Classification of Autism Spectrum Disorder on Functional Magnetic Resonance Imaging Using Convolutional Neural Networks

Aketza Romaniega Bilbao

MU Bioinf. i Bioest.

Área de trabajo final

Nombre Tutor/a de TF

Romina Astrid Rebrij

Profesor/a responsable de la asignatura

Carles Ventura Royo

Fecha Entrega

Enero 2024

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Classification of Autism Spectrum Disorder on Functional Magnetic Resonance Imaging Using Convolutional Neural Networks.*
Nombre del autor:	*Aketza Romaniega Bilbao*
Nombre del consultor/a:	*Romina Astrid Rebrij*
Nombre del PRA:	*Carles Ventura Royo*
Fecha de entrega (mm/aaaa):	*01/2024*
Titulación o programa:	Máster universitario en Bioinformática y Bioestadística
Área del Trabajo Final:	*Bioinformática Estadística y Aprendizaje Automático*
Idioma del trabajo:	*Inglés*
Palabras clave	*Convolutional neural networks, autism, image classification*

Resumen del Trabajo

El trastorno del espectro autista es diagnosticado en el 2% de la población sin importar la raza, etnia y estatus socioeconómico. Aun así, se ha demostrado que esta neurodivergencia es diagnosticada 4 veces más en niños que en niñas. Un diagnóstico y tratamiento temprano muestran eficacia en la mejora de las habilidades lingüísticas y cognitivas de los niños que padecen autismo.

Las imágenes de resonancia magnética de cerebros y el uso de *Machine Learning* han sido ampliamente empleadas en la última década para el diagnóstico y detección de enfermedades relacionadas con el cerebro.

En este trabajo, se plantea emplear imágenes de resonancia magnética de cerebros con autismo y neurotípicos para construir modelos de redes neuronales convolucionales capaces de clasificar estas imágenes correctamente. Este proyecto se ha desarrollado empleando el lenguaje de programación Python y principalmente con las librerías TensorFlow, Keras y NumPy.

El conjunto de datos se ha obtenido de un proyecto derivado de la base de datos *Autism Brain Imaging Data Exchange* por Yang et al. De este modo, se

ha obtenido un modelo capaz de clasificar correctamente cerebros con autismo con una precisión del 68%.

Abstract

Autism spectrum disorder is diagnosed in 2% of the population regardless of race, ethnicity and socioeconomic status. Yet, it has been shown that this neurodivergence is diagnosed 4 times more often in boys than girls. Early diagnosis and treatment show efficacy in improving the language and cognitive skills of children with autism.

Magnetic resonance imaging of brains and Machine Learning have been widely employed in the last decade for the diagnosis and detection of brain-related diseases.

In this work, we propose to use magnetic resonance images of autistic and neurotypical brains to build convolutional neural network models capable of classifying these images correctly. This project has been developed using the programming language Python and mainly the libraries TensorFlow, Keras and NumPy.

The dataset has been obtained from a project derived from the Autism Brain Imaging Data Exchange database by Yang et al. Thus, a model capable of correctly classifying brains with autism with an accuracy of 68% has been obtained.

Index

List of figures

List of tables

1. Introduction

1.1. Context and justification of the work

1.1.1. General description

The aim of this work is to develop an algorithm based in Convolutional Neural Networks (CNN) that allows brain magnetic resonance image (MRI) processing and proper classification of autism spectrum disorder (ASD). For this purpose, different deep learning models will be developed, trained and evaluated. In order to show the applicability of the algorithm a webpage will be elaborated, where after the user uploads of an MRI image a prediction will be made whether ASD was detected or not.

1.1.2. Context

The autism spectrum disorder is diagnosed in about 2% of population regardless of race, ethnicity and socioeconomic status [1]. However, this illness usually involves boys 4 times more frequently than girls during the first decade of life [2]. It has been shown the efficacy of early detection and treatment in children's language and cognitive abilities [3]. Clark et al. concluded that early diagnosed children were more likely to attend mainstream school due to early treatment compared to those diagnosed after 3 years [4]. According to the American Psychiatric Association's 5th edition of the Diagnostic and Statistical Manual of Mental Disorders, in order to be diagnosed with ASD the following five points must be fulfilled: persistent deficits in social communication and interactions, restricted and repetitive patterns of behaviour, seen in early development, deficits that impair one's functioning in society, and exclusion of other disorders [5].

1.2. Objectives

1.2.1. General objectives

The general objective of this project is to develop an algorithm that is able to correctly classify ASD from brain MRIs and to build a webpage in order to use the algorithm more user-friendly.

1.2.2. Specific objectives

- To correctly classify ASD in brain MRIs.
- To improve the accuracy of the algorithm compared to what has been developed previously.
- To develop a user-friendly webpage for the algorithm, that allows users to upload an MRI image and that shows the prediction.

1.3. Ethical and global behaviour

The Sustainable Development Goals (SDG) are challenges proposed by the United Nations [6]. Their objectives are based on poverty, inequality, climate, environmental degradation, prosperity, peace and justice. This thesis supports the following SDG project objectives:

- **SDG3: Good health and well-being.** This work tries to improve the quality and accuracy of ASD classification through the use of MRIs. As previously shown, early diagnose and treatment improves the quality of life in children.
- **SDG5: Gender equality.** Relevant studies in the field of the thesis have been and will be referenced in this paper regardless of the author's gender identity. Even though the ratio of MRIs used in this work is almost 6:1 (male:female) and is close to that reported van't Hof et al. (4:1), it has been claimed that there is evidence of increased camouflaging in autistic females, which could contribute to delay in the recognition of difficulties and provision of support, including early diagnosis [7]. Therefore, there is still much work to be done in terms of diagnosis of ASD in women.
- **SDG10: Reduced inequalities.** This work is intended to aid in the detection of autism and therefore also aid in the intervention and support of the affected people. In this way, these people could have more equal opportunities to face life.

1.4. Approach and method to be followed

MRI has been widely employed for the detection of different brain related illnesses, such as: ASD, Alzheimer and Parkinson [8-9]. Deep learning has been widely used for this purpose; however, the classification performance is difficult to compare across studies [10]. There are four main types of approaches to MRI usage in CNN: 2D slice-level, 3D patch-level, ROI-based and 3D subject level.

For this study, the dataset released by Yang et al. will be used [11]. This dataset derives from the Autism Brain Imaging Data Exchange (ABIDE) where subjects are classified into controls (healthy) and patients (suffering from ASD). From 1025 MRIs, 537 correspond to healthy brains and 488 to ASD brains. As reported by the authors, there are 873 male and 152 female participants. The dataset also includes 259 teenagers and 96 children. Among these MRIs, 312 have been obtained with the patient having their eyes closed, which has impact on which areas of the brain are activated [12].

The data is presented in Figure 1, where different 2D-slices of the same MRI 3D scan are shown.

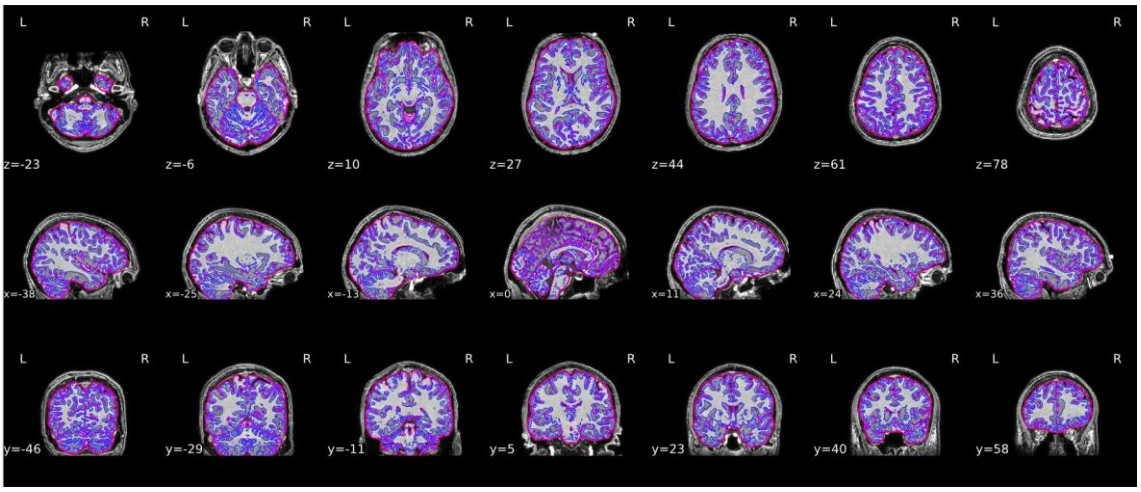


Figure 1. Data obtained from Yang and coauthors. Source: self-made.

The next methodology will be followed:

- Data acquisition:

As mentioned beforehand, data was obtained from Yang et al.'s work; which is derived from the ABIDE dataset.

- Data exploration:

Deshpande et al. determine that the motor cortex and the mid cingulate cortex are the most reproducible resting-state functional brain networks to separate autism and control groups [13]. Therefore, to pursue the objective images at coordinate z=0 are needed from every MRI. These areas on interest are shown in Figure 2.

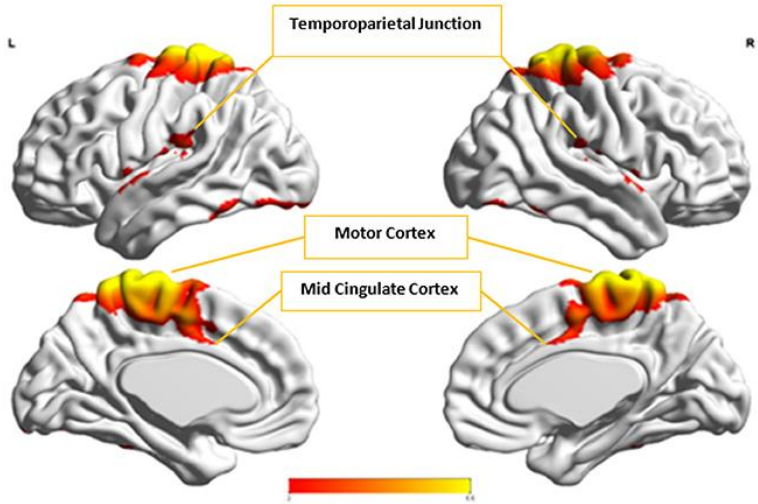


Figure 2. Areas of brain of interest to study ASD. Source: Deshpande et al. [13].

- Data preprocessing:

In order to get the desired 2D slice, images will be cropped employing either Pillow or OpenCV in Python. As RGB channel values are in between 0 and 255, they must be standardized to values between 0 and 1.

- Model building and training:

In order to build our model, a state of art will be studied to investigate what has been done for image classification of patients with ASD. Therefore, different approaches and layers will be considered to build the model. To train the model, 80% of the images will be randomly selected.

- Evaluation of the model:

Once the model is trained, we will proceed to evaluate it with the 20% rest of images.

- Model improvement:

Different parameters, such as layers, will be changed in order to improve the accuracy of the model.

- Evaluation of the improved model:

Once better parameters have been found, where the accuracy is higher, the model will be evaluated with the 20% rest of the images.

1.5. Planning

1.5.1. Tasks

Task 1: “Develop an algorithm based on CNN that classifies brain MRIs and is capable of differentiating healthy brains and those with ASD”.

- Obtain the dataset.
- Preprocess the data.
- Design the architecture of the model.

Task 2: “Obtain a higher accuracy than those reported in the literature”.

- Train and evaluate the first model.
- Change the architecture of the model.
- Evaluate the new model.

Task 3: “Develop an interactive webpage that allows data import for classification and that shows the results”

- Learn to develop a web using Flask.
- Develop the web.
- Test if it works correctly

1.5.2. Calendar

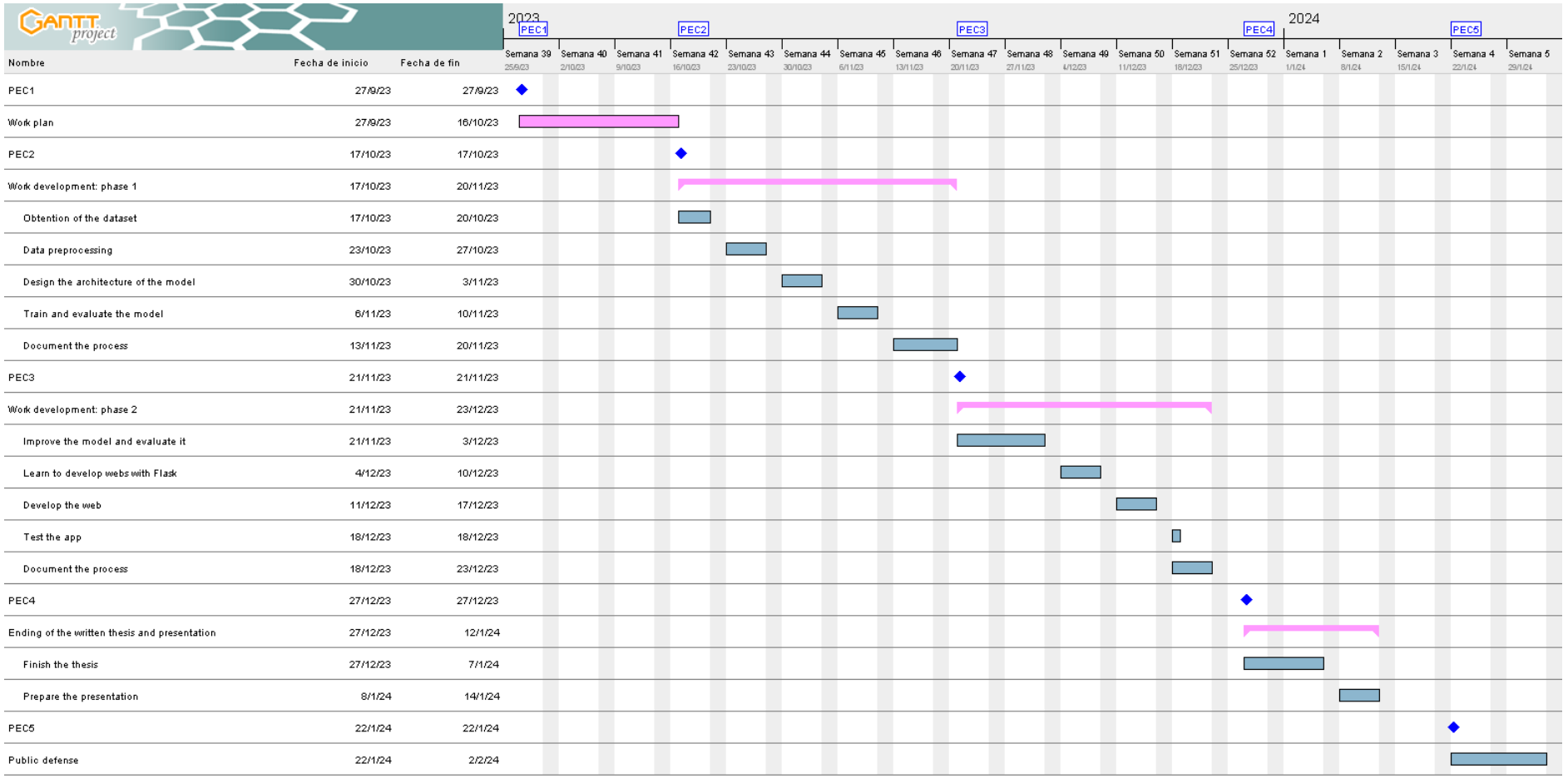


Figure 3. Grantt Diagram: calendar for the completion of the project. Source: self-made.

