates the parameters of a network using the computed loss values from the entire dataset, this is referred to as one epoch of training.

Backpropagation

The backpropagation algorithm is the method used by neural networks to compute gradients that will be used by gradient descent to update the network's parameters [40].

Since $\hat{y}$ is part of the loss function, the contribution of the model's prediction in the amount of loss can be found by calculating the partial derivative $\frac{\partial \mathcal{L}}{\partial \hat{y}}$ and since the prediction $\hat{y}$ is a function of the models' parameter, therefore the derivative of the loss function can also be expressed in terms of the model's parameters by computing $(\frac{\partial \mathcal{L}}{\partial W_0}, \frac{\partial \mathcal{L}}{\partial W_1}, \frac{\partial \mathcal{L}}{\partial W_2})$ using the derivative chain rule:

$$\frac{\partial \mathcal{L}}{\partial z_3} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_3}$$

$$\frac{\partial \mathcal{L}}{\partial W_3} = \frac{\partial \mathcal{L}}{\partial z_3} \cdot \frac{\partial z_3}{\partial W_3} \qquad \frac{\partial \mathcal{L}}{\partial b_3} = \frac{\partial \mathcal{L}}{\partial z_3} \cdot \frac{\partial z_3}{\partial b_3}$$

$$\frac{\partial \mathcal{L}}{\partial h_2} = \frac{\partial \mathcal{L}}{\partial z_3} \cdot \frac{\partial z_3}{\partial h_2} \qquad \frac{\partial \mathcal{L}}{\partial z_2} = \frac{\partial \mathcal{L}}{\partial h_2} \cdot \frac{\partial h_2}{\partial z_2}$$

$$\frac{\partial \mathcal{L}}{\partial W_2} = \frac{\partial \mathcal{L}}{\partial z_2} \cdot \frac{\partial z_2}{\partial W_2} \qquad \frac{\partial \mathcal{L}}{\partial b_2} = \frac{\partial \mathcal{L}}{\partial z_2} \cdot \frac{\partial z_2}{\partial b_2}$$

$$\frac{\partial \mathcal{L}}{\partial h_1} = \frac{\partial \mathcal{L}}{\partial z_2} \cdot \frac{\partial z_2}{\partial h_1} \qquad \frac{\partial \mathcal{L}}{\partial z_1} = \frac{\partial \mathcal{L}}{\partial h_1} \cdot \frac{\partial h_1}{\partial z_1}$$

$$\frac{\partial \mathcal{L}}{\partial W_1} = \frac{\partial \mathcal{L}}{\partial z_1} \cdot \frac{\partial z_1}{\partial W_1} \qquad \frac{\partial \mathcal{L}}{\partial b_1} = \frac{\partial \mathcal{L}}{\partial z_1} \cdot \frac{\partial z_1}{\partial b_1}$$

These partial derivatives will be used to update the model parameters by the means of gradient descent to minimize the loss:

$$W_0 = W_0 - \alpha \cdot \frac{\partial L}{\partial W_0}$$

$$W_1 = W_1 - \alpha \cdot \frac{\partial L}{\partial W_1}$$

$$W_2 = W_2 - \alpha \cdot \frac{\partial L}{\partial W_2}$$

The value α in the previous equations denotes the learning rate, a hyperparameter that could be used to scale down the magnitude of updating the parameter. The learning rate value is usually less than one. Otherwise, gradient descent could skip a minimum point by updating the parameters with a high magnitude. Another

hyperparameter is the learning rate decay, which is a value used to decrease α. The motivation behind the decay rate is that at the beginning of the training process the parameters need to take big update steps. Later on, the parameters are close to the values that minimize the loss and need small update steps. The following equation describes how the decay rate d updates the learning rate at every iteration i where α_0 is the initial learning rate:

$$\alpha_i = \alpha_0 \cdot \frac{1}{1 + d \cdot i}$$

The name backpropagation describes that the algorithm propagates partial derivatives back through the network's layers. The use of backpropagation enforces that the loss function and all the activation functions to be differential, otherwise the derivatives can not be calculated.

3.2.5 Activation functions

As mentioned in subsection 3.2.1, an activation function is function applied to the output of the neuron before it gets propagated to the next layer. The activation function used in the SLP example in subsection 3.2.1, is known as the linear activation function and is defined by the following equation:

$$f(x) = x$$

When using the linear function as activation in a neural network, not only does that limit the network to find a linear relation between the input features and label outputs, but it also limits the network from gaining more predictive ability by increasing it's depth. For example the MLP from subsection 3.2.2, if the activation functions f_1 and f_2 are linear activations, the network could be simplified to a single layer network

with one weight matrix $W^{'}$:

$$z_1 = X \cdot W_1$$

$$z_2 = X \cdot W_1 \cdot W_2$$

$$\hat{y} = X \cdot W_1 \cdot W_2 \cdot W_0$$

$$\hat{y} = X \cdot W^{'} \text{ where } W^{'} = W_1 \cdot W_2 \cdot W_0$$

It has also been proven that neural networks with one hidden layer with a finite number of units and a non-linear activation function are universal functional approximators [41], which means that a neural network can approximate any measurable function [42]. These facts gave motivation for using non-linear activation functions to improve a neural network's performance. There are three non-linear activation functions used in this thesis:

- **Sigmoid**: An S-shaped curve that has range of (0, 1). It is often used as the activation function of the output layer of classification problem, because the output of these networks can be thought of as a probability score. Figure 9a shows the plot of the sigmoid function. The sigmoid function and its derivative are defined by the following equations:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\frac{d\sigma}{dx} = \sigma(x) \cdot (1 - \sigma(x))$$

- **ReLU**: Rectified Linear Units. The function behaves like the linear function when the input value is positive, otherwise the output is zero. The ReLU plot is shown in Figure 9b. Known for solving the vanishing gradient problem [43]. The ReLU function is not differential when the input is zero, however neural networks implementation assume that it is one [30]. The ReLU function is

defined using the following equations:

$$ReLU(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

$$\frac{d}{dx}ReLU(x) = \begin{cases} 1, & \text{if } x\ 0 \\ 0, & \text{otherwise} \end{cases}$$

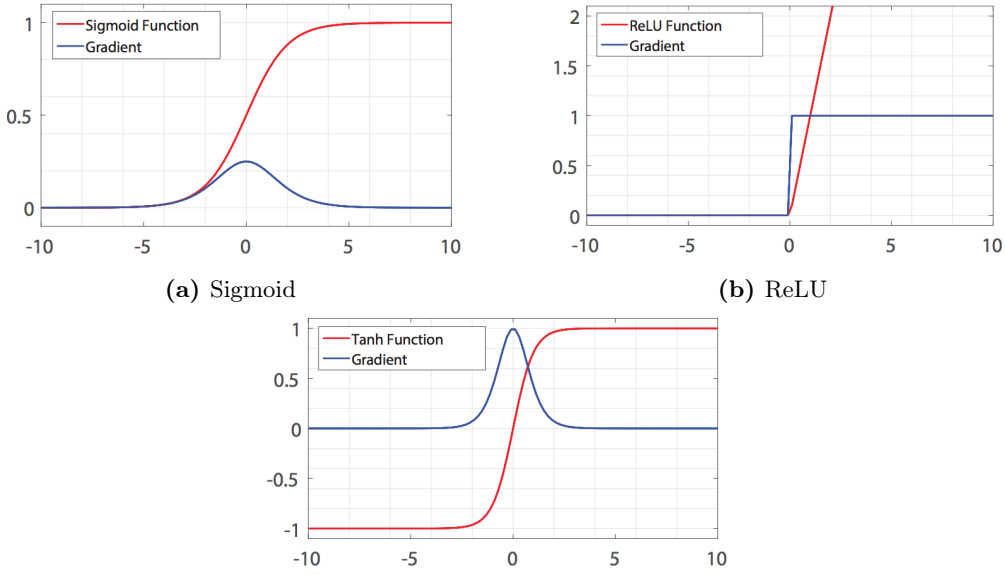
- **Tanh**: The Hyperbolic Tangent function is a function that has a zero centered output and a range of (-1, 1). Figure 9c illustrates the tanh plot. The tanh and it's derivative are defined using the following equations:

$$tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

$$\frac{d}{dx}tanh(x) = 1 - tanh^2(x)$$

3.2.6 Dropout

One inexpensive and simple technique used to avoid over-fitting a neural network is called dropout [45]. To apply dropout a hyperparameter p needs to be picked, a probability value of range (0, 1). Next, dropout is applied to neural network layer such that for every unit in the layer there is a probability of p that this unit is going to be dropped, which means that it's output is going to be multiplied by zero and the output of the remaining undropped units will be scaled by a factor $1 - p$ fill the place of dropped units. This technique is only applied in the training phase and is repeated with every forward propagation so that each time different units get dropped and different weight parameters get updated. Dropouts decrease the chance of overfitting because they decrease the co-adaptation between units [46]. Co-adaptation in neural networks means that neurons depend on each others' outputs, this could lead to



(a) Sigmoid **(b)** ReLU

(c) tanh

Figure 9: Example plots of activation functions and their derivatives. Reprinted from [44].

overfitting as if one neuron weights are overfitted that means that all neurons that are dependant on it will output inaccurate values. Since the dropout technique neutralizes neurons randomly, the dependencies between neurons decrease.

3.2.7 Hyperparameters

As mentioned in section 3.1, hyperparameters are the parameters of the model that do not change during the training. Neural networks have the following hyperparameters:

- **Layer size**: The number of units in a layer. Since layer maps input into new space, the number of units decides the number of variables representing the input in the new space.

- **Network depth**: The number of layers in a network.

- **Dropout rate**: The probability of dropping a unit in a layer that has dropout applied to it.

- **Activation functions**: Each layer has an activation function applied to its computed values. The choice of an activation function depends on the type of features to extracted in layer.

- **Optimizer parameters**: Mini-batch size, learning rate and decay rate.

- **Optimizer algorithm**: Gradient descent is not the only algorithm that could train the parameters of a neural network, there is also Rprop [47] and ADAM [48].

3.3 Convolutional neural networks

One problem that arises when using fully-connected networks is the increase of the number of network parameters with the input size. For example, an MLP with one hidden layer of 512 units that gets as an input a small RGB image of size (320, 180). The size of the input layer will have 172,800 units (320×180 pixels for each one of the three primary colors) and the first hidden layer will have more than 88 million parameters ($320 \times 180 \times 3 \times 512$). Another drawback of using fully-connected networks is that the extracted features are not translation invariant, in other words, the extraction of a certain feature is dependant on it's location in the input. This section will describe Convolutional Neural Networks (CNNs), a variation of neural networks that avoids the dimensionality problem and can process data with a grid-like shape, like images or time sequences [30].

In subsections 3.3.1 and 3.3.2, convolutional and pool layers will be explained, these layers are the building blocks of CNNs. Followed by a discussion of the reason behind

CNNs exceeding performance in subsection 3.3.3 and finally transfer learning, a technique to utilize pre-trained CNNs will be discussed in subsection 3.3.4.

3.3.1 Convolutional layer

The main difference between using a fully-connected network and a CNN is that instead of applying a matrix multiplication between input vectors and model parameters, CNNs use the convolution operation. A layer that applies the convolutional operation is called convolutional layer. Before illustrating how the convolution operation works, a component of the convolution operation needs to be illustrated. Assuming x and w are two 1-dimensional discrete functions of the same scalar variable a, the magnitude of the overlap between the two functions could be computed by adding the product of the two functions at every possible value of the variable a:

$$\sum_{a} x(a) \cdot w(a)$$

The convolution operation given two functions x and w, would shift the function x with the value t, and for every possible value of t, the convolution operation would compute the overlap between x and w:

$$(x * w)(t) = \sum_{a} x(a) \cdot w(t - a)$$

In CNNs, the function x is the input to the convolutional layer and the function w is the parameter of the layer known as kernel or filter. The convolutional operation can also extend to more than 1-dimensional functions. For example, When using two-dimensional input and kernel the convolution operation uses two variables (i,j) to convolve on both axes:

$$(I * K)(i, j) = \sum_{m} \sum_{n} I(m, n) K(i - m, j - n)$$

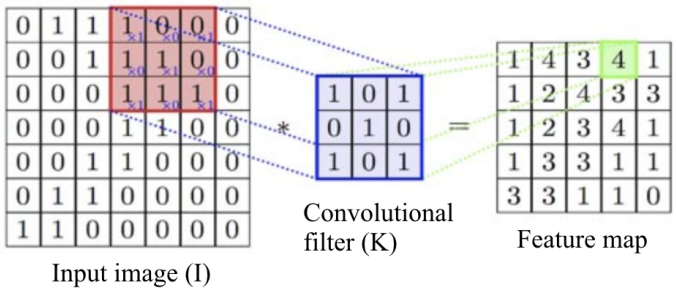


Figure 10: An example of the convolutional operation. Every value in the feature map is computed by a summation of the element-wise multiplication between the kernel and the overlapped area of the input. Reprinted from [49].

Figure 10 is an illustration of the convolutional operation. The convolutional operation could be described as sliding the convolutional filter over the input from top left to bottom right and each time the filter overlaps with the input, the sum of element-wise multiplication overlapping between the filter and the input is calculated to output one value in the output feature map. In Figure 10 the convolutional filter K is sliding over the image i, the convolution operation is calculated using k and the highlighted red box in I to output the value highlighted in green in the output feature map.

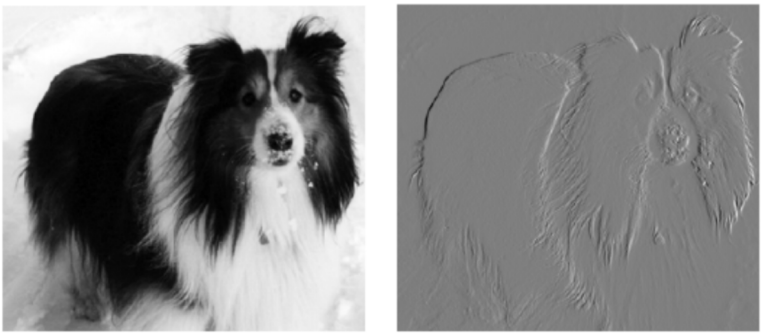
Technically, since the kernel size is smaller than the input and it convolves on different parts of the image and the convolution operation is commutative [30], the convolution function could be expressed as:

$$(I * K)(i, j) = \sum_m \sum_n I(i - m, j - n)K(m, n)$$

For simplicity, neural networks frameworks would implement the cross-correlation operation [30], which is equivalent to the convolution operation:

$$(I * K)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n)$$

Convolutional filters can extract features from points with respect to the relation between them and their neighbors. Figure 11 shows an image and it's output feature



$$K = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Figure 11: An image and the output feature map of applying an edge detector filter to the image and an example of a vertical edge detector K. Reprinted from [30]

map after applying an edge detecting filter. The figure also shows a convolution filter detects horizontal edges by subtracting the values to the right of a pixel from the left values.

