UCLouvain

École polytechnique de Louvain

Machine Learning for fMRI brain data

Authors: **Simon Iavarone, Nicolas Kerkhofs**
Supervisors: **Olivier Collignon, Michel Verleysen**
Readers: **Remi Gau, Stefania Mattioni**
Academic year 2021–2022
Master [120] in Computer Science and Engineering

Abstract

The goal of this thesis is to explore different Machine Learning techniques in order to improve fMRI decoding on a specific dataset. This is part of the more global research aiming to understand how the human brain works. The examined dataset contains motion direction information induced by visual and auditory stimuli for 23 subjects. The perception of motion direction stimuli is handled by the V5/hMT+ area of the visual cortex, while motion direction for auditory stimuli is encoded by the Planum Temporale (PT). The aim of the trained classifier is to predict motion direction (i.e. left, right, up or down) using spherical Regions Of Interest (ROI) to select useful data. We analysed the effect of various methods such as feature selection, data augmentation as well as the results obtained with different classifiers. In addition to this, we examined the implications of a change in the size of said ROIs and the type of data used (beta-maps or t-maps). Even though we observed some significant accuracy changes in our results, we did not observe major improvements for decoding. However, we noticed both the LR and SVM classifiers behave in similar fashions when trained on our data, LR being the best.

***keywords**: Machine Learning - fMRI - motion direction - V5/hMT+ - Support Vector Machine - Logistic Regression*

Acknowledgements

With these few lines, we would like to thank everyone who supported us along this thesis.
Firstly, we would like to thank our supervisors Michel Verleysen and Olivier Collignon for their guidance and advice whenever we needed it.
We also would like to express our gratitude to our daily contacts Remi Gau and Mohamed Rezk, who answered all our questions, even the dumbest ones, and gave us interest in fMRI decoding. Also, we thank Mohamed for his many explanations of his study and the data and codes he gave us.
Finally, we would like to thank our friends and families for their moral support through this adventure.

Contents

List of abbreviations

ADHD	Attention Deficit Hyperactivity Disorder
ANOVA	ANalysis Of VAriance
BOLD	Blood Oxygenation Level Dependent
ELM	Extreme Learning Machine
fMRI	functional Magnetic Resonance Imaging
GAN	Generative Adversarial Network
GLM	General Linear Model
hMT+/V5	human Middle-Temporal cortex
HRF	Haemodynamic Response Function
KNN	K-Nearest Neighbors
LASSO	Least Absolute Shrinkage and Selection Operator
LR	Logistic Regression
ML	Machine Learning
MRI	Magnetic Resonance Imaging
MVPA	Multivoxel Pattern Analysis
PT	Planum Temporale
RFE	Recursive Feature Elimination
ROI	Region Of Interest
SMOTE	Synthetic Minority Oversampling TEchnique
SVM	Support Vector Machine
TR	Repetition Time

Introduction

Scientists have been trying to understand the human brain for a very long time. In turn, this gave rise to neuroscience, a broad field which focuses on both the structure of the brain and the way it works. In fact, this science nowadays combines many disciplines such as biology, chemistry, mathematical modeling, and even computer science. With the development of these other fields, the tools used by neuroscientists evolved a lot through time. This tool diversity enables researchers to perform a variety of analyses at different scales of the nervous system. While some study the behavior of single neurons or small cell aggregates, others try to understand where and how the brain encodes information such as emotions, behaviors, etc. The problem dealt with in this thesis belongs to this second category.

More precisely, this work takes place in the context of motion direction decoding. This implies a study of the brain's activity when a person is subjected to visual or auditory stimuli that go in a specific direction. Previous research demonstrated motion direction is encoded in two different areas of the brain: PT for audition and hMT+/V5 for vision. It was also demonstrated that hMT+/V5 contains both visual and auditory information (1)! After having made this observation, neuroscientists wondered whether the way information is represented in V5 is the same for vision and audition. In the past, they were able to answer positively to similar questions for monkeys by putting electrodes in their brains (2).

The next step is to answer the question for humans. However, as it is nowadays forbidden to put electrodes on brains (or at least much less encouraged), scientists required a non invasive technique. As a result, the tool used to gather data for such a study is functional Magnetic Resonance Imaging (fMRI). The idea behind this technique is to capture a video of brain activity while a person is performing a task in the fMRI scanner. In this case the task at hand is simply listening to a sound or seeing a cloud of dots on a screen that goes in a specific direction.

fMRI is a whole field in and of itself. It is concerned with both the acquisition and analysis of brain activity. In fact, there are many steps between the capture of

the raw fMRI data and the creation of brain maps, which constitute the analyzable data. When inspecting such data, the objective is to build a mathematical model that describes neural activity depending on external stimuli. This is where Machine Learning can come in.

Machine Learning (ML) is a subset of the Artificial Intelligence field. It focuses on building and analyzing mathematical models through use of data. ML methods enable a computer to learn the model that best fits a dataset. This field comes with a large variety of models, from simple regressions to neural networks. It also provides a lot of different methods to enhance and evaluate the performance of said models.

What is the goal of this thesis? This thesis is based on a fMRI study published by Rezk (1). Our goal is to provide an in-depth exploration of the Machine Learning side of things. The main question is: which ML technique(s) can significantly and reliably improve fMRI decoding, in the context of motion direction decoding in V5?

