End of degree thesis / Trabajo fin de grado
Degree in Electronic Engineering / Grado en Ingeniería Electrónica

Deep Learning Algorithms for Mental Illness Detection and Emotion Analysis

Author/ Autora:
Ane Varela Etxeberria

Supervisor/Directora:
Raquel Justo

Leioa, 22 June 2023 / Leioa, 22 de Junio de 2023

Index

1 Introduction and objectives

The aim of this work is to understand and put into practice some of the most important concepts that have been developed lately regarding machine learning, specially in the mental healthcare field. In order to do so, two models, BERT and HuBERT, will be used.

In the last years, the published papers regarding machine learning, and, more specifically, deep learning for health informatics applications, have dramatically increased in number [1]. There are also many papers in the specific application of machine learning in mental health issues [2], [3]. In our university, works like [4] and [5], which are based in speech processing, have been developed. This has greatly helped my own research and experiments.
Currently, the trend is to use Natural Language Processing (NLP) and Deep Learning (DL) in order to achieve intelligent models. The types of algorithms used are usually Neural Networks (NN), based on the architecture of the human nervous system, and Support Vector Machines (SVM).

Some applications of machine learning in healthcare, and specifically targeting the field of mental health and social work, could be sign language interpretation, suicide prevention, addiction treatment, Alzheimer diagnosis, cyber-harassment recognition, and detection of several mental health issues such as depression, bipolarity and anxiety [6]. Although much work is still needed in this field and many challenges arise, great expectations and hopes are placed in these applications. However, more intercommunication and work between health care or mental health professionals and data scientists is needed in order to develop better models, and it is important to note that machine learning is currently mostly used as helping or supporting human intervention of doctors or psychological mentors.

Therefore, the main aim of this work will be to achieve emotion and mental illness classifications with both text and audio data processing. A differentiation algorithm between people with anxiety and/or depression and people with no mental illnesses will be developed, and, moreover, two emotional parameters called valence (positivity) and arousal (excitation) will be computed. In the long run, this work would serve to provide mental healthcare workers with a support system in order to lessen their workload and make decision-making faster. This could be achieved by helping to detect patients with mental illnesses, and also by classifying the emotions of said patients so as to monitor their improvement, mood switches, and such.

This paper is divided in four sections regarding the carried out work; firstly, the theoretical basis for machine learning will be briefly explained. The proposed models will also be defined. In the following section, an explanation of the data used and its characteristics will be presented. Afterward, in the development section, the experiments and their results will be described. Finally, some conclusions will be drawn for the experiments, and future lines of work will be highlighted.

2 Theoretical background

2.1 Machine learning

Machine learning is the science of programming computers so that they can learn from data without needing human supervision [7]. There are also several more definitions, like machine learning as the study of giving computers the ability to learn without explicitly programming it, or the study of making a computer learn from data in order to perform a task, making its performance better after that training. There are many machine learning applications in our daily lives, like spam filter for e-mails or image analysis in social media.
Machine learning algorithms can be classified using various criteria, but the most common one is whether they are trained with human supervision or not. In this form of classification, we may find four different kinds of algorithms.

Supervised learning. It consists of a computer learning from data that has been previously labeled so as to conduct a classification or a regression. It requires data preprocessing in order to achieve viable data.

Unsupervised learning. The items to be considered in training are unlabeled. Thus, it is normally used to cluster or organize data by similarities.

Semi-supervised learning. It is a mixture of the two classifications previously defined, and it consists of a data group in which most items are unlabeled, but that contains some labeled ones. It can be used for image recognition, among other applications.

Reinforcement learning. Instead of offering labels, the program is trained using trial and error; if an error is committed, there is a penalization, and similarly, if a task is achieved, it receives a reward. It is most commonly used in robot training.

The most common algorithms in machine learning, and the ones this work will base itself on, are support vector machines (SVM) and neural networks (NN). SVMs are used in order to create divisions between different groups that the algorithm identifies, and different hyperparameters can be tuned in order to give more or less flexibility to the limits of the classification. Neural networks, on the other hand, are used in more complex tasks and are inspired by the way in which the brain functions. In these models, a lot of hyperparameters, which will be explained as the project experiments are developed, have to be taken into account.

The main challenges in machine learning come from the insufficient amount of data, which is perhaps as important as an effective algorithm. There are also risks of the training data not representing the actual, real data. Noisy data, or even too-detailed data with non-relevant features, also hinder efficiency. Algorithms, on the other hand, often suffer from under or overfitting, meaning they adapt too little or too much to the training data in comparison to the testing one.

Preparing the data

In order to conduct a machine learning project, it is important to not only train the algorithm, but also to firstly clean the data and then test the model. Cleaning the data usually means clustering information that may be too general, or filling blanks with mean or mode values if they are missing in some instances; getting rid of those instances is also a possibility. If the scales are really different in the input data, normalizing it is also a key step to make a more effective algorithm.

When testing the model generated from this optimized data, usually around 20% of the data is put aside at the beginning of the project and used for testing. The data scientist may have to be very careful when splitting this data, so as to achieve similar distribution of relevant parameters in both the training and the test group. A different approach is to split the data into groups to use methods such as cross-validation, so as all the data can be used to train the algorithm and the end results do not depend on the particular test groups. Algorithm testing using cross-validation is a great tool to really assess the efficiency of a program.
Aside from the test group, another data group is usually separated from the training data. This is called the validation data and is used to choose the hyperparameters that will make the model more accurate, as it will be seen later on.

Testing the model

Generally, when the functioning of an algorithm has to be visualized, a confusion matrix is used. This shows the fractions of successes and fails; even though axis choice can vary, one axis will show the algorithm guesses, while the other one will demonstrate real values. Note that the matrix does not have to be, and will usually not be, symmetrical. Thus, in a two-dimensional problem in which the program guesses whether an instance belongs to a certain group, the confusion matrix would be constructed as Figure 1 indicates. Some

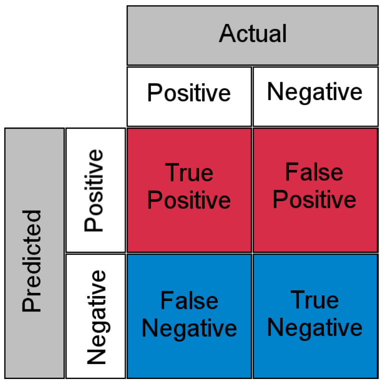


Figure 1: Confusion matrix [8]

numerical values can be defined in order to assess efficiency of the algorithm. The most widespread ones are, on the one hand, precision, the ratio between true positives (TP) and all predicted positives (including false positives, FP); the other one is recall, the ratio between true positives and all actual positives (including false negatives, FN):

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

There is usually a trade-off between the two of them, with one increasing while the other decreases. Nevertheless, there is a value in which the combination of the two is optimal. This is usually determined by the F_1 score, a harmonic mean of the two parameters, which is thereby defined:

$$\text{F}_1 = 2\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{3}$$

This is normally the parameter to measure the effectiveness of an algorithm.
As usually in classification tasks the choice for positive or negative is arbitrary, or depends on the type of the algorithm, sometimes the average of all the F_1-scores (with each of the F_1-scores having a different positive class) is taken into account.
Another parameters to determine the performance of the model can be the accuracy or the Matthews Correlation Coefficient (MCC).

2.1.1 SVMs

A type of algorithm usually used in machine learning and which is suitable for both classification and regression tasks is a Support Vector Machine (SVM). This algorithm is based on hyperplanes that divide the data, creating groups or classes between them. SVMs can be precisely tuned using various hyperparameters and thus are a great solution for various machine learning problems.
As stated, the aim of SVMs is to classify groups using a hyperplane, whose dimension depends on the sample representation, fitting the widest possible street. That means we want to separate the classes as much as possible. In order to obtain this result, however, *margin violations* have to be taken into account. SVMs with hard margins force the fitting to get all training instances out of that separating street, while soft margins allow outliers to be left out and thus be a part of that street. These last ones are more flexible, and a balance between the two has to be obtained in order to obtain a good performing algorithm. A good fitting example for a two-dimensional model can be found in Figure 2.

SVM can be used for regression, instead of classification, with one small change; the instances are aimed to be put on the street instead of outside of it.

2.1.2 Neural networks

Another very common and versatile algorithm is that of a neural network. This architecture is tailorable and thus very practical for complex tasks such as image recognition, and it can be effectively used for mental health applications.

Neural networks are architectures inspired by the way in which data is processed in

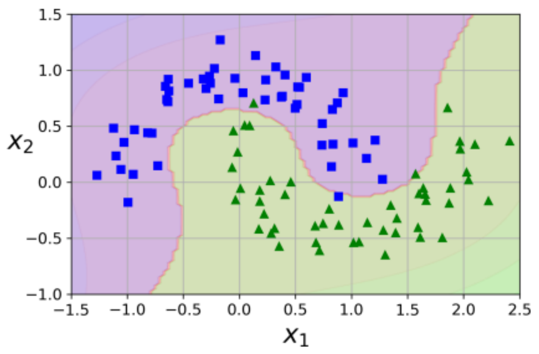


Figure 2: Working principle of a SVM [7]

the brain. They consist of different layers with many basic logic units called neurons; there, information from other neurons (or from the input layer) is weighted and summed, and then a result is computed using an activation function that varies depending on the effectiveness and the output information that the user wants to achieve.
The simplest neural network is called a multi-layer perceptron. Many more architectures exist, such as Convolutional Neural Networks (CNN), which are based on mathematical convolution instead of multiplication, or Recursive Neural Networks (RNN), which uses the same weight matrices over and over. However, the multi-layer perceptron, depicted in Figure 3, can give us a basic idea behind the concept of neural networks. Each one of the neuron in a layer takes data from the previous layer or the input data group, multiplies each instance by a weight, and computes the output using a certain activation function. Some neurons can also contain a value named as bias, which is summed to the rest of the weight times data values. The process of each neuron can be summarized in these formulas:

$$z = (\sum_{i=1}^{n} w_i * x_i) + b \\ a = g(z) \tag{4}$$

Here, a is the output of the neuron, each w_i denotes a weight corresponding to the feature x_i from a total group of size n, and b is the bias. Each neuron follows this formula, so the output a of a certain neuron, computed by the activation function $g(z)$, will be one of the samples x_i of the neurons in the next layer.