A convolutional layer could have multiple convolutional filters and the values of each filter are the parameters to be trained by the network. However, the number and size of the filters are hyperparameters. Another hyperparameter in CNNs is called the stride which is the size of the step when moving the filters.

Another important concept about convolutions is how do convolutions interact with three-dimensional inputs; for example, colored images have the dimensionality width, height, and color. As mentioned before a convolutional kernel has the same dimensionality as the input data, however the size of the kernel is not necessarily equal to the input except for the final dimension. Figure 12 shows a CNN that has an RGB image as an input. The figure shows that an image can be broken down into three two-dimensional arrays (one for each primary color) and that each feature map

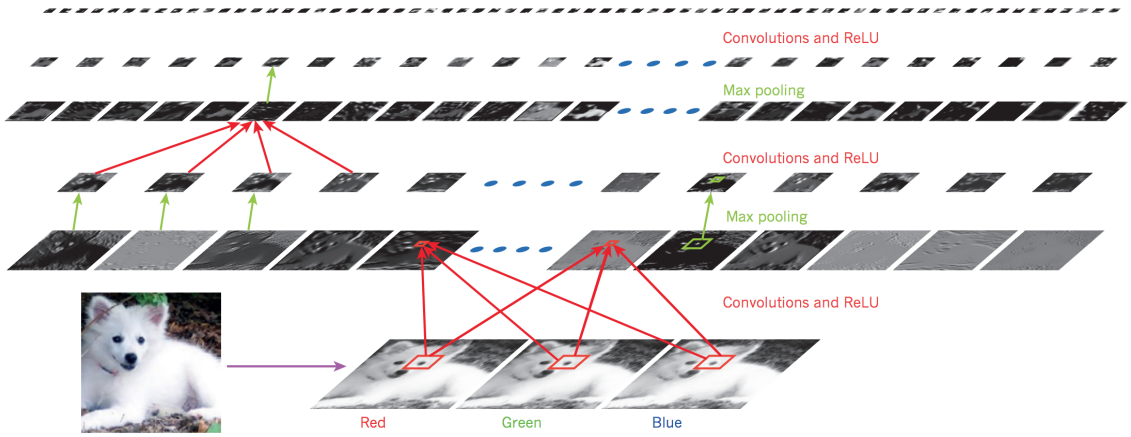


Figure 12: An illustration of the filters of a CNN processing each one of three image colors to output one value in the feature map. Reprinted from [50]

output value is connected to all three arrays, which means that the convolutional kernels convolve on the three arrays at same time, that's because they have the same depth value as the input image. This is described by the following function:

$$(I * K)(i, j) = \sum_m \sum_n \sum_k I(i + m, j + n, k) K(m, n, k)$$

3.3.2 Pooling layers

Pooling layers are another type of layers used often after convolutional layers [51]. Pooling layers are used to decrease the dimensionality of its inputs, while also summarizing the features extracted from a convolutional layer. The parameters of a pooling layer are the size and step of a window that slides over the input image.

The pooling window slides on the input data in a similar way to the convolution kernel and applies a reduction function to the overlapping area from the input that will transform these values into one value in the output feature map. There are two types of functions applied by pooling layers:

- **Average pool**: The kernel outputs the mean of the input values.

- **Max pool**: The kernel outputs the maximum of the input values.

Figure 13 shows an example of max pooling and average pooling layers being applied to an input matrix.

3.3.3 Explaining CNNs performance

A benchmark in object category classification and detection from images is the ImageNet Large Scale Visual Recognition Challenge(ILSVRC) [53], a competition where competitors are asked to detect hundreds of different objects from thousands of images. Winners of this competition have achieved very accurate performances like AlexNet [54], GoogLeNet [55] and ResNet that has better classification accuracy than humans in solving the ILSVRC dataset [56].

This exceeding performance of CNNs can be justified by that, CNNs take advantage of several techniques to achieve better performance:

- **Sparse interactions**: Unlike fully-connected networks that have a weight matrix to describe the relationship between each input and output values, CNNs use kernels that would use only a subset of the input to compute an

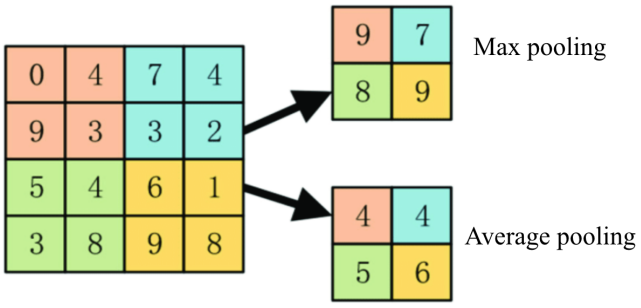


Figure 13: An illustration of max and average pooling operations. The boxes of the same color contribute to the same output feature. Reprinted from [52]

output value. In that sense, the computations needed to compute the output in a CNN are much less than a feed-forward network [30].

- **Parameter sharing**: CNNs use the same parameters to extract low-level features from all parts of the input. Not only does that make CNNs more space-efficient as fewer parameters need to be saved, but also fewer parameters needs to be trained [30].

- **Equivariant representations**: Since CNNs use kernels to extract features from an input, the location of the feature in the input is invariant as the CNN will convolve the kernel on all of the input [30].

- **Stacking layers**: A convolutional layer has the ability to extract low-level features. Stacking convolutional layers and feeding the output feature maps to convolutional layers gives CNNs the ability to extract higher features as the network goes deeper [57]. So for example, a deep CNN that processes image data will start by extracting low-level features, such as edges and lines, in the early layers. As the features are fed to more convolutional layers the extracted features will be shapes and complicated objects. Figure 14 shows a CNN that extracts hierarchical features. The first layer extracts edges, the edges feature map is then passed to another convolutional layer to detect if there are edges that can form shapes. Finally, the shapes feature map is then passed to another layer that could detect if these shapes form objects.

- **Receptive fields**: Not only do the pooling layers decrease the dimensionality of the data, but also they summarize the output feature maps of the convolutions so that convolutions from the next layers could build new features using less parameters. [51].

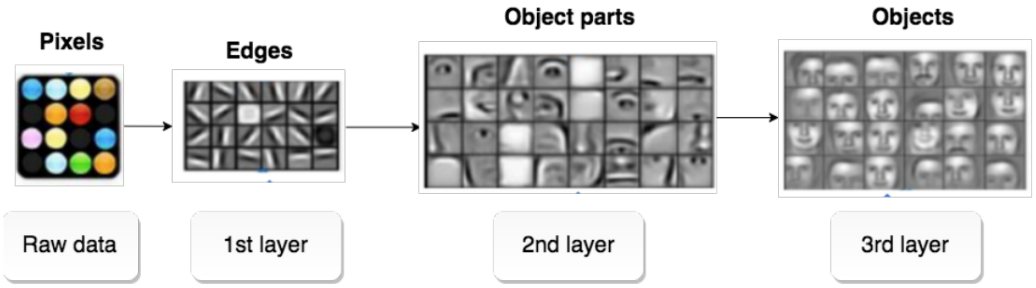


Figure 14: An illustration of a CNN that extracts hierarchical features. The network starts by extracting low-level features in the first layer like edges. The second layer extracts object parts based on the edges extracted from the previous layer and make the features extracted more high-level and so on. Reprinted from [58]

3.3.4 Transfer learning

As mentioned in subsection 3.3.3, there exists CNNs that have very high performances when it comes to image recognition. These works have used deep CNNs that need huge datasets like the one from the ILSVRC to train; however, the trained parameters of these networks are available for public usage.

This gave motivation for the transfer learning concept, which is instead of training a deep neural network from scratch to solve a specific problem, use a pre-trained network that has good performance solving another problem and add just a few layers on top of it that will relate to the specific problem. Transfer learning takes advantage of the fact that the early layers in a deep CNN extract general low-level features that are necessary to solve many problems, while the last few layers could be related to solving many practical problems because they use the high-level features differently to solve the final task.

3.4 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a variation of neural networks that are designed to process data that has a sequential form like $x^{(1)}, x^{(2)}, ..., x^{(t)}$. Similar to CNNs, RNNs also take advantage of parameter sharing to extract the same features from different parts of data, so as CNNs can expand to extract features from images with large width and height, RNNs can expand and extract features from long sequences without increasing the number of parameters. Also, RNNs have some similarities with 1D convolutions, where the output at each point of a convolution operation is a function of the input and it's neighbors, while in RNNs the output at each point is a result of processing the input and the previous outputs.

The output of an RNN can be described using the classical form of a dynamic system [30]:

$$s^{(t)} = f(s^{(t-1)}; \theta)$$

The former equation states that the output of the system at time t is a function of the output of the system at time $t-1$ and the parameters θ of the system and since the output is computed recursively, $s^{(t-1)}$ is a representation of the outputs of all the previous states. This function could be broken down to show the relationship between the outputs of each time step; for example, the output at time $t = 3$ is:

$$\begin{aligned} s^{(3)} &= f(s^2; \theta) \\ &= f(f(s^1; \theta); \theta) \end{aligned}$$

The same concept could be applied to an RNN, only RNNs also have inputs or possibly just one input, that will affect the output computation. The first RNN model was introduced in [59] and is known as vanilla RNN. Figure 15 shows an illustration of unfolding a vanilla RNN through time. The left part of Figure 15 shows an RNN with a recurrent unit that uses the hidden layer computed from previous steps to compute the output and the right side shows the unfolded version of the same network.

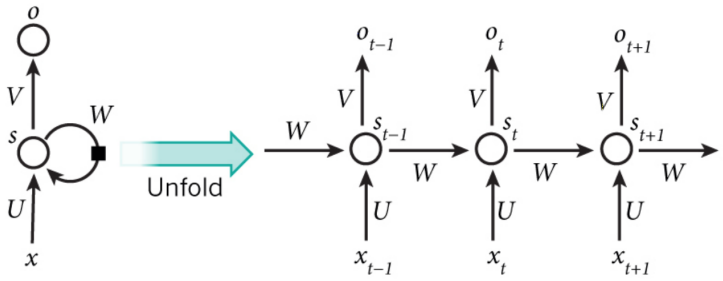


Figure 15: Unfolding of an RNN through time. Reprinted from [50]

In Figure 15, x is the input sequence, U, W and V are parameter matrices of the network, h_t represents the hidden state of the network at time t and o is the output sequence. An RNN could also have bias vectors denoted as b and c and an activation function f. The forward propagation of an RNN starts by initializing the first hidden state $h^{(0)}$ and for each time step t the following equations are computed:

$$a^{(t)} = b + W \cdot h^{(t-1)} + U \cdot x^{(t)}$$
$$h^{(t)} = f(a^{(t)})$$
$$o^{(t)} = c + V \cdot h^{(t)}$$

The RNN in Figure 15 is a synced many to many RNN because both inputs and outputs of the network are sequential and have the same sequence size. However, there are other types of RNNs and each one of them is suitable for a different type of problem:

- **One to many**: Only the output is sequential. This could be used for example in the image captioning problem where the input is an image (non-sequential) while the output is text (sequential) [60].

- **Many to one**: Only input is sequential. This could be used for solving a sentiment analysis problem, when having a sequential input for example text and the RNN will output the predicted sentiment of the input [61].

- **Many to many**: The network process variable sequence size input and output a variable size sequence. This could be used for translation, where it is not necessary that the input and output have the same sequence size [62].

- **Synced many to many**: A good application for these RNNs could be object tracking, where for each input image frame the network outputs the location of the object of interest [63].

A variation of RNNs will be explained in subsection 3.4.1. The drawbacks of using RNNs will be discussed in subsection 3.4.2 and a variation of RNN that solves them is explained in subsection 3.4.3.

3.4.1 Bidirectional RNNs

In some problems, the output of an RNN at a certain time step does not depend only on the input and the previous state of the RNN, but also the inputs from the following time steps. For example, understanding the meaning of a word that has two meanings can only be resolved by reading the whole sentence. To solve this problem the Bidirectional RNN (BRNN) has been introduced [64]. A BRNN uses two RNNs, one that processes an input sequence forward and another one to process it backward. At each time step t a BRNN will have two hidden states, one for the forward RNN $h^{(t)}$ and one for the backward RNN $g^{(t)}$ and both of these states will be concatenated, then fed to the output unit $o^{(t)}$. Figure 16 shows an example of a BRNN.

3.4.2 Drawbacks of RNNs

There are two drawbacks of using RNNs:

- **Long term dependencies**: Sometimes to predict an output in a time step depends on an input that was fed to the network far from the current time

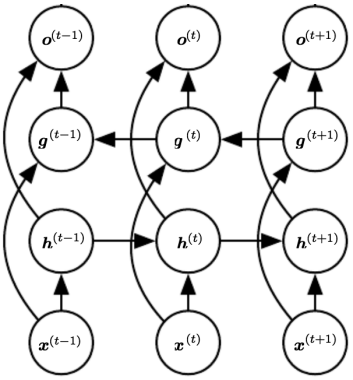


Figure 16: An illustration of a Bidirectional RNN. The figure shows how the input is processed forward and backward before being propagated to the next layer. Reprinted from [30]

step. For example, an RNN that given a sentence and should predict the next word could get an input sentence like "I grew up in France ... I speak fluent ", the next word to be predicted by the network should be the "French", but this prediction depends on the word "France". Although RNNs can use data from the previous time steps using $h^{(t-1)}$; however, the processed data from all the input sequence is decoded into this one vector and the network propagates these information with no option of focusing on some parts more than others, but rather all data is treated the same .

- **Exploding and vanishing gradients**: When the gradients are computed for updating the parameters of a neural network the backpropagation algorithm carries out multiple multiplication operations as illustrated in subsection 3.2.4. However in a deep neural network or an RNN, the length of the multiplication series is high and the updating value for the early parameters could be very low in case of activation values being less than one through the series of multiplication (vanishing gradient), or very high in case of high activation values (exploding gradient) [30].