The chapter 1 of this document presents the knowledge necessary to understand the rest of this work for both the fMRI field and Machine Learning. Right after, comes the state of the art in chapter 2 which presents various studies of Machine Learning with fMRI data. Following this, comes the chapter 3 that describes the dataset used for the experiments presented in this thesis. Before diving into said experiments, the implemented Machine Learning framework will be explained in depth in chapter 4. Chapter 5 details the performed experiments as well as their results and the corresponding interpretations before concluding.

Chapter 1

Background knowledge

In this chapter, we will give the basis needed to understand the following thesis. It explains what is functional Magnetic Resonance Imaging (fMRI) and gives an overview of the standard preprocessing steps for fMRI data. These consist of univariate data processing. Next we will talk about multivariate analysis, i.e. Machine Learning. The reader is supposed to already have a basic knowledge of what Machine Learning is.

1.1 functional Magnetic Resonance Imaging (fMRI)

One may already know Magnetic Resonance Imaging (MRI), which is a kind of non invasive imaging technique that provides structural information of the body with a good spatial resolution. When applied to the brain, this technique enables visualization of the brain's structure. MRI is widely used in the medical field, for example to diagnose diseases or evaluate the evolution of a treatment. Now what if we wanted to study cognitive processes? We would need to visualize evolution of brain activity across time. This is where functional MRI (fMRI) comes into play.

1.1.1 What is fMRI

fMRI data is acquired in a similar fashion as MRI data except that in the case of fMRI, what is generated is not a single brain volume but rather a "video" of brain activity. This means the result of an fMRI experiment on one person is a sequence of 3D images. In consequence, the obtained data is in 4 dimensions: 3 dimensions of space and a dimension of time. To create each of these 3D images, the scanner takes 2D images of slices of the brain to later reassemble them in a 3D volume. The following paragraph explains how these 2D images are captured. The explanations and some of the illustrations given here are based on the excellent

course "Principles of fMRI" given by Martin Lindquist and Tor Wager (3).

(f)MRI is based on the magnetic properties of the hydrogen nuclei (i.e. protons) contained in water molecules. These protons can be seen as spinning positively charged spheres. Each of these generates a net magnetic moment along the axis of the spin. Initially, the direction of this axis is random for each proton. Now let's put the subject into a (f)MRI scanner. The machine generates a powerful (1.5T - 7T) magnetic field that aligns the hydrogen nuclei in the direction of the field. All protons now spin around an axis that is in the same direction as the magnetic field. This induces a net magnetization in the direction of the field. A Radio Frequency (RF) pulse is then used to flip the protons by a certain angle. Now the net magnetization is in a direction perpendicular to the one of the magnetic field. When the RF pulse is removed, the protons want to go back to their original configuration. In doing this, the net magnetization changes direction, it grows exponentially in the direction of the magnetic field. The exact shape of this exponential depends on the kind of tissue the protons are in. This is what enables visualization of the brain's structure. The different steps of data acquisition are represented in figure 1.1

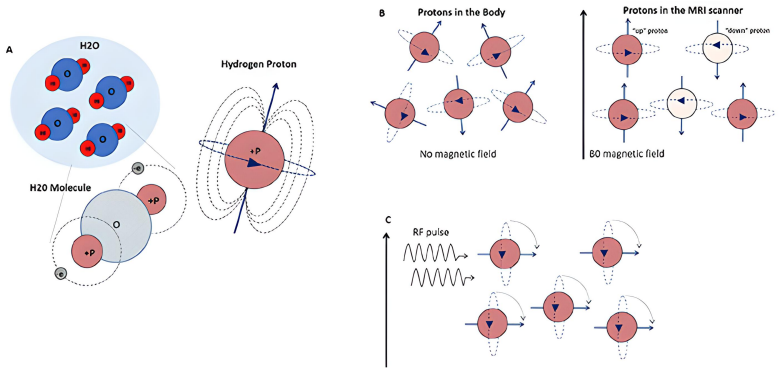

Figure 1.1: MRI steps (4)

The way brain activity is measured is via the Blood Oxygenation Level Dependent (BOLD) contrast. It is important to note that this contrast does not measure neural activity directly! It measures oxygen demand in the brain. Indeed, oxygen rich blood does not have the same magnetic properties as blood that carries less oxygen. Moreover, highly active brain regions have a higher oxygen demand. How exactly does a change in neural activity reflect in the fMRI signal? This is described by the Haemodynamic Response Function (HRF) (fig. 1.2). When the person in the scanner perceives a stimulus, the neurons that handle the reponse to this stimulus will consume oxygen. In short, this will cause the concentration

in oxygenated blood to drop and the fMRI signal to do the same. Just after this "initial dip", oxygenated blood flows towards the concerned regions leading to a peak in the fMRI signal. This peak is observed more or less 5 seconds after the person is stimulated. The magnitude in signal change is actually quite small, between 0 and 5%.