The first layer, in which the data that is processed is contained, is called the input layer. The output layer is the one that computes the result or results the model wants to obtain. The layers in between, consisting normally of hundreds of neurons each, are called hidden layers.
Training on neural networks is based on gradient descents, which calculate the difference between the predicted result and the real one through a loss function and aim to minimize it through backpropagation. This means that the weights w_i of each neuron and layer are altered in each iteration or epoch to reach the minimum loss.
Each neural network can be altered through many parameters, including the number of layers, number of neurons, activation function, loss function or learning rate. We can aim

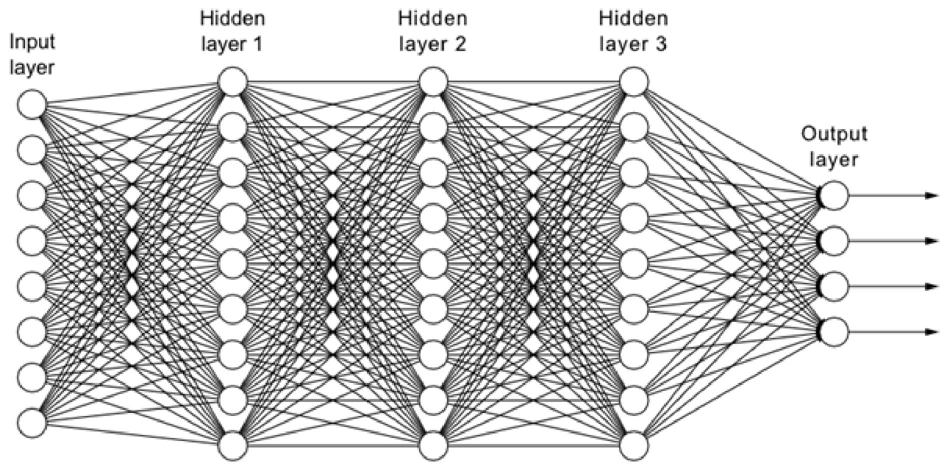


Figure 3: Architecture of a multi-layer perceptron [9]

to tune our model and achieve great efficiencies by changing these hyperparameters. This, however, also increases the risk of overfitting the training data, as we will discuss later on.

2.1.2.1 Performance enhancement

In order to enhance the performance of the neural network, there are several parameters that have to be taken into account. If these parameters are chosen poorly, there is a risk of overfitting or underfitting the data.
Overfitting the data means that an algorithm is too fitted to a limited training data and will not generalize well. The model is too complex and fits the noise also present in the training data, so a simpler model, or more data, could solve this problem. Underfitting, on the other hand, arises when the model is too simple to learn the underlying structure of the data, and thus, its predictions will be inaccurate.
Thus, the hyperparameter choice has to be meticulous so as to fit the data properly. Different parameters can be tailored to do so, and they are usually chosen separating a small group from the training data. This is called the validation group, and it serves the purpose of evaluating the trained model while tuning the hyperparameters. It is very different from the test data, which is used only at the end, once the model is tuned.

Firstly, the number of epochs, which establish the number of training steps in the general algorithm, can be changed. The loss and the weights will be calculated and altered n times, being n the number of epochs we choose. Another important parameter is the learning rate, which controls the backpropagation speed or the ratio at which the weights are updated in each learning step via the formula $w_{i+1} = w_i + \nu * \Delta y$. In this formula, w is a certain weight in the neuron, which changes via a loss denoted by Δy and the learning rate ν. Increasing the learning rate too much leads to suboptimal results, but decreasing it means the training will be much slower. The batch size, which groups the data in different sections in order to optimize the results by taking different data values to be trained at the same time, has also to be taken into account. It normally is a small power of two value like 16 or 32, but in computers with great capacity much larger batch sizes can be taken.

Other hyperparameter choices are not numerical and depend on a set of options. For instance, there are many activation functions that compute the final output of a neuron. Depending on the task and the nature of the output data, one or another could be a better choice, although historically the $g(z) = tanh(z)$ function has been one of the most used ones. The loss function, which computes the difference between the desired and the predicted output, also has to be chosen. This choice is, however, determined by the type of task to be developed. For instance, for classification, the cross-entropy loss is computed. There are also algorithm optimizers in order to tweak some hyperparameters at the same time of the training; changing the learning rate, for example, is pretty common, and can be done by different algorithms, being one of the most well-known ones the Adam optimizer, which will be used in the models we propose.

Other factor that has to be taken into consideration is the number of training data. When the data is imbalanced or scarce, the neural network may perform poorly because of the lack of variety or the predisposition to predict the most prevalent data. In order to avoid this, different solutions exist; oversampling is one of them, and here some data points are repeated in order to obtain a balanced dataset for classification. Another option is the SMOTE algorithm (Synthetic Minority Oversampling TEchnique), which synthesizes new examples for the minority class by selecting examples that are close in the feature space and creating a new one from the line that is thus created [10].

2.1.3 BERT

Before diving into the data used in this project, a most interesting algorithm has to be explained. BERT, standing for Bidirectional Encoder Representations from Transformers, is a language preprocessing algorithm that mind-blowingly set records on language and communication technology algorithm performances.
It is based on the encoder of a transformer neural network. These algorithms were proposed in order to substitute Long Short-Term Memory (LSTM) algorithms, as they were slow and did not learn the language bidirectionally or contextually [11]. Transformer networks map words based on similarity in an environment called an embedding space, and use a positional encoder to determine whether the position of a certain word in the sentence may cause changes on its meaning based on contextual information. Transformer neural networks basically try to understand language, although their applications can be very varied.
A transformer neural network's encoder consists of, firstly, adding positional information to the word embedding. Then, a block containing a multi-head attention and a feed forward is applied n times. A multi-head attention determines how relevant a word is in the sentence with respect to the other words; a feed forward, on the other hand, transforms this output, called attention vector, into more interpretable data. Finally, after this block repetition, the output is obtained.

Thus, based on this model, the BERT algorithm learns first, and then is fine-tuned in order to obtain the desired output, which can be different depending on the desired application; question answering, translation, summarization. . .
The learning phase consists of the algorithm learning simultaneously on two unsupervised

tasks: masked language modeling (MLM), which is kind of a fill-in-the-blanks exercise, and next sentence prediction (NSP). The input is a two-sentence paragraph with masked or hidden words, and the algorithm has to both predict the missing words and determine if the second sentence is a logical following of the first one. Each word or token is transformed into an embedding in order to do so; each embedding has three vectors. Simplifying the algorithm, it can be said that the first vector value contains the position of the word in a data pool of vocabulary, the second one dictates if the word is part of the first or the second sentence, and the third just marks the position of the token in the input.
After this two-tasked training, the algorithm has to be fine-tuned to the specific task it will be applied on. This part is usually very fast, with only one neural layer, it is done in a supervised way, and only requires minimal editing of the original programming. The fine-tuning depends greatly on the task to be performed. An overlook of the general process can be seen in Figure 4.

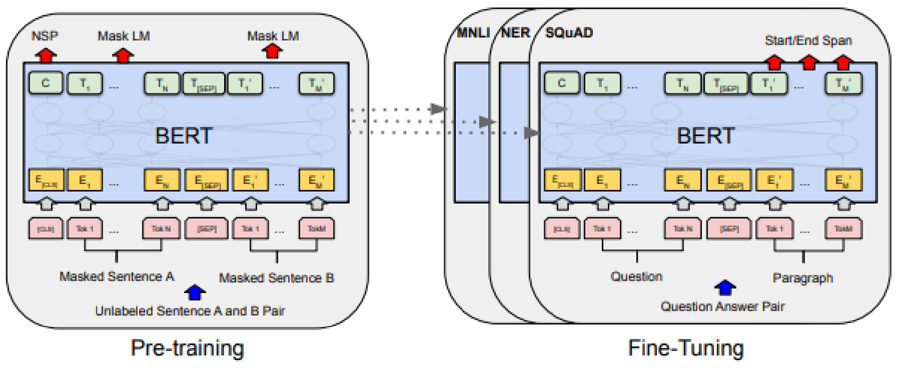


Figure 4: BERT functioning mechanism [11]

BERT is usually talked about in a general way, but really, two algorithms exist; one based on 12 encoders, called the base model, which is faster, and other one based on 24 encoders, the large model, with much more parameters to tune and thus slower but with a greater performance. For each task, one or the other may be more suitable.

2.1.4 HuBERT

Audio processing can be done in several ways, but if it contains spoken data, it is labeled as speech processing and it becomes more complex. A model very similar to BERT, which has good performance in speech processing, was recently developed. It is called HuBERT, which stands for Hidden-unit BERT [12], and it consists of forcing the model to learn both from the acoustic and the language model. As audio data does not depend only on words, there are many more factors to take into account; some of them could be interruptions to denote hesitation, tone to detect speaker identity, trembling of the voice, laughter or background noise.
In order to process both non-linguistic and linguistic data, the model uses both a Convolutional Neural Network (CNN) and a transformer which masks data instead of pseudo-labeling them. Pseudo-labeling was widely used before this audio model was created,

and it was more task-oriented, meaning that it was difficult to train for different but similar tasks. The process was also costly and faulty, as the model learned from previous versions called teacher models, in which only the data from the first one contained supervised labels. With this new approach of using masking, the model can capture the long-range temporal relations instead of only instantaneous ones, improving the previously existing wav2vec model.
Hugging Face created several models based on this principle, enabling the public use of HuBERT very similarly to BERT. However, the options are still limited comparing to the text counterpart.

The functioning of HuBERT can be seen in Figure 5; the audio file is taken as a wav, in numerical array form, and its acoustic features are extracted from the wave using a CNN. Then, different inputs are masked, like it was done on BERT. Afterward, a transformer, whose functioning was previously explained, is applied to the data. Therefore, the principle is similar to BERT, hence the name, even though the application can be quite different.
HuBERT, like its textual counterpart, has a base and a large model, and also a toy tiny model, and it may be trained not just in English but in other languages such as Chinese. In this work, I will focus mainly on the base English model of both BERT and HuBERT.

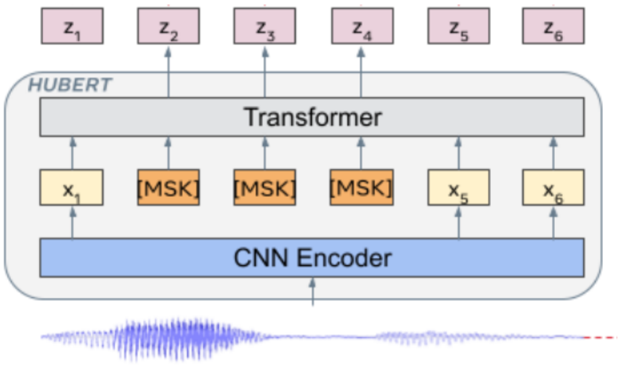


Figure 5: HuBERT operating system [12]

2.1.5 Multi-modal algorithms

When we define modalities as the different elements that can be used in order to define or describe the data, multi-modal algorithms arise [13]. These not only try to fit the model to the different data types existent, but they also seek to make connections between them and achieve a context or underlying information that connect the different elements. By analyzing the data using different modalities simultaneously, the nuances of each can be assessed and the model can fit the data better than if only one modality was used.
In this work, the multi-modality will consist of processing text and audio data simultaneously. Speech processing is particular in this sense, because audio files normally contain different information than text-based data. Text data consists of semantic information, while, with audio, other characteristics like the tone of the voice or the breathing sounds are taken into

account. Therefore, it can be important to combine the two types of data so as to obtain the maximum information available.
In this work, a pretrained BERT and HuBERT model will be used to process the text and audio data separately, and then the inputs will be concatenated. This combined data will be used to train neural networks and SVMs so that the wanted output is achieved. This architecture can be seen on the left part of Figure 6, where the preprocessed data is fed to a multi-layer perceptron. Nevertheless, much more complex architectures can be built for multi-modal processing.