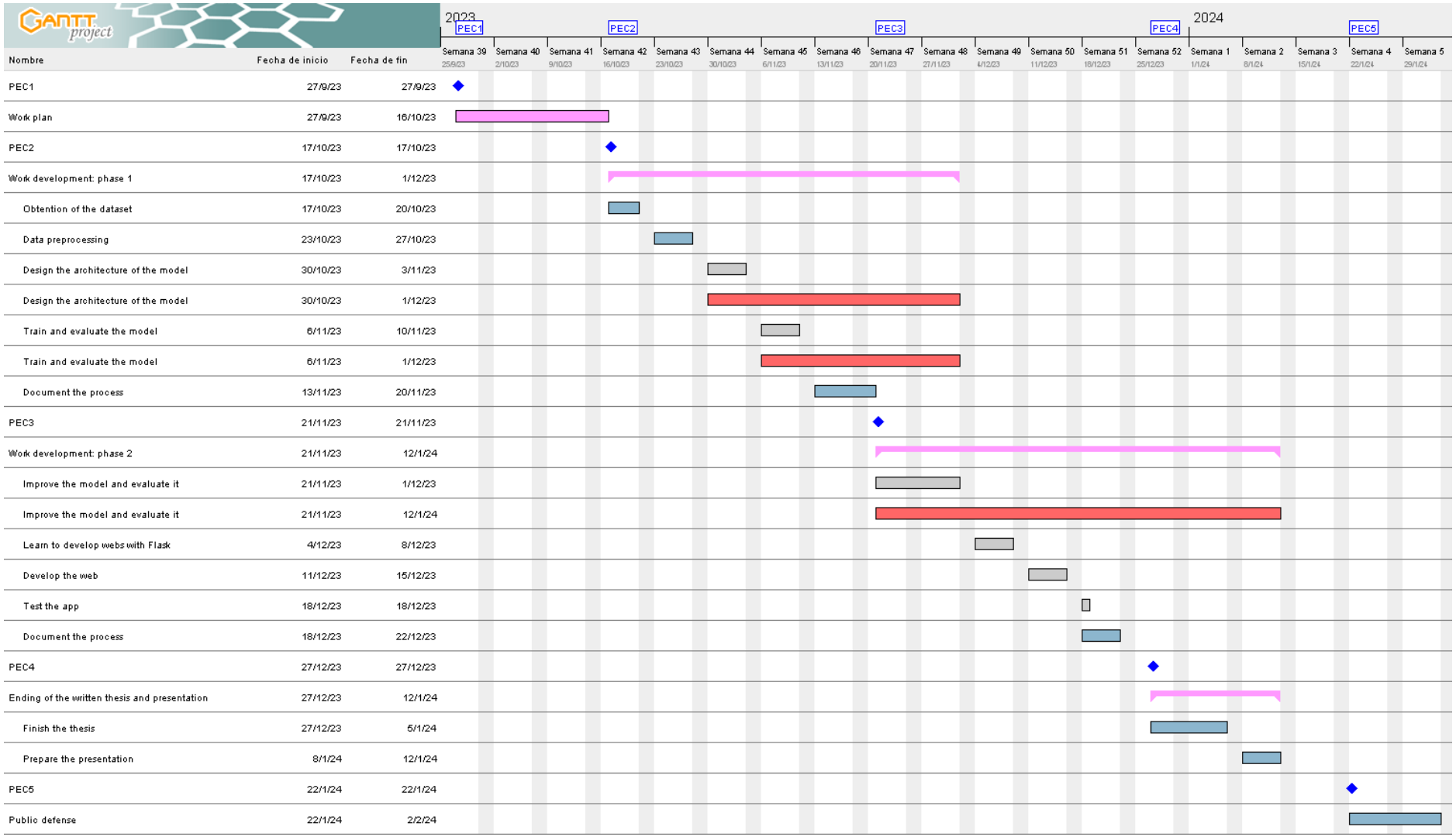


Figure 4. Grantt Diagram: updated calendar for the completion of the project. Source: self-made.

1.5.3. Milestones

1st milestone: PEC1. Delivery of the work plan (16/10/23).

2nd milestone: Get the data ready to start the project (27/10/23).

3rd milestone: Get the initial model for the algorithm developed (10/11/23).

4th milestone: PEC2. Delivery of the progress done (20/11/23).

5th milestone: Improve the algorithm (03/12/23).

6th milestone: Develop the web (18/12/23).

7th milestone: PEC3. Delivery of the progress done (23/12/23).

8th milestone: PEC4. Handle the thesis and presentation (14/01/24).

9th milestone: PEC5. Public defence of the thesis (22/01/24).

1.5.4. Risk analysis

The following risks may be encountered during the performance of the project:

- Problems with data extraction and data preprocessing: data can be found to be downloaded as Yang et al. reported. The data is divided in 14 fractioned zip files that weight 527 GB in total. The data must be extracted from the files, organized and the region of interest must be cropped.
- Problems with hardware capacity: as working with images requires high-performance hardware, there could be some limitations.
- First time developing an app: as a first time developing an app in Python, there is a risk of the final product not being delivered.
- Time to develop all the milestones: due to time limitations, there is a risk that all the milestones might not be completed. However, an attempt will be made to keep to the proposed schedule.

1.6. Expected results

- Thesis. The whole process will be documented on a written thesis, from the planification of the project to the delivery of the final product. The state

of the art and an introduction will be included to give context and justification to the project.

- GitHub repository. All the code that has been developed for the project will be found here.
- App. An app will be delivered that allows the user to upload an image, so that a prediction can be made, whether the MRI shows ASD. This app will be hosted in PythonAnywhere.

1.7. Brief description of the chapters

- State of art. Literature published to date that follows the same line of research will be reviewed and researched. In addition, an introduction to key concepts that have been used during the development of the work will be made.
- Methodology. During this chapter we will proceed to explain in detail how the project has been developed, from the working environments used to the specifications of the algorithms.
- Results. The results obtained following the procedures previously explained in the methodology section will be shown and explained.
- Conclusion. The most relevant results obtained in the project will be commented and the conclusions will be presented. In addition, the variations in the project plan will be explained and future lines of work will be indicated.

2. State of art

2.1. Published work

In 2018, Parison et al. presented an algorithm based in Graph Convolutional Networks that involves the representation of the populations as a sparse graph, where its nodes are associated with imaging-based feature vectors and the phenotypic information is integrated as edge weights [14]. To train the algorithm, they selected 403 autistic and 468 neurotypical MRI data form the ABIDE database, obtaining a 70.4% accuracy for the classification of autism spectrum disorder.

In the same year, Heinsfeld et al. studied the regions of interest (ROI) *via* functional connectivity to differentiate subjects with autism and healthy [15]. They included 505 autistic individuals and 530 typical controls. This way, they were able to extract the anti-correlated (underconnected) areas and the highly correlated (connected) areas in autistic brains, successfully classifying them with a 70% accuracy.

A year later, Kong. et al. constructed an individual brain network as feature representation for each data to later perform the classification using Deep Neural Networks [16]. This way, using only 78 and 104 MRI images of autistic and neurotypical people, they obtained an accuracy of 90.4%.

In 2020, Shrivastava et al. followed a ROI based study, obtaining a symmetric matrix containing the Pearson correlation in between regions-of-interest [17]. They obtained a 76% accuracy using 1035 images, divided in 505 autistic and 530 neurotypical. Sherkatghanad et al. followed a similar approach as a ROI based study [18]. They compared the performance of the CNN to other machine learning algorithms like Support Vector Machines, K-Nearest Neighbours and Random Forest. By building a parallel architecture of seven pairs of 2D convolutional and max pooling layers and then inputting sequentially to a dense layer with a dropout of 0.25, they were able to obtain a 70% accuracy. Another approach was taken by Thomas et al. by summarizing the temporal dimension of the resting state functional MRI data and training a three-dimensional convolutional neural network, obtaining a 66% of accuracy [19].

The following year, the methods proposed by Leming et al. showed an accuracy of 69.7% when classifying by only structural similarity, 67.7% when classifying by only functional connectivity and 66.4% accuracy when classifying by univariate grey matter volumes [20]. The combination of structural similarity and functional connectivity yielded in a 69.4% accuracy. By combining the convolutional neural network and individual structural covariance network, Gao et al. proposed a gradient-weighted class activation mapping to characterize the weight of the features contributing to the classification [21]. Thus, obtaining a 71.8% accuracy across different sites. Moreover, they found out that the discriminative features were found to be mainly located in the prefrontal cortex and cerebellum.

2.2. Theoretical background

2.2.1. Machine Learning

Machine Learning is the set of methods and algorithms that allow a machine to learn automatically based on past experience. Normally, a machine learning algorithm must be trained using a training dataset. During this phase, the algorithm compares the predicted output with the idyllic one in order to adjust the model and increase the accuracy. There are two types of learning processes: supervised and non-supervised. Supervised learning requires an external component that compares the outcome of the model and the expected one and provides the model with feedback in order to adjust. On the other hand, in non-supervised learning, the algorithm learns about the input data itself by discovering and grouping patterns, features, correlations…

Depending on the objective of our analysis, we can differentiate three different tasks: classification, regression and clustering. In this work, we will focus on classification algorithms. A classification task, as the name implies, is about assigning labels to a set of classes. In this way, a correctly labelled data set will be used for model building. A common classification task is usually binary classification, a problem where you have a data set belonging to two classes.

2.2.2. Deep Learning

The first working Artificial Intelligence (AI) program was written in 1951 to run on the Ferranti Mark 1 machine of the University of Manchester to play checkers and chess [22]. It was not until almost 25 years later that the Deep Blue chess machine from IBM defeated the then world chess champion Garry Kasparov [23]. In just another 25 years since that event, artificial intelligence has grown and expanded in our everyday life. From deep learning models like AlexNet, to image analysis programs like Google Lens, to a tool that shakes the current educational system like ChatGPT.

Deep learning belongs inside the concept of machine learning, where the aim is to create algorithms that can learn by themselves complex and abstract concepts. Starting from simple concepts, these algorithms can define hierarchically more complex ones. In this hierarchy of concept layers lies the concept of deep learning.

One of the most common deep learning algorithms are neural networks. Artificial neural networks (ANN) are inspired by the communication mechanism of the biological neuron. Even though the concept of ANNs has been around since the 1950s, it was not until the early 2000s that their utility was shown.

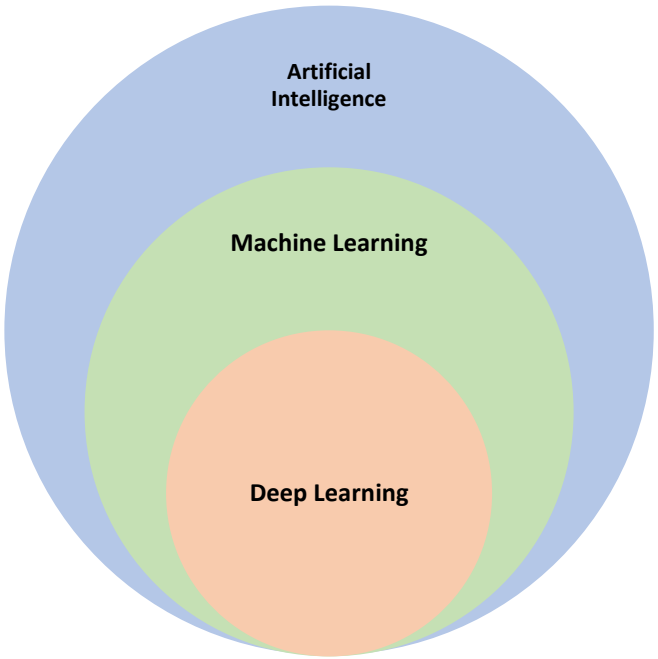


Figure 5. Venn diagram showing the hierarchy of AI, ML and DL. Source: self-made.

2.2.3. Convolutional Neural Networks

Convolutional neural networks (CNN) are a special type of neural networks for processing gridded data, such as images. These algorithms are used in our everyday life. For example, one of these applications is shown in self-driving cars, where the algorithm is capable of detecting cars and people in order to avoid accidents. Facial recognition is also a usual task for these kinds of algorithms when we unlock our phones or even with TikTok and Instagram filters. Another example is Google Lens, which is a software that allows us to recognize an item and it will show us online references regarding that object.