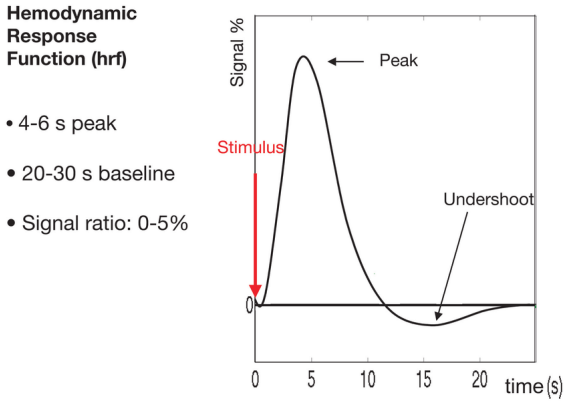

Figure 1.2: Hemodynamic Response Function (5)

The BOLD contrast has a nice property which makes it a popular technique for fMRI data: it enables to capture images of the brain with a spatial resolution of some millimeters to a centimeter. But it also has drawbacks. Indeed, the exact shape of the HRF varies across subjects and regions of the brain and the fact that the response is delayed and slow makes extraction of temporal information tricky.

1.1.2 Trade-off between spatial and temporal resolution

Before talking about temporal resolution let us dig deeper into the image format of fMRI data. fMRI images are in 3 dimensions. The base unit used to represent a brain in 3 dimensions is the voxel. A voxel (volume element) is the 3D equivalent of a pixel (picture element). As the resolution is often of some millimeters, the information is contained in voxels of a few millimeters of side. The number of voxels is often very high, a brain volume contains around 50000 voxels (3 x 3 x 3 mm voxel resolution). One could be asked to examine only a certain area of the brain. In this case, a Region Of Interest (ROI) can be extracted from the brain volume at hand. This implies selecting a set of voxels depending on structural or functional information (more details on this in (6)). Figure 1.3 shows an illustration of a brain separated in voxels.

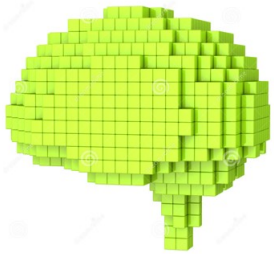

Figure 1.3: Brain volume made of voxels

To better understand the implications of a variation in the spatial resolution it is necessary to dig a bit more into the functional organisation of the brain. These different levels of organisation of the brain are laid out in figure 1.4. The highest organisational level of the brain is the large-scale networks level. These networks occupy patterns across multiple systems of the brain down to more or less 1 cm. The second level contains the functional maps. While they usually span several voxels, the smallest of these maps can fit within 1 voxel in a standard neuroimaging experiment. Going deeper, the organisational structure that is encountered is called "functional columns". As these columns are observed at a finer scale, many of them can fit into one voxel. Taking an even finer resolution would enable the visualisation of cell assemblies. To sum it up, the spatial resolution influences the kind of structure that can be analysed with an fMRI experiment.

As said before, time is also involved in fMRI. Indeed, for each voxel of the studied brain the data gathered by the experiment is a time series. So the higher the sampling frequency, the better the signal! Unfortunately, with BOLD fMRI, the quicker an image is acquired the lower the spatial resolution. So the temporal resolution of an fMRI experiment depends on how spatially accurate the image has to be. This is actually the main difference between MRI and fMRI. What is important when doing an MRI is the spatial resolution, because this type of imaging is used to extract structural information. Time resolution does not matter in MRI. While in fMRI the desired output is a "video" of neural activity depending on the stimuli perceived by the subject.

1.1.3 Experiment design

When designing an fMRI experiment, there are many things to consider. Usually such an experiment is conducted on several individuals and each individual does the experiment a certain number of times. Each of these passages in the scanner

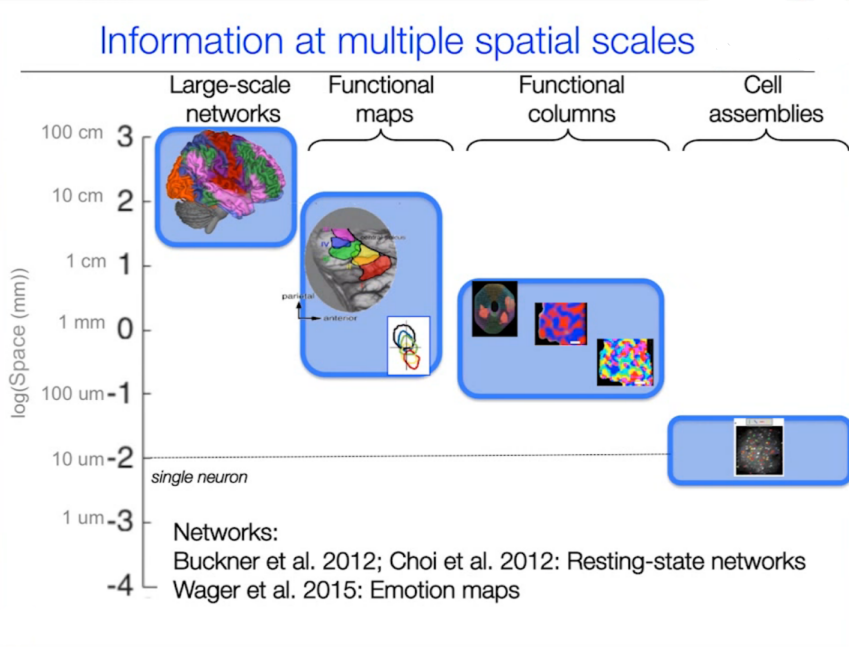

Figure 1.4: Levels of brain organization (from (3))

is called a run. The purpose of the experiment's design is to induce the desired psychological state in the subject. The two main ways of organizing the stimuli that will be presented to the subjects are block design and event related design. In block design, stimuli that belong to the same category are grouped together, while in event related design stimuli of different categories are intermixed (see figure 1.5). There are much more things to consider when designing an fMRI experiment but they are outside the scope of this thesis.