3.4.3 Gated RNNs

Gated RNNs are variations of RNNs that are used to avoid the drawbacks mentioned in subsection 3.4.2 by adding more functions and parameters trained by the network that will determine if the information encoded in the network should be kept or removed, such functions are known as gates. In their work, Hochreiter et al. [65], have introduced the gated RNN, Long Short Term Memory (LSTM), which is the RNN used in this work. The structure of an LSTM is different than a vanilla RNN. Figure 17 illustrates the internal structure of an LSTM. An LSTM has one more state than a vanilla RNN, which is known as the memory state $c^{(t)}$ that has two weight matrices U_c, W_c and a bias vector b_c. An LSTM also has three gate functions: the forget gate f, the input gate i and the output gate o. Each one of these gates has two weight matrices and a bias vector. All of these gates are fed the input from the current time step $x^{(t)}$ and the hidden state from the previous time step $h^{(t-1)}$. The outputs of the gates are computed by the following equations:

$$f_t = \sigma(b_f + U_f \cdot x^{(t)} + W_f \cdot h^{(t-1)})$$

$$i_t = \sigma(b_i + U_i \cdot x^{(t)} + W_i \cdot h^{(t-1)})$$

$$o_t = \sigma(b_o + U_o \cdot x^{(t)} + W_o \cdot h^{(t-1)})$$

All the gates values are computed using sigmoid activations so the output has the range of (0, 1). These gates control the flow of information through the LSTM, so zero means that no information would pass and one means that all of the information should move on in the network. The forget gate f is used to compute at each step how much of the memory state from the last time step $c^{(t-1)}$ remain in the memory of the current time step $c^{(t)}$. The input gate i determines how much of $x^{(t)}$ and $h^{(t-1)}$ should contribute in the current memory state $c^{(t)}$. The output gate determines the contribution of the memory state in the output and hidden state $h^{(t)}$. To compute

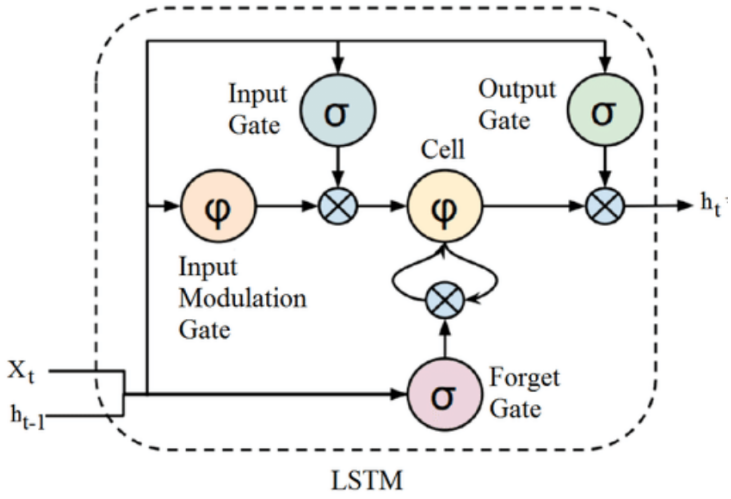


Figure 17: A figure of an LSTM cell. The input and previous hidden state are fed to the input, forget, and output gates. They are also fed to the input modification gate to compute their contribution in the new memory state. Reprinted from [66].

the forward propagation, $c^{(0)}$ and $h^{(0)}$ need to be initialized. After that for each time step the following equations are used:

$$\tilde{c}^{(t)} = tanh(b_c + U_c \cdot x^{(t)} + W_c \cdot h^{(t-1)})$$
$$c^{(t)} = f_t \cdot c^{(t-1)} + i_t \cdot \tilde{c}^{(t)}$$
$$h^{(t)} = o_t \cdot tanh(c^{(t)})$$

Another gated RNN is the Gated Recurrent Unit (GRU) [62]. GRUs have a simpler architecture than LSTM, but also was found to have a comparable performance for few tasks [67].

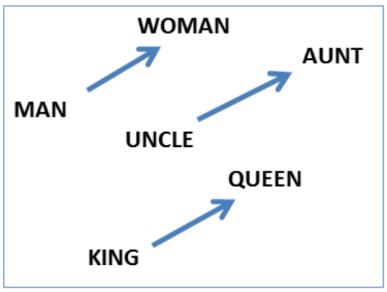


Figure 18: A visual representation of words in vector space. The figure shows that the differences between the vectors can represent information, in this case it represents gender. Reprinted from [70].

3.5 Word embeddings

Word embeddings are vectors used as features that represent words in neural networks. Not only are these features unique for each word, but also they also capture meaning and similarities between words [30]. The values of these vectors are constructed using neural networks that are trained to understand words from the contexts they appear in [68][69]. In this work, the Word2vec word embeddings model introduced in the research work by Mikolov e.t a.l. [68], is used to represent text data as input features for predicting personality. Figure 18 shows how the word2vec embeddings can capture the meaning of gender as the distance between the vectors representing the words "man" and "woman" is equal to the distance between the words "uncle" and "aunt" and also "king" and "queen".

4 Approach

In this chapter, the details of the deep learning models used to predict the OCEAN personality traits will be discussed. Each subsection of this chapter will handle the models that take a certain modality as input and after that, the fusing techniques that predict personality from several modalities will be discussed. The problem is a regression problem with five output labels, one for each OCEAN personality trait and the range of each is [0, 1]. The models that take text as input are described in section 4.1, section 4.2 shows the audio model and the models that process images are explained in section 4.3. Multimodal models are shown in section 4.4.

4.1 Text modality

This section considers the problem, that given some text a neural network should be used to predict the OCEAN personality traits of the author of the text. The problem is similar to text sentiment analysis [71]; however, instead of predicting positive or negative sentiment, OCEAN traits are predicted. Two text neural network models were implemented in this thesis: a word-based model and a character-based model.

4.1.1 Word model

A word-based neural network model is a neural network that takes input in the form of text and uses word embedding vectors to transform each word into a vector and

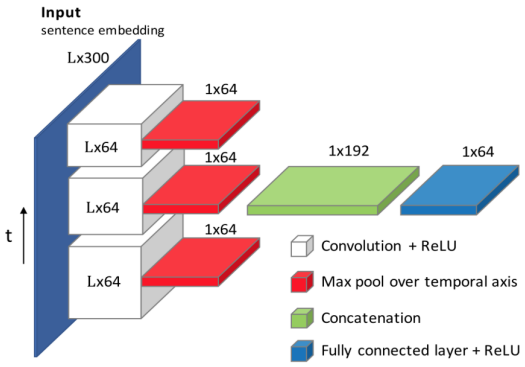


Figure 19: A visual representation of the word-based neural network implemented by [1]. Reprinted from [1].

thus an input sentence into a matrix. In this work, the word-based neural network model by Onno Kampman et al. [1] has been re-implemented. Figure 19 shows the neural network architecture of the model. Word2vec is used as the word embedding layer, each word is modeled as a vector of size 300 and these vectors were pre-trained on Google News data. The output word vectors are then fed in parallel to three 1D convolutional layers with kernel sizes of 3, 4, and 5 words and a stride of one word. Each one of the convolutional layers extracts features from different sequence lengths (3, 4, 5) of words, which is equivalent to extracting features from n-grams where for each convolution layer n is the kernel size. A global max-pooling layer is used to summarize these n-grams features into three vectors of the size 64. The vectors are then concatenated and passed to a fully-connected layer with 64 units and a ReLU activation. Finally, an output layer with sigmoid activation function will transform the feature vector into the OCEAN personality traits space.

4.1.2 Character model

One problem that arises when using a neural network model with word embeddings is that no matter how many words the word embeddings matrix has, there will always be new words that were not used in the embeddings' training, leading to

an out-of-vocabulary problem. Additionally, when collecting text data from social media websites, a significant number of words are noisy which will also lead to the same problem, which gave the motivation of using a character-based model. Unlike word-based models, character-based models encode each character with a unique vector and feed the character vectors of a word to a recurrent neural network and the output vector is the word encoding.

A research paper published by Xerox [72] has introduced a model named C2W2S4PT (Character to Word to Sentence for Personality). Figure 20 illustrates how the model works, an input sentence S is divided into a sequence of words $w_1, w_2, ..., w_n$. To get corresponding vectors of these words, each word is broken down to a sequence of characters, so e.g. the word w_i will be the sequence $c_{i_1}, c_{i_2}, ..., c_{i_n}$, each element in this sequence has a unique one-hot vector that will be fed to a bi-directional LSTM to provide the encoding e_{w_i} of the word w_i. The word encoding vectors are then fed into another bi-directional LSTM that will output a feature vector representing the whole sentence. These vectors will be fed then to a fully-connected layer with a ReLU activation function and finally an output layer to map the sentence's feature vector to the OCEAN traits.

4.2 Audio modality

The focus of this section will be on the representation and the model used to analyze audio data to predict the OCEAN personality traits of a person using their voice.

In the physical world, the source of any audio or sound is the vibration of an object that sends a mechanical longitudinal wave that travels in space, these waves represent the change of pressure in the space they are traveling in [73]. However, a limitation arises for utilizing sound waves in the digital world is that these waves are continuous and have different pressure values at every point on a time scale, to solve this problem sampling is used. Sampling is the measurement of pressure of the wave at equally

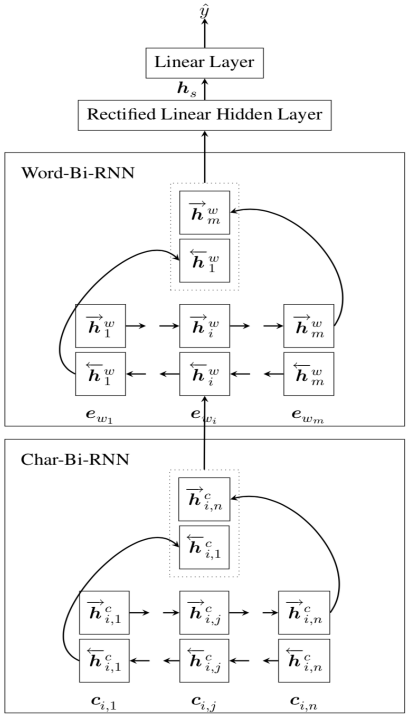


Figure 20: A visual representation of the C2W2S4P network. The networks has bi-directional LSTM that computes the vector representation of words using a sequence of characters, these vectors are fed to another bi-directional LSTM to generate a vector that represents a whole sentence. Finally the sentance feature vector is fed to a full-yconnected layer and an output layer to predict OCEAN traits. Reprinted from [72].

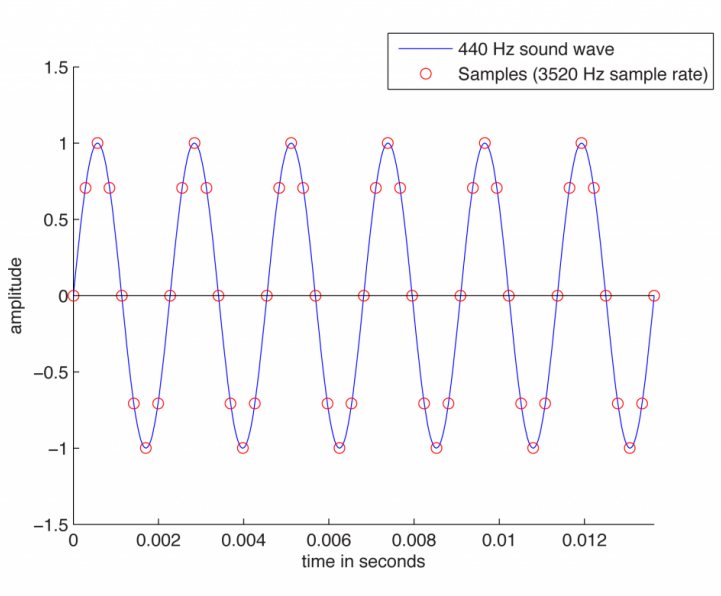


Figure 21: An example of samples extracted from a sound signal with a sampling rate of 3520 sample/sec. Reprinted from [74].

distanced points on the timescale and then grouping these measurements in a list that will represent the wave, this list is known as raw audio data. The distance between the measured points is determined by the sample rate (the number of points measured per second), the higher the sample rate the higher the quality of the audio. Figure 21 shows an example of wave sampling, where the blue line is the wave and the red points are the samples.

In their work, Tzirakis et al. introduced an end-to-end deep learning model, that given raw audio data would predict the emotions of the speaker [75]. In this thesis, the architecture of the neural network model introduced in [75] has been implemented to solve the personality problem, but few hyper-parameters have been changed to yield better performance. In particular, the authors of [75] have introduced a new methodology to choose the pooling size of the max-pooling layers coming after convolutional layers using the following equation :

$$R = \frac{K - 1}{K + P - 1}$$

Where K is the kernel size, P is the pooling size and R is a variable called the overlap rate. The assumption is that P should always have a value that will make R less than 0.5, otherwise, the network will extract and analyze similar features from close frames and the values used for the experiments yielded R with values around 0.4. In the experiments performed in this work, P and k were set to values that made R approximately 0.25 which had slightly better performance.

A visual representation of the neural network is found in Figure 22. The input of the network is raw audio data with a sample rate of 16KHz (16,000 samples/sec), then this data is fed to a stack of convolutional and max-pooling layers. The first part of this block is a convolutional layer with 32 filters and a kernel size of 8, followed by a max-pooling layer with kernel size 20 and the overlap rate R is 0.259. In order to extract more high-level features, another convolution layer is used with twice the number of filters of the previous convolution layer 64 and a kernel size of 6. To preserve the value of R, the max-pooling kernel size is decreased to 16. The last convolution layer has also the double amount of filters as the second convolution layer 128, but with the same kernel size and since the convolution kernel size did not change so the max-pooling size is also the same having an overlap rate of 0.238. Like with images, the first convolutional layer will extract low-level features, use a pooling layer to summarize the features extracted, and pass the output to another convolutional layer to extract higher-level features and so on. The output features from the last pooling layer are 48 vectors of size 128, representing 128 features extracted from 48 sub-parts of the audio. These vectors are fed to two bi-directional LSTMs to extract temporal relations between these features and then a fully-connected layer with 256 units with *tanh* as the activation function and finally an output fully-connected layer with five units (one for each personality trait) and a sigmoid function for activation. To prevent the network from overfitting dropout layers with a dropout rate of 0.5 are added after the last max-pooling layer and the fully-connected layer.

There are only two differences in the implementation of this model in [75] and the

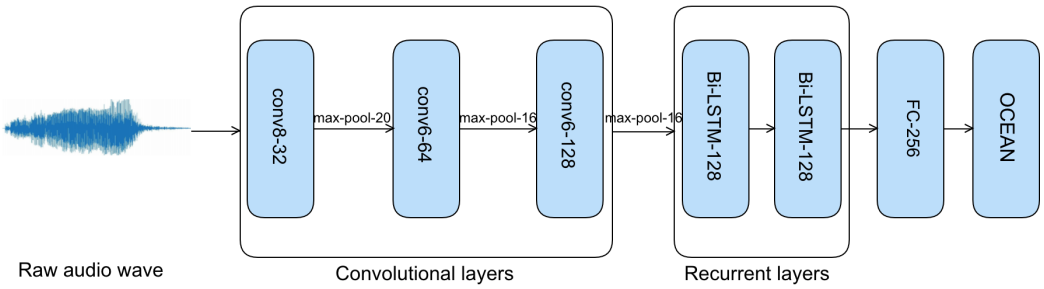


Figure 22: End-to-end neural network for audio input. Reprinted from [75].

implementation of this model in this thesis: the different overlap rate R and in this work bi-directional LSTMs are used instead of regular LSTMs.

4.3 Image modality

In this section, the last data type will be discussed, images. This data type could be utilized for the problem in two distinct ways, either as a single image containing one person and in that case the problem would be analyzing the facial expressions and background features and using them to predict the personality traits of the person in the image or as a video where there is a stream of images from which these features could be extracted from. In this work, experiments were only conducted on videos; however, one of the models used was inspired from a work that is solving the problem using images [27].

4.3.1 Image based model

In the work from Xiu-Shen Wei, et al.[27], several models that use a person's image as an input to predict their personality traits were introduced, the one with the highest performance was a modification of another model created by the same authors called Descriptor Aggregation Networks (DAN) and the modified version is called DAN+.