However, these processes have a huge computational cost. Imagine if we had and image with 100-pixel height and a 100-pixel wide. If the image has colour, the input shape would be 100 x 100 x 3 = 30.000 pixel, due to the three RGB colour channels. The bigger the image, the higher the computational cost will be.

2.2.3.1. Convolutional layers

Convolutional layers are the core building block of CNN, consisting on an input data, a filter and a kernel size. Neurons on convolutional layers are connected to pixels in their respective receptive fields. This way, the second convolutional layer will only be connected to neurons located in their respective receptive fields of the first layers. This kind of architecture allows the network to concentrate on small low-level features in the first layer, assembling them into larger higher-lever features in the following ones.

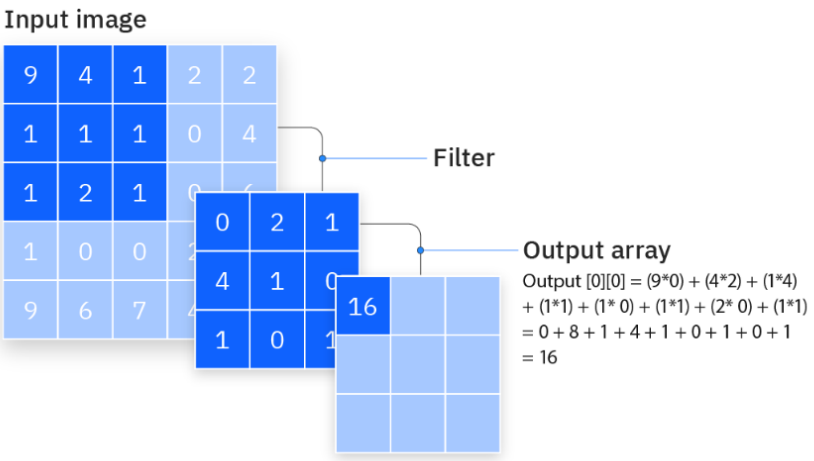


Figure 6. Example of a convolutional layer with a filter and a kernel size. Source: IBM [24].

A neuron's weights can be represented as a small image the size of their filters. A layer full of neurons using the same filter will output a feature map, which highlights the areas in an image that activate the filter the most.

2.2.3.2. Pooling layers

Pooling layers, also known as *downsampling* layers, reduce the dimension of the input in order to reduce computational load, memory usage and number of parameters (also helping with the risk of overfitting). Similar to the convolution layer, a filter is swept across the input without any weights. In this case, the kernel applies an aggregation function to the values within the receptive field. There are two main types of pooling:

- Max pooling: it selects the pixel with the maximum value withing the receptive field to send to the output array.
- Average pooling: it calculates the average of the receptive field to send to the output array.

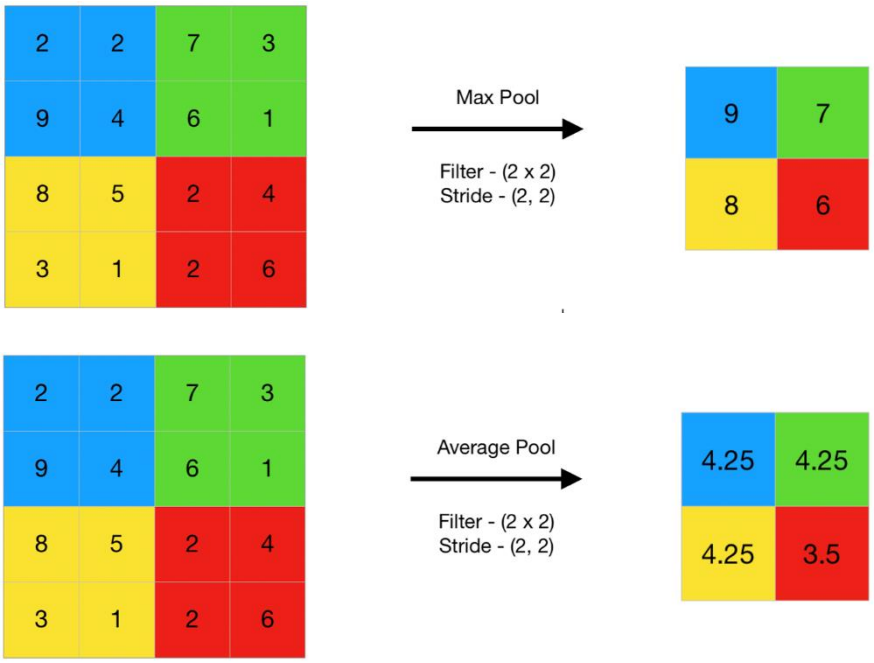


Figure 7. Illustrated example of max pooling and average pooling. Source: Geeksforgeeks [25].

2.2.3.3. ReLU activation

Each convolutional layer will have an activation function. One of these activation functions is the Rectified Linear Units (ReLU) function [26]. The purpose of this function is to introduce nonlinearity to the system. The ReLU function applies the following function to all values: $f(x) = \max(0, x)$, or:

$$f(x) = \begin{cases} 0 \; for \; x < 0 \\ x \; for \; x \geq 0 \end{cases}$$

Basically, the activation function ReLU changes negative values to 0, increasing this way the nonlinear properties of the model.

2.2.3.4. Softmax activation

The softmax activation function is usually used in the output layer as it calculates the relative probabilities of each class. The following function is being used to calculate the probabilities:

$$softmax(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_i)}$$

where x_i represents the value *i* from the total *j* neurons of the output layer [27]. The exponential acts as a nonlinear function.

2.2.3.5. Sigmoid activation

Sigmoid activation function is commonly used in the output layer when the problem requires to solve a binary classification [28]. It is usually denoted by σ(x) and expressed in the following day:

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

As it ranges from 0 to 1, it guarantees that the output of this layer will always be in that margin. As it is a non-linear function, the output would also be a non-linear function of the weighted sum of inputs.

2.2.3.6. Dropout layers

Dropout layers are one of the most popular regularization techniques [29]. Even though it is a simple method, it usually yields in outstanding results. At each training stage there is a *p* probability for each neuron of being “dropped out”, meaning that it will be ignored during that training stage but it may be active during the next step. The hyperparameter *p* is called the dropout rate and can take a value from 0 to 1. However, it is typically set between 10% and 50%. After the training is done, neuros don’t get dropped anymore.

2.2.3.7. Batch normalization

Batch normalization layers automatically standardize the inputs to the next layer. This process allows the researchers to use much higher learning rates and also works as a regularizer, sometimes even eliminating the need for a dropout layer. Szegedy et al. employed this technique in the architecture of their deep neural networks to train models with saturating nonlinearities, allowing them to obtain results with the same accuracy as state-of-art image classification models with 14 times fewer training steps [30].

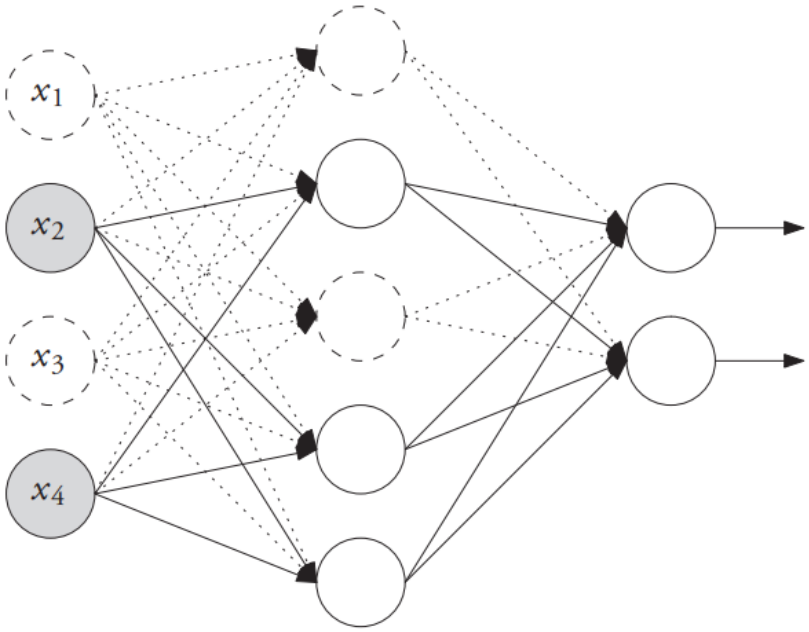


Figure 8. Illustration of how dropout works. Neurons shown in dashed line are deactivated. Source: Bosch et al. [27].

2.2.3.8. Transfer learning

As explained by Keras themselves "Transfer learning consists of taking features learned on one problem, and leveraging them on a new, similar problem" [31]. This process allows us to re-use the model weights from a pre-trained model that has been employed for a similar task. These models are usually pre-trained on the ImageNet dataset, which is a large dataset containing over 1.4 million images and 1000 classes. In Keras Applications different deep learning models can be found, which are available to be used with their pre-trained weights [32]. In this site, models like XCeption by Google or VGG and ResNet. Transfer learning is usually performed for tasks where the dataset has too little data to train. Moreover, transfer learning also increases the learning speed of the algorithm; with less new things to learn, the algorithm can generate high-quality output quicker.

3. Methodology

3.1. Dataset

The Autism Brain Imaging Data Exchange has gathered functional and structural brain imaging data collected from more than 24 different laboratories to accelerate the understanding of the neural bases of ASD [33]. In 2022, Yang et al. released a functional brain network collection to the public at a large scale [11], containing 1025 pre-processed rs-fMRI. Out of the 1025 MRIs, 488 correspond to patients (autistic), whereas 537 correspond to controls (neurotypical). As reported by the authors, there are 873 male and 152 female participants, containing 259 teenagers and 96 children.

3.2. Work environment

This project has been developed in a personal computer with the following hardware and software specifications:

- Processor: Intel® Core™ i5-8400 CPU @2.80 GHz.
- RAM: 16.0 GB (3000 MHz).
- GPU: GeForce GTX 1070 Ti (8 GB).
- System type: 64-bit operating system.
- Operating system: Windows 10 Home, 22H2 version.

3.3. Programming language

Python 3.11.2 has been used to develop this project due to the vast availability of libraries for Machine Learning and image processing.

3.4. Programming environment

A virtual environment was created with the *venv* module, which allows to create lightweight virtual environments, each with their own independent set of Python packages installed in their site directories [34]. All the used libraries were installed in the virtual environment. The script was written using Visual Studio Code (VS Code) software 1.84.2 in a ".py" file. Test runs were run with VS Code; however,

time- and capacity-demanding tasks were run directly through the command prompt.

3.5. Used libraries

The following libraries and functions were used:

- aspose.words was used to convert ".svg" data to ".png" data.
- OpenCV was used to import the data.
- matplotlib was used to make 2D and 3D graphs.
- NumPy was used to deal with the numeric data.
- os was used to iterate through the paths.
- TensorFlow was used to build the model, train it and evaluate it.

3.6. Data import

The data was obtained in 14 fractioned zip files with a total weight of 527. Once the data was decompressed, 1025 folders, one for each patient, was found. In every folder, four more folders where found: "anat", "figures", "func" and "log". Inside the folder "figures" the data was found as "sub_*[label][number]*_dseg.svg".

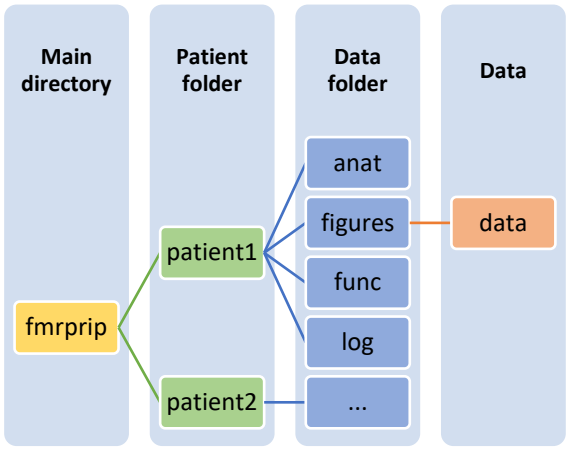


Figure 9. Illustration of the paths. Source: self-made.

In order to get the directories of all the images *os.walk()* function from *os* library was used. This way a variable named *paths,* containing the paths of each image that would end with "dseg.svg", and another variable called *file_names* containing

the names of the files were stored. The length of these variables was 1024, meaning that out of 1025 subjects, only one of them did not have the desired file.

3.7. Exploratory analysis of the data

Each data is presented as a Scalable Vector Graphics (SVG) file type containing 2D slices of the same 3D MRI. These files contain 21 T1-weighted images with coloured contours delineating the detected brain mask and brain tissue segmentations.

A manual check was done through every image, noticing that those with index 638, 634, 628, 627, 625, 624, 623, 80 and 70 did not have the same template as the rest. All but the abovementioned data, had the desired image in the centre. Comparing the data to a 3 row and 7 column matrix, the desired data would be at position (2, 4). However, in the abovementioned images, this slice was found at position (1, 4).

3.8. Data preprocessing

In order to input our data to the algorithm, we need a tensor containing the information of each pixel in each RGB channel. However, our data comes as a ".svg" file containing vectors. With the aim of converting these files to PNG files, *aspose.words* library was employed, obtaining PNG images with a size of (353, 831, 3) corresponding to height, width and number of channels respectively. These images were imported in colour with the function *cv2.imread()* from the *cv2* library in colour with *cv2.IMREAD_COLOR*.

Images were then cropped *via* a for loop with the *enumerate()* function. Those different images were cropped at (15:115, 364:464,) positions, whereas the rest at (130:230, 364:464,) positions leading to (100, 100, 3) sized data. The data was then converted to a NumPy array and normalized diving by 255 as pixel values range from 0 to 255. This data was then stored as a ".npy" filetype to make the process faster and less expensive.

3.9. Labels

The dataset is divided into two categories: patient (autistic) and control (neurotypical). Therefore, a NumPy array containing zeros with a length of 1024 was created with *np.zeros()*. “0” was assigned to control patients, whereas the array was filled with “1” whenever the file name contained the “patient” word. Afterwards, the labels were one-hot encoded.