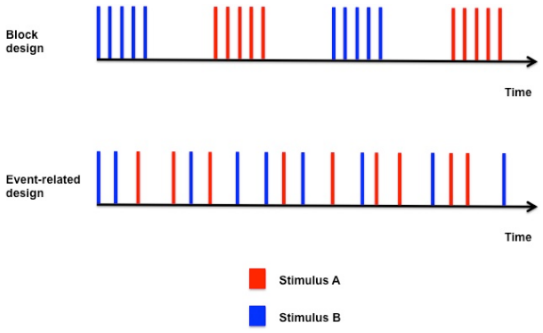

Figure 1.5: Comparison between block and event-related designs (7)

1.1.4 Data preprocessing pipeline

This pipeline has two main goals.

- Minimize influence of noise from the acquired data. Noise in fMRI data is not negligible at all and comes from many sources.
- Standardize the location of brain regions across individuals. The anatomy of the brain is not exactly the same from one person to another.

When processing the data of an fMRI experiment, it is useful to have both the "video" of brain activity described earlier and a structural image of the brain of the subject. This structural image is simply the result of a classical MRI scan.

The first step of the preprocessing pipeline is a very common one in data science, visualization and artifact removal. In essence this step can be summarized as taking a look at the data and remove data that does not make sense. This can be achieved with different data visualization tools.

Now remember that the acquired data are 2D images (slices of the brain) that are reassembled to form a brain volume. Each of these 2D images is captured at a different time point. This is why slice time correction is needed. This technique shifts each voxel's time series, using interpolation, so that it seems like they have all been sampled simultaneously.

What happens if the subject moves his head during the scan? The captured image will be distorted compared to other ones in the same run. In order to avoid this kind of problem, all images undergo a rigid body transformation that aligns them to a reference image. This reference is often the first or the mean of all captured images.

The next step is known as co-registration. The structural image mentioned earlier is registered to the fMRI images. This enables visualization of the neural activity of a subject overlaid onto its anatomical information. This technique also simplifies the mapping of fMRI images to a standard coordinate system that will come later.

It was stated that the shape and anatomy of the brain can vary from person to person. This is where normalization enters the picture. This technique is all about stretching, squeezing and warping each individual's brain to make it match a standard brain. Although it may introduce errors in the data and reduces the spatial resolution, normalization enables to compare or average results across subjects and studies. It also allows for consistent interpretation of spatial regions.

Normalization is done on the structural image and then on the functional images with the same parameters.

The last step is smoothing. This is another technique that reduces spatial specificity of the data. The advantages it offers are the following: it may increase statistical validity and signal to noise ratio as well as overcome limitations of the normalization by blurring anatomical differences. A certain level smoothness is also necessary for some of the corrections for multiple comparison used in univariate analysis.

1.1.5 The General Linear Model (GLM)

This is the first step in data analysis and also the last step in the fMRI data (pre)processing pipeline. The GLM will produce the data that will be fed to the Machine Learning process. The goal here is to model the activity of each voxel with a mathematical model. The model has the following equation:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_k x_{ik} + \epsilon_i \tag{1.1}$$

Remember, the value y observed at each voxel is a time series. Hence y_i represents activity at each timepoint i. β_0 is the intercept, whose role is to capture the baseline activation level and ϵ_i the error. The β's are the parameters of the model and the x's are the expected values for neural activity depending on the fMRI experiment's design.

Let us take an example to better illustrate how the GLM works. Imagine an experiment where the person in the scanner is subjected to pictures of famous and non famous persons. Let us say the researcher implemented the event-related design. In this context, neural activity is expected to vary depending on two different stimuli. Therefore, the model for a single voxel would include two parameters in addition to β_0. β_1 in the case where the subject sees the photo of a famous person and β_2 if the person is not famous.

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \epsilon_i \tag{1.2}$$

The general equation (1.1) can also be written as:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{1.3}$$

Where $\boldsymbol{Y}$ and $\boldsymbol{X}$ are known and $\boldsymbol{\beta}$ has to be calculated by minimizing the sum of the squared errors $\sum \epsilon_i^2 = \sum (y_i - x_i\boldsymbol{\beta})^2$. The first column of $\boldsymbol{X}$ is filled with ones to match the intercept β_0 ($\boldsymbol{X_{i0}} = 1$ for all i). Figure 1.6 shows more visually how

equation 1.2 looks like.

There is a question that has not been covered yet, how is $\boldsymbol{X}$ defined? $\boldsymbol{X}$ is called the design matrix and $\boldsymbol{X_{ij}}$ is the convolution between the HRF (figure 1.2) and the neural response function, at time point i and stimulus $j > 0$. The HRF is assumed to be the same across all individuals and brain regions although this is an approximation. The neural response function is a signal that is 1 when the subject is presented with stimulus j and zero otherwise. Figure 1.7 shows an illustration of this.