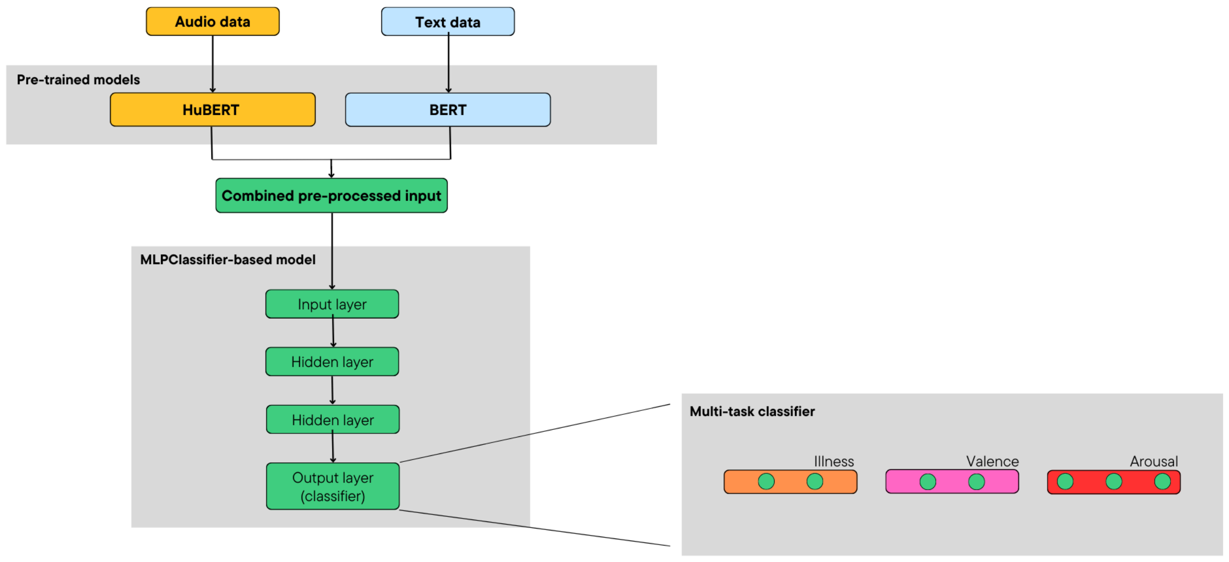


Figure 6: Architecture of the multi-modal multitask pretrained model

2.1.6 Multitask algorithms

Multitask algorithms, opposite of the multi-modal models, are architectures that compute different tasks from the same input data [14]. Normally, these different outputs are related or have some kind of connection, and the similarities and differences between tasks can be taken into account to increase efficiency. However, unrelated tasks can also be computed at the same time, and this will at least improve the speed of the training. Architecture-wise, once the input data is ready, a common body is shared between the tasks, but different heads are used for each one of them.
A particular multitask model type, and the one that I will use in this work, is the multi-output multi-class one. In this case, each of the different outputs that are predicted belong to a classification task. The problem is relatively simplified, as the heads also can have layers in common, excepting a last layer that computes the output for each task. Thus, the output layer is a multi-output classifier.
In this work, an architecture very similar to the multi-modal algorithm was used. The data was processed and put through the pretrained BERT and HuBERT models, and then a multi-layer perceptron-like architecture was used, with a specialized output layer that computed three classifications at the same time. This can be seen on Figure 6.

3 Task and corpus

Before processing and feeding the data to different models, it is important to analyze the characteristics of the dataset that is going to be used. The data used for this work was gathered within the framework of the MENHIR project [15], which strives to help people cope with mental illnesses. It is based on 60 interviews made to people from Ireland. Some of them were clients from a mental institution who had been mostly diagnosed with depression and anxiety, and will be referred to as the AMH group. The other group of people belonged to a control group, and had no diagnosis of mental health issues.
There were originally 32 AMH interviews and 28 control interviews, but due to transcription and computational issues, less information has survived the processing. The issues include audio file corruption, lack of transcribed textual data and lack of emotion transcription. Therefore, only 51 interviews (26 Control, 21 AMH) were left.
Each interview is divided in turns in which either the interviewer or the interviewee speaks. The participation of the interviewer has not been taken into account in this work, as it has no relevant information on how the patient responds; the interviewer just made the questions and directed the interview. Additionally, a great part of the AMH group responses were long, so in the transcriptions, the interventions have been sliced up in order for the dataset to be greater and the items more specific. In text, there are 6824 interventions in total; in the emotion detection part, which is the part we are using for this work, there are 3645 transcriptions (2541 AMH, 1104 control). In audio, however, there are only 1852 interventions, making these data less abundant, which could be detrimental.

In the interviews, two characteristics involving emotion are noted. The emotional framework recorded in this dataset is the two-dimensional space created by valence and arousal [16].
Valence would denote how much of a positive or negative experience the person is having, and it ranges from highly negative to highly positive. Arousal, on the other hand, ranges from agitated to calm, and it characterizes the excitement of the person. The two-dimensional classification is presented in Figure 7, where different emotions have been put into the graph for a better understanding of emotion classification using this method.

3.1 Corpus analysis

People having anxiety or depression have an 85% probability of containing the other illness in our dataset; only three people of each illness group does not have the other one. Taking that into account, we consider a group of people in which $39,22\%$ people have anxiety, $39,22\%$ have depression, $33,33\%$ have both and $13,73\%$ have other mental issues apart from anxiety or depression that have not been thoroughly investigated in this report. In general, about $45,10\%$ of the interviewed people had some type of mental issue.

On the other hand, as the valence and arousal value in each turn of the interviewee was annotated, their percentages can be analyzed.

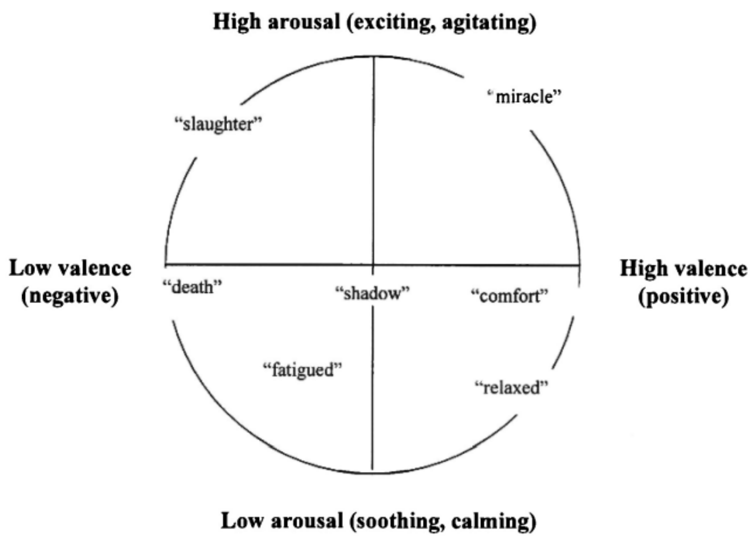


Figure 7: Emotion classification via valence-arousal [16]

Among all the text interview chunks, most of the ill patient interviews (52%) identify with mostly negative emotions, while 91% of sane interview sections had more positive feelings. These numbers can be better observed in Figure 8. In terms of arousal, most fragments (more than 62% in both groups) were categorized into the most neutral section, as Figure 9 shows.
The percentages of valence and arousal are similar in audio, even seen that the interview chunks were much less; sick patients had negative valence in 47% of the fragments and a neutral arousal in 69% on them. In sane patients, the negative emotions were around 12%, and the neutrality of arousal more prominent, with almost 75%. This can be seen in Figures 10 and 11.

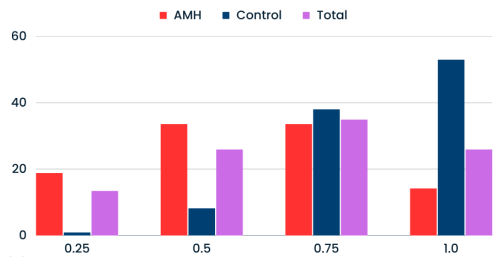


Figure 8: Percentages of valence in text data for each data group

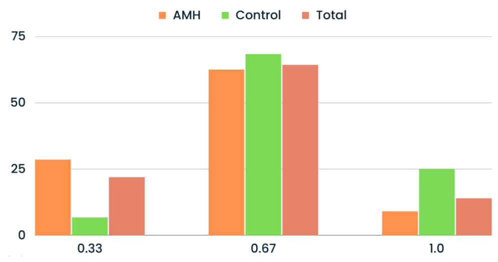


Figure 9: Percentages of arousal in text data for each data group

As there is a clear correlation between valence, arousal and mental illnesses, it was calculated through the Pearson coefficient, defined by the following formula [17]:

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2} \tag{5}$$

In this equation, two values are computed with respect to one another, through n points of the sample. Calculating this coefficient gives us an approximate 46% absolute correlation

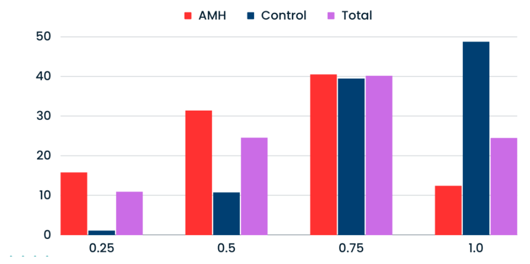


Figure 10: Percentages of valence in audio data for each data group

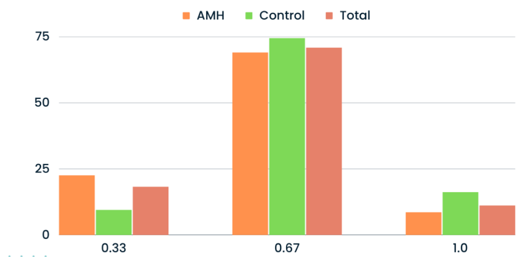


Figure 11: Percentages of arousal in audio data for each data group

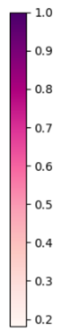
between an illness and the valence in text data and 43% in audio data, it suggests that such a relation must be considered in order to diagnose a patient. The relations between illness presence and different valence and arousal values can be clearly seen in Tables 1 and 2 for both the text and audio data. The most strong one suggests a connection between valence and mental illness, but it is clear that, in general, a relationship between emotion and mental illness has to be taken into account.