A visual representation of DAN+ is shown in figure 23. The architecture of DAN+ is based on the VGG-16 network [76], a network well known for it's high performance in object recognition and classification. However, instead of connecting the last pooling layer with fully-connected layers, each of the last pooling layer ($pool\_5$) and the activation layer before the last convolution layer ($relu\_5\_2$), are fed to global average-pooling and global max-pooling layers to output four 512 dimensional vectors. Each one of these vectors is normalized using the $l2$ function[1], then they are all concatenated and fed to one fully-connected output layer with sigmoid activation. The DAN+ model uses transfer learning by initializing the weights of the network using the pre-trained weights of the VGG-16 network that solves the object classification problem, to solve the APP problem. The VGG-16 weights were trained from the ImageNet dataset and are available and could be used by the deep learning framework Keras[2]. The difference between the architecture of the VGG-16 and DAN+ gives DAN+ few advantages. Since the DAN+ architecture does not use fully-connected layers before the output layer, the model has less parameters making it easier and faster to train and also the final representation of the image before the output layer is smaller.

Conversely, in this thesis the DAN+ model was extended to predict personality using video instead of a single image. In order to do that, six frames are extracted randomly from a video and each image will be input to a separate DAN+ network with the same architecture as in [27]. Furthermore, the outputs of the six networks are concatenated and fed to a single fully-connected output layer also with sigmoid activation. This model will be referred to as DAN+V in the rest of the thesis.

[1] $l2(x) = \sqrt{\sum_n x_n^2}$[30]
[2] Keras version 2.2.4 `https://keras.io/`

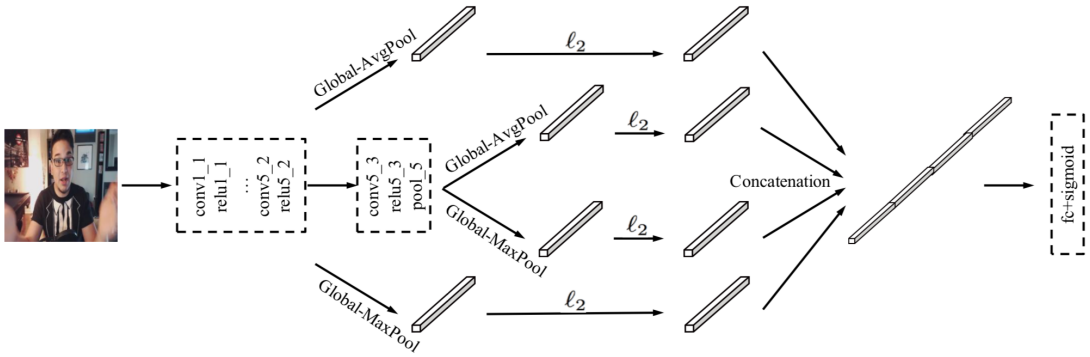


Figure 23: A visual preview of DAN+. Reprinted from [27].

4.3.2 Video-based model

The work of Subramaniam et. al [28] has introduced a multimodal neural network model that participated in the First Impressions competition in 2016 and won second place. The model processes both audio and visual data. However, in this section the visual neural network is the subject of interest. As input, the model is fed six image frames extracted from a video, the images are grouped together in one 4-dimensional tensor with the dimension (color channels, number of images, image width, image height). The vector is then input to a 3-dimensional convolution layer with a ReLU activation function, followed by a 3-dimensional max-pooling layer. The output tensor is then passed twice to 3-dimensional convolution and max-pooling layers, but with kernels of smaller size and the final output vector is then flattened and concatenated with the audio features and passed to a fully-connected layer. The architecture of this model is using 3-dimensional convolutional layers, that train 4-dimensional kernels to extract visual features from several images. The intuition is that features extracted from several images using the same kernels could have a deeper meaning or information than features extracted from several images separately, thus solving the problem more accurately.

In this thesis, another video-based convolution model has been implemented based on the same motivation as the previous model. Figure 24 shows a visual representation

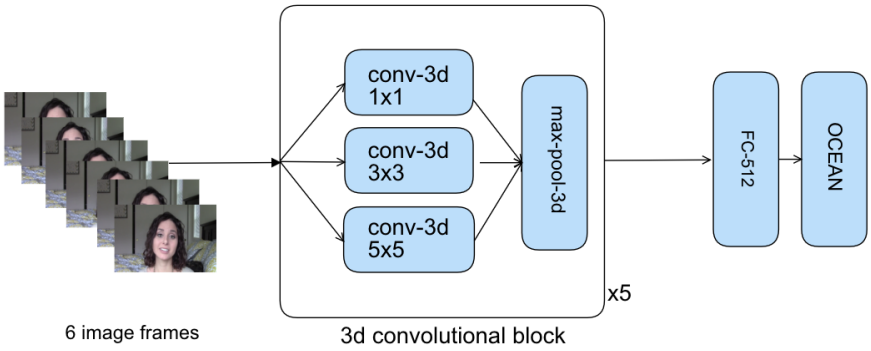


Figure 24: A visual representation of the video-based model. The model consists of a stack of five convolutional-pooling blocks and the output is fed to a fully-connected layer and then an output layer to make OCEAN predictions.

of the model. The model aligns the six input frames the same way as [28]. After that, the input vector is fed in parallel to three 3-dimensional convolution layers with three different kernel sizes, the outputs are then concatenated and fed to a 3-dimensional max-pooling layer. The parallel convolution layers were inspired by the architecture of the Inception network [55]. The convolution max-pooling block is repeated five times before being fed to a fully-connected layer and finally the output layer with sigmoid activation function.

4.4 Multimodality

In this section, the models that predict OCEAN traits using information from more than one modality will be discussed. There were two strategies used to build these models: early fusion (mentioned as middle fusion in some literature) and late fusion, also known as ensembles.

4.4.1 Early fusion

The early fusion technique is based on the idea of fusing features from different data types and training a model that will learn and utilize the correlations between high level features of each modality[77]. In order to test the early fusion technique, several models were implemented for each combination of modalities.

The fusion of several neural networks is done by picking a subset of layers from each network. This subset is the networks' layers excluding the last fully-connected layers. Next, the outputs of the last layer of each sub-network are merged into one vector that represents the high level features extracted from all modalities. The high level features are then fed to fully-connected layers to map these features to the OCEAN traits.

Figures 25-29 show the architectures of the multimodal neural networks implemented in this thesis.

4.4.2 Late fusion

The late fusion technique work as follows: several models that take a single modality as input are trained separately and after that certain heuristics are used, e.g. majority vote or averaging [77]. In this work the final prediction was calculated by averaging the predictions of the trained models. The use of this technique was motivated by the work of [27], as they have used this technique while participating the First Impressions competition and it has yielded the best performance in the competition. Also, ensembles can be expected to have less generalization error than one neural network [78]. In addition, ensembles can be easily extended to multiple models to add more diversity to the predictions by using several diverse models.

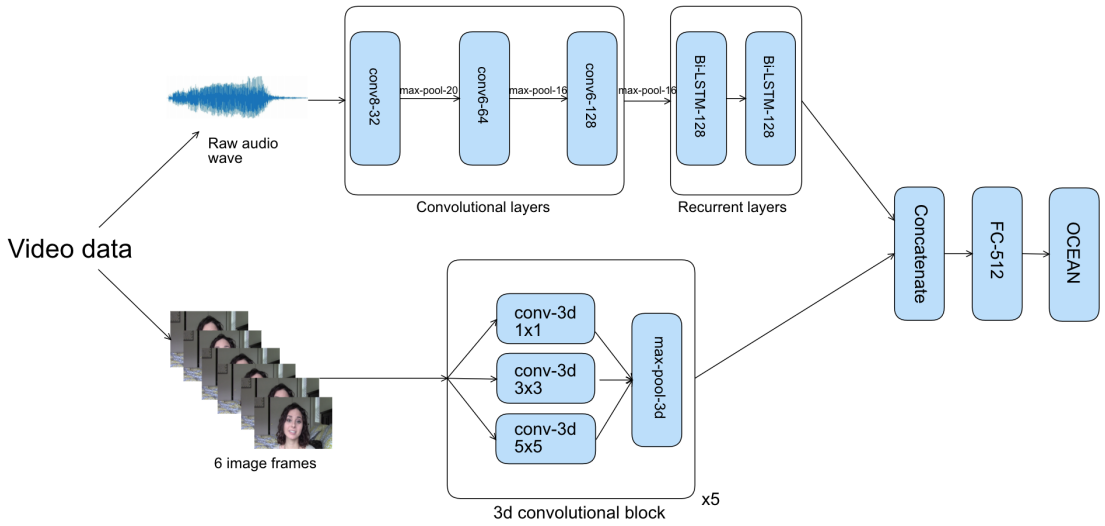


Figure 25: Fusion between audio end-to-end model and the video-based model. From the video data, the raw audio data and six image frames are extracted. The raw audio data is fed to the end-to-end audio model to extract audio features. The six image frames are fed to the video-based conv-3d model to extract visual features. Audio and visual features are merged and fed to a fully-connected layer and output layer to predict OCEAN traits.

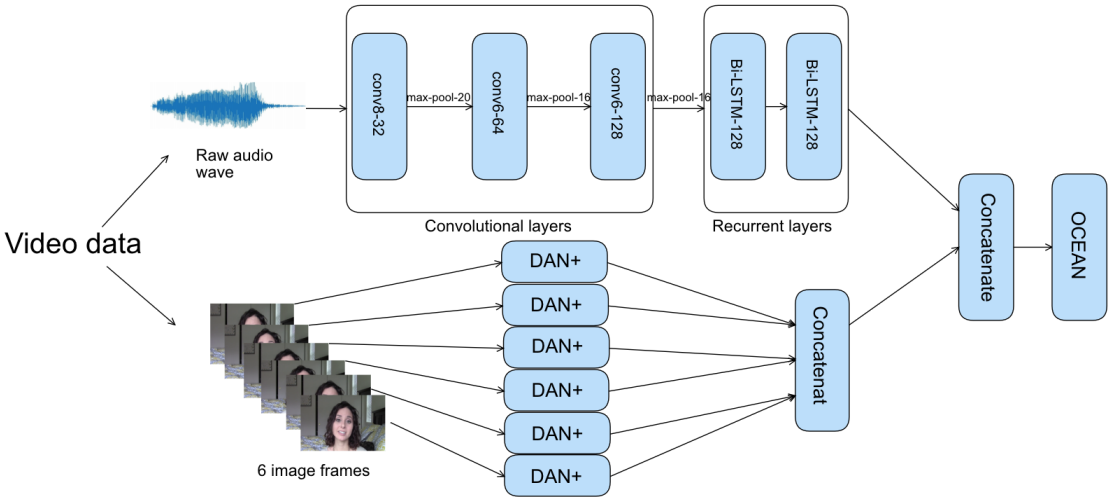


Figure 26: Fusion between audio end-to-end model and DAN+V model. From the video data, the raw audio data and six image frames are extracted. The raw audio data is fed to the end-to-end audio model to extract audio features. The six image frames are fed to the DAN+V model to extract visual features. Audio and visual features are merged and fed to the output layer to predict OCEAN traits.

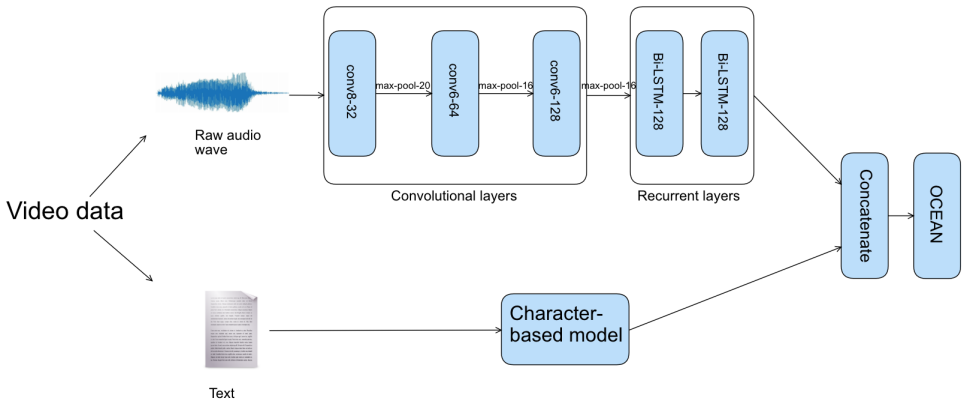


Figure 27: Fusion between audio end-to-end model and C2W2S4P model. From the video data, the raw audio data and text are extracted. The raw audio data is fed to the end-to-end audio model to extract audio features. The text is fed to the C2W2S4P model to extract text features. Audio and text features are merged and fed to the output layer to predict OCEAN traits.

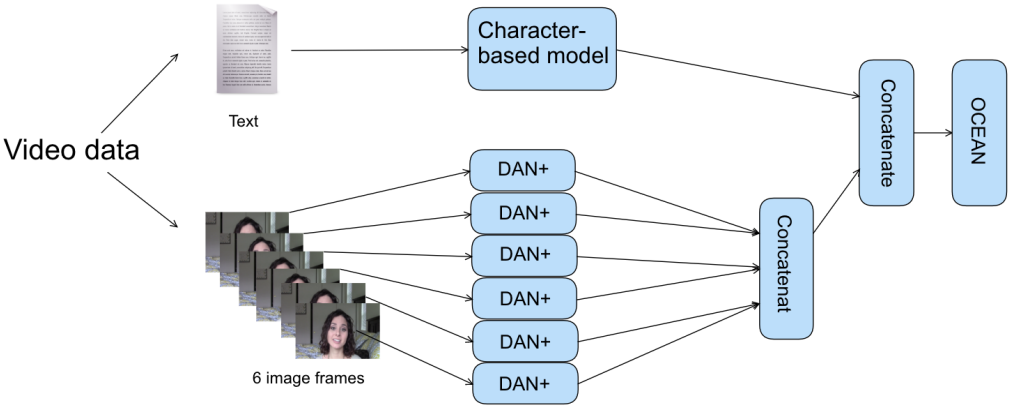


Figure 28: Fusion between DAN+V model and C2W2S4P model. From the video data, six image frames and text are extracted. The six image frames are fed to the DAN+V model to extract visual features. The text is fed to the C2W2S4P model to extract text features. Visual and text features are merged and fed to the output layer to predict OCEAN traits.