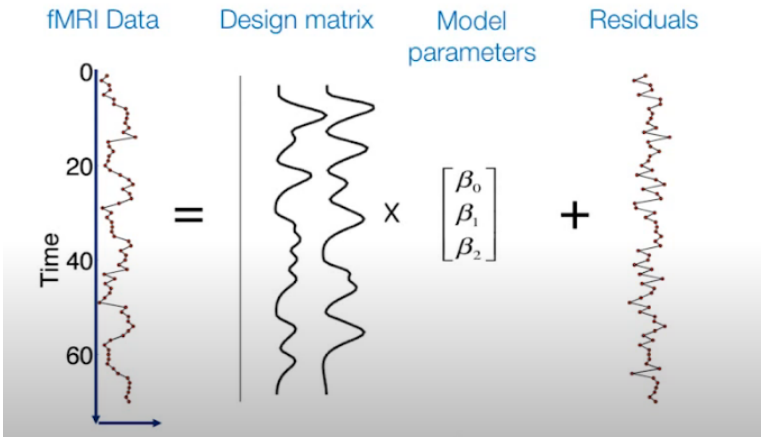

Figure 1.6: Visual representation of equation 1.2 (3)

Now let's see what the data extracted from this model looks like. After fitting the GLM to the fMRI data, beta-maps can be obtained by multiplying the learned $\boldsymbol{\beta}$ by a contrast vector called c. This gives a scalar value for every voxel. The goal of this multiplication is to observe the effect of a linear combination of conditions on the subject. Let us take the example mentioned before. If the researcher wants to assess the significance of the response to the "famous face" stimulus, he would take $c = (0, 1, 0)$ because the parameter that accounts for this event is β_1. One might also want to have a beta-map that presents the difference between the two conditions or the sum of the two conditions. In these situations, c would respectively be equal to $(0, 1, -1)$ and $(0, 1, 1)$. With the obtained beta-map it is possible to create a t-map simply by performing a t-test on the values of the beta-map. Beta-maps and t-maps are usual inputs to further Machine Learning processes.

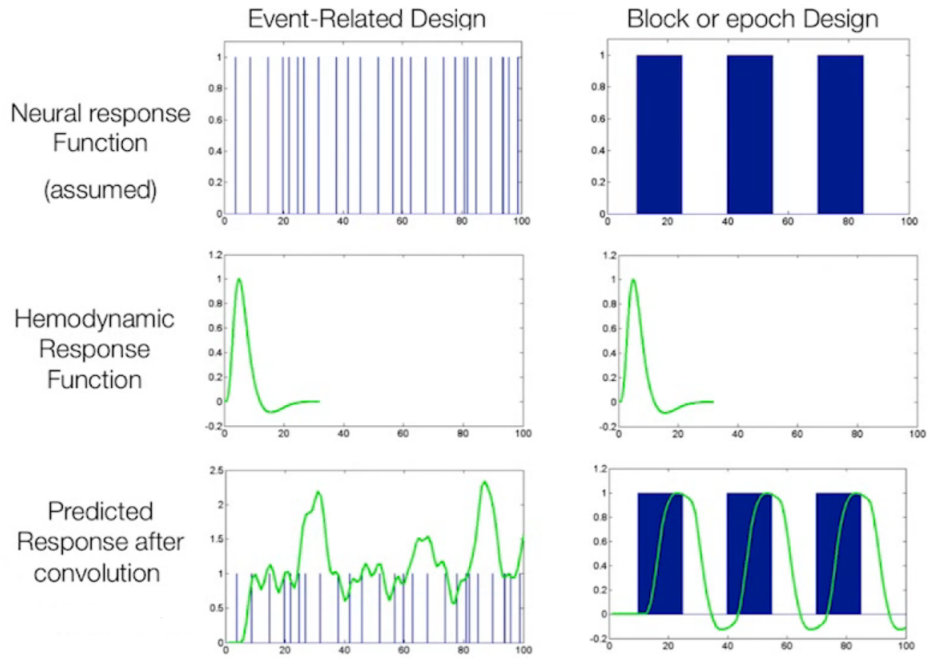

Figure 1.7: Definition of $\boldsymbol{X}$ (3) ($\boldsymbol{X}$ is on the last row for both possible designs)

1.2 Machine Learning

Machine learning is now a very well established domain of computer science. The general goal is to elaborate mathematical models that can improve their predictions through experience gained on data (a process called fitting). These models are intended to make reliable predictions on previously unseen data. In this section we will of course not explain every concept because it is simply impossible. The focus will rather be on the parts that are be key to understand the analyses performed in this thesis.

1.2.1 Classification

Machine learning can take multiple shapes: regression, classification, clustering, etc. In this case the focus is on classification, that is predicting a discrete output from a set of (discrete or continuous) variables/features. On figure 1.8 there is an example where one tries to predict if a subject is diabetic from biomedical data (e.g. age, sex, smoking habit, sports habit, height).
Often, classification translates into a dual choice (yes or no) and many ML models exist to tackle this challenge. But it does not have to be a dual choice and sometimes

one may want to choose between more than 2 possibilities. For example in the diabetes table there could be the 3 following diagnostics: healthy, type 1 diabetic and type 2 diabetic. That is called multinomial or multi-class classification. It is a little bit trickier and needs a dedicated strategy because often ML models proper nature is to make a binary choice and they are not easily extendable to more than 2 possible outcomes.