3.10. Training and testing datasets

The original dataset was split in two sub-datasets: one for training the algorithm and another one to evaluate it. We allocated 67% of the data for training and the rest for validation. The same process was followed for the labels.

3.11. CNN architecture building

First, a basic model was built with a 2D convolution layer with 16 filters, a kernel size of (2, 2), *relu* as activaton function and with padding as *same*. Afterwards a 2D max pooling layer was added. Finally, the data was flattened and the outcome was presented in a dense layer with *softmax* activaton and *categorical crossentropy* function as loss function.

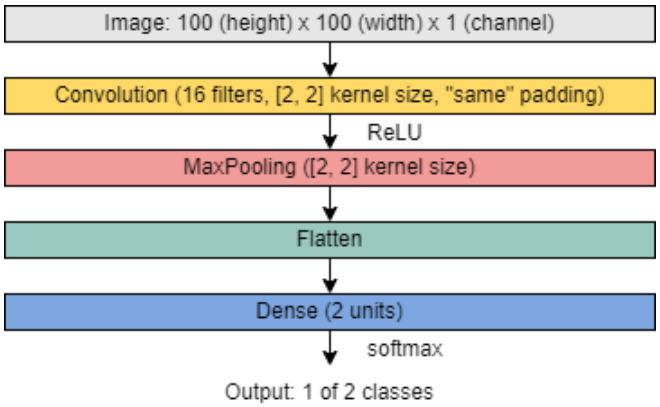


Figure 10. Architecture of the simple model with grayscale images. Source: self-made.

In order to accomplish this, we used the *Sequential()* function from *Keras* library that stacks layers where each one has exactly one input tensor and one output tensor. Layers were then added with the method *add()* to the sequential model. A

2D convolutional layer was used as an input with *Conv2D()*. 16 filters, a (2 x 2) kernel size and “same” padding were chosen as hyperparameters. ReLU function was also chosen as activation function. Aftwerwards, we added a MaxPooling layer as a downsampling layer, with a kernel size of (2 x 2) with *MaxPooling()*. Next, we introduced a flatting layer with the function *Flatten()*. Finally, the output layer was added as a Dense layer with 2 units with *Dense()* and softmax activation function.

Once the architecture was chosen, we proceeded to compile the model with *compile()*. We chose *rmsprop* as optimizer and the categorical crossentropy as loss function.

3.12. Training and validation of the model

The model was then trained with the *fit()* function with the training dataset and their respective labels. A batch size of 50 images was set, with 10 epochs and a validation split of 10%.

Afterwards the accuracy of the model was evaluated with the *evaluate()* function, using this time the testing dataset and their labels.

3.13. Improvement of the model

In order to improve the model, different approaches were taken. These approaches include changing the architecture in order to get a better and more complex learning, using RGB data to include three channels and therefore more information, tunning the hyperparameters of the layers and data augmentation.

Architecture change

In order to improve the previous model, the architecture of the model was changed. A pair of layers consisting on a 2D convolutional layer and a MaxPooling layer were added. The convolutional layers had a kernel size of (2 x 2) and the padding set as “same”. With each addition of layer, the number of filters would increase in a 2^{4+n} fashion, being n the number of layers added and n = 0, …, 4. This way, the first layer had 16 filters, whereas the second one had 32 and so on.

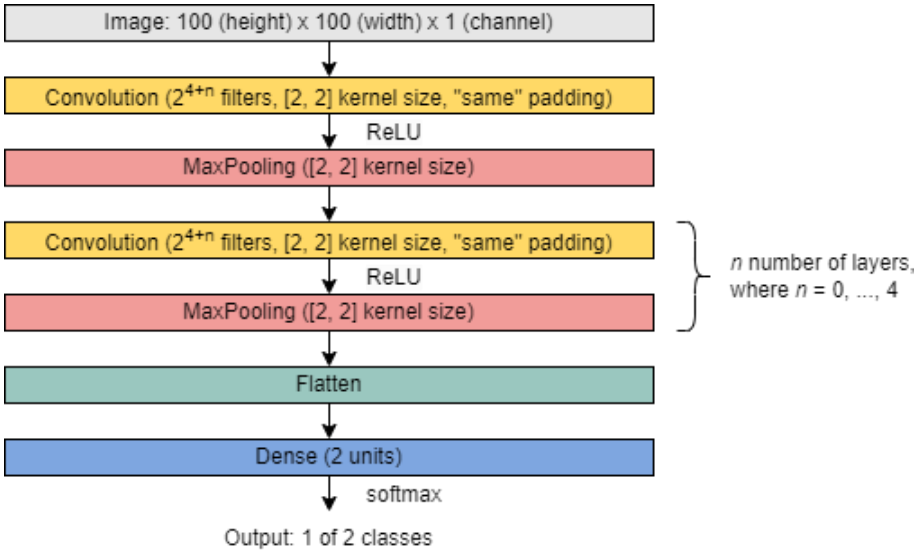


Figure 11. Architectures of the more complex models with grayscale images. Source: self-made.

RGB data

Instead of using the data in greyscale, we decided to input the data as RGB; therefore, it would contain three times more information. For this purpose, a simple model containing a 2D convolutional layer as input with 16 filters, (2 x 2) kernel size and padding as "same" with ReLU activation followed by a MaxPooling layer with a (2 x 2) kernel size. The data was then flattened and the output was obtained with a dense layer with 2 units and softmax activation.

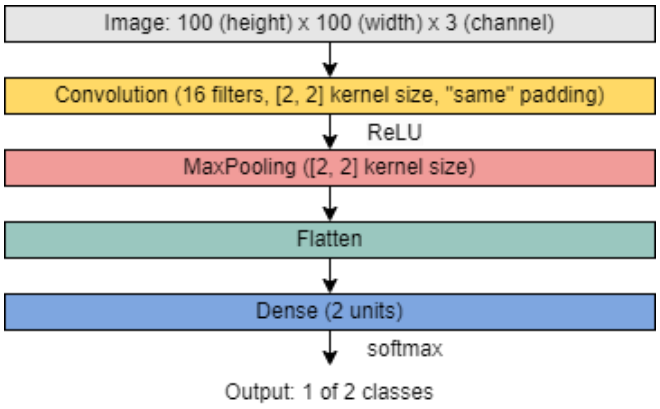


Figure 12. Architecture of the simple model with RGB images. Source: self-made.

As an attempt to increase the learning capability of the algorithm a more complex architecture was built consisting of four pairs of 2D convolutional and MaxPooling

layers. The filter size increased in a 2^{4+n} manner in each layer added. However, the rest of the hyperparameters stayed constant with a (2 x 2) kernel size for both layer types and padding set as “same” in the convolutional layers.

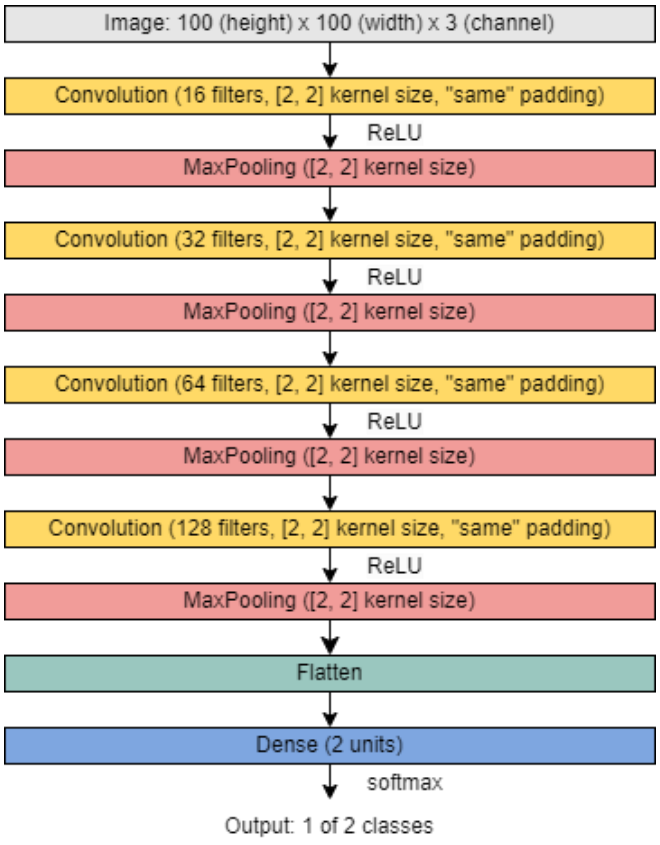


Figure 13. Architecture of the more complex model with RGB images. Source: self-made.

Tunning of hyperparameters

In order to get a better idea of which role hyperparameters play in this scenario, the following parameters were set to optimize: number of layers, number of filters, size of the kernel, dropout rate and addition of dense layers before the output layer.

Hence, we started evaluating the impact that the three first mentioned parameters would have in performing this task. We proposed the combination of number of layers that would vary from 1 to 5 (therefore, $i = 0, \ldots, 4$), number of filters from 16 to 256 and kernel size ranging from (2 x 2) up to (6 x 6). The combination of these variables with different values would yield a total of 125 models.

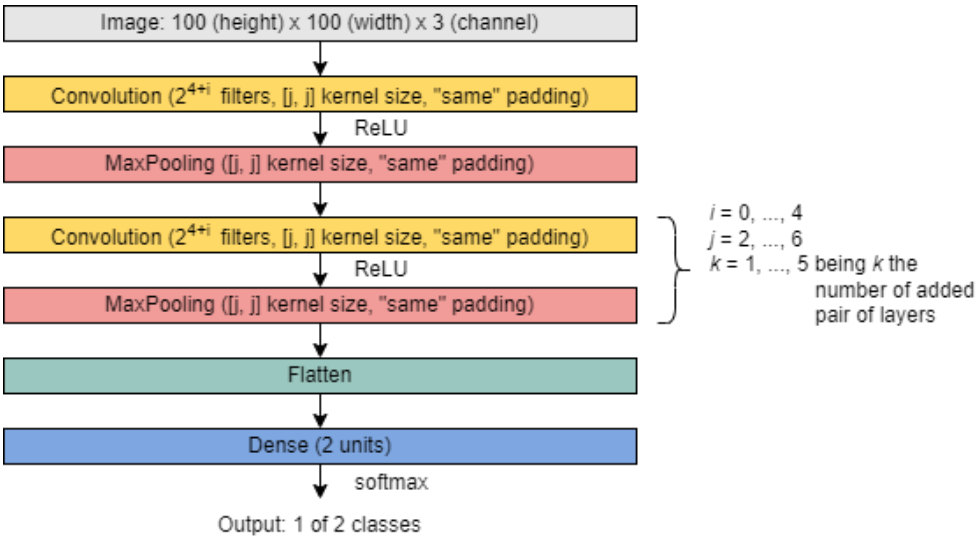


Figure 14. Architecture of the hyperparameter tunning for layers, filters and kernel sizes. Source: self-made.

Afterwards, the kernel size was maintained constant for all 2D convolutional and MaxPooling layers. Therefore, we introduced a dropout layer with the *Dropout()* function. This hyperparameter ranged from 0.1 up to 0.5 with a 0.1 step. The combination of the three hyperparameters yielded 125 models.

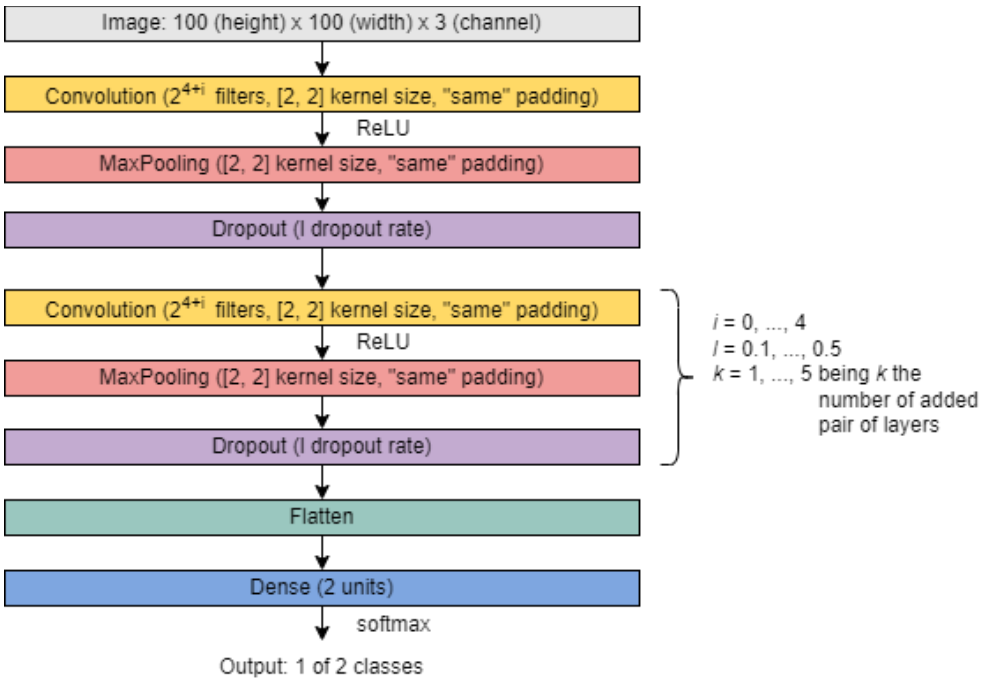


Figure 15. Architecture of the hyperparameter tunning for layers, filters and dropout rates. Source: self-made.

Finally, a dense layer was added in between the flattening and output layers. The units of this layer were set from 16 to 256. At the same time, number of layers and number of filters were also explored, the combination leading to 125 models.