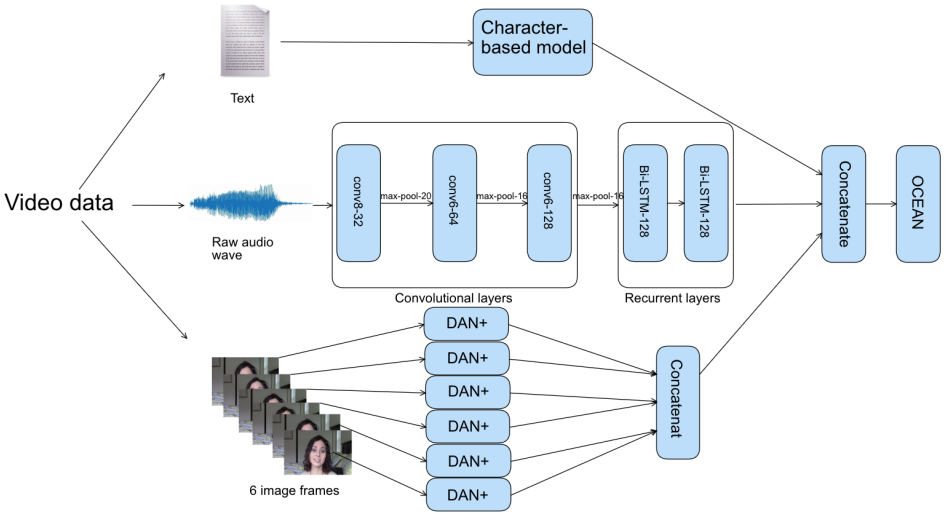


Figure 29: Fusion between audio end-to-end model, DAN+V and char model. From the video data, raw audio data, six image frames, and text are extracted. The six image frames are fed to the DAN+V model to extract visual features. The text is fed to the C2W2S4P model to extract text features. The raw audio data is fed to the end-to-end model to extract audio features. All features are merged and fed to the output layer to predict OCEAN traits.

5 Experimental Setup

This chapter will discuss the details of the experiments, what kind of data was used, and how performance was measured. A description of the datasets and their underlying distribution will be shown in section 5.1. The metrics used to evaluate the implemented models are explained in section 5.2.

In this thesis, the code used to carry out the experiments was in Python 3. For reading and splitting the datasets the library pandas was used, the library Keras was used to implement all the neural network models and the libraries NumPy and SciPy where used for calculating some of the evaluation metrics.

5.1 Datasets

This section describes the datasets that were used to train and test the performance of the models described in chapter 4. Subsection 5.1.1 will describe the First Impressions dataset, while the PAN dataset will be described in subsection 5.1.2.

5.1.1 First Impressions

The First Impressions dataset was introduced by the organizers of the competition First Impressions (ECCV '16, ICPR '16) [7]. The dataset consists of 10,000 video clips extracted from the video sharing website YouTube. The organizers of the competition

found Q&A videos to be the most suitable type of videos for the problem as there is only one person in the video and the subject person of the video is centered in front of the camera with very few movements. Therefore, all the video clips are 15 second tracks cut out of Q&A videos of people who have YouTube channels. The 10,000 clips were extracted from 3,060 unique videos across 2,764 Youtube channels; however, a channel may have one or more video makers. Additionally, for each channel, the organizers considered at most three videos and the clips were extracted such that a maximum of six clips were cut from the same video. These limiting decisions could be justified as forcing more diversity in the dataset, due to the fact that the same speaker will have the same personality traits. Furthermore, the dataset comes with transcribed text and the OCEAN personality traits of the subject person for each clip. The split of the dataset was 8,000 videos for training and 2,000 videos for testing. A Bradley Terry Luce (BTL) model, which is a probabilistic model that uses pairwise comparisons between objects to predict a latent trait, was used by the organizers to obtain labels for the videos. The model was fitted using Maximum Likelihood and had a sigmoid function as the output making the predictions follow a logistic distribution. The pairwise comparisons where gathered using Amazon Mechanical Turk. More details about the BTL model are described at [79]. The reason for using the BTL model instead of doing interviews or asking the subjects of the video to fill surveys was that, these methods suffer from biased and variable results. Additionally, using such techniques is not feasible with huge amounts of diverse data.

Data distribution

In the First Impressions dataset, each video clip is labeled with five numbers corresponding to each personality trait, these number range from 0.0 to 1.0, where one represents the highest value a personality trait could have and zero represents the lowest value. In this section, a simple analysis of each personality trait labels will be discussed. Figure 30 shows the data distributions of the labels for each personality

Trait	Mean	Standard deviation
Openness	0.57	0.15
Conscientiousness	0.52	0.15
Extraversion	0.48	0.15
Agreeableness	0.55	0.13
Neuroticism	0.52	0.15

Table 2: Mean and standard deviation of the First Impressions dataset for each personality trait.

trait using histograms, while Table 2 shows the mean and standard deviation for each trait. The histograms show that all the personality traits labels have a distribution resembling a normal distribution. This can be expected due to the BTL model that outputs predictions with a logistic distribution that is close to normal distribution. Table 2 shows that all of these distributions are centered approximately around the middle point of the value spectrum 0.53 and with approximately a standard deviation of 0.15.

5.1.2 PAN

Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection (PAN) is a set of competitions that co-locate with the conference CLEF (Conference and Labs of the Evaluation Forum), a conference that is helping the development of multilingual data research [80]. In 2015, PAN hosted an author profiling competition through which participants develop machine learning algorithms that given a piece of text would predict the OCEAN personality traits of the author of the text. The dataset used in the competition is multilingual, however in this thesis only the English dataset was used. The dataset was extracted from the social media website Twitter. The English dataset used data from 294 Twitter users, each user had approximately 100 tweets (a sequence of text less than 140 characters), with a total of 27,344 tweets. The OCEAN personality traits labels where collected using surveys completed by the twitter users [81]. The data was split such that 152 users data was used as training

Trait	Mean	Standard deviation
Openness	0.76	0.15
Conscientiousness	0.67	0.15
Extraversion	0.67	0.16
Agreeableness	0.63	0.16
Neuroticism	0.64	0.23

Table 3: Mean and standard deviation of the PAN dataset for each personality trait.

and 142 for testing. The values of the labels range from -0.5 to 0.5, but to make comparisons with the First Impressions dataset easier, the labels where shifted to be from 0 to 1.

Data distribution

The histograms in Figure 31 show the data distribution of each trait and Table 3 shows the mean and standard deviation of each trait. The figures show few essential differences than those from the First Impressions dataset. First, all of the traits appear to lack balance between the frequency of low values and high values as all the traits do not have values below 0.2 and the openness trait does not have a value less than 0.4. This is also verified in Figure 3 as the mean value for all traits is greater than 0.6. Second, Figure 31a shows that the openness trait does not follow a normal distribution.

5.2 Metrics

This section will describe the evaluation metrics used to evaluate the performances of the models discussed in chapter 4.

5.2.1 Mean absolute accuracy

Mean Absolute Accuracy (MAA) is the metric used to evaluate the accuracy of a model's predictions for the test dataset. It is defined by the following equation:

$$MAA(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(1 - |y_i - \hat{y}_i|)$$

Since the output labels y have a range of [0.0, 1.0], the range of the MAA is also [0.0, 1.0], where 1.0 indicate that the predictions are identical to the labels.

5.2.2 Root Mean Square Error

Root Mean Square Error (RMSE) is the square root value of the MSE. It is defined by the following equation:

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

5.2.3 Confusion matrix

A confusion matrix is a table used to evaluate the performance of binary classification model based on a labelled dataset. A confusion matrix is shown in Table 4. The content of the matrix shows accurately the number of correct and wrong predictions for each class. The elements of the confusion matrix will be used to compute the metric explained in subsection 5.2.4.

		Actual	
		+ve	-ve
Pred	+ve	True Positive(TP)	False Positive(FP)
	-ve	False Negative(FN)	True Negative(TN)

Table 4: Confusion matrix

5.2.4 Area Under ROC Curve

The Area Under ROC Curve (AUC) is a classification metric used to evaluate binary classification models that use a probabilistic score and a threshold to classify objects as one of two classes. The value of this metric is calculated by drawing the Receiver Operating Characteristic (ROC) curve for the classification model and measuring the area under this curve, hence the name area under ROC curve. This curve is drawn by plotting the model's True Positive Rate (TPR) against False Positive Rate (FPR) for every possible classification threshold. To define TPR and FPR we need to use terms from the confusion matrix as follows :

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

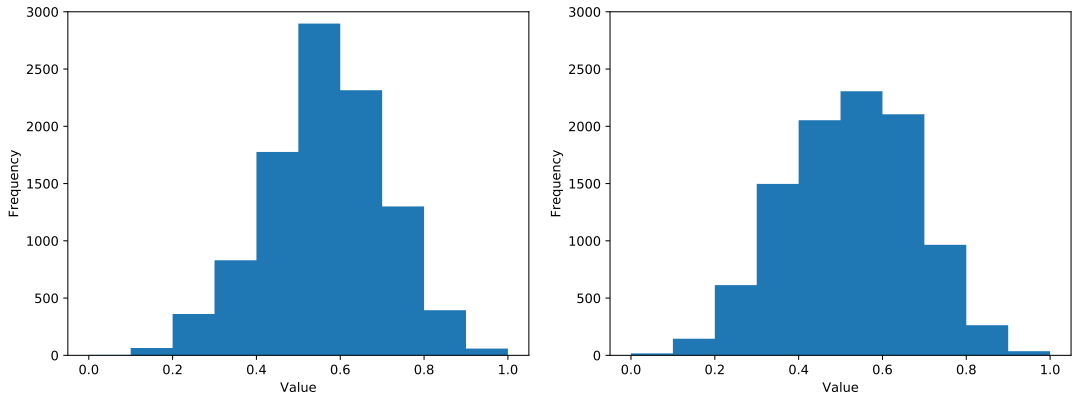
$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

TPR also known as recall, is the ratio between the number of correctly classified positive examples to the total number of positive examples in the dataset. FPR is the ratio of negative examples classified as positives to the total number of negative examples.

These two variables are directly proportional to the number of examples the model classifies as positive. In other words, for a classification model, when the classification threshold is high both TPR and FPR will have low values because the model classifies very few examples as positive, while if the same model has an extremely low classification threshold, TPR and FPR will have high values as more examples are classified as positive.

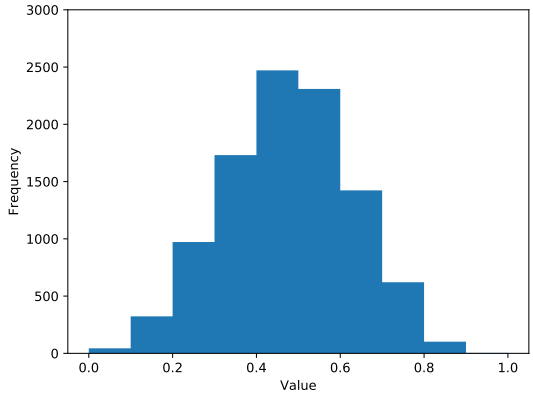
Figure 32 shows four ROC curves with different AUC values. All the plots start at the point (0, 0), at this point the threshold value is set to the maximum value such that all examples are classified as negative, therefore TPR and FPR are equal to 0. For the remaining points in the curve, the threshold is gradually lowered to yield

different classification value for one example in the dataset. If the newly classified point is a true positive then the curve moves vertically as only the TPR will change, otherwise, it is a false positive and the curve will move horizontally. In the case of a perfect classifier, the curve will only start to move horizontally when TPR is already maximum, i.e. the lowest probability score assigned to a positive example is higher than all probabilities assigned by the classifier to negative examples and in that case the ROC curve will cover the whole area of the plot like in Figure 32a. For a less accurate classifier, the ROC curve would start classifying negative examples as positive while the threshold is still higher than the probability score of positive examples, which will make the ROC curve move horizontally before it reaches the maximum vertical point, leading to less area under the ROC curve, similar to Figure 32b. Figure 32c describes the performance of a classifier which can not distinguish between positive and negative. The classifier shown in Figure 32d classifies every positive example as negative and every negative example as positive the exact opposite of the classifier in figure 32a. In summary, AUC represents the probability that a random positive example will have a higher probability prediction than a random negative example [82].

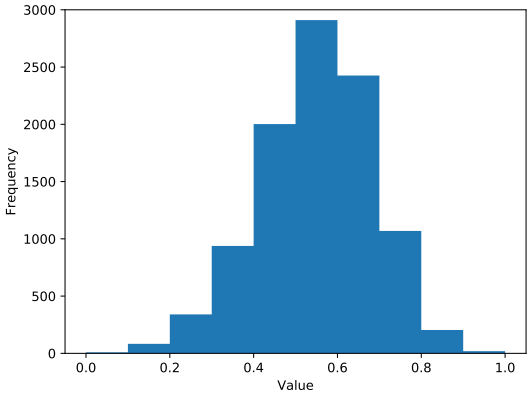


(a) Openness

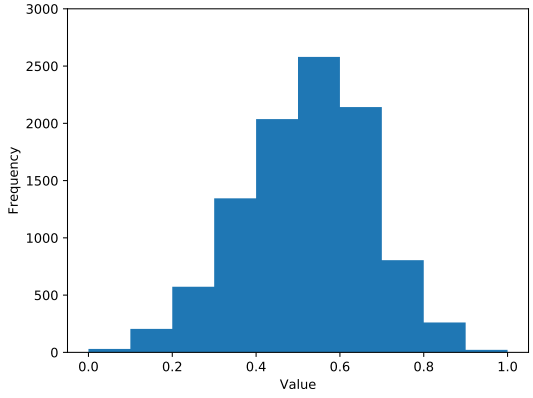
(b) Conscientiousness



(c) Extraversion

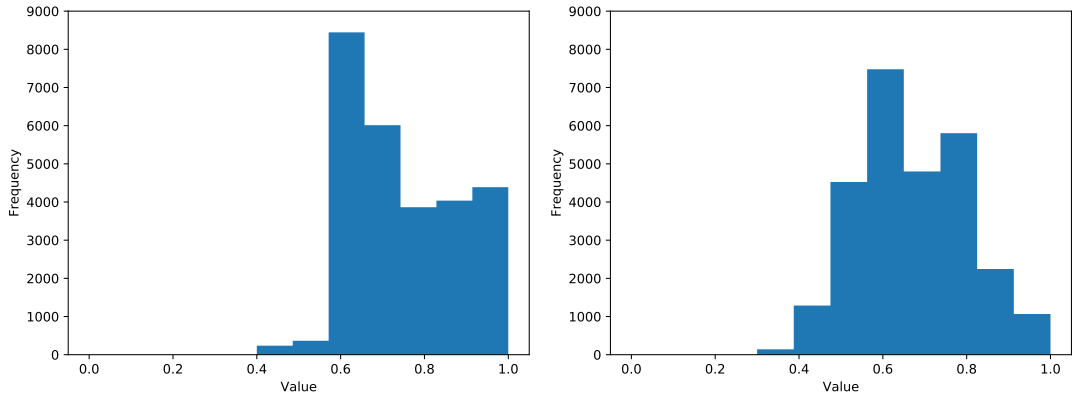


(d) Agreeableness



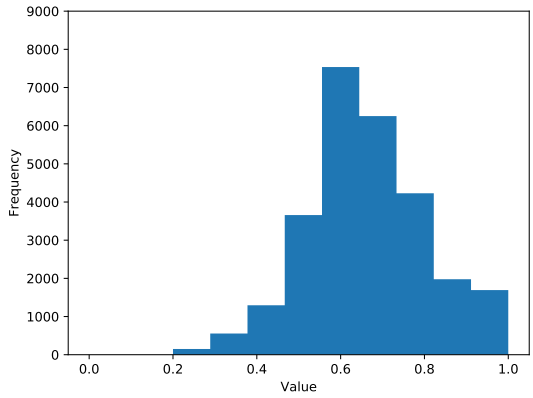
(e) Neuroticism

Figure 30: Histograms that show the distribution of each OCEAN trait of the First Impressions dataset. All the traits follow a normal distribution.