	Illness	Valence	Arousal
Illness	1.0	-0.46	-0.29
Valence	-0.46	1.0	0.23
Arousal	-0.29	0.23	1.0

Table 1: Correlation for text data

	Illness	Valence	Arousal
Illness	1.0	-0.43	-0.18
Valence	-0.43	1.0	0.22
Arousal	-0.18	0.22	1.0

Table 2: Correlation for audio data



4 Development

Firstly, the data had to be organized in order to link the emotional data and the illness label with the input text and/or audio. The audio data was already somewhat tidy, but the text data had to be manipulated to link all interview chunks to the corresponding emotions.
Before starting the experiments, four interviews, two pertaining to the control group (Control3 and Control6) and two identified as illness-bearing patients (AMH4 and AMH25), were separated in order to form a test group that would be used in the final stage, so as to prove the veracity and applicability of the models. These interviews were not used in training nor in validation, in order to make the testing more fair.
These interviews were also chosen so as to alter as little as possible the valence and arousal percentages; in text, the test positive valence percentage was less prominent than in the training data, and although neutral arousal remained mostly neutral, it had more prevalence than in training data. This can be seen in Figures 12 and 13. In audio, the differences were slightly larger, as it can be seen in Figures 14 and 15. This data choice is more similar in valence than in arousal so it might perform better in valence classification than in arousal classification.

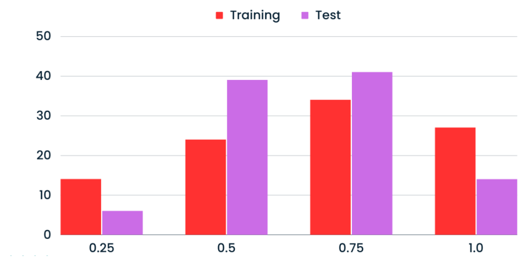


Figure 12: Percentages of valence in text data for training and test

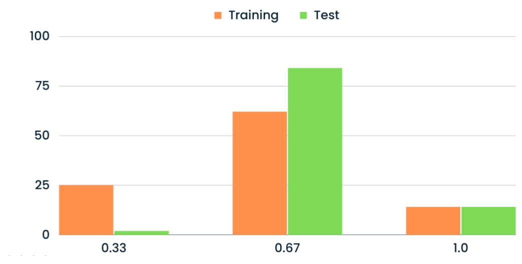


Figure 13: Percentages of arousal in text data for training and test

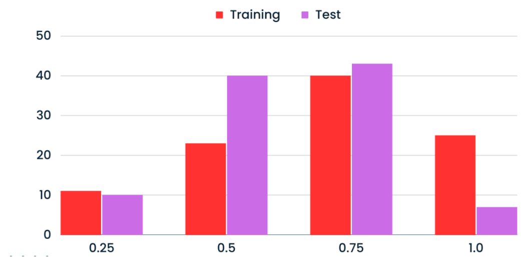


Figure 14: Percentages of valence in audio data for training and test

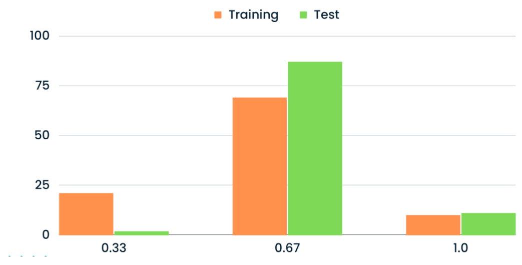


Figure 15: Percentages of arousal in audio data for training and test

For each experiment, a validation set of size 10% was also used. This was chosen randomly and served to check the validity of the model throughout the tuning process. The test data was only used on the final algorithms.

4.1 Experiment definition

In this section, a general outline of the programs that were developed will be defined[1]. The different architectures that were developed in this work will also be presented.
The model building process started with the selection of python packages, which included numpy [18] and pandas [19] for mathematical computation and scikit-learn [20], keras [21] and transformers [22] for machine learning functions and classes. After this, a GPU device from a university server was selected to process the data faster. When these prerequisites were filled, the data was uploaded, the test group separated, and the relevant data was selected. Both the input data (text, audio or both) and the label that was intended to be predicted (illness, valence, arousal, or all of them) had to be picked.
The text or audio, which would be the input data, had to be preprocessed in order to feed it to the networks. Text data was tokenized, which means each word was transformed into a number so that it could be processed numerically, while audio data was transformed using processors that extracted relevant information from it. Afterward, it was necessary to shuffle and split the data into train and validation groups.
Once the data was ready, the model was created with its hyperparameters, and a training loop was defined. The data was then fed to the algorithm in batches. After this, the the validity of the model was checked.

Three different model architectures were developed. Firstly, a fully trained model based on BERT or HuBERT was trained in order to process text and audio data, respectively[2]. Secondly, a model based on the pretrained BERT and HuBERT algorithms was developed, and this was used to process not only text or audio, but also their combined input. Finally, a multitask algorithm that computed illness detection and emotion classification at the same time was developed.

4.2 Fully trained models

Firstly, the models that used text or audio separately were developed.

4.2.1 Text processing

Neural networks based on text were developed using BERT. In the Hugging Face API (*Application Programming Interface*), different models and networks are listed, with simple examples, in order to see the functioning and applications of the networks. I fundamentally used BertForSequenceClassification, a network with a linear classifier as a last layer, in order to perform classification tasks.
An initial configuration with a learning rate of $2e-5$ and an Adam epsilon of $1e-8$ was used for the BERT model. The training was initially done in batches of 32 inputs throughout 3 epochs.
This configuration gives the results shown in Table 3 for validation. In the case of 4-labeled

[1]Some of these programs, particularly the final ones, are included as an appendix.
[2]This code was mostly based on kaggle code https://www.kaggle.com/code/praveengovi/classify-emotions-in-text-with-bert

valence detection, the results presented are not limited to the 'direct' 4-labeled outcome. The second value corresponds to the classification after simplification, where the negative labels are merged into a single label and the positive labels into another one.
It is clear that hyperparameter tuning and an oversampling technique, specially in arousal, could lead to a much better performance. Examining the results closely, it could be seen that arousal fitted only the most prevalent class, the neutral one. Therefore, until an oversampling method was introduced, arousal was not taken into consideration for fine-tuning.

	F_1 macro average
Illness	0.71
2-labeled valence	0.59
4-labeled valence	0.31
	0.61
Arousal	0.25

Table 3: Initial validation scores for text-based models

4.2.1.1 Tuning process

In order to enhance the performance of the program, different hyperparameters and variables were tweaked and tested.

BERT models

The data was trained on the BERT base model, as it has fewer instances than the large one and is thus faster. Nonetheless, three different variants were applied at the beginning: the cased model, the uncased model, and the uncased model for emotion detection. In general, there was no significant difference, although the cased set worked a bit better than the other two. This was selected to be used hereunder.
The BERT large model also could have been used in order to see whether the results improved. As it was far more complex and slow, and thus more difficult to fine-tune, some tests were made, but the results are not included in this work. It suffices to say that training more complex models is much more time- and space-costly, and that the hyperparameter tuning is a much more delicate process.

Batch size

The first hyperparameter to be taken into account was the batch size. As it alters the number of trained instances per second, increasing it makes the algorithm faster, but also slightly more unstable. I tested both 16 and 32 batch sizes, both of them are recommended for the BERT model, with the smallest one obtaining slightly better results, as it can be seen on Table 4.

	16	32
Illness	0.72	0.72
2-labeled valence	0.61	0.60
4-labeled valence	0.36	0.32
	0.62	0.63

Table 4: Scores for text-based models depending on batch size

Data choices and oversampling

So as to make the results more practical, the input data had to be cleansed. The aim was to remove the irrelevant data. There were plenty of sentences that gave no real insight of valence or illness, as they were monosyllabic: ”yes”, ”no”, ”mmm”, or even some instances like ”⟨silence⟩”. They constituted roughly one third of the data I had been using, and it confused the training algorithm, as the same word could be labeled as negative and positive in different instances. To avoid this, I removed every one-worded sentence in the data group. As this left me with only two thirds of the original data, I applied the SMOTE technique in order to fabricate new data points[3]. This not only creates more data, but also allows the ratio on the dataset to change, in order to balance the difference between them. As it significantly improved the algorithms' performance, from this now on, SMOTE was applied to all algorithms.

Number of epochs

Altering the number of epochs in which the data would be trained in could lead to more improvement in the results. Table 5 shows the results. It can be seen that, generally, the best results are achieved with 4 epochs.

	2	3	4
Illness	0.83	0.81	0.84
2-labeled valence	0.79	0.81	0.81
4-labeled valence	0.35	0.34	0.47
	0.70	0.75	0.73
Arousal	0.57	0.56	0.59

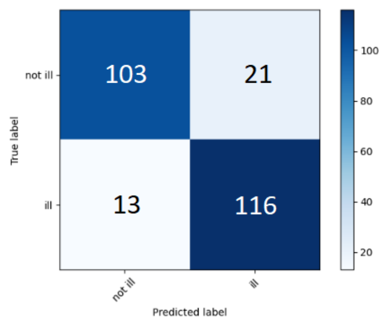
Table 5: Scores for text-based models depending on number of epochs

Learning rate

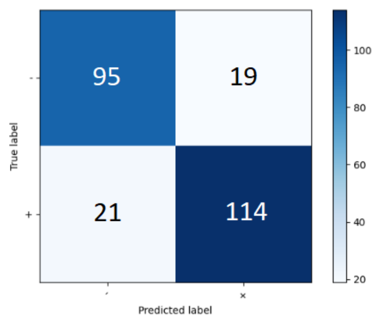
In order to solve the decreasing accuracy problem, the learning rate was altered. Both the 2-labeled valence and the illness detection were used in order to choose the best possible starting value. Arousal was not taken into consideration for choosing the learning rate, seen as in previous instances valence had had much better results. 2-labeled valence detection was also chosen over 4-labeled valence, given its better performance up until now.

[3]Oversampling was later tasted and seen to have slightly worse results.

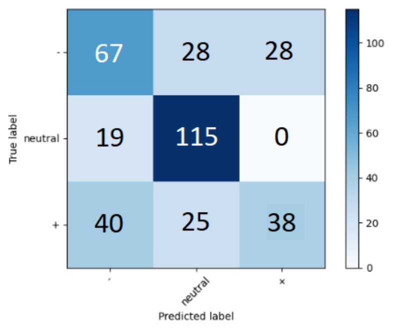
Picking the same value for both tasks proved to be difficult, as a constant or increasing performance should be needed in both valence classification and illness detection. Decreasing the learning rate from $2e-5$ to $14e-6$ improved the illness detection, but worsened the valence classification; decreasing it to $1e-5$ worsened illness detection, but improved valence classification.
After many trials, a learning rate of $11e-6$, close to the one that improved valence classification, was finally chosen. The F_1-score macro average of the valence classification was 0.84 (the starting value being 0.81), and the illness detection had a value of 0.87, compared to the previous 0.84. Even if the results in itself are not that better, the algorithm did reach its best point in the last epoch, which was the main objective.
With this decision, the final algorithm for the text-based fully trained model was ready. The confusion matrices for the illness detection and the 2-labeled valence classification are shown in Figure 16. Arousal was also computed at this point, and it obtained a F_1-score macro average of 0.58 with this final algorithm, as can be seen in the third image of Figure 16; the 4-labeled valence classification gave a F_1-score of 0.51, which with clustering went up to 0.76. As this result is worse than the one with the direct 2-labeled classification, the final confusion matrix is not included.