Subject n°	Age	Sex	Smoker	Sporting	Height(cm)	**Diagnosis**
1	23	Male	Yes	Yes	176	**Healthy**
2	16	Female	No	No	165	**Diabetic**
3	38	Male	No	Yes	192	**Diabetic**
4	64	Male	No	No	163	**Healthy**
5	35	Female	Yes	Yes	187	**Healthy**

Figure 1.8: Example of diabetes classification

1.2.2 Cross-validation

As already said, Machine Learning aims at making predictions from new data, based on experience on training data. Therefore it would have no sense to measure how efficient a model is by observing its predictions on training data (this is called over-fitting). It is better to observe predictions on another, independent set of data (called test set). But of course a dataset's size is finite, and a trade-off will have to be determined between a bigger train set, which should improve the model capacities, and a bigger test set, aiming at more reliable performance measures.
To overcome this difficulty, one can do what is called (k-fold) cross-validation. This is cutting the full dataset in k independent folds, and successively train on all folds except one that are kept away for testing. In the end each fold is used once for testing. Test scores are then averaged over each fold to obtain a measure of the performance of the model. However, even if it avoids a problem, it creates another: an increase in computation time, because it multiplies it by k. It is therefore necessary to wisely choose the k parameter depending on the situation.

But models often have (discrete) hyper-parameters that need to be tuned as well (such as the number of neurons for a neural network, or the number of iterations, etc.). The problem is that if the best value is chosen based on the training data, there is absolutely no guarantee that it will be the best value on new data. And if we choose the best value based on testing data, that is over-fitting and we have no more idea on how it would perform on really new data. This is why a 3rd

independent data set called validation set is needed. We will thus use the training set to tune the parameters of the model (continuous weights values), the validation to tune the hyper-parameters (often discrete values), and the test set to see how the model performs.
Again, to make a full use of all the data available, we will use cross-validation, and more precisely nested cross-validation, as we now have 2 nested loops: one outer loop to combine different (train + validation) set - testing set, and another inner loop to combine different train set - validation set. To make it clearer, the principle of nested cross-validation is explained on figure 1.9.

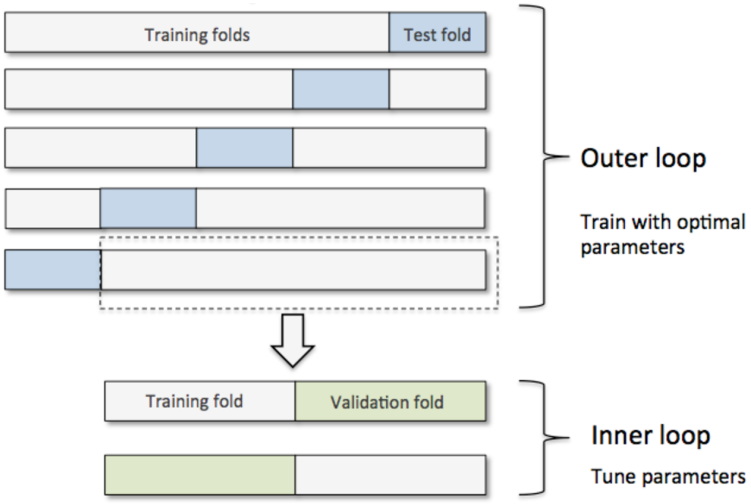

Figure 1.9: Nested cross-validation principle (source: (8))

1.2.3 Feature scaling

ML models are nothing less than mathematical expressions. Consequently, considering fixed coefficients, a high variation of a feature will influence more the outputs of the model than a small variation of the same feature. It is thus very often beneficial to ensure that features vary in a roughly same scale. That is done through feature scaling.
There exist multiple ways to do this. Let us focus on standard scaling. This technique transforms the data to ensure that for each feature, the mean over all the samples is 0 and the variance is 1. It is achieved by simply removing the mean of the feature and dividing by its standard deviation $X'_i = \frac{X_i - \mu_X}{\sigma_X}$, where X'_i is the scaled value of the feature X for the i^{th} sample, X_i its original value, μ_X the mean of the feature X across all the samples, and σ_X the standard deviation of

the feature X across all the samples.

1.2.4 Feature selection/extraction

Some ML models are sensitive to the presence of irrelevant features (features that do not provide useful information for the prediction), as the learning algorithms of the models might not have the ability to set the corresponding coefficients to 0 and will give influence to features that should not have any.
Also, some ML models can be biased by redundant features (features that express almost exactly the same thing, like height in meters and height in inches for example). In that case, the models might give a double weight to a single piece of useful information, and will end with a decision that is biased by this information.
To counteract these effects, a solution is to try to select only some features from the full set that is available. Of course, it is not an easy task to tell if a feature is useful or not, and there exists many different techniques to determine this (linear correlation, mutual information, etc.). Each of these could generate a different selection. If the selection really removes useless or redundant features, it is reasonable to hope for better results.