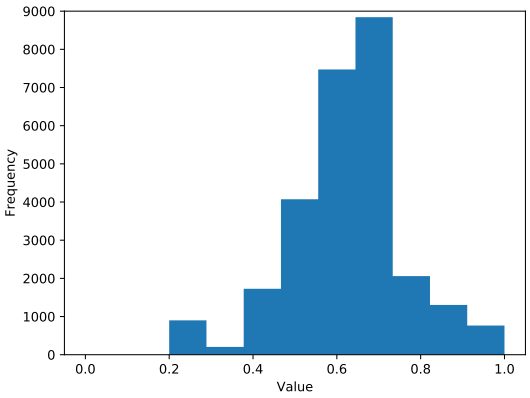


(a) Openness

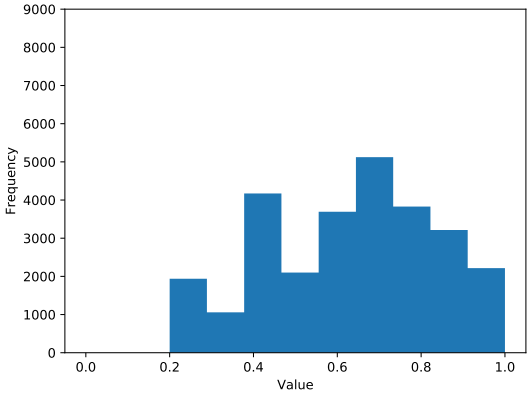
(b) Conscientiousness



(c) Extraversion

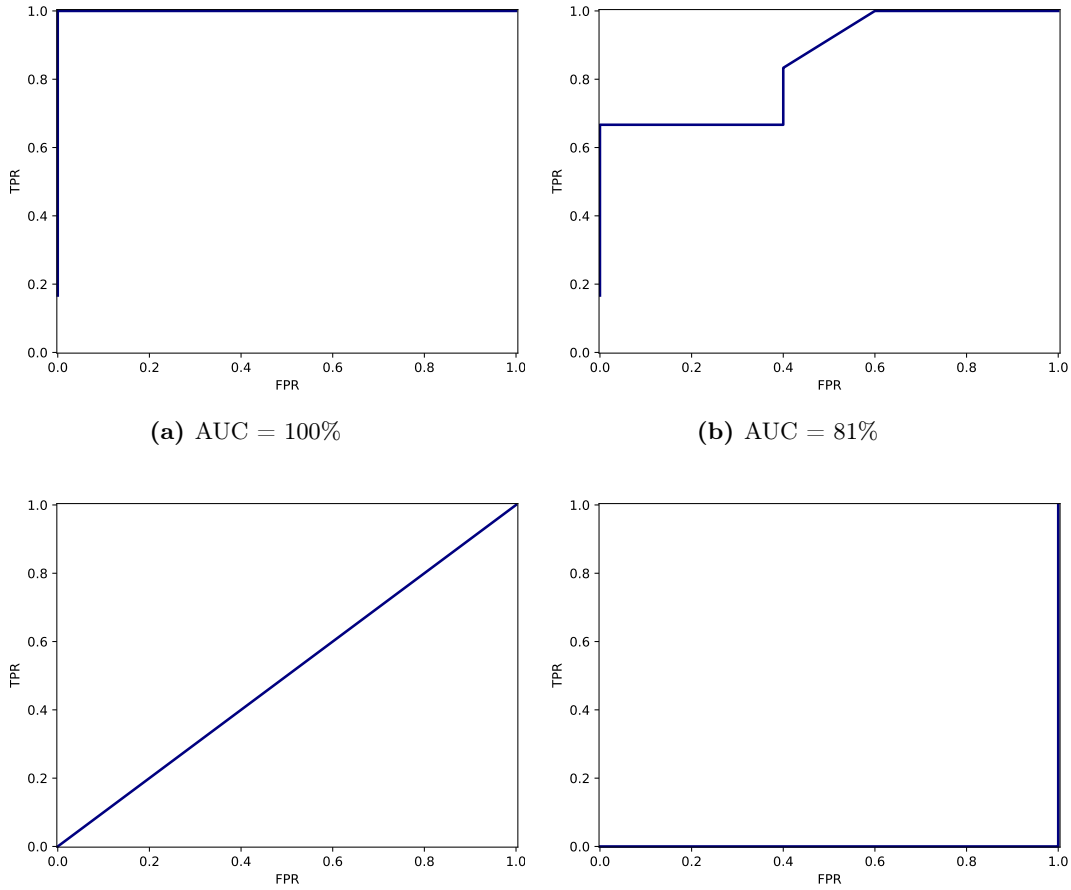


(d) Agreeableness



(e) Neuroticism

Figure 31: Histograms that show the distribution of each OCEAN trait of the PAN dataset. All of the traits do not have any examples with values bellow 0.2. Only the neuroticism trait follows a normal distribution.



(a) AUC = 100%

(b) AUC = 81%

(c) AUC = 50%

(d) AUC = 0%

Figure 32: Plots of ROC curve with different values. Plot (a) shows the ROC curve of classifier that classifies all the examples correctly. Plot (b) shows a classifier with an 81% AUC. Plot (c) represents a classifier that classifies only of half the examples correctly, often resembles a random classifier. Plot (d) is from a classifier that classifies all the examples incorrectly.

6 Results

In this chapter, the performance of the models described in chapter 4 will be evaluated using the MAA and AUC metrics explained in section 5.2. The performance of the models will also be compared to a baseline model. The baseline model used for each dataset, is a model that predicts one value no matter the input. The value predicted by the baseline model is the average of the output labels y of the test dataset. The mean of the output labels is the value that yields the minimum MSE for a constant predictor, this is further discussed in subsection 6.3.1. The baseline model sets a performance threshold, where a model with predictions less accurate than that threshold will imply that the model is not learning from the input features.

In section 6.1, the performance of the models trained using the PAN dataset will be shown, while the performance of the models trained using the First Impressions dataset will be shown in section 6.2. The discussion of the results will take place in section 6.3 and interpretations about the features extracted by the models will be discussed in section 6.4.

6.1 PAN evaluation

Due to the nature of the PAN dataset, it can only be used to train text models. It was mainly used to ensure the validity of the implementation of the character-based model introduced in 4.1.2. Table 5 shows the results of evaluating both implementations of

Model	AVG	E	A	C	N	O
Baseline model	0.1700	**0.1599**	**0.1526**	0.1502	0.2313	0.1558
C2W2S4P in Thesis	0.1700	0.1600	0.1543	0.1540	0.2268	0.1550
C2W2S4P in [72]	**0.1655**	0.1665	0.1647	**0.1483**	**0.2059**	**0.1419**

Table 5: A comparison of the RMSE scores between the implementation of C2W2S4P in the thesis and in [72] and the baseline model of the PAN test dataset.

the C2W2S4P model (from [72] and this thesis) and the baseline model using the RMSE metric and the PAN test dataset

6.2 First Impressions evaluation

In this section, a comparison between the performance of the top four competitors of the First Impressions competition (NJU [83], evolgen [28], DCC [29], ucas [84]) and the models implemented in this thesis will be shown. The team ucas is using random forests to predict the OCEAN traits. Evlogen have used an external library called pyAudioAnalysis to extract audio features [85] and they implemented a 3-dimensional CNN to extract visual features. The audio and visual features are then joined and fed to a fully-connected network to predict OCEAN traits. DCC has used deep CNNs to extract visual and audio features from the input video to predict the OCEAN personality traits, the networks that extract the features are trained jointly. The NJU team has used several models:

- **Image modality**: They used VGG-16, ResNet and also their implementation of the image-based DAN and DAN+ models to predict OCEAN traits from a single image. All of these models' weights were initialized with pre-trained weights from the VGG-16 network or the ResNet network.

- **Audio modality**: NJU implemented a linear regression model that uses audio features as input, which were extracted using an external library called

python_speech_features[1]. The library extracts MFCC and logfbank features from the audio.

- **Multimodal**: A late fusion model of all the image models and the audio model was implemented. This is the model that NJU used to compete in the competition.

One year after the competition, NJU published a follow-up paper [27], where they changed a few hyperparameters that improved the predictions of the image models and they also implemented an end-to-end audio model, also based on the work in [75]. NJU also implemented a late fusion model using the improved image models and both audio models.

Also, the same competition was held one year later, but there was only one team able to surpass the baseline performance [86]. The team name is heysky and they have used CNNs to extract visual features from images and used an external tool to extract audio features called openSMILE [87]. Finally, these features are fed to a random forest to output the final personality predictions. This team did not report the AUC values of this model.

Tables 6 and 7 show the MAA and AUC of: the models trained using the First Impressions dataset, the top four competitors in the First impressions competition, the model from the NJU follow-up paper [27] and the model from heysky. The results from the paper that introduced the word-based model were not reported because they did not use the First Impressions test dataset to evaluate their results, but rather used the pre-defined validation set [1]. The tables are divided into several sections, where each section lists the models that use the same modalities as inputs. For the multimodal models, the name of the sub-models is written and next to it an indication if it is an early or late fusion.

[1] `https://github.com/jameslyons/pythonspeechfeatures`

Model	AVG	E	A	C	N	O
Baseline	0.8835	0.8806	0.8991	0.8739	0.8791	0.8847
Audio, images and text						
DAN+V,EtEA,C2W2S4P(E.F.)	0.9131	0.9141	0.9106	0.9193	0.9103	0.9109
DAN+V,EtEA,C2W2S4P(L.F.)	0.9080	0.9067	0.9101	0.9072	0.9062	0.9096
Audio and images						
NJU follow-up [27]	**0.9212**	–	–	–	–	–
heysky [86]	0.9173	–	–	–	–	–
DAN+V,EtEA(E.F.)	0.9146	0.9152	0.9127	0.9191	**0.9129**	0.9131
DAN+V,EtEA(L.F.)	0.9135	0.9137	0.9132	0.9145	0.9111	0.9151
NJU [83]	0.9130	0.9133	0.9126	0.9166	0.9100	0.9123
evolgen [28]	0.9121	0.9150	0.9119	0.9119	0.9099	0.9117
DCC [29]	0.9109	0.9107	0.9102	0.9138	0.9089	0.9111
ucas [84]	0.9098	0.9129	0.9091	0.9107	0.9064	0.9099
conv-3d,EtEA(E.F.)	0.9094	0.9081	0.9073	0.9120	0.9090	0.9109
conv-3d,EtEA(L.F.)	0.9065	0.9050	0.9080	0.9056	0.9054	0.9086
Text and Images						
DAN+V,C2W2S4P(E.F.)	0.9116	0.9116	0.9080	0.9179	0.9089	0.9116
DAN+V,C2W2S4P(L.F.)	0.9094	0.9079	0.9110	0.9123	0.9068	0.9095
Audio and images						
EtEA,C2W2S4P(E.F.)	0.8978	0.8957	0.9028	0.8921	0.8980	0.9003
EtEA,C2W2S4P(L.F.)	0.8956	0.8929	0.9027	0.8893	0.8947	0.8982
Images						
DAN+V	0.9154	**0.9169**	**0.9129**	**0.9219**	0.9109	**0.9146**
conv-3d	0.9029	0.9009	0.9024	0.9081	0.9007	0.9025
Audio						
EtEA	0.8974	0.8957	0.9034	0.8899	0.8975	0.9006
Text						
C2W2S4P	0.8880	0.8827	0.8987	0.8844	0.8858	0.8882
Word-based model	0.8840	0.8803	0.8946	0.8778	0.8807	0.8866

Table 6: The table shows the MAA scores of models predictions on the First Impressions test dataset. The value of MAA for each OCEAN trait and the average of the MAA scores of all the traits is reported. The table shows the scores of the top four teams in the First Impressions competition (NJU, evolgen, DCC, ucas) and the models implemented in the thesis. In this table the end-to-end audio model from subsection 4.2 is referred to as EtEA and the video-based model from 4.3.2 is referred to as conv-3d. E.F. and L.F. indicate if a multimodal model is built using early or late fusion.

Model	AVG	E	A	C	N	O
Baseline	0.5096	0.4988	0.5129	0.5161	0.5010	0.5193
Audio, images and text						
DAN+V,EtEA,C2W2S4P(E.F.)	0.8170	0.8260	0.7590	0.8703	0.8185	0.8110
DAN+V,EtEA,C2W2S4P(L.F.)	0.8354	0.8420	0.7900	0.8707	0.8407	0.8337
Audio and images						
DAN+V,EtEA(E.F.)	0.8317	0.8408	0.7788	0.8744	0.8392	0.8254
DAN+V,EtEA(L.F.)	**0.8407**	**0.8479**	**0.7912**	**0.8802**	**0.8448**	**0.8391**
ucas [84]	0.8277	0.8421	0.7767	0.8569	0.8338	0.8290
NJU [83]	0.8227	0.8391	0.7634	0.8696	0.8199	0.8217
evolgen [28]	0.8207	0.8376	0.7771	0.8492	0.8260	0.8135
DCC [29]	0.8111	0.8178	0.7528	0.8579	0.8131	0.8138
conv-3d,EtEA(E.F.)	0.8034	0.7943	0.7451	0.8344	0.8233	0.8201
conv-3d,EtEA(L.F.)	0.8081	0.8024	0.7567	0.8384	0.8201	0.8230
Text and images						
DAN+V,C2W2S4P(E.F.)	0.8154	0.8292	0.7460	0.8670	0.8130	0.8219
DAN+V,C2W2S4P(L.F.)	0.8323	0.8438	0.7803	0.8771	0.8338	0.8265
Audio and text						
EtEA,C2W2S4P(E.F.)	0.7408	0.7470	0.7033	0.7427	0.7612	0.7501
EtEA,C2W2S4P(L.F.)	0.7404	0.7423	0.7128	0.7349	0.7600	0.7518
Images						
DAN+V	0.8271	0.8424	0.7649	0.8795	0.8249	0.8239
conv-3d	0.7684	0.7621	0.7089	0.8169	0.7711	0.7831
Audio						
EtEA	0.7418	0.7493	0.7126	0.7248	0.7638	0.7586
Text						
C2W2S4P	0.6532	0.6301	0.6527	0.6748	0.6689	0.6400
Word-based model	0.5914	0.5823	0.5783	0.5981	0.6037	0.5947

Table 7: The table shows the AUC scores of models predictions on the First Impressions test dataset. The value of AUC for each OCEAN trait and the average of the AUC scores of all the traits is reported. The table shows the scores of the top four teams in the First Impressions competition (NJU, evolgen, DCC, ucas) and the models implemented in the thesis. In this table the end-to-end audio model from subsection 4.2 is referred to as EtEA and the video-based model from 4.3.2 is referred to as conv-3d. E.F. and L.F. indicate if a multimodal model is built using early or late fusion.