Illness detection

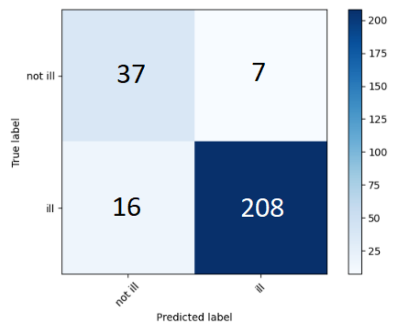


Valence classification

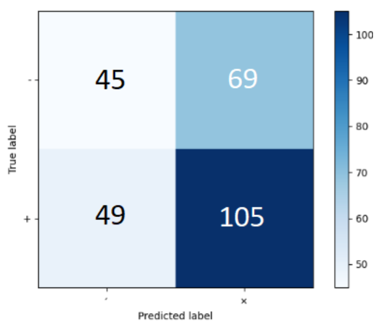


Arousal classification

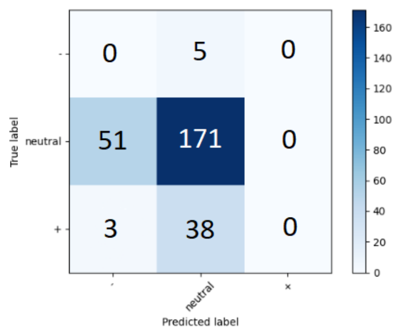
Figure 16: Final validation confusion matrices for text-based models



Illness detection



Valence classification



Arousal classification

Figure 17: Final test confusion matrices for text-based models

4.2.1.2 Testing

The final algorithms were ready, and so I retrieved the four interviews I had previously left aside and I used them to calculate the reliability of the models in patients not used for model training. Worse results were expected, but even so, it was surprising to see the great difference.
Firstly, the test of the original neural networks, the one with no hyperparameter tuning, was calculated. The 2-labeled valence classification obtained a 0.54 F_1-score macro average and the illness detection had a 0.77 F_1-score macro average. The arousal obtained a more humble 0.30 score. Then, the improved algorithm was tested. The final scores are recorded in Table 6. Throughout this work, the final test scores will be highlighted in a green color if they obtained the highest score of each classification when comparing it with the pertinent models.

	Validation	Test
Illness	0.87	0.86
2-labeled valence	0.84	0.54
4-labeled valence	0.51	0.38
	0.76	0.55
Arousal	0.58	0.26

Table 6: Final scores for text-based models

In the case of illness detection, a 0.86 F_1 score was obtained in the test set[4]. This result can be seen on the first image in Figure 17.
Arousal classification was trickier, as it involved three groups instead of being binary. Moreover, it had not really been used in the learning rate choice, so the final algorithm was suboptimal. In testing, it obtained F_1-score macro averages of 0.26. This is due to the fact that the non-balanced test data contains very few non-neutral values, so each mistake is costly. This can be seen on the third image of Figure 17. The F_1 weighted average was 0.65, and the accuracy 0.64, so these results are not as bad as they seem; they are just overwhelmed by the quantity of data in the neutral section for testing. However, it can be seen that this arousal model is not very applicable to real-life subjects.
The valence results were not so good either. 4-labeled valence classification showed non-optimal conditions even in validation comparing to its binary counterpart, even if the final 4-labeled result was clustered to only 2 labels. Moreover, even though validation gave great results in binary classification, testing had a F_1-score macro average of 0.54, as can be seen on the second image in Figure 17. It is clear that this model has no real-world applicability if the emotions of the person have not previously been studied.

[4]If the data was not discarded by length, a F_1-score macro average of 0.79 would be obtained. The theory that one-word sentences are not useful for text classification stands.

4.2.1.3 Error management

In order to visualize where the valence error could come from, several trials were made. Firstly, the control and ill groups were divided to see whether the emotions coming from each of them were so different that the algorithm found it confusing. This did seem to be the problem, to some extent; the control group had mostly positive emotions, as seen in the section 3, and in fact the test control patients had no negative emotions whatsoever, so that could be a decisive factor. The data obtained a high score in control patients, having a F_1-score macro average of 0.93, but the AMH patients scored 0.49 in testing. Most interviews partain to the AMH group, but that SMOTE was implemented in order to try and balance the data, so it makes sense that the algorithm will get confused when the test data is imbalanced. Maybe the use of SMOTE is unjustified in this case, as the classes were somehow balanced, and this is certainly something to be taken into account.
Therefore, SMOTE was removed, then, to no avail; a worse validation F_1-score macro average was obtained, 0.69, which was logical, but the test data still fell way behind, at 0.56. This shows, however, more similarity between validation and test; the result can simply be not that good, and maybe overfitting is happening when SMOTE is applied to the training data. The same happens when SMOTE is not applied to arousal classification either; accuracy and weighted average were still good, but both validation and test F_1 macro averages are disappointing, even if they had more similarity.
It seems clear, however, that apart from the class imbalance, there is a strong overfitting over the train and validation data. In order to prove this, a cross-validation-like validation technique was manually made, with 5 different test data selected, as seen in Table 7. They were selected to have similar train and test valence and arousal scores[5]. By doing the average of each one, a F_1-score macro average of 0.796 in validation and 0.532 in test data was obtained.
Therefore, although there is a bit of overfitting to the validation data, the real problem seems to be the great difference between both groups, and the possibly overfitting caused by SMOTE.

	1	2	3	4	5	Average
Test interviews	C3, C6, A4, A25	C17, C23, A10, A23	C8, C19, A2, A29	C21, C22, A9, A18	C7, C25, A19, A21	
Validation	0.84	0.77	0.77	0.80	0.80	0.796
Test	0.54	0.61	0.52	0.45	0.54	0.532

Table 7: 5-folds validation of valence classification. Due to lack of space, AMH has been referred to as A, and control group as C.

Finally, the phrases that created conflict in the test data were analyzed. Decoding the tokenized vectors, the phrases that were actually negative but that were labeled as positive were taken into account, as that seemed to be the main problem. Most of them were quite neutral-sounding, although some were phrases that out of context could possibly be detected

[5]Although this seems to be the logical choice, and the one recommended by the bibliography, the groups with less similarity to the training data had a better score in most of the cases.

as positive, like expressions of gratefulness and luckiness, like *I'm very lucky to be in that situation* or *well thank you very much cause I'm trying*, or even *my mum provided a lot of love*; of course, that could depend a lot on the tone of the patient or the context. There were also phrases that could be labeled as negative, such as allusions to ex relationships, to not fitting in, and preferring their own space: *my ex-girlfriend and stuff I would've just been close to*, *so it doesn't feel like I'm intruding now or I've overstayed my welcome*, etcetera. However, I do believe most of these sentences could be better classified with the audio analysis of the work, because they do not carry real emotional meaning, like *I'd say some of the time*, *I'm staying at her house*, or *um it does*. With so few instances, however, it is difficult to say that this is the problem.

4.2.2 Audio processing

Audio-based networks were developed following the text model, and taking into account that a HuBERT model had to be used instead of the BERT one. The inputs, too, had to be processed differently: the audio in wav format had to be converted to a more suitable array of numbers by feeding it to a tokenizer-like preprocessor.
Since the beginning, some classifiers and preprocessors, like GeMAPS (Geneva Minimalistic Acoustic Parameter Set), UniSpeech, and Opensmile, were discarded for lack of time and/or appearance in previous works [4], [5].

4.2.2.1 Tuning process

As in the text models, hyperparameters had to be chosen in order to fit the data as much as possible without risking overfitting. Previous experience and results were extended so as to avoid spending so much time in trials. For instance, SMOTE was also applied since the first experiments, and a unique HuBERT pretrained model was used (the facebook wav2vec 2.0 base model, trained in 960h of audio data, specialized in keyword spotting: "superb/hubert-base-superb-ks"). The large model of HuBERT also exists, but given the length of the audio data (each of them having 3500 instances, when text data was less than 300), the GPUs I used to train the model could not handle both the large dataset and the large model at the same time, so it was discarded.
In the beginning, 4 epochs, a batch size of 16, a learning rate of $11e-6$ and the wav2vec feature extractor were used.

Batch size

Batch size was chosen at the beginning to see whether a 16 or a 32 batch size would be better. It was validated on the four classification tasks and the results can be seen in Table 8. Even though the results are very similar, the 16 batch size was selected, seen that its results were slightly better and that the text model had the same batch size.

Number of epochs

In order to optimize the results as much as possible, different epochs were used to train the model. As the BERT model recommends between 2 and 4 epochs, and HuBERT is largely

	16	32
Illness detection	1.00	1.00
2-labeled valence	0.70	0.68
4-labeled valence	0.53	0.49
	0.73	0.72
Arousal	0.75	0.75

Table 8: Scores for audio-based networks depending on batch size

based on it, these numbers were used for the trials. The results are shown in Table 9, with 4 epochs being the most logical choice for the final algorithm.

	2	3	4
Illness detection	1.00	1.00	1.00
2-labeled valence	0.71	0.68	0.70
4-labeled valence	0.44	0.50	0.53
	0.71	0.72	0.73
Arousal classification	0.69	0.74	0.75

Table 9: Scores for neural networks depending on number of epochs

Learning rate

Audio networks showed to be much more capricious than text-based networks. The previously selected $11e-6$ learning rate was not optimal for audio data, as it created overfitting to one of the labels. Thus, after a bit of back and forth, a learning rate of $2e-5$ was chosen. This choice was then contrasted for two different preprocessors.

Feature extractor

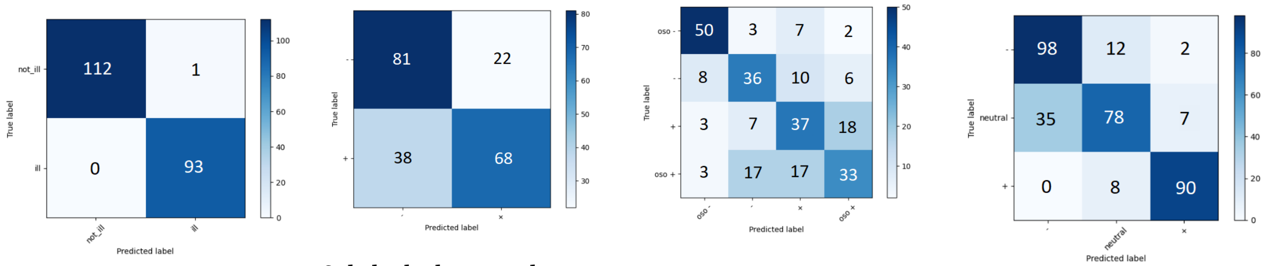
Two different preprocessors were used: Wav2Vec2FeatureExtractor and AutoFeatureExtractor. The second one gave slightly better results, which was to be expected, as it was pretrained in the same data as the HuBERT model. However, the downgrade is that, because it was pretrained in other sampling frequency, the dataset I used had to be downsampled, and thus the algorithm was much more time-costly for this variant. Seeing the difference between the results in Table 10, which are trained with the updated learning rate value, the trade-off may not be worth it.