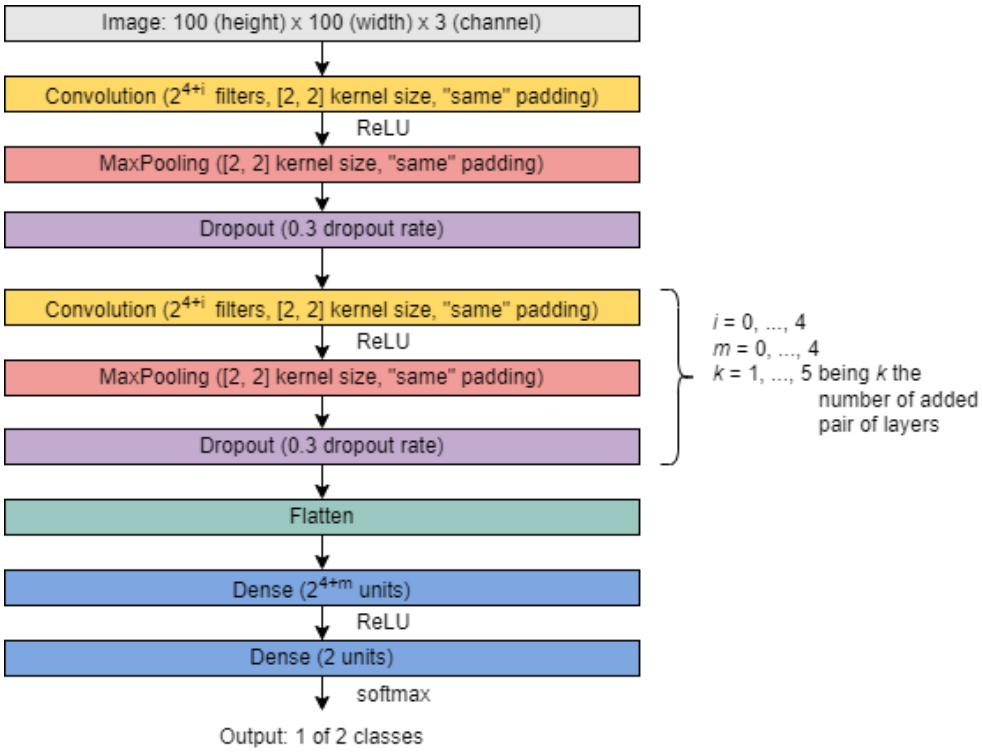


Figure 16. Architecture of the hyperparameter tunning for layers, filters and units. Source: self-made.

Data augmentation

As the title implies, data augmentation relies in increasing the amount of data by performing small transformations in the dataset [33]. One of the most commonly used image augmentation approaches is geometric transformation. By randomly flipping, cropping rotating, stretching and zooming the original images, new images are created to be used in our tasks.

In order to accomplish this, two folders were created containing the autistic data in RGB as “patient” and the neurotypical data as “control”. To access these folders the method *flow_from_directory()* was employed. Using the function *ImageDataGenerator()*, the following transformations were performed in our data: a shear range of 0.2, which performs random cutting transformation; a zoom range of 0.2, where the image is zoomed in or out in a 20% range; and a horizontal flip, which randomly flips half of the images horizontally.

Change of area of interest

Literature shows by region-of-interest-based volumetry that adults with ASD have reduced corpus-callosum, whereas surface-based morphometry studies show increased cortical thickness in the parietal lobes [36].

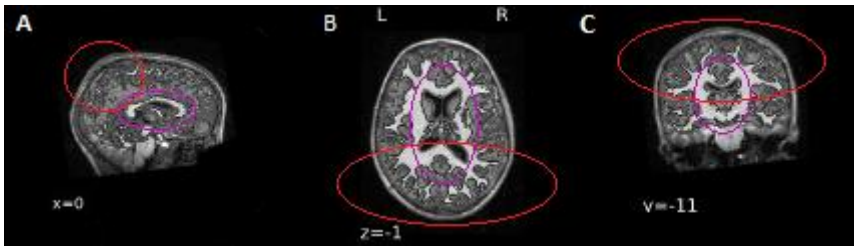


Figure 17. Areas of interest to be studied. A) Already used slice in the algorithm training, B) slice where the corpus-callosum can be better seen and C) slice where the parietal lobe can be observed (corpus-callosum is highlighted in purple, whereas the parietal lobes are shown in red). Source: self-made.

Batch normalization

With the idea of taking a new approach a batch normalization layer was added in the architecture after every MaxPooling layer. Hence, 3 added pair of 2DConv and MaxPooling layers were added with kernel-sizes of (3 x 3) and 32 filters.

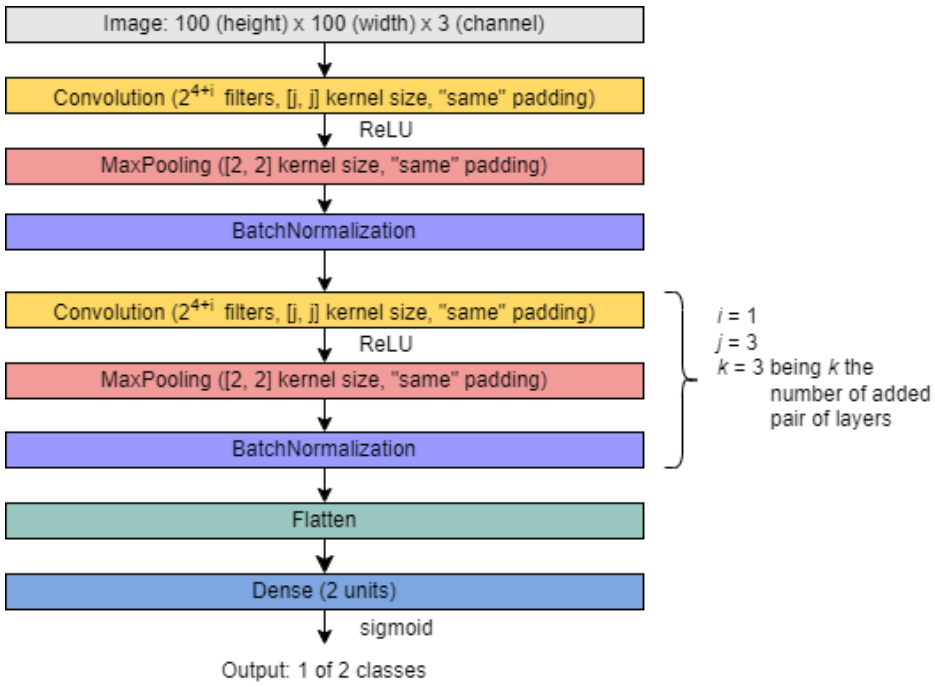


Figure 18. Architecture of the model employing batch normalization layers. Source: self-made.

Moreover, a similar approach was also followed, where the BatchNormalization layer would be followed by a Dropout layer with a 0.3 dropout rate.

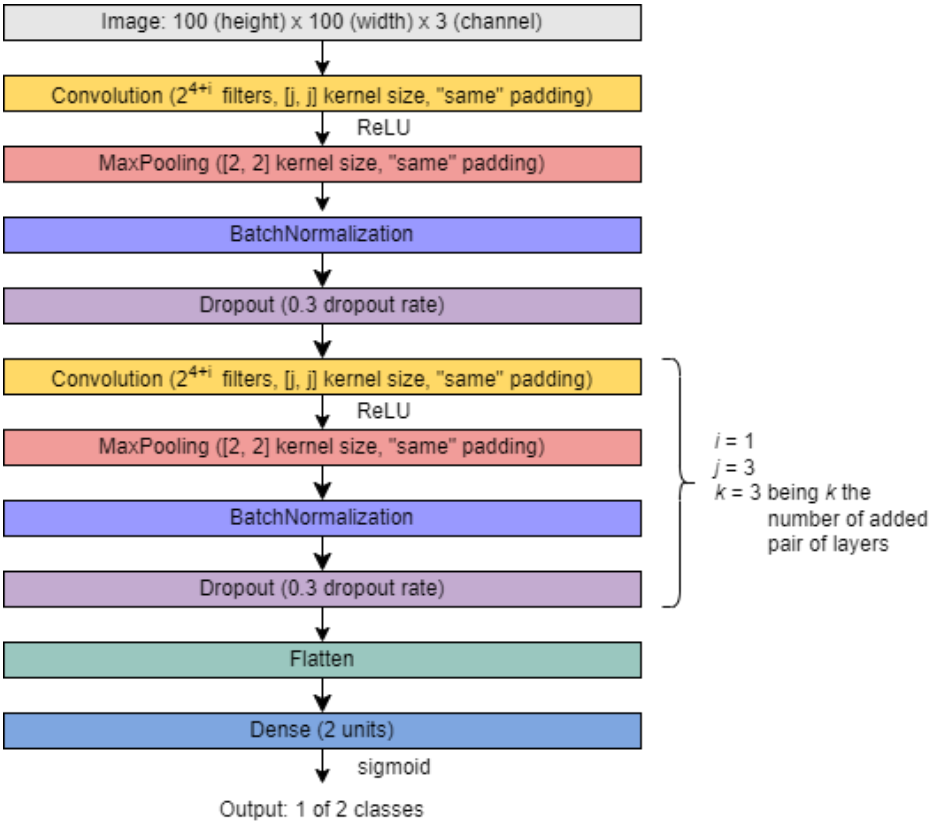


Figure 19. Architecture of the model employing batch normalization and dropout layers. Source: self-made.

Transfer learning

Among the available pre-trained models at Keras Applications, VGG16 model was selected. Hence, the model was imported from *keras.applications* and used with *VGG16()* function. The input shape was then selected to match out images (100 x 100 x 3) and *include_top* was set to "false" in order to not include the 3 fully-connected layers at the top of the network. It was specified that there were two classes (*classes = 2*) and that the weights were set to "ImageNet" to download those pre-trained with that databank.

Afterwards, the layers arguments for the pretrained model *trainable* were set to "false" so that they would not be trained. Finally, in a sequential manner, a *Flatten()* layer and a dense layer with 1 unit and sigmoid activation were added as output. The model was then compiled using *rmsprop* optimizer, with *BinaryCrossentropy()*

as loss function and *binary_accuracy* as metric. A callback function was also added in order to stop the process after no improvement in the loss for 10 epochs, starting from epoch 10. It was fitted with 100 epochs and a batch size of 256, with a validation split of 0.2. Finally, the model was evaluated with new data.

4. Results

In this section we will show the training and evaluation accuracy and losses for each model in order to explore whether the model is learning or not. Furthermore, the viability of the algorithm will also be discussed

4.1. Greyscale data containing models

The first made model containing a simple architecture was trained, observing overfitting, high validation loss and a low validation accuracy.

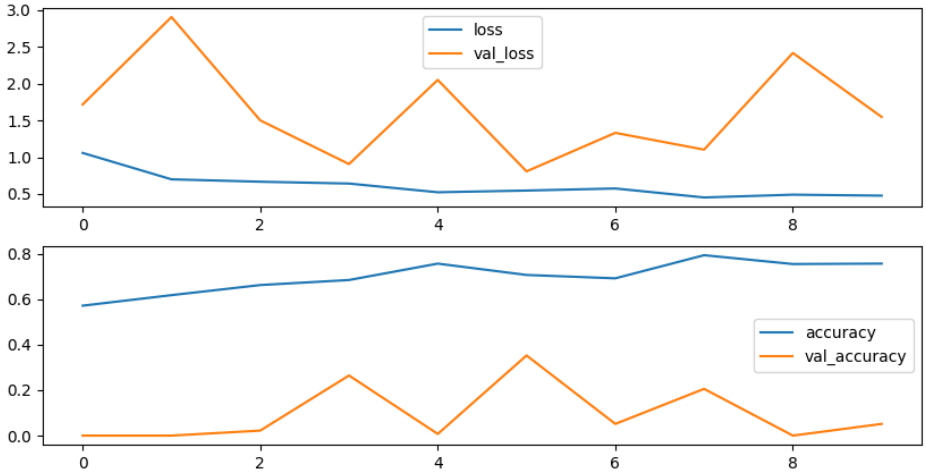
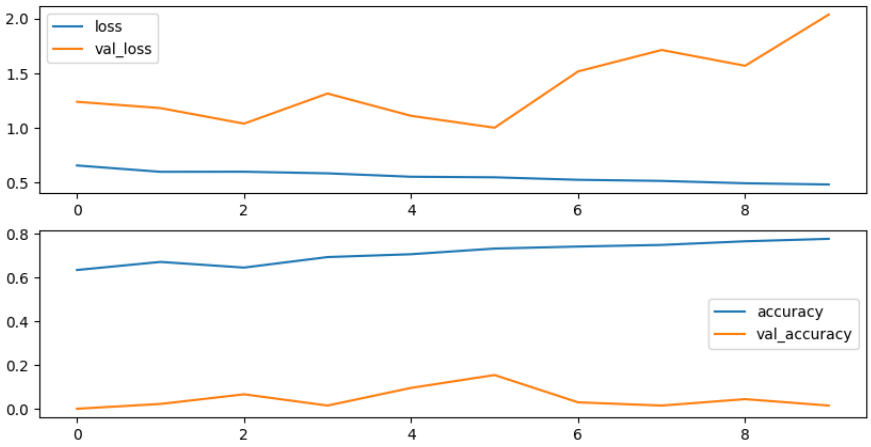


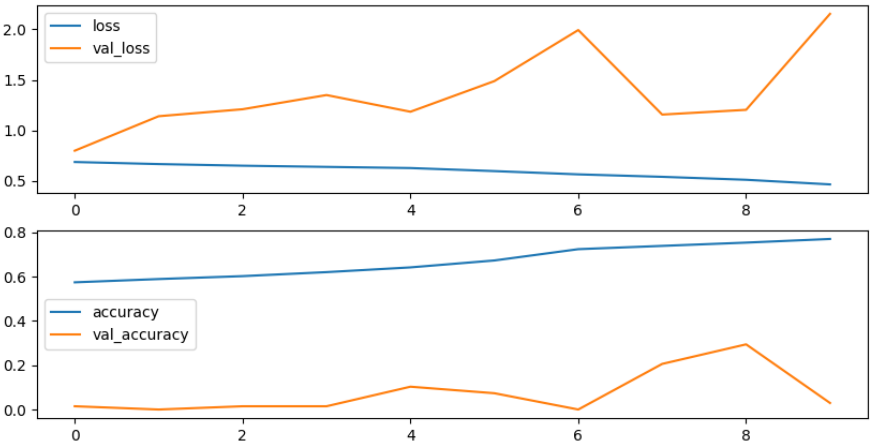
Figure 20. Training results from the simple greyscale data containing model. Source: self-made.