Model	MAA	AUC
Images		
DAN+V	**0.9154**	**0.8239**
DAN+(NJU) [83]	0.9111	–
DAN(NJU) [83]	0.9100	–
ResNet(NJU) [83]	0.9080	–
conv-3d	0.9029	0.7831
Audio		
End-to-End (thesis)	0.8974	0.7586
End-to-End (NJU) [27]	0.8950	–
Linear Reg. (NJU) [83]	0.8900	–
Text		
C2W2S4P	0.8880	0.6532
Word-based model	0.8840	0.5914

Table 8: A comparison between the MAA and AUC scores of the models that take only one modality as input using the First Impressions test dataset. The reported scores are the average over the five OCEAN traits.

Table 8 shows a comparison between all the models implemented in this thesis and models from the NJU team that use one modality as input, using the MAA and AUC metrics. For the models implemented by NJU, only the MAA is used in this comparison as they did not mention the AUC values in their work [27].

6.3 Discussion

In this section, the results presented in sections 6.1 and 6.2 will be discussed. Each subsection will interpret a different part of the results.

6.3.1 Baseline model

The MAA for the baseline model in Table 6 is relatively high for a model predicting a constant value. This leads to ambiguity because when a model has an MAA value of 88 for example, it is not clear if the model is predicting accurately or is it just

predicting a constant value. The high MAA score of the baseline model could be justified by the following facts:

- When using a constant predictor, the value that would minimize the MSE is the mean of the output labels. This could be proved by the differentiating the MSE function in terms of the constant prediction and solving for minimizing the error. While the value that would minimize the mean absolute error (MAE) and maximize the MAA, is the median of the output labels. The histograms from Figure 30 show that the mean and the median of the data are very close.

- The histograms from Figure 30 show that almost half of the labels are centered around the mean. Since MAA is a regression metric that punishes predictions that are far from the output labels, then just predicting the mean value will most of the cases be very close to the output label. For example, if a dataset labels range is [0.4,0.6], a model that predicts 0.5 constantly will have an MAA ≥ 0.9.

These facts justify the high MAA value for the baseline model, but they also show that, although the MAA is a regression metric, it is not enough to evaluate the performance of regression models for this dataset.

On the other hand, the AUC score of the baseline model in Table 7 is very low. This is due to the fact that AUC describes how good a model is at distinguishing between labels that belong to different categories. So the baseline is equivalent to a model that is doing random predictions.

Hence, using both metrics is a good evaluation of the model, as a high MAA will indicate that a model can predict OCEAN traits accurately and a high AUC will assure that the MAA value is high because the model can understand which examples have high or low personality traits and not just predicting a value close to the mean.

6.3.2 Comparison between single modalities

In this subsection, the results of models that use the same modality will be discussed. In addition to a discussion, that makes a comparison between the modalities and the information they hold with respect to personality.

Text

In table 8, the average of the MAA and AUC metrics over the OCEAN traits for the text-based and character-based models are shown. The table shows that the character-based model has better scores for both metrics, this is explainable considering the nature of the First Impressions dataset. The First Impressions dataset is collected from a social media platform where people post videos. In the context of social media it will be common for people to speak in an informal way or even mispronounce few words. In both cases, a word-based model could not utilize these words. As described in section 4.1.1, a word-based model represents words using pre-computed word embeddings, which will not have a representation for these words. On the other hand, the character-based model would represent words using an LSTM, which will produce a vector representing a word by processing it's characters as a sequence.

However, the predictive performance of the text models is not as good as the other two modalities. Not only are the MAA scores close to the baseline, but also the AUC scores of both models are less than the AUC scores of any other model. One way to explain this is that these models were trained using 15 seconds videos and this is a short time for a person to choose words that could reflect their personality.

For the PAN dataset, the implementation of the C2W2S4P model in this thesis has a close RMSE to the original implementation in [72]. However, the RMSE of both models is very close to the baseline, which indicates that both do not have good predictive performance. When examining the results of the PAN competition [13], it

is found that the competition had 22 competitors and only 10 were able to achieve better than the baseline. One of the justifications could be the unbalanced dataset distribution. As the histograms in Figure 31 show, all the traits do not have a good balance of examples below and above 0.5.

Audio

Concerning the audio modality, there are three models to compare: the end-to-end audio model from section 4.2, the linear regression model and the end-to-end audio model by NJU [27]. Table 8 shows the MAA scores of the three models. The table shows that the end-to-end audio model implemented in this thesis has a higher MAA, this could be justified for three reasons:

- Since the neural network's parameters are used to extract features from input audio, an end-to-end model can optimize it's parameters to extract the features that minimize the loss, opposed to a model that extracts a static set features that might not be the best to model the problem.

- These models extract features from the audio input, either using a neural network or an external library. Each one of the models extracts their set of features from several parts of the audio, for example, the end-to-end audio models use convolution filters to find features in various parts of the input. However, after the feature extraction, the end-to-end audio models process these features sequentially using LSTMs to map the temporal relation between these features.

- One difference between the end-to-end audio models, is that the one implemented in this thesis uses bi-directional LSTMs, so it can find more temporal relations between the extracted features.

The AUC and MAA scores of the end-to-end audio model are relatively higher than the scores of the text models, nevertheless lower than all of the image models. This indicates that images have the most information that reflects a person's personality.

Images

Table 8 shows the MAA scores of five models that use images as input: the three image models implemented by NJU [83] and the models from subsections 4.3.1 and 4.3.2.

The table shows that the conv-3d video-based model has the least MAA, this shows the effect of transfer learning because even though the conv-3d video-based model gets six images as input and the other models (three using only one image) the manage to have better accuracy because of transfer learning.

In Table 8, the implementation of the DAN+V model has the highest MAA across all models that predict OCEAN traits from one modality. This can be justified by:

- It is an image model and the overall results show that images have more features related to personality than text and audio.

- It combines two techniques that the rest of the image modality models used only one of them: transfer learning and it takes as input several images instead of one.

6.3.3 Comparison between multiple modalities

This section will discuss and compare the performances of multimodal models using Tables 6 and 7 that show the MAA and AUC scores of the models.

Subsection 6.3.2 has shown that not only the text models have the least MAA and AUC scores, but also that their scores are far from the audio and image models. These conclusions align with the results from Tables 6 and 7, as all the models that were trained jointly with the character-based model (which has better performance than the word-based) had worse results compared to when they were trained alone; with an exception for the end-to-end audio model that has a slightly better MAA.

Conversely, the end-to-end audio model has significantly improved the results of the conv-3d video-based model on both MAA and AUC, when the models were trained jointly. While slightly decreasing the MAA score of the DAN+V model, the end-to-end audio model has increased the AUC of the DAN+V model. This shows that the features from audio have information that could even help the DAN+V model (the model with the third highest MAA) to improve at classifying the OCEAN traits. An interpretation of this is, that jointly training audio and image models leads to co-dependence between the features and each model can focus on extracting the features that are more informative in it's modality. Since the best two MAA score are from models that are not end-to-end (the first uses ensemble of five models and the second extract audio features using an external library), the early fusion of DAN+V and the end-to-end audio model is the best end-to-end model with an MAA score of 0.9146.

When comparing between multimodal models that use early and late fusion techniques, a visible pattern is found. All of the early fusion models have better MAA, while the late fusion models have better AUC, making the late fusion between the DAN+V model and the end-to-end audio model, the model with the highest AUC of 0.8407. This shows that models that use late fusion may not be the most accurate at predicting the exact OCEAN traits values, but they are the best at distinguishing different personalities.

6.4 Interpretations

In this section, techniques that were used to interpret the features extracted by the neural networks will be shown. The techniques used to interpret features extracted from images are discussed in subsection 6.4.1 and those used to interpret audio features are in subsection 6.4.2.

6.4.1 Image features

Saliency heat maps are one of the techniques used to interpret the trained parameters of a CNN [88]. A saliency heat map is a heat map of an image that is input to a CNN, where the pixels that contribute more to the output of the CNN are highlighted. In the work by Yang et al. [88], a ResNet network is trained using the First Impressions dataset to predict OCEAN traits from images. They have also visualized saliency heat maps for their trained ResNet, which are shown in Figure 33.

The figure shows that for input images, the pixels with the most relevance for the network are around the face of the subject person in the image. This assures that the CNN uses facial features to compute the OCEAN traits of a person.

6.4.2 Audio features

PyAudioAnalysis is an open-source python library that can be used for various applications related to audio processing [85]. One application that is of interest to this thesis is audio features extraction, as this library could get as input an audio file and output interpretative audio features and representations. Table 9 shows the features that the library extracts. The library could also be used to extract these features from several parts of the audio input and then would output the mean and standard deviation of the extracted features in a 68-dimensional vector.

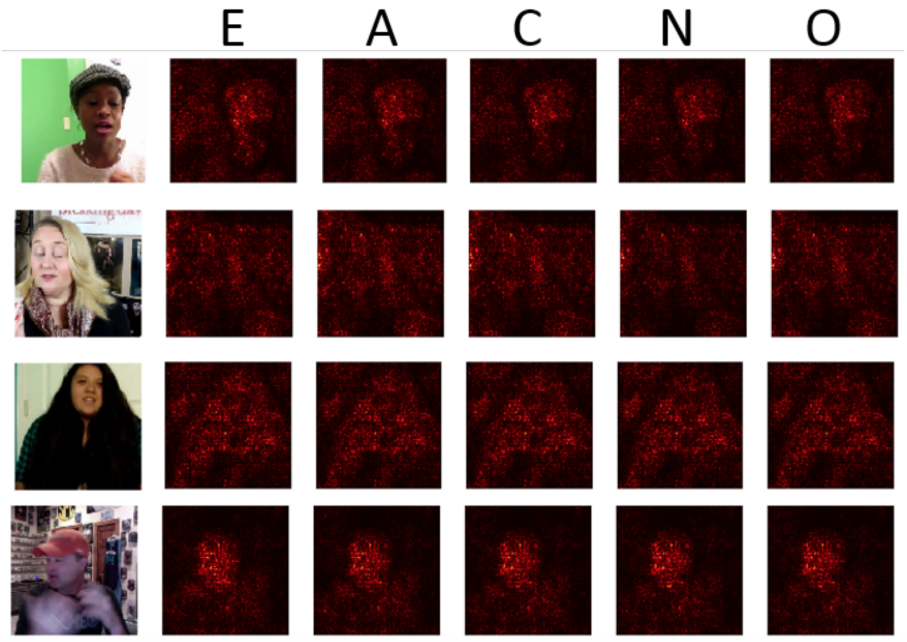


Figure 33: Saliency heat maps of a ResNet network that predicts OCEAN traits from images. Reprinted from [88].

In order to interpret the features extracted by the end-to-end audio model, pearson correlation coefficient is computed between every feature extracted by the end-to-end audio model and pyAudioAnalysis, from the First Impressions training dataset. These values has shown that out of the 256-dimensional vector extracted by the deep neural network there are four features that have a pearson correlation higher than 0.5 with the MFCC features, while five other features have a pearson correlation less than -0.5 with the MFCC features. According to [89], a pearson correlation coefficient more than 0.5 shows a moderate positive correlation and less than -0.5 is a moderate negative correlation.

Trait ID	Trait name	Description
1	Zero Crossing Rate	The rate of sign-changes of the signal during the duration of a particular frame.
2	Energy	The sum of squares of the signal values, normalized by the respective frame length.
3	Entropy of Energy	The entropy of sub-frames' normalized energies. It can be interpreted as a measure of abrupt changes.
4	Spectral Centroid	The center of gravity of the spectrum.
5	Spectral Spread	The second central moment of the spectrum.
6	Spectral Entropy	Entropy of the normalized spectral energies for a set of sub-frames.
7	Spectral Flux	The squared difference between the normalized magnitudes of the spectra of the two successive frames.
8	Spectral Rollof	The frequency below which 90% of the magnitude distribution of the spectrum is concentrated.
9-21	MFCCs	Mel Frequency Cepstral Coefficients form a cepstral representation where the frequency bands are not linear but distributed according to the mel-scale.
22-33	Chroma Vector	A 12-element representation of the spectral energy where the bins represent the 12 equal-tempered pitch classes of western-type music (semitone spacing).
34	Chroma Deviation	The standard deviation of the 12 chroma coefficients.

Table 9: Audio features extracted by the pyAudioAnalysis library.

7 Conclusion

In this thesis, an attempt at modeling the Automatic Personality Perception problem has been made. The personality model used as the underlying basis of this work modeled personalities as five apparent traits: Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism (and is thus referred to as OCEAN). In this work, deep neural networks were used to predict personality from three modalities: text, audio, and images.

Chapter 2 has shown other works that modeled the APP problem. These works used either only one modality to predict personality or a combination of multiple modalities. Chapter 2 also introduced two competitions wherein participants were asked to predict personality from text or video data.

The technical background of the algorithms used in this thesis was explained in chapter 3. The steps of the pipeline used have been illustrated and neural networks and their special variations that were used to model the APP problem have been demonstrated.

Chapter 4 is where the neural network models implemented in this thesis were explained. There were two models implemented to predict personality using text as input: a character-based and a word-based model. Regarding the audio modality, an end-to-end audio model was used to predict the OCEAN traits. To utilize images for modeling the APP problem, two CNNs were implemented, one of them uses transfer learning.

There were also models that used more than one modality to predict OCEAN traits. These models were implemented by fusing the models that use one modality either by jointly training them together or by doing an average of their predictions. The architecture of each model implemented in this thesis is shown and intuition of why each one is suitable to extract useful features based on the input modality is discussed in chapter 4.

The experimental setup for this thesis is shown in chapter 5. The specifications of the datasets used and their underlying distributions were discussed. Two datasets were used, namely PAN and First Impressions, most of the results were reported on the First Impressions dataset and they were evaluated using two metrics Mean Absolute Accuracy (MAA) and Area Under ROC curve (AUC).

Finally, chapter 6 shows the results of evaluating the trained models from chapter 4, using the metrics and datasets introduced in chapter 5. The results have shown that, regarding the models that use one modality to predict personalities, the models implemented in this thesis have the highest scores. For multimodal models, one of the models implemented in this thesis has the third best MAA score while surpassing all results of the models that participated in the first round of the First Impressions competition. Furthermore, this model has the best MAA score of the models that could be trained using only one neural network without using any other external libraries. This is beneficial from an industrial point of view, as it is easier to deploy and has fewer hyperparameters to tune.

The team that got the best AUC score in the First Impressions competition was the team ucas with a score of 0.8277. In this thesis, three of the implemented models have reached higher scores with a maximum of 0.8407, making this model the best AUC score that the author knows of for predicting the First Impressions test dataset.

A discussion also of the results has been made, which concluded that the best modality that could be used to predict personality is the image modality and the

best combination of modalities is image and audio, as the fusing of the audio model with the best image model has yielded better AUC score. Additionally, the results showed that early fusion techniques have better MAA scores, while late fusions yield better AUC scores. Furthermore, an interpretation of the audio and image features extracted by the models was made, concluding that the visual features are mainly facial features of the subject person and that some of the audio features have a good correlation with MFCC.