There are multiple ways of performing feature selection, including filter methods and wrapper methods.
The first one selects features before using any classifier. It often uses a certain criterion (linear correlation with output target, mutual information with output target, etc.) to measure how much a feature is associated to a class. These filter methods are often quite fast, but do not often really improve the results as they do not take the classifier into account.
This is why wrapper methods are useful. They use a classifier (often the one that will be used ultimately for the classification) to assess which features contribute the most to the accuracy. Recursive Feature elimination (RFE) is probably the most used wrapper method. Its principle is illustrated on figure 1.10. It iteratively fits a pseudo-linear classifier (i.e. a classifier whose decision function uses a linear mixture of the features) and evaluates the usefulness of each feature as the absolute value of the weight the classifier attributes to this feature. Doing so, at each iteration the x most useless features are removed, until reaching the desired amount of features. Wrapper methods are often more effective than filter methods, but at the cost of a larger computation time.

It is also possible to extract new features from a dataset, that is creating new features based on original ones. One of the most used technique for doing so is the Principal Component Analysis (PCA). PCA aims at extracting features (called

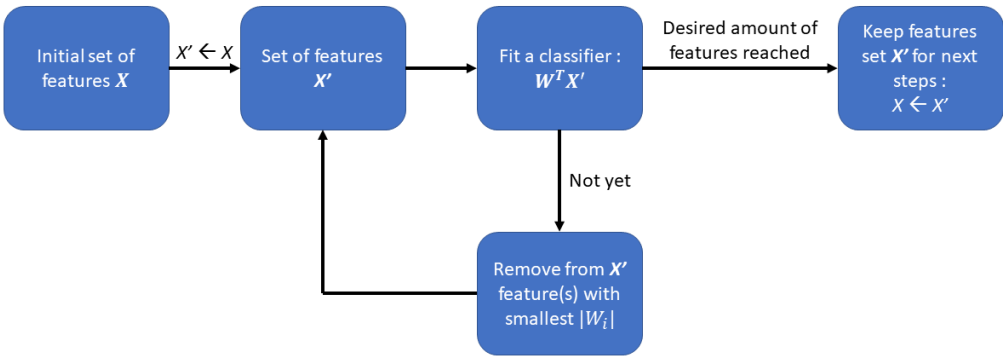

Figure 1.10: RFE mechanism

components) that explain most of the variance in the dataset. Depending on how much components one decides to keep, it ends up with a certain fraction of the original dataset variance.

1.2.5 Machine Learning models

The model is the mechanism that will ultimately product a prediction. It can take many different shapes but always corresponds to a mathematical formula.

K-Nearest Neighbors

KNN is one of the most simple ML models and was invented in 1985 by Keller et al. (9). In order to classify an unseen sample, the KNN classifier will evaluate which k points of the training dataset are closest to the unseen sample. It will then predict a class based on that of the neighbors. Often, this will simply be the majority class, but it can also be a more complex assignment, such as weighted vote based on the distance to each neighbor. The k hyper-parameter will depend on the amount of available samples. A large k will reduce the variance, at the risk of under-fitting, and a small k will induce sensitivity to particular patterns, at the risk of over-fitting. Lantz et al. suggest to set k equal to the square root of the amount of samples in the training dataset (10).
The big advantage of this model is that it does not require any training: it only has to store all samples (which can become problematic in case of a large dataset). However, it has to go through its whole training set in order to find the nearest neighbors of each unseen sample. Therefore the classification of an unseen sample often takes more time than for another ML classifier.

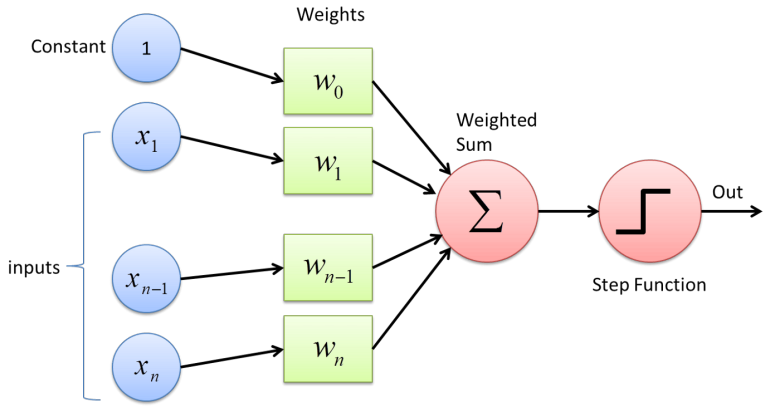

Figure 1.11: Perceptron mechanism (source: (12))

Perceptron

The perceptron is quite an old concept, introduced by Rosenblatt in 1958 already (11). This model is the basis on which neural networks were built, and is therefore often considered as the ancestor of Machine Learning.

The perceptron is a linear (binary) classifier that aims at discovering a separation hyper-plane between data samples of 2 classes. As illustrated on figure 1.11, class assignment is done using the function $class = sign(\mathbf{W}^T\mathbf{X} + w_0)$ where $\mathbf{W}$ is the vector of coefficients assigned to each feature, $\mathbf{X}$ the vector of input features, and w_0 a constant term. It tunes the $\mathbf{W}$ vector and w_0 by iterating over all training samples multiple times and applying stochastic gradient descent (see this document for further explanations).