Therefore, 2D convolutional and MaxPooling layers were added pairwise, obtaining more complex architectures. The obtained training and evaluation losses and accuracies are shown in the following graphs:

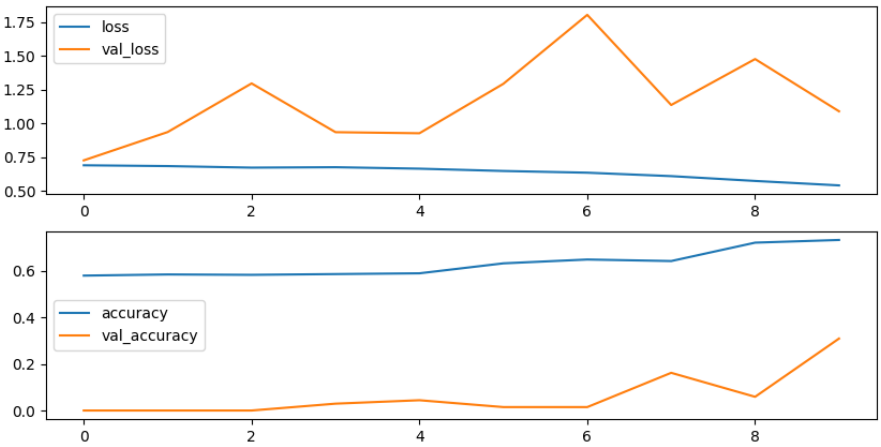
1


loss
val_loss
accuracy
val_accuracy

2


loss
val_loss
accuracy
val_accuracy

3


loss
val_loss
accuracy
val_accuracy

4

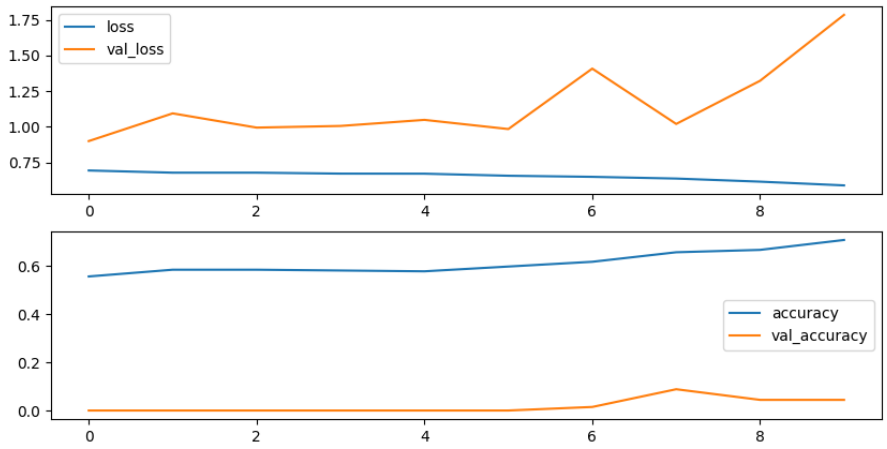


Figure 21. Training results from the more complex greyscale data containing models. Source: self-made.

As it can be observed in the graphs, neither of the architectures was able to yield a validation accuracy above 20%. The learning process was halted and overfitted. Furthermore, the loss was really high.

4.2. RGB data containing models

With all this information, we decided to import the original RGB data in order to input more information and check whether the algorithm would be capable of learning the task.

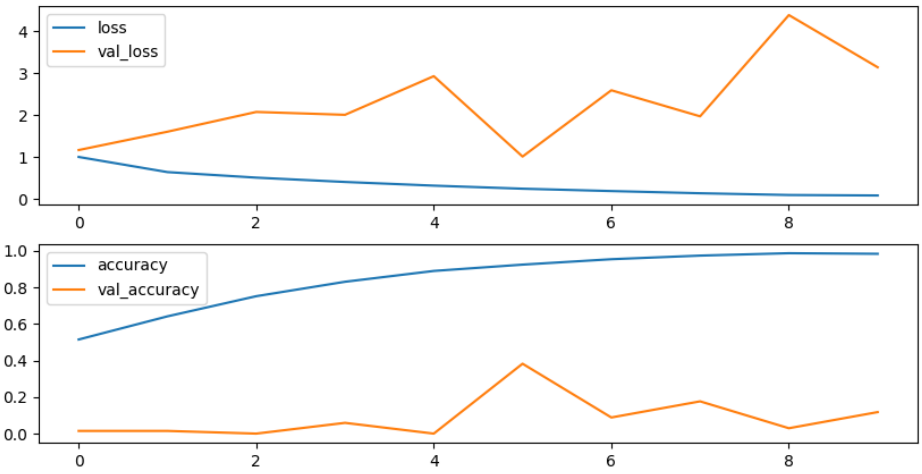


Figure 22. Training results from the simple RGB data containing model. Source: self-made.

Employing a simple architecture, containing only a 2D convolutional and a MaxPooling layer, lead to a better learning curve. However, the algorithm was still not able to perform the task correctly.

Therefore, four pairs of 2D convolutional and MaxPooling layers were added to the architecture. The deeper the layer, the higher the number of filters was in a 2^{4+n}, being *n* the number of layers.

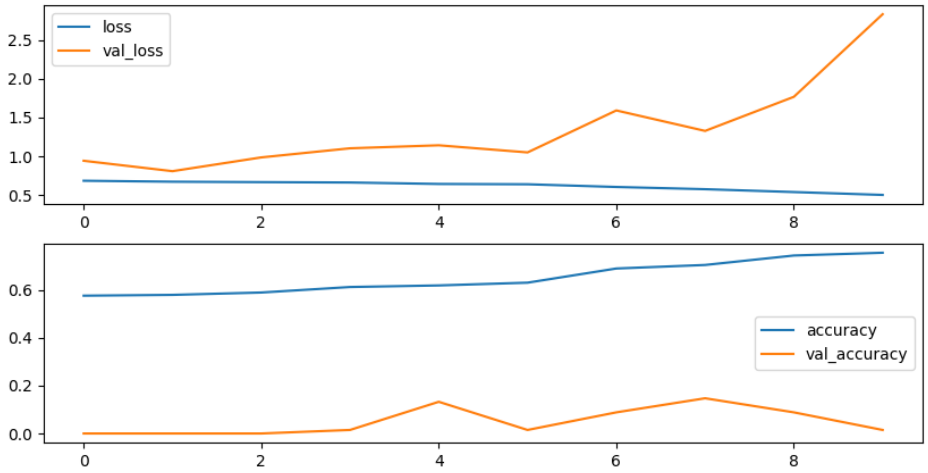


Figure 23. Training results from the more complex RGB data containing model. Source: self-made.

As it can be observed above, no improvement was observed increasing the complexity of the architecture. The accuracy was almost in a *plateau*, whereas the validation accuracy was as low as 5% by the end of the training.

4.3. Tunning of hyperparameters

With the aim of understanding how the hyperparameters interact with the learning capability of the algorithm different combinations were tested. Before training all of the following models, the training and testing data ratio was changed from 2:1 to 4:1, using this way 80% of the data for training and 20% for testing. Number of epochs was also increased from 10 to 30.

First, we evaluated the impact that the number of layers, number of filters and size of the kernel would have. Therefore, we trained 125 models with 1 to 5 layers, 16 to 256 number of filters and kernel sizes from 2 to 6.

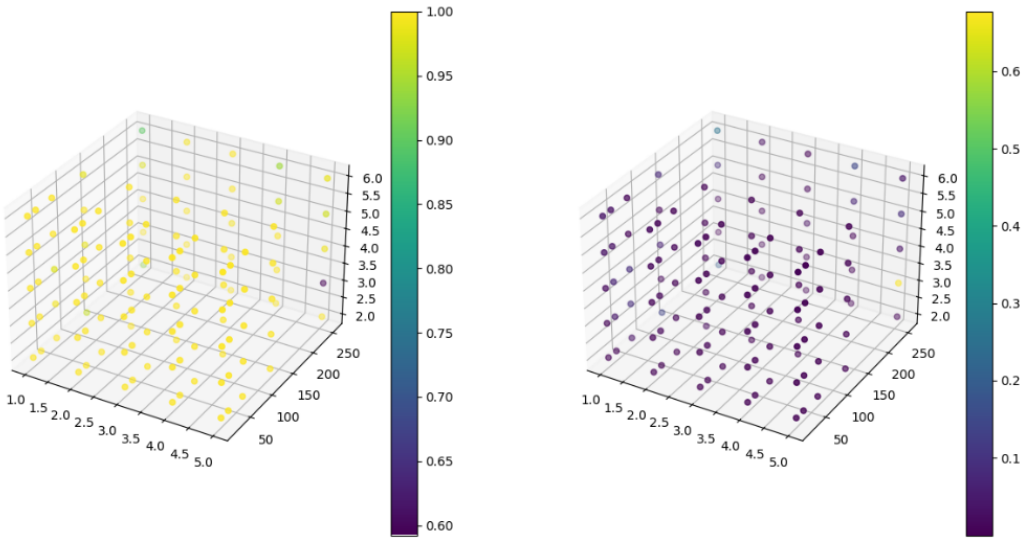


Figure 24. Training results from the 125 models: comparing number of layers in the x axis, number of filters in the y axis, kernel size in the z axis and accuracy (left) and loss (right) as heatmap. Source: self-made.

As it can be observed in the 3D scatterplots above, the training accuracy (Figure 19, left) showed high values with probable overfitting, whereas the training loss (Figure 19, right) showed lower loss values for smaller kernel sizes and number of filters.

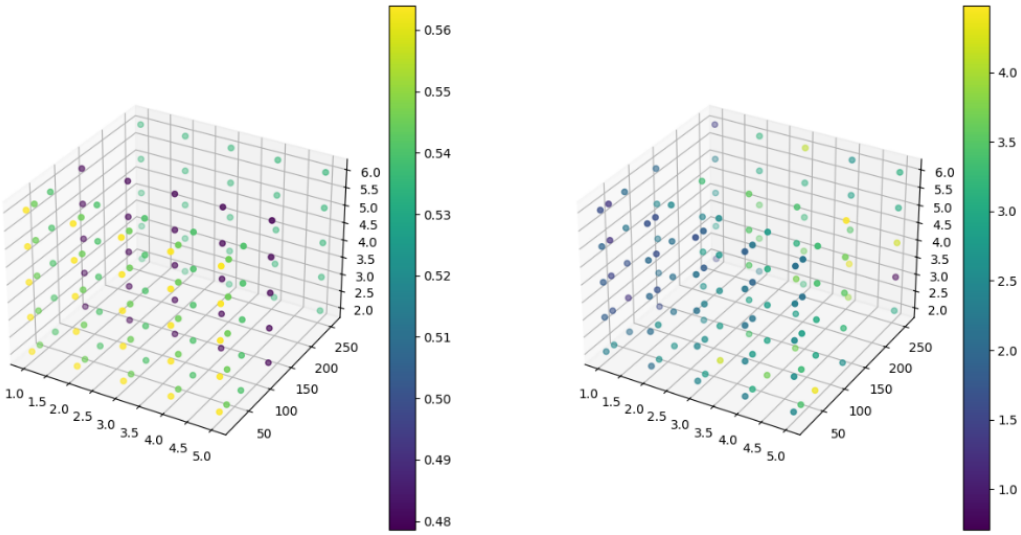


Figure 25. Evaluation results from the 125 models comparing number of layers in the x axis, number of filters in the y axis, kernel size in the z axis and accuracy (left) and loss (right) as heatmap. Source: self-made.

On the other hand, the evaluation scatterplots showed more interesting information. The accuracy (Figure 20, left) showed values above 55% for lower values for filters. It was also observed that higher kernel sizes led to lower accuracy values. The loss (Figure 20, right) did not show any straightforward conclusion across the axes.

Afterwards, a new combination of hyperparameters was prepared. The kernel size was kept constant as (2 x 2). Therefore, number of layers, number of filters and dropout rate was investigated. These dropout layers were introduced after each MaxPooling layer and before the following 2D convolutional layer (or flatten layer for the last one).

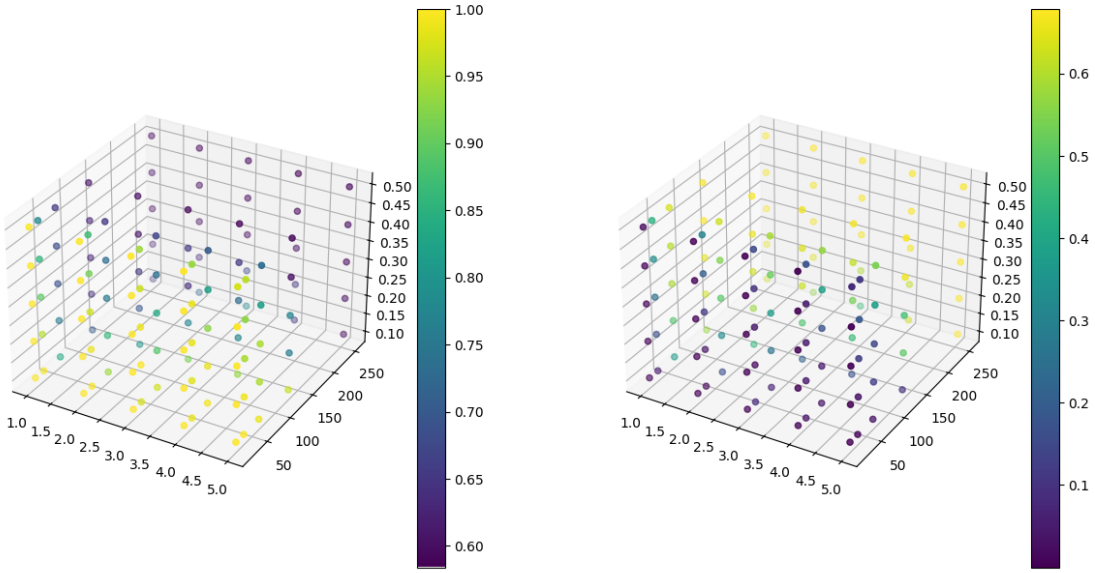


Figure 26. Training results from the 125 models: comparing number of layers in the x axis, number of filters in the y axis, dropout rate in the z axis and accuracy (left) and loss (right) as heatmap. Source: self-made.