4.2.2.2 Testing

The final results for audio-based neural networks are recorded in Table 11. These results are only comparable with the text scores in Table 6, and audio obtained better results in every classification. Illness detection could not be used for tuning hyperparameters, as the validation data had almost always a 100% exactitude when detecting the label. This result is given by the left confusion matrix in Figure 18, and was unaltered through most

	wav2vec	AutoFeatureExtractor
Illness detection	1.00	1.00
2-labeled valence	0.70	0.71
4-labeled valence	0.53 0.73	0.61 0.79
Arousal	0.75	0.81

Table 10: Scores for audio-based networks depending on feature extractor



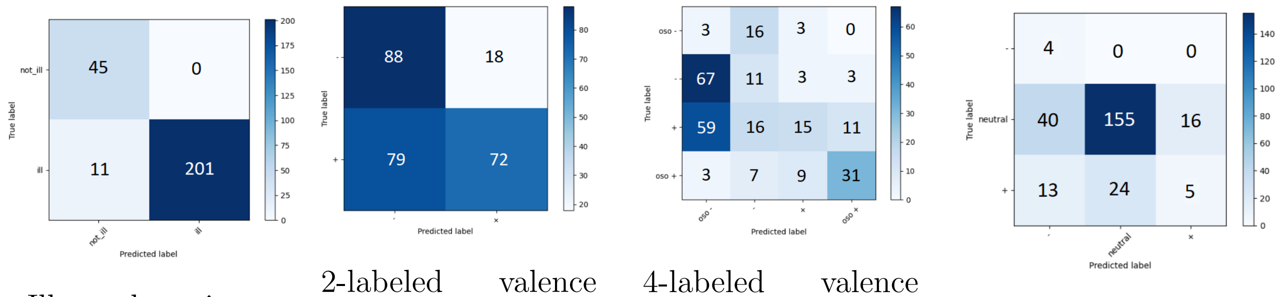
Illness detection

2-labeled valence classification

4-labeled valence classification

Arousal classification

Figure 18: Final validation confusion matrices for audio-based models



Illness detection

2-labeled valence classification

4-labeled valence classification

Arousal classification

Figure 19: Final test confusion matrices for audio-based models

	Validation	Test
Illness	1.00	0.93
2-labeled valence	0.71	0.62
4-labeled valence	0.61 0.79	0.27 0.63
Arousal	0.81	0.36

Table 11: Final scores for audio-based networks

experiments. It was later tested, as the people in the validation set were the same as in the train set, and this could be decisive in audio, because of the similar tone of the voice; but the test data still gave a F_1-score macro average of 0.93, as seen on the left image of Figure 19, so we can say the audio neural network for illness detection is quite good, even if it's not infallible. The starting algorithm gave a test macro average of 0.70 for illness detection, so we seem to have found a good non-overfitting spot comparing to the difference between validation and test in that beginning.

In 2-labeled valence detection, a 0.71 F_1-score macro average was achieved in the validation data, while in 4-labeled valence detection, a F_1-score macro average of 0.79 was achieved through clustering, even if the 4 labels gave a direct macro average of 0.61. In test data, the macro average worsened, but to a more logical extent than in text; 2-labeled valence gave a macro average of 0.62, and 4-labeled valence gave results of 0.27 and 0.63 without and with clustering. In this case, computing the 4-labeled valence and then clustering gave a slightly better result. The comparison between 2-labeled valence classification and 4-labeled classification can be seen in Figure 18 for validation and 19 for test. Taking into account that the starting algorithm gave a 0.46 test F_1-score macro average for 2-labeled valence classification, and 0.17 and 0.39 for 4-labeled valence without and with clustering respectively, we can say that even though the validation data was not much improved, the testing data obtained better results for our fine-tuned algorithm, and thus the model seems to be better.

In arousal detection, a 0.81 F_1-score macro average was achieved in the validation data, and in test data, a macro average of 0.36 was achieved. These results can be contrasted in Figures 18 and 19. The contrast between validation and test is much more notable in this case, as previously described; the central label is much more prominent in reality, and thus, when testing in real data, the other two instances worsen the macro average. Weighted macro average would give 0.68, which is much more logical and acceptable. The testing of the original network gave us a F_1-score macro average of 0.27 with a weighted macro average of 0.62, which is worse than the final one.

Error management

The issue was the same as before; in emotion detection, and specially in arousal, results in validation seemed to be much better than in testing. There may be two problems here. On the one hand, overfitting may be happening, although it is true that the problem seems to be less prominent than in text processing. In audio, validation scores are much more close to the final test scores.
On the other hand, the test set may not be appropriate for every task; the valence and arousal percentages in the audio data are slightly more imbalanced than in the text data. However, the error is not as noticeable as in the text data either. However, computing the valences separately for each group confirms that the AMH group is the one that causes the classification mismatch, both in validation, with a 0.66 score, and in test, with a score of 0.49. The control group scores 0.98 in both instances.

4.2.3 Model comparison

The results summarized in Table 12, where the best results for each case have been highlighted in green, seem to show that the performance has improved from text processing to audio processing. The emotional information contained in the tone of the voice or the pace of the

speech cannot be contained in textual data, which is more semantic. Moreover, as mentioned earlier, the audio data seems to have a lower disagreement between validation and test data, thus making it, in principle, more applicable.
It is expected that the combination of the two inputs, text and audio, will improve the classification performance, as more features will be simultaneously taken into account.

	Text		Audio	
	Validation	Test	Validation	Test
Illness	0.87	0.86	1.00	0.93
2-labeled valence	0.84	0.54	0.71	0.62
4-labeled valence	0.51	0.38	0.61	0.27
	0.76	0.55	0.79	0.63
Arousal	0.58	0.26	0.81	0.36

Table 12: Final results for fully trained algorithms

4.3 Pretrained models

The pretrained models allow text and audio data to be preprocessed through their corresponding BERT or HuBERT algorithm, and then fed to a functional classifier. This allows to use both audio and text data for classification at the same time.
All of the models are based on the same multi-layer perceptron, which was fine-tuned using multi-modal data.

4.3.1 Text processing

In this model, an algorithm was built to process only textual data through the pretrained BERT model. Then, this input was fed to a classifier. The results that were obtained are shown in Table 13.
As, in general, the results are worse than in multi-modal processing, and also due to lack of space, the corresponding confusion matrices have not been included.

	Validation	Test
Illness	0.79	0.68
2-labeled valence	0.70	0.55
4-labeled valence	0.47	0.25
	0.73	0.54
Arousal	0.64	0.35

Table 13: Scores for the pretrained text-based algorithm

4.3.2 Audio processing

In this model, the audio data was preprocessed though the HuBERT pretrained model and then fed a classifier neural network. Thus, the results seen in Table 14 were obtained.

As the results are worse than both in text and in multi-modal processing, and also due to lack of space, the corresponding confusion matrices have not been included.

	Validation	Test
Illness	0.95	0.86
2-labeled valence	0.78	0.57
4-labeled valence	0.74	0.30
	0.96	0.53
Arousal	0.88	0.33

Table 14: Scores for the pretrained audio-based algorithm

4.3.3 Multi-modal processing

In this section, a model that would take both text and audio into account was developed.

Firstly, audio and text data had to be combined. Text had more instances, as long phrases had been split into more manageable sizes; for this experiment, I recombined them, and linked each of them to the corresponding audio. Taking the documents for audio and text data I had previously developed, this task was not difficult, but proved to be time-consuming.
On the algorithm itself, audio data was first processed with the wav2vec transformer, but then sent through the pretrained HuBERT model. The same thing was done with text-based data; first it was tokenized, and then fed to the pretrained BERT model. The pretrained text model was selected to be the cased model, as it had shown better performance previously.
After this preprocessing, the output of each sentence was combined in a single array containing both audio and text data. This combined array was the input of a model that would classify the pertinent output; valence, arousal or illness.

4.3.3.1 Tuning process

Several models were developed in order to attain the best classification possible. These were much simpler and quicker to train than both HuBERT and BERT. Other hyperparameters, such as batch size for feeding the pretrained models and attention mask presence, and audio feature extractor, were also taken into consideration. SMOTE was applied since the beginning, and no phrases were discarded for their lengths, because combining phrases and audio lead to a logical interpretation anyway.

Model selection

Firstly, and taking into account that the hyperparameters would change significantly due to the type of model selected, one of them was chosen. Three initial models were taken into account: SVCs (Support Vector Classifiers), which are linear SVMs, a multi-layer perceptron for classification with initial values of 2 hidden layers with 50 neurons each, and a self-constructed keras sequential neural network with 2 hidden layers with 20 neurons each. The differences between the multi-layer perceptron and the neural network was not so noticeable, but the SVC performed worse than the other two in almost every instance. Watching Table 15, the multi-layer perceptron was selected, as it was easier to tune than the complex neural network, and it showed better results in most instances. It is noticeable that, even without

	Multi-layer perceptron	Neural network	SVC
Illness	0.95	0.94	0.86
2-valence	0.82	0.83	0.67
4-valence	0.73	0.63	0.50
	0.86	0.82	0.69
Arousal	0.92	0.82	0.60

Table 15: Scores for multi-modal classification depending on model

any fine-tuning, the results for the multi-layer perceptron are quite good on validation data. It is to be expected that the test scores will decrease. From now on, I will refer to the selected model, the multi-layer perceptron, as 'network'.

Batch size

Before fine-tuning the model layer and neuron sizes, the batch sizes for feeding the pretrained models was chosen. As before, the recommended 16 and 32 sizes were contrasted. In this case, the biggest one proved to be a quite better choice, as Table 16 shows, though 16 did perform better on illness detection.