Future work

The future work for this thesis can be summarized in the following points:

- The MAA and AUC scores of the text models implemented in this thesis were not high. However, when examining the winners of the PAN competition, it is found that they did not use deep neural networks, but they used classical machine learning algorithms like SVMs and random forests. One way to interpret this is that the features extracted from short term text should be less complex than the ones extracted by neural networks, therefore using classic machine learning algorithms might be beneficial for predicting personality from text, as complex features are more vulnerable to overfitting

- Watson Personality Insights is a tool developed by IBM, that predicts peoples' personality from the text they wrote[1]. On the demo webpage of the tool, it is written that the input text to the tool "should contain words about every day experiences, thoughts, and responses" in order to predict the personality of the author [2], also the bigger the amount of text, the more accurate the results are. The examples from the demo pages are blocks of text that consist of at least

[1] `https://www.ibm.com/watson/services/personality-insights/`
[2] `https://personality-insights-demo.ng.bluemix.net/`

6,000 words. This shows that implementing text models that get longer and more personal input could yield better predictions.

- The NJU team has implemented the model with the best MAA score using an ensemble of five models. Although ensembling experiments were applied in this thesis, these experiments have used a maximum of three models due to the fact that for each modality, there was one model that has much better performance than the others. However, NJU implemented several models with good performance and applied ensemble. Applying ensemble this way with more diverse models could yield better results. Additionally, this can be investigated in early fusions

- The team that implemented the model with the 2^{nd} best MAA has used deep neural networks to extract visual and acoustic features and fed these features to a random forest to do the final prediction. This technique of using classical machine learning and deep learning together for the same task needs to be more explored.

- The correlation technique used in section 6.4.1 has shown that only a few of the features extracted by the deep end-to-end audio mode have similarities with MFCC. The same technique could be used with other audio features to get an idea about what are the rest of the features. For example pitch, loudness, or spectrograms.

Bibliography

[1] O. Kampman, E. J. Barezi, D. Bertero, and P. Fung, "Investigating audio, visual, and text fusion methods for end-to-end automatic personality prediction," *arXiv preprint arXiv:1805.00705*, 2018.

[2] S. Kedar and D. S. Bormane, "Automatic personality assessment: A systematic review," *2015 International Conference on Information Processing (ICIP)*, pp. 326–331, 2015.

[3] D. Zumstein and S. Hundertmark, "Chatbots–an interactive technology for personalized communication, transactions and services.," *IADIS International Journal on WWW/Internet*, vol. 15, no. 1, 2017.

[4] G. Mohammadi and A. Vinciarelli, "Automatic personality perception: Prediction of trait attribution based on prosodic features," *IEEE Transactions on Affective Computing*, vol. 3, no. 3, pp. 273–284, 2012.

[5] O. P. John, S. Srivastava, *et al.*, "The big five trait taxonomy: History, measurement, and theoretical perspectives," *Handbook of personality: Theory and research*, vol. 2, no. 1999, pp. 102–138, 1999.

[6] A. Vinciarelli and G. Mohammadi, "A survey of personality computing," *IEEE Transactions on Affective Computing*, vol. 5, no. 3, pp. 273–291, 2014.

[7] V. Ponce-López, B. Chen, M. Oliu, C. Corneanu, A. Clapés, I. Guyon, X. Baró, H. J. Escalante, and S. Escalera, "Chalearn lap 2016: First round challenge on first impressions-dataset and results," in *European Conference on Computer Vision*, pp. 400–418, Springer, 2016.

[8] M. R. Minar and J. Naher, "Recent advances in deep learning: An overview," *arXiv preprint arXiv:1807.08169*, 2018.

[9] T. Yarkoni, "Personality in 100,000 words: A large-scale analysis of personality and word use among bloggers," *Journal of research in personality*, vol. 44, no. 3, pp. 363–373, 2010.

[10] P. Dandannavar, S. Mangalwede, and P. Kulkarni, "Social media text-a source for personality prediction," in *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, pp. 62–65, IEEE, 2018.

[11] F. Alam, E. A. Stepanov, and G. Riccardi, "Personality traits recognition on social network-facebook," in *Seventh International AAAI Conference on Weblogs and Social Media*, 2013.

[12] T. Tandera, D. Suhartono, R. Wongso, Y. L. Prasetio, *et al.*, "Personality prediction system from facebook users," *Procedia computer science*, vol. 116, pp. 604–611, 2017.

[13] F. M. Rangel Pardo, F. Celli, P. Rosso, M. Potthast, B. Stein, and W. Daelemans, "Overview of the 3rd author profiling task at pan 2015," in *CLEF 2015 Evaluation Labs and Workshop Working Notes Papers*, pp. 1–8, 2015.

[14] M. A. Alvarez-Carmona, A. P. López-Monroy, M. Montes-y Gómez, L. Villasenor-Pineda, and H. Jair-Escalante, "Inaoe's participation at pan'15: Author profiling task," *Working Notes Papers of the CLEF*, 2015.

[15] L. M. Werlen, "Statistical learning methods for profiling analysis," in *Proceedings of CLEF*, 2015.

[16] O.-M. Sulea and D. Dichiu, "Automatic profiling of twitter users based on their tweets: Notebook for pan at clef 2015.," in *CLEF (Working Notes)*, 2015.

[17] G. Farnadi, G. Sitaraman, S. Sushmita, F. Celli, M. Kosinski, D. Stillwell, S. Davalos, M.-F. Moens, and M. De Cock, "Computational personality recognition in social media," *User modeling and user-adapted interaction*, vol. 26, no. 2-3, pp. 109–142, 2016.

[18] G. Cucurull, P. Rodríguez, V. O. Yazici, J. M. Gonfaus, F. X. Roca, and J. Gonzàlez, "Deep inference of personality traits by integrating image and word use in social networks," *arXiv preprint arXiv:1802.06757*, 2018.

[19] T. Polzehl, S. Moller, and F. Metze, "Automatically assessing personality from speech," in *2010 IEEE Fourth International Conference on Semantic Computing*, pp. 134–140, IEEE, 2010.

[20] M.-A. Carbonneau, E. Granger, Y. Attabi, and G. Gagnon, "Feature learning from spectrograms for assessment of personality traits," *IEEE Transactions on Affective Computing*, 2017.

[21] N. Al Moubayed, Y. Vazquez-Alvarez, A. McKay, and A. Vinciarelli, "Face-based automatic personality perception," in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 1153–1156, ACM, 2014.

[22] T. Fernando *et al.*, "Persons' personality traits recognition using machine learning algorithms and image processing techniques," *Advances in Computer Science: an International Journal*, vol. 5, no. 1, pp. 40–44, 2016.

[23] F. Celli, E. Bruni, and B. Lepri, "Automatic personality and interaction style recognition from facebook profile pictures," in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 1101–1104, ACM, 2014.

[24] J. Joshi, H. Gunes, and R. Goecke, "Automatic prediction of perceived traits using visual cues under varied situational context," in *2014 22nd International Conference on Pattern Recognition*, pp. 2855–2860, IEEE, 2014.

[25] F. Alam and G. Riccardi, "Predicting personality traits using multimodal information," in *Proceedings of the 2014 ACM multi media on workshop on computational personality recognition*, pp. 15–18, ACM, 2014.

[26] G. An and R. Levitan, "Lexical and acoustic deep learning model for personality recognition.," in *Interspeech*, pp. 1761–1765, 2018.

[27] X.-S. Wei, C.-L. Zhang, H. Zhang, and J. Wu, "Deep bimodal regression of apparent personality traits from short video sequences," *IEEE Transactions on Affective Computing*, vol. 9, no. 3, pp. 303–315, 2017.

[28] A. Subramaniam, V. Patel, A. Mishra, P. Balasubramanian, and A. Mittal, "Bi-modal first impressions recognition using temporally ordered deep audio and stochastic visual features," in *European Conference on Computer Vision*, pp. 337–348, Springer, 2016.

[29] Y. Güçlütürk, U. Güçlü, M. A. van Gerven, and R. van Lier, "Deep impression: Audiovisual deep residual networks for multimodal apparent personality trait recognition," in *European Conference on Computer Vision*, pp. 349–358, Springer, 2016.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[31] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[32] Y. M. Ngoma, *Analysis of Control Attainment in Endogenous Electroencephalogram Based Brain Computer Interfaces*. PhD thesis, Tshwane University of Technology, 2017.

[33] V. Kecman, *Support Vector Machines – An Introduction*, vol. 177, pp. 605–605. Springer, 05 2005.

[34] N. Horning, "Introduction to decision trees and random forests," *Am. Mus. Nat. Hist*, vol. 2, pp. 1–27, 2013.

[35] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[36] J. Hoffman, "Cramnet: Layer-wise deep neural network compression with knowledge transfer from a teacher network," *arXiv preprint arXiv:1904.05982*, 2019.

[37] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[38] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[39] L. Prechelt, "Early stopping-but when?," in *Neural Networks: Tricks of the trade*, pp. 55–69, Springer, 1998.

[40] D. E. Rumelhart, "Hinton, ge and williams, r," *Learning representations by backpropagation error. nature*, vol. 323, pp. 533–536, 1986.

[41] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[42] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[43] H. Ide and T. Kurita, "Improvement of learning for cnn with relu activation by sparse regularization," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2684–2691, IEEE, 2017.

[44] G. Roffo, "Ranking to learn and learning to rank: On the role of ranking in pattern recognition applications," *arXiv preprint arXiv:1706.05933*, 2017.

[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[46] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 8609–8613, IEEE, 2013.

[47] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," in *Proceedings of the IEEE international conference on neural networks*, vol. 1993, pp. 586–591, San Francisco, 1993.

[48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[49] A. Gupta, P. J. Harrison, H. Wieslander, N. Pielawski, K. Kartasalo, G. Partel, L. Solorzano, A. Suveer, A. H. Klemm, O. Spjuth, *et al.*, "Deep learning in image cytometry: a review," *Cytometry Part A*, vol. 95, no. 4, pp. 366–380, 2019.

[50] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[51] M. A. Nielsen, *Neural networks and deep learning*, vol. 25. Determination press San Francisco, CA, USA:, 2015.

[52] S. Wang, C. Tang, J. Sun, J. Yang, C. Huang, P. Phillips, and Y.-D. Zhang, "Multiple sclerosis identification by 14-layer convolutional neural network with batch normalization, dropout, and stochastic pooling," *Frontiers in Neuroscience*, vol. 12, p. 818, 11 2018.

[53] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[54] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[55] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[56] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[57] A. Laxman Katole, K. Prasad Yellapragada, A. Kumar Bedi, S. Singh Kalra, and S. C. Mynepalli, "Hierarchical deep learning architecture for 10k objects classification," *Computer Science Information Technology*, vol. 5, 09 2015.

[58] M. Cherti, *Deep generative neural networks for novelty generation: a foundational framework, metrics and experiments*. PhD thesis, 2018.

[59] D. E. Rumelhart, G. E. Hinton, J. L. McClelland, *et al.*, "A general framework for parallel distributed processing," *Parallel distributed processing: Explorations in the microstructure of cognition*, vol. 1, no. 45-76, p. 26, 1986.

[60] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3156–3164, 2015.

[61] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proceedings of the conference on empirical methods in natural language processing*, pp. 151–161, Association for Computational Linguistics, 2011.

[62] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[63] D. Zhang, H. Maei, X. Wang, and Y.-F. Wang, "Deep reinforcement learning for visual object tracking in videos," *arXiv preprint arXiv:1701.08936*, 2017.

[64] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[65] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[66] M. Fayyaz, M. H. Saffar, M. Sabokrou, M. Fathy, F. Huang, and R. Klette, "Stfcn: spatio-temporal fully convolutional neural network for semantic segmentation of street scenes," in *Asian Conference on Computer Vision*, pp. 493–509, Springer, 2016.

[67] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[68] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[69] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[70] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 746–751, 2013.

[71] E. Haddi, X. Liu, and Y. Shi, "The role of text pre-processing in sentiment analysis," *Procedia Computer Science*, vol. 17, pp. 26–32, 2013.

[72] F. Liu, J. Perez, and S. Nowson, "A language-independent and compositional model for personality trait recognition from short texts," *arXiv preprint arXiv:1610.04345*, 2016.

[73] R. A. Serway and C. Vuille, *College physics*. Cengage Learning, 2014.

[74] J. Burg, J. Romney, and E. Schwartz, *Digital Sound & Music: Concepts, Applications, and Science*. Franklin, Beedle & Associates, 2017.

[75] P. Tzirakis, J. Zhang, and B. W. Schuller, "End-to-end speech emotion recognition using deep neural networks," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5089–5093, IEEE, 2018.

[76] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[77] K. Liu, Y. Li, N. Xu, and P. Natarajan, "Learn to combine modalities in multimodal deep learning," *arXiv preprint arXiv:1805.11730*, 2018.

[78] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 10, pp. 993–1001, 1990.

[79] B. Chen, S. Escalera, I. Guyon, V. Ponce-López, N. Shah, and M. O. Simón, "Overcoming calibration problems in pattern labeling with pairwise ratings: application to personality traits," in *European Conference on Computer Vision*, pp. 419–432, Springer, 2016.

[80] F. M. Rangel Pardo, F. Celli, P. Rosso, M. Potthast, B. Stein, and W. Daelemans, "Pan workshop 2015." `https://pan.webis.de/clef15/pan15-web/index.html`.

[81] B. Rammstedt and O. P. John, "Measuring personality in one minute or less: A 10-item short version of the big five inventory in english and german," *Journal of research in Personality*, vol. 41, no. 1, pp. 203–212, 2007.

[82] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[83] C.-L. Zhang, H. Zhang, X.-S. Wei, and J. Wu, "Deep bimodal regression for apparent personality analysis," in *European Conference on Computer Vision*, pp. 311–324, Springer, 2016.

[84] B. Aydin, A. A. Kindiroglu, O. Aran, and L. Akarun, "Automatic personality prediction from audiovisual data using random forest regression," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 37–42, IEEE, 2016.

[85] T. Giannakopoulos, "pyaudioanalysis: An open-source python library for audio signal analysis," *PloS one*, vol. 10, no. 12, 2015.

[86] H. Kaya, F. Gurpinar, and A. Ali Salah, "Multi-modal score fusion and decision trees for explainable automatic job candidate screening from video cvs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1–9, 2017.

[87] F. Eyben, M. Wöllmer, and B. Schuller, "Opensmile: the munich versatile and fast open-source audio feature extractor," in *Proceedings of the 18th ACM international conference on Multimedia*, pp. 1459–1462, ACM, 2010.

[88] K. Yang and N. Glaser, "Prediction of personality first impressions with deep bimodal lstm," 2017.

[89] M. M. Mukaka, "A guide to appropriate use of correlation coefficient in medical research," *Malawi Medical Journal*, vol. 24, no. 3, pp. 69–71, 2012.