Training accuracy showed high values for lower kernel-sizes (Figure 21, left), whereas loss was lower with the same conditions. However no remarkable conclusion could be taken on the dropout rate.

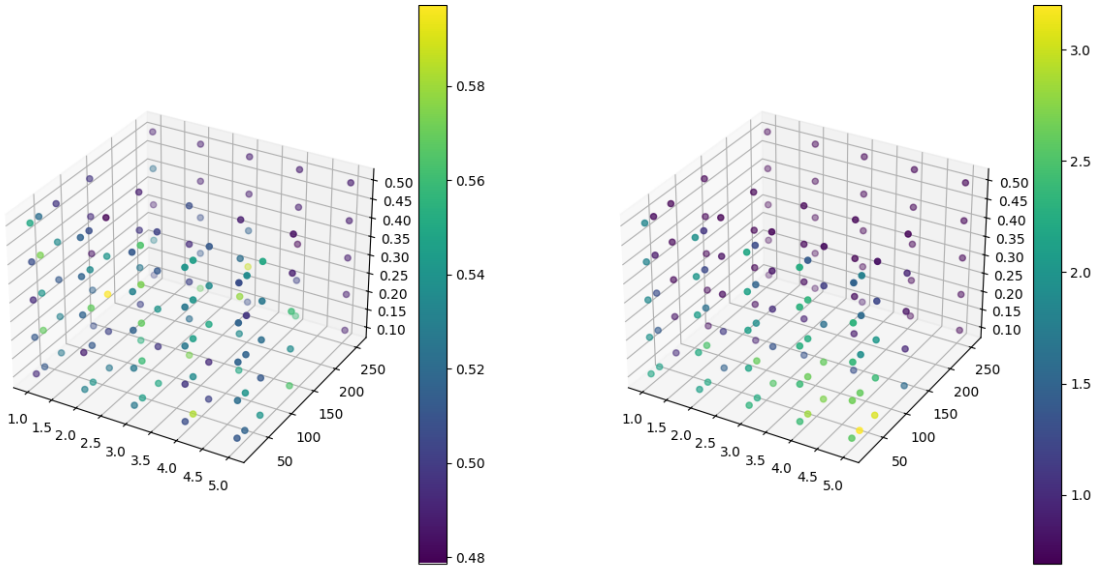


Figure 27. Evaluation results from the 125 models comparing number of layers in the x axis, number of filters in the y axis, dropout rate in the z axis and accuracy (left) and loss (right) as heatmap. Source: self-made.

According to the upper scatterplots, higher validation accuracy of 59.71% was obtained with 3 layers, 32 filters and a dropout rate of 0.3 (Figure 22, left). At the same time the evaluation loss was 74.41%.

Finally, by maintaining the dropout rate constant at 0.3, a dense layer was added before the output layer containing 2^{4+n} units, being n = 0, …, 4.

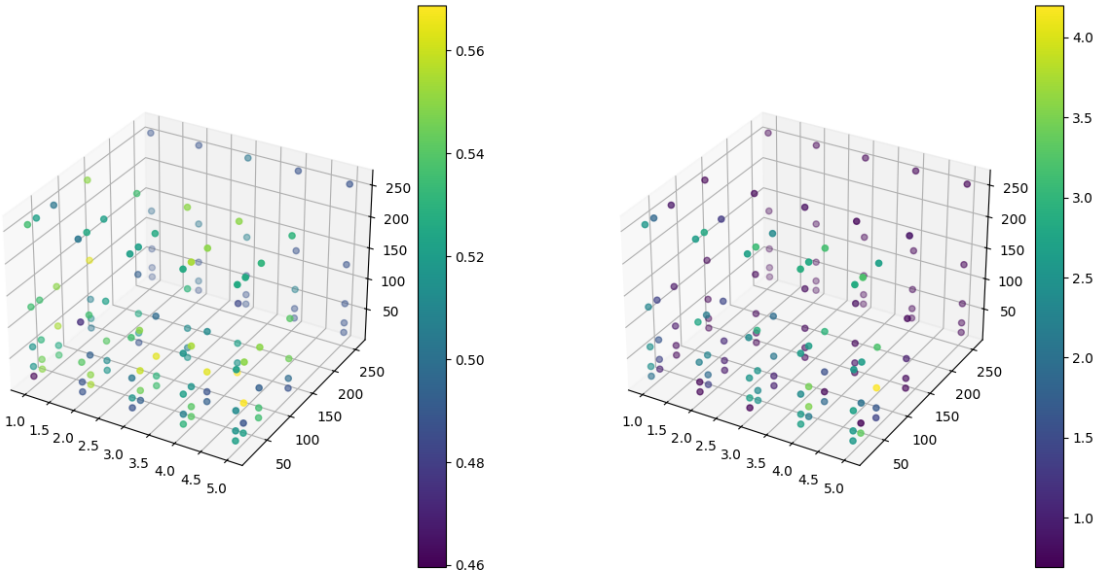


Figure 28. Evaluation results from the 125 models comparing number of layers in the x axis, number of filters in the y axis, number of units in the z axis and accuracy (left) and loss (right) as heatmap. Source: self-made.

As it can be observed in the upper 3D scatterplots, there was no obvious improvement in the accuracy of the model by inserting a dense layer before the output. However, best results were observed when this layer had 128 units.

Therefore, we employed the best conditions to date with the architecture shown in Figure 14. Containing three layers, with 32 filters, (2 x 2) kernel size and dropout layers with a dropout rate of 0.3, obtaining a prediction accuracy of 48.34% with a loss of 70.07%. As we could see during the training, the algorithm was not capable of learning the classification task, as shown in Figure 24:

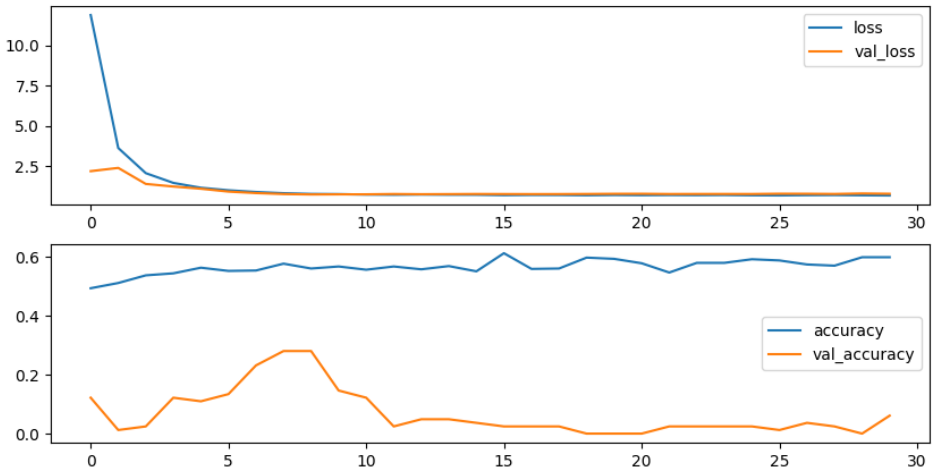


Figure 29. Training results from the best model to date, employing softmax as last activation layer function and CategoricalCrossentropy() as loss function. Source: self-made.

As the learning was not appropriate, we changed the activation function from the last dense layer (output layer) from *softmax* to *sigmoid* due to only being two categories. We also changed the loss function from *CategoricalCrossentropy()* to *BinaryCrossentropy()*. Maintaining the other hyperparameters as before, these new conditions were tested obtaining a 49.51% prediction accuracy and a 70.61% loss. As we could see from the training curves, the algorithm did not do a good job:

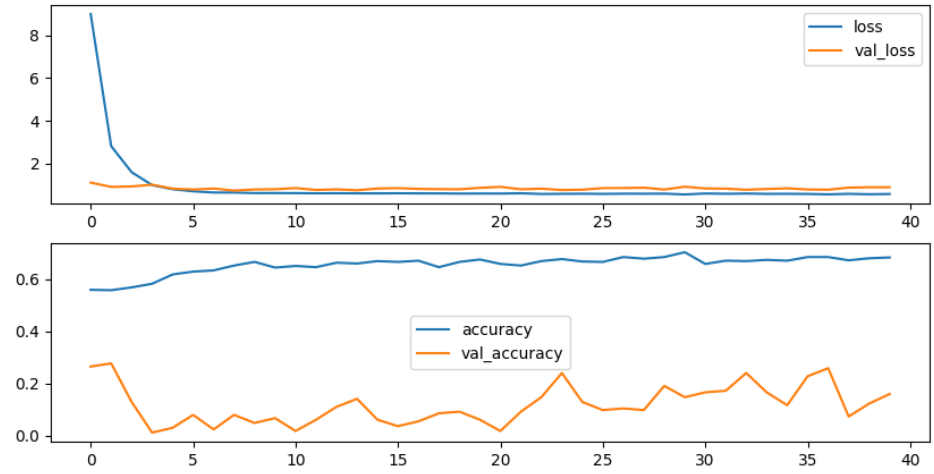


Figure 30. Training results from the best model to date, employing sigmoid as last activation layer function and BinaryCrossentropy() as loss function. Source: self-made.

4.4. Data augmentation

Data augmentation was performed for the three 2D slices, leading to 10 augmented images per original image. As it can be observed in Figure 24, different transformations were performed randomly. In this case, we employed the following geometric transformations: 20% zooming, 20% shear range and horizontal flipping.

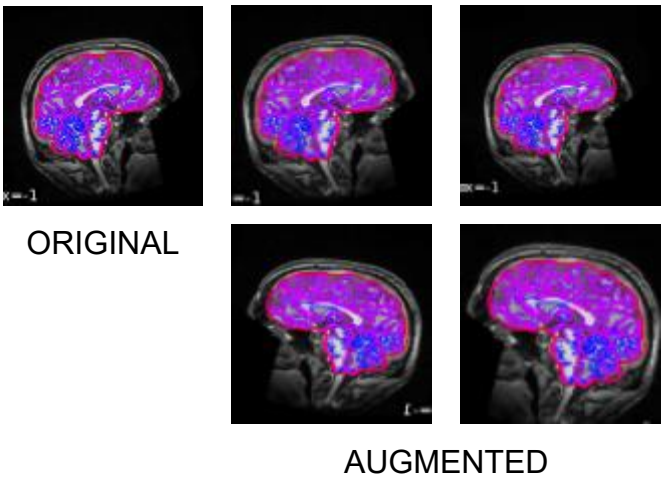


Figure 26. Example of data augmentation. Source: self-made.

Therefore, employing the previous conditions a training dataset containing 10x the original images were loaded, obtaining a prediction accuracy of 58.17% and a loss of 68.23%.

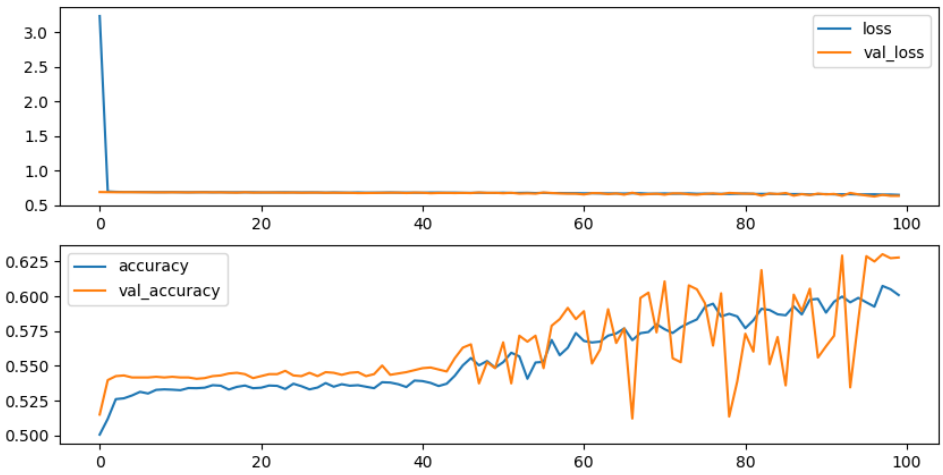


Figure 31. Training results from the best model to date with data augmentation. Source: self-made.

As we can see in Figure 26, the learning capability of the algorithm improved because of the data augmentation. We then tested a kernel-size of 3 due to odd sizes being preferred due to symmetrically dividing the previous layer pixels around the output layer [37]. This way, a smoother training curve was obtained as shown in Figure 28. However, the obtained prediction accuracy was low, a 53.84%; whereas the loss was 72.44%.

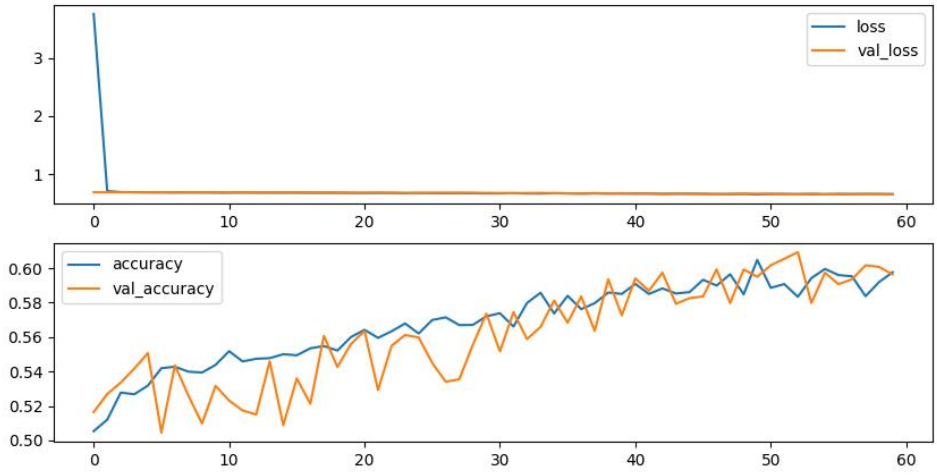


Figure 32. Training results from the best model to date with data augmentation, but changing the kernel size to (3 x 3). Source: self-made.