	16	32
Illness	0.97	0.95
2-valence	0.77	0.82
4-valence	0.66	0.73
	0.76	0.86
Arousal	0.86	0.92

Table 16: Scores for the multi-modal network depending on batch size

Model layers and neurons

Then, the model had to be adjusted. Different number of layers and neurons were taken into account, but only to a certain extent. The default model with one hidden layer of 100 neurons was used, along with several two hidden-layered networks and three-layered networks taken from the best results of the two-layered models: $(30, 30)$ and $(50, 50)$. The validation results can be seen in Table 17.
The models with three hidden layers seemed to fit the data worse than the two-layered networks. At the end, the $(30, 30)$ model was chosen for three reasons: it showed slightly better performance in illness, the performance difference in emotion detection was not very

high, and emotion tends to overfit, so maybe better results were obtained by reducing the validation scores a bit, like happened in audio versus text. The general architecture of the

	Default (100)	(10,10)	(20,20)	(30,30)	(40,40)	(50,50)	(100,100)	(30,30,30)	(50,50,50)
Illness	0.96	0.95	0.94	0.96	0.95	0.95	0.95	0.93	0.95
2-valence	0.82	0.73	0.78	0.81	0.79	0.82	0.81	0.83	0.81
4-valence	0.71	0.71	0.71	0.66	0.65	0.73	0.71	0.67	0.72
	0.82	0.84	0.84	0.82	0.79	0.86	0.84	0.85	0.85
Arousal	0.88	0.83	0.87	0.88	0.83	0.92	0.96	0.87	0.82

Table 17: Multi-modal network scores depending on layer and neuron quantity

selected model was shown previously on the left part of Figure 6, with the output classifier changing for each task.

Attention masks

The $(30, 30)$ model was then processed again applying attention masks to the BERT text inputs. I hadn't previously applied them due to the computational cost and seemingly unhelpful task, seen as they only say whether the ID of a word is 0 or not and that text does not seem to contribute a lot of data. However, applying them improved the results slightly. This may be due to the fact that it can ease the computation of outputs in both the BERT and HuBERT models by simply asking them to ignore those data points. The difference in scores can be seen in Table 18.

	Without masks	With masks
Illness	0.96	0.96
2-valence	0.81	0.85
4-valence	0.66	0.73
	0.82	0.85
Arousal	0.88	0.89

Table 18: Multi-modal network scores depending on attention mask presence

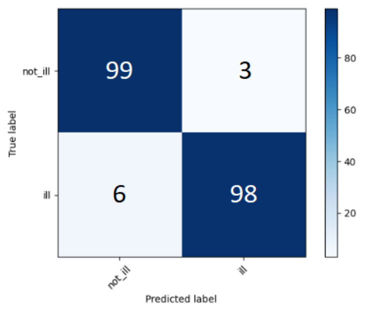
Feature extractor

Lastly, the AutoFeatureExtractor was applied instead of the wav2vec one. As I explained in the audio section, this one takes much more time to compute, and thus only the final algorithm was given to it so as to avoid testing many different options with it. The results did improve a bit, but not very noticeably, and thus the wav2vec model, which was a bit more humble in scores, but compiled much faster, was chosen for the final algorithm [6]. The validation scores for the two processors can be seen in Table 19.

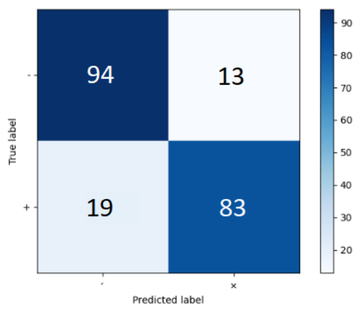
[6]Later, the two variants were tested, and wav2vec showed slightly better results in testing than AutoFeatureExtractor did.

	AutoFeatureExtractor	wac2vec
Illness	0.96	0.96
2-valence	0.85	0.85
4-valence	0.77	0.73
	0.86	0.85
Arousal	0.90	0.89

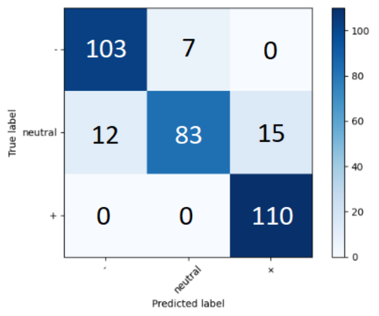
Table 19: Multi-modal network scores depending on feature extractor



Illness detection

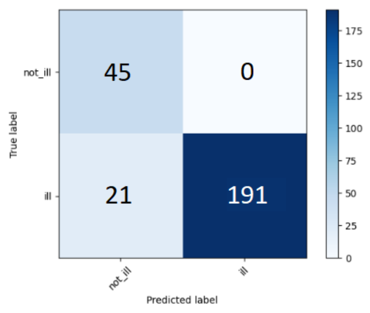


Valence classification

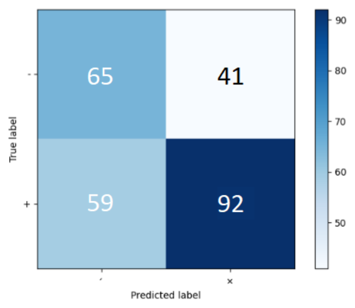


Arousal classification

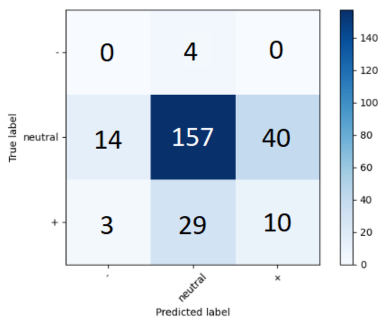
Figure 20: Final validation confusion matrices for multi-modal models



Illness detection



Valence classification



Arousal classification

Figure 21: Final test confusion matrices for multi-modal models

4.3.3.2 Testing

The initial and final algorithms were then tested to see how the models performed in unseen data. The $(50, 50)$ multi-layer perceptron was considered to be the first model, with SMOTE, a batch size of 32, no attention masks and wav2vec processing. For the final model, the validation versus testing results were not surprising; emotion detection worsened, specially arousal. These results are summarized in Table 20.

	Validation	Test
Illness	0.96	0.88
2-labeled valence	0.85	0.61
4-labeled valence	0.73	0.31
	0.85	0.57
Arousal	0.89	0.33

Table 20: Final results for multi-modal algorithm

Illness detection scored 0.88 in testing, as can be seen in the left images of Figures 20 and 21. Initial testing would have given 0.85.
In testing, the valence scores decreased to 0.61 for 2-labeled valence and 0.31 and 0.57 for 4-labeled valence. In the initial model, these results were not worse, at 0.62 for 2-labeled valence and 0.31 and 0.58 for the 4-labeled valence. It seems like, in general, direct binary classification is better than the clustering technique. However, it is clear that there has not been an improvement in valence classification for the test data, even when fine-tuning. Results for 2-labeled valence can be seen in the middle of Figures 20 and 21.
Arousal had a test average score of 0.33 and a weighted average of 0.68, which can be seen in the right images of Figures 20 and 21. The initial arousal model had a testing score of 0.32, with 0.65 being the weighted macro average, so only a slight improvement has been made in this case.

Error management is not included because it is the same as before; the AMH and Control patients show very different valences, and arousal is too polarized.

As this model was trained on a multi-layer perceptron, but not trained in BERT and HuBERT models themselves, comparing results with previously obtained data did not make much sense. Therefore, the same algorithm was built using only text and audio data, to see how it affected both validation and test scores.

4.3.4 Model comparison

Watching the three input results side by side in Table 21, it can be seen that most of the scores are better for the multi-modal algorithm, even if the arousal is slightly better in text processing. However, the weighted average was better in multi-modal processing, at 0.68, comparing to 0.66 on text processing. The peak scores have been highlighted in green. It can also be seen that the audio scores are really similar to the multi-modal scores, as was expected; more information is given by audio data, even if text data also contributes, seemingly in a positive way. However, the difference between validation and test in emotion detection persists; even if very good validation scores have been obtained, the testing seems to not improve from approximately %60 in valence, and %30 in arousal.

	Text		Audio		Multi-modal	
	Validation	Test	Validation	Test	Validation	Test
Illness	0.79	0.68	0.95	0.86	0.96	0.88
2-labeled valence	0.70	0.55	0.78	0.57	0.85	0.61
4-labeled valence	0.47	0.25	0.74	0.30	0.73	0.31
	0.73	0.54	0.96	0.53	0.85	0.57
Arousal	0.64	0.35	0.88	0.33	0.89	0.33

Table 21: Final results for pretrained single-output models

4.4 Multitask models

As mentioned earlier, the final goal would be that of classifying emotion and illness at the same time, to see whether one would help the other, given their correlation. 2-labeled valence was chosen to be detected, as it did not make much sense to detect 4-valence and 2-valence at the same time, and the binary one achieved, in general, better results in testing and was quicker than its multi-class counterpart. It was expected that the results would improve a bit; however, there was a problem, regarding the impossibility to use regular SMOTE in multitask. Therefore, fewer data was used in order to train and validate the data, resulting in slightly worse results than what could be expected. Three different multitask classifiers were constructed: based on text, based on audio, and multi-modal ones. The three of these were constructed using the pretrained models[7] for comparison purposes.
The multi-layer perceptron used in the previous 4.3.3 section was used as a base for the algorithm in the multitask models. The output neural network had to be altered as to classify illness presence, valence and arousal at the same time. In this case, I used the multitask classifier from scikit-learn based on the multi-layer perceptron that was already fine-tuned. Another models, like a multi-lasso linear model or a classifier chain, were also considered, they were discarded due to the lack of compatibility with the current problem; multi-lasso would be a regression and thus not suited for classification, and the classifier chain would not support multi-label models like arousal[8].

4.4.1 Text processing

As in the fully trained and pretrained models, the text-based algorithm was due to have the smaller efficiency of the three input data. This can be seen in Table 22, in which the results for the multitask text model are shown for both validation and test.

[7]Even if it is not included in the main work, a fully trained multitask model could be constructed and is included as an appendix.

[8]Even if it is not part of the main work, a ClassifierChain model for detecting illness and valence was developed, and it is included as an appendix. It obtained the best results in valence testing, and even if the difference was not so big, a mention should be made.

	Validation	Test
Illness	0.70	0.69
2-labeled valence	0.68	0.50
Arousal	0.38	0.38

Table 22: Scores for text-based multitask classification

4.4.2 Audio processing

Applying the pretrained HuBERT model to wav2vec preprocessed audio files gives us the scores mentioned in Table 23. The scores itself are not that good compared to previously obtained results, but the most significant thing about this model is that, although not good, it really matches validation and test.
As the results are generally better for the multi-modal algorithm, the audio and text confusion matrices have not been included.

	Validation	Test
Illness	0.84	0.85
2-labeled valence	0.64	0.55
Arousal	0.41	0.32

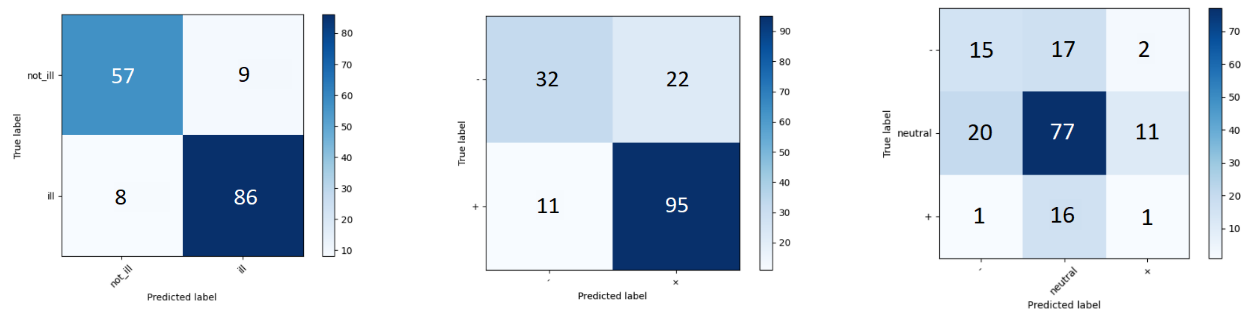
Table 23: Scores for audio-based multitask classification

4.4.3 Multi-modal processing

This multitask algorithm's only change from the original multi-modal model was the output neural network, which was now a multitask multi-class classifier.
The results obtained with this model can be seen in Table 24. The validation and test confusion matrices are also showed in Figures 22 and 23, respectively.

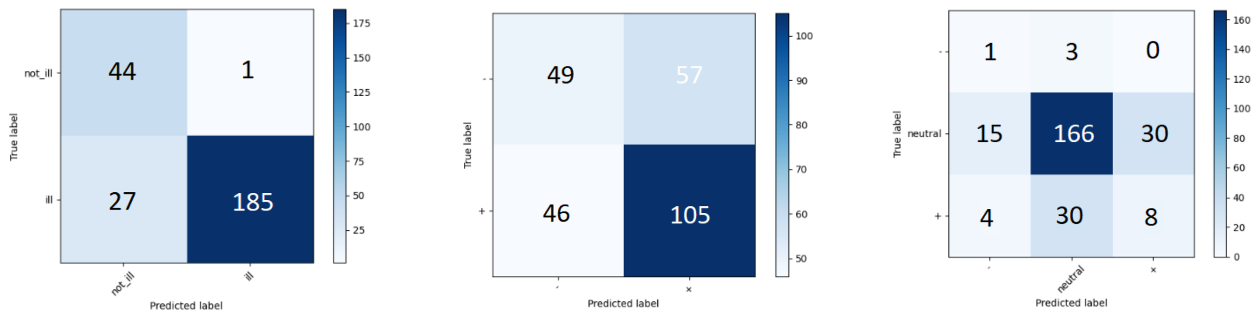
	Validation	Test
Illness	0.89	0.84
2-labeled valence	0.76	0.58
Arousal	0.40	0.36

Table 24: Scores for multi-modal multitask classification



Illness detection Valence classification Arousal classification

Figure 22: Final validation confusion matrices for multi-modal multitask models



Illness detection Valence classification Arousal classification

Figure 23: Final test confusion matrices for multi-modal multitask models

4.4.4 Model comparison

The results for the multitask models, shown in Table 25, are similar to what we had already seen in the other sections, though slightly worse; it seems like training each task on their own obtains a better performance. This may be due to the SMOTE effect, but maybe more fine-tuning of the model could be a way of improving the jointly obtained results.
The multi-modal algorithm seemed to be the best performing model, and it actually showed the most similarity between validation and test. In general, this multi-modal algorithm gives us better scores than the text-based or audio-based multitask algorithms.

	Text		Audio		Multi-modal	
	Validation	Test	Validation	Test	Validation	Test
Illness	0.70	0.69	0.84	0.85	0.89	0.84
2-labeled valence	0.68	0.50	0.64	0.55	0.76	0.58
Arousal	0.38	0.38	0.41	0.32	0.40	0.36

Table 25: Final results of multitask models

5 Conclusions

In this work, mental illness has been detected using textual and audio data. A connection between mental illness presence and emotional data, summarized in valence and arousal, has also been pursued. Even though a relation between them exists, more research is needed, both machine learning-wise and psychologically, in order to understand emotion, specially in people with mental illnesses.
The main conclusion of this work is that the combination of various methods is crucial in order to create a model that performs as well on real data as it does during training. In this work, audio and text data are both important, because each of them has a better performance in a particular section, but it is the combination of both which gives the best performance. Multi-modal algorithms which compute the outputs separately seem to be the ones with the best performance. Additionally, seeing the results, it is somehow hinted that having previous information of a person's emotion makes it easier to monitor them. Therefore, the interaction between machine learning and healthcare workers is crucial in order to achieve the best diagnose and patient treatment. These algorithms could probably never substitute a healthcare worker, at least for now, and specially for emotion detection, but they could definitely help diagnose mental illnesses.

There are many things to take into account regarding data and algorithm choices. First of all, I would like to mention that the data may be too artificial. The control and AMH groups were sufficiently different in ages and environments for it to make an impact in their illnesses and their interview performance. Therefore, there is a great probability that the models I developed would work much worse in the 'real life', especially the ones that detect illness. Using real data also poses challenges like the data being unbalanced and the emotions being more subtle than in artificially created databases.
Another crucial note is the importance of relevant data that is not only representative, but also clean. In this work, having more data would probably improve the results. Overall, it can be said that more data means better performance, when this data is representative.
The problem of emotion detection is also one that has to be taken into account, especially in text models. As it was seen on the error management sections, possibly the significant difference between control and AMH interviewees made the model perform worse than what it would be desirable. The AMH group by itself is also very unpredictable. This requires further studying.

Regarding the algorithms, there are several points I would like to state. Firstly, using pretrained models, so highlighted in neural networks, can be a difficult task. Even though using pretrained models can help you understand more complex models and their functioning, any model, however similar, has to be adapted a great deal out of it so as it fits. The uncased-emotion BERT base model is a great example: although its purpose is to detect emotion, the task in this work is sufficiently different so that using the cased model is preferred instead.
On the other hand, and in a quite different note, I would like to state that the understanding of neural networks does not equate the understanding of all the underlying parameters, transformers and functions. Therefore, this project has, by no means, achieved is greatest

potential. There are simply too many factors that can be wrangled and infinite possibilities of model architectures or performance enhancement functions which had to be discarded since the beginning. Nevertheless, I believe the data was neatly processed to compute an applicable and accurate yet simple model, at least for illness detection.

So as to wrap this work, future lines of work can be recommended. The first would be to create a transformer-like model in order to train the previously processed audio and text data, so that the model applied to this combined data can be trained through and through. Completely trainable multitask BERT and HuBERT models are also something to be taken into account.

Finally, and as mentioned earlier, more data is crucial in order to improve the performance of any model. Data from other mental institutions would be interesting to compute in order to see if the models respond well to people from other places who may have other accents, mannerisms and other characteristics. More data could also be important in order to make a distinction between depression and anxiety, which has not been pursued in this work.

In summary, the models that were developed could be improved by either increasing the complexity of the algorithms, by a more thorough fine-tuning, by understanding the underlying emotional theory, or by increasing the data instance numbers. In general, much work is needed in order to master this field.

6 Bibliography

[1] D. Ravi *et al.*, "Deep learning for health informatics," *IEEE journal of biomedical and health informatics*, vol. 21, no. 1, 2017. https://doi.org/10.1109/JBHI.2016.2636665.

[2] A. L. Glaz *et al.*, "Machine learning and natural language processing in mental health: systematic review," *Journal of Medical Internet Research*, vol. 23, no. 5, 2021. https://doi.org/10.2196/15708.

[3] A. B. R. Shatte, D. M. Hutchinson, and S. J. Teague, "Machine learning in mental health: a scoping review of methods and applications," *Psychological medicine*, vol. 1, no. 23, 2019. https://doi.org/10.1017/S0033291719000151.

[4] I. Zubiaga, I. Menchaca, M. de Velasco, and R. Justo, "Mental Health Monitoring from Speech and Language," in *Proc. Workshop on Speech, Music and Mind*, pp. 11–15, 2022. https://doi.org/10.21437/SMM.2022-3.

[5] I. Menchaca, "Impacto del contenido emocional en un sistema de monitorización de salud mental," Bachelor's Thesis, Euskal Herriko Unibertsitatea, 2021.

[6] R. Pillai, P. Oza, and P. Sharma, "Review of machine learning techniques in health care," in *Proceedings of ICEIC 2019* (P.K.Shing *et al.*, eds.), vol. 597, pp. 41–50, Springers Nature Switzerland, 2019. https://doi.org/10.1007/978-3-030-29407-6_9.

[7] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow.* Sebastopol: O'Reilly Media, 2nd ed., 2019.

[8] ThresholdTom, "Confusion matrix," 2019. https://commons.wikimedia.org/wiki/File:ConfusionMatrixRedBlue.png.

[9] O. Dürr, B. Sick, and E. Murina, *Probabilistic Deep Learning with Python.* Manning, 1st ed., 2020.

[10] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, jun 2002. https://doi.org/10.1613/jair.953.

[11] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, p. 4171–4186, Association for Computational Linguistics, 2019. https://doi.org/10.18653/v1/N19-1423.

[12] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "HuBERT: Self-supervised speech representation learning by masked prediction of hidden units," jun 2021. https://doi.org/10.48550/arXiv.2106.07447.

[13] P. Mehta, “Multimodal deep learning. fusion of multiple modalities using deep learning,” *Towards Data Science*, 2018. https://towardsdatascience.com/multimodal-deep-learning-ce7d1d994f4.

[14] S. Khan, “Multi-task learning with transformers: Transformers with multiple prediction heads,” Sept. 2021. https://medium.com/@shahrukhx01/multi-task-learning-with-transformers-part-1-multi-prediction-heads-b7001cf014bf.

[15] “MENHIR. mental health monitoring through interactive conversations.” https://doi.org/10.3030/823907.

[16] E. A. Kensinger, “Remembering emotional experiences: The contribution of valence and arousal,” *Reviews in the Neurosciences*, vol. 15, pp. 241–251, 2004. https://doi.org/10.1515/REVNEURO.2004.15.4.241.

[17] T. Turney, “Pearson correlation coefficient (r). guide and examples,” 2022. https://www.scribbr.com/statistics/pearson-correlation-coefficient/.

[18] C. R. Harris *et al.*, “Array programming with NumPy,” *Nature*, vol. 585, pp. 357–362, Sept. 2020. https://doi.org/10.1038/s41586-020-2649-2.

[19] Wes McKinney, “Data Structures for Statistical Computing in Python,” in *Proceedings of the 9th Python in Science Conference* (Stéfan van der Walt and Jarrod Millman, eds.), pp. 56 – 61, 2010. https://doi.org/10.25080/Majora-92bf1922-00a.

[20] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[21] F. Chollet *et al.*, “Keras,” 2015. https://github.com/fchollet/keras.

[22] T. Wolf *et al.*, “Huggingface’s transformers: State-of-the-art natural language processing,” 2019. https://doi.org/10.48550/ARXIV.1910.03771.