Literature shows by region-of-interest-based volumetry that adults with ASD have reduced corpus-callosum, whereas surface-based morphometry studies show increased cortical thickness in the parietal lobes [32]. Therefore, the same strategy was followed for the other two data slices. However, as shown in Table 1, even though training curves looked better than before data augmentation, no good accuracy scores were obtained.

Table 1. Prediction accuracy and loss for the three images with augmented data. Source: self-made.

IMAGE	Accuracy	Loss
A	0.5384	0.7244
B	0.5433	0.6917
C	0.4951	0.6952

4.5. Batch normalization

In order to try to increase the accuracy of the algorithms two protocols were tested: first, batch normalization layers were inserted where the dropout layers belonged; second, both batch normalization layers and dropout layers were employed in this order.

As we can see in the results in Table 2, prediction accuracies were not high overall for images B and C. However, we could observe in image A that both procedures yielded a higher accuracy.

Table 2. Prediction accuracy and loss for the three images with augmented data. The addition of dropout layer is also shown with a "+" sign in the "Dropout layer" column.

IMAGE	Dropout	Accuracy	Loss
A		0.6490	2.8565
	+	0.6779	0.7423
B		0.5529	4.2097
	+	0.5288	1.0323
C		0.5385	2.3216
	+	0.4519	0.7914

4.6. Transfer learning

In order to test whether we could achieve a higher accuracy, transfer learning technique was tested on the augmented image A dataset and the pre-trained model VGG16.

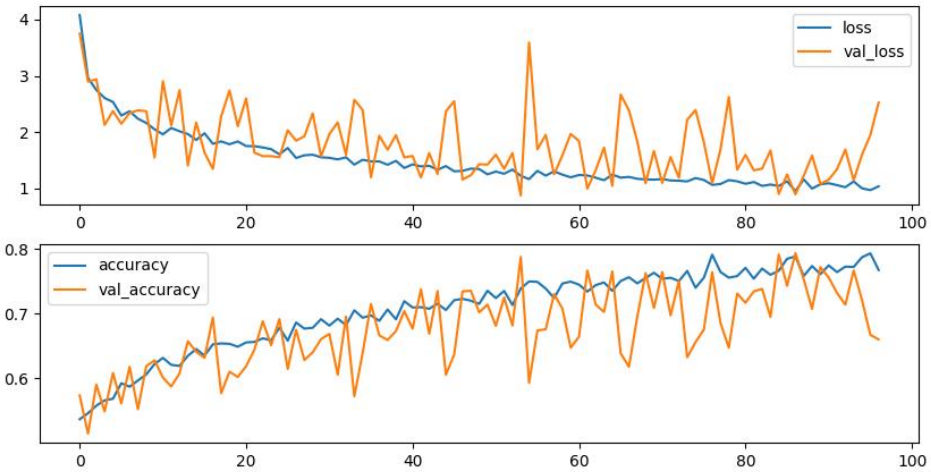


Figure 33. Training curves for transfer learning methodology. Source: self-made.

As we can see in the training curves, the model did indeed learn; however, when new data was presented, the algorithm yielded a low accuracy of 51.92% and a high loss of 378.60%.

5. Conclusion

Different algorithm architectures have been developed in order to achieve the correct classification of magnetic resonance images of patients with autism spectrum disorder and neurotypical ones. Among the procedures that have been carried out, the best method has been to use magnetic resonance imaging with a vertical slice showing both the corpus-callosum and the parietal lobe. In addition, the use of BatchNormalization layers followed by Dropout layers resulted in a model with an accuracy of 68% and a loss of 74%.

Although there is room for improvement, decent accuracy has been achieved with easily obtainable data. This type of data does not require any special pre-processing unlike those mentioned in the state of art. Furthermore, to the best of our knowledge, there is no precedent for direct employment of 2D brain MRI for classification of neurotypical brains and patients with autism spectrum disorder.

The planning could not be carried out as expected. Due to the complexity of the proposed problem, the model improvement required a lot of time for research and code execution due to the large volume of data and limitations in the hardware. In addition, at the beginning, Anaconda and Jupyter Notebook were used to run and edit the code. However, the kernel died unexpectedly several times when trying to run more complex algorithms. Therefore, it was necessary to install a virtual environment on Windows, use Virtual Studio Code and run the scripts from the terminal. On top of all this, the website was left undeveloped due to lack of time.

Even though satisfactory results were obtained, there is still work to be done. As a future plan, it is planned to gain access to more images and work with them to improve the models to achieve better accuracy. In addition, it is also planned to create a website for future use.

6. Glossary

ABIDE: Autism Brain Imaging Data Exchange

AI: Artificial Inteligence

ASD: Autism Spectrum Disease

ANN: Artificial Neural Network

CNN: Convolutional Neural Network

ML: Machine Learning

MRI: Magnetic Resonance Image

ReLU: Rectified Linear Unit

ROI: Region-of-interest

RGB: Red Green Blue

rs-fMRI: resting-state functional Magnetic Resonance Image

SDG: Sustainable Development Goals

SVG: Scalable Vector Graphics

VS Code: Virtual Studio Code

7. Bibliography

[1] Maenner, M., Shaw, K., Baio, J., Washington, A., Patrick, M., DiRienzo, M., Christensen, D. L., Wiggins, L. D., Pettygrove, S., Andrews, J. G., Lopez, M., Hudson, A., Baroud, T., Schwenk, Y., White, T., Rosenberg, C. R., Lee, L. C., Harrington, R. A., Poynter, J. N., Hallas-Muchow, L., Constantino, J. N., Fitzgerald, R. T., Zahorodny, W., Shenouda, J., Daniels, J. L., Warren, Z., Vehorn, A., Salinas, A., Durkin, M. S., Dietz, P. M. (2020). Prevalence of autism spectrum disorder among children aged 8 years – Autism and developmental disabilities monitoring network, 11 sites, United States, 2016. *MMWR Surveill. Summ.*, 69 (4), 1-12.

[2] van't Hof, M., Tisseur, C., van Berckelear-Onnes, I., van Nieuwenhuyzen, A., Daniels, A. M., Deen, M., Hoek, H. W., Ester, W. A. (2021). Age at autism spectrum disorder diagnosis: a systematic review and meta-analysis from 2012 to 2019. *Autism*, 25 (4), 862-873.

[3] Rogers, S. J., Vismara, L., Wagner, A. L., McCormick, C., Young, G., Ozonoff, S. (2014). Autism treatment in the first year of life: a pilot study of infant start, a parent-implemented intervention for symptomatic infants. *J. Autism Dev. Disord.*, 44 (12), 2981-2995.

[4] Clark, M. L. E., Vinen, Z., Barbaro, J., Dissanayake, C. (2018). School age outcomes of children diagnosed early and later with autism spectrum disorder. *J. Autism Dev. Disord.*, 48 (1), 92-102.

[5] Rafiee, F., Habibabadi, R. R., Motaghi, M., Yousem, D. M., Yousem, I. J. (2022). Brain MRI in autism spectrum disorder: narrative review and recent advances. *J. Magn. Reson. Imaging*, 55 (6), 1613-1624.

[6] United Nations. (2023, October 14). Take action for the sustainable development goals –united nations sustainable development. United Nations. https://www.un.org/sustainabledevelopment/sustainable-development-goals/

[7] Wood-Downie, H., Wong, B., Kovshoff, H., Mandy, W., Hull, L., Hadwin, J. A. (2021). Sex/gender differences in camouflaging in children and adolescents with autism. *J. Autism Dev. Disord.*, 51, 1353-1364.

[8] van Oostveen, W. M., de Lange, E. C. M. (2021). Imaging techniques in Alzheimer's disease: a review of applications in early diagnosis and longitudinal monitoring. *Int. J. Mol. Sci*, 22, 2110-2144.

[9] Pyatigorskaya, N., Gallea, C., Garcia-Lorenzo, D., Vidailhet, M., Lehericy, S. (2014). A review of the use of magnetic resonance imaging in Parkinson's disease. *Ther. Adv. Neurol. Discord*, 7 (4), 206-220.

[10] Wen, J., Thibeau-Sutre, E., Diaz-Melo, M., Samper-González, J., Routier, A., Bottani, S., Dormont, D., Durrleman, S., Burgos, N., Colliot, O. (2020). Convolutional Neural Networks for classification of Alzheimer's disease: overview and reproducible evaluation. *Med. Image Anal.*, 63, 101694.

[11] Huang, D. T. J., Gururajapathy, S. S., Ke, Y., Qiao, M., Wang, A., Kumar, H., Yang, Y. (2022). Data-driven network neuroscience: on data collection and benchmark. arXiv:2211.12421v2.

[12] Marx, E., Deutschländer, A., Stephan, T., Dieterich, M., Wiesmann, M., Brandt, T. (2004). Eyes open and closed as rest conditions: impact on brain activation patterns. *Neuroimage*, 21 (4), 1818-1824.

[13] Syed, M. A., Yang, Z., Hu, X. P., Deshpande, G. (2017). Investigating brain connectomic alterations in autism using the reproducibility of independent components derived from resting state functional MRI data. *Front. Neurosci.*, 11, 459.

[14] Parison, S., Ktena, S. I., Ferrante, E., Lee, M., Guerrero, R., Glocker, B., Rueckert, D. (2018). Disease prediction using graph convolutional networks: Application to Autism Spectrum Disorder and Alzheimer's Disease. *Med. Image Anal.*, 48, 117-130.

[15] Heinsfeld, A. S., Franco, A. R., Craddock, R. C., Buchweitz, A., Meneguzzi, F. (2018). Identification of autism spectrum disorder using deep learning and the ABIDE dataset. *Neuroimage Clin.*, 17, 16-23.

[16] Kong, Y., Gao, J., Xu, Y., Pan, Y., Wang, J., Liu, J. (2019). Classification of autism spectrum disorder by combining brain connectivity and deep neural network classifier. *Neurocomputing*, 324, 63-68.

[17] Shrivastava, S., Mishra, U., Singh, N., Chandra, A., Verma, S. (2020) "Control or Autism – Classification using Convolutional Neural Networks on Functional MRI", 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kharagpur, 1-3 July.

[18] Sherkatghanad, Z., Akhondzadeh, M., Salari, S., Zomorodi-Moghadam, M., Abdar, M., Acharya, U. R., Khosrowabadi, R., Salari, V. (2019). Automated

detection of autism spectrum disorder using a Convolutional Neural Network. *Front. Neurosci.*, 13, 1325.

[19] Thomas, R. M., Gallo, S., Cerliani, L., Zhutovsky, P., El-Gazzar, A., van Wingen, G. (2020). Classifying autism spectrum disorder using the temporal statistics of resting state functional MRI data with 3D Convolutional Neural Networks. *Front. Psychiatry*, 11, 440.

[20] Leming, M. J., Baron-Cohen, S., John, S. (2021). Single-participant structural similarity matrices lead to greater accuracy in classification of participants than function in autism in MRI. *Mol. Autism*, 12, 34.

[21] Gao, J., Chen. M., Li., Y., Gao, Y., Li, Y., Cai, S., Wang., J. (2021). Multisite autism spectrum disorder classification using Convolutional Neural Network Classifier and individual morphological brain networks. *Front. Neurosci.*, 14, 629630.

[22] Russell, Stuart J.; Norvig, Peter. (2021). Artificial Intelligence: A Modern Approach (4th ed.). Hoboken: Pearson.

[23] Hsu, F.-H. (2002). Behind Deep Blue: Building the Computer That Defeated the World Chess Champion. Princeton: Princeton University Press.

[24] IBM. (n.d.). *What are Convolutional Neural Networks?* https://ibm.com/topics/convolutional-neural-networks

[25] GeeksforGeeks. (2023, April 21). *Introduction to Pooling Layer.* https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

[26] Tensorflow. (2023, September 27). *tf.keras.activations.relu.* https://www.tensorflow.org/api_docs/python/tf/keras/activations/relu

[27] Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep Learning*. MIT Press. pp. 180-184.

[28] Machine Learning Mastery. (2021, August 18). *A gentle introduction to sigmoid function.* https://machinelearningmastery.com/a-gentle-introduction-to-sigmoid-function/

[29] Bosch, A., Casas, J., Lozano, T. (2019). *Deep Learning. Principios y fundamentos.* Editorial UOC. p. 96.

[30] Szegedy, C., Ioffe, S. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Interval Covariate Shift. arXiv:1502.03167.

[31] Keras. (2023, June 25). *Transfer learning & fine-tuning.* https://keras.io/guides/transfer_learning/

[32] Keras. (n.d.). *Keras Applications.* https://keras.io/api/applications/

[33] Autism Brain Imaging Data Exchange. (2017, March 27). *Introduction.* https://fcon_1000.projects.nitrc.org/indi/abide/

[34] Python. (n.d.) *venv – Creation of virtual environments.* https://docs.python.org/3/library/venv.html

[35] DataCamp. (2022, November). *A complete Guide to Data Augmentation.* https://www.datacamp.com/tutorial/complete-guide-data-augmentation

[36] Yun Jiao, R. C., Herskovits, E. H. (2011). Structural MRI in Autism Spectrum Disorder. *Pediatr. Res.*, 69 (5 Pt 2), 63-68.

[37] Medium (2020, June 23). *How to choose the size of the convolution filter or Kernel size for CNN?* https://medium.com/analytics-vidhya/how-to-choose-the-size-of-the-convolution-filter-or-kernel-size-for-cnn-86a55a1e2d15