Nowadays, many artificial neural networks are based on this, and perceptrons are stacked on top of one another to create what is called multi-layer perceptron.

(Linear) Support Vector Machine

SVM is a model that was created in 1992 by Boser et al. (13). A linear SVM aims at finding an hyper-plane that best separates the data points, that is with a maximal margin between the 2 prediction classes, as on figure 1.12. However, it goes beyond the perceptron's capacities. The SVM model uses the same linear function of the features: $class = sign(\mathbf{W}^T\mathbf{X} + b)$ where $\mathbf{W}$ is the vector of coefficients assigned to each feature, $\mathbf{X}$ the vector of input features , and b a constant term. But in this case the goal is not only to have a separating hyper-plane, but it is to have the maximal margin hyper-plane, and this is achieved by minimizing the scalar product between the coefficients vector and itself $< \mathbf{W}, \mathbf{W} >$.

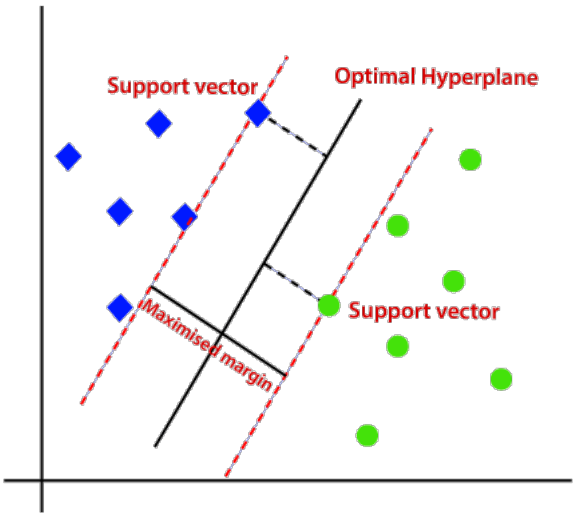

Figure 1.12: SVM example (source: (14))

However, when data is not linearly separable, it is impossible to find any hyperplane that perfectly separates the data. The solution is to use a kernel trick, this implies transferring the data from the original feature space into a new space where the data is now linearly separable. Functions have to respect some properties to be considered as kernels, and the most commonly used ones are linear or Gaussian.
Also, when the obtained maximal margin is tiny, one may want to include some tolerance to misclassified samples, in order to have a wider margin, even if that causes some classification errors. The loss function to minimize becomes $< \mathbf{W}, \mathbf{W} > +C \sum_i \xi_i^2$ where ξ_i corresponds to how far the i^{th} sample is from the margin, and C is a constant. This last parameter has to be tuned with regard to how much tolerance is wanted for misclassified data. A higher C value tells the model to give more importance to the accuracy of the model on training data, hence less importance to weights regularization.
Without going into technical details, the main advantage of SVMs is that their computation time is not proportional to the amount of features, but to the amount of samples, which is very interesting in the biomedical field where the data available is often very limited, but the amount of features can be huge.
However one of the disadvantage of the SVM is that it does not have a proper multiclass strategy. Therefore, when there are more than 2 possible outcomes, it becomes necessary to use a trick, which often takes 2 possible shapes: first, one can train an SVM to discriminate each possible combination of classes, and then combine these results to see which class outperforms the others. That is called one-vs-one strategy. The other possibility is to train an SVM to discriminate each class against all the others, and in the end select the class that outperforms the

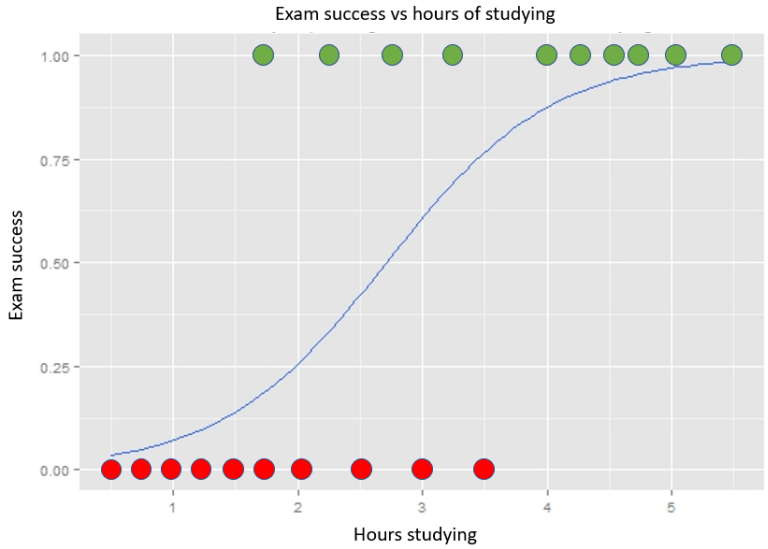

Figure 1.13: LR classification example

others. This is the one-vs-rest strategy.

Logistic Regression

LR was invented in 1995 by Write (15). This model is intended to build an estimation of each class' probability for a sample. Its mathematical formula is $class = sign(\frac{1}{1+e^{-(\mathbf{W}^T\mathbf{X}+b)}})$ where $\mathbf{W}$ is the vector of coefficients assigned to each feature, $\mathbf{X}$ the vector of input features and b a constant term. There is an LR classification example on figure 1.13, where it tries to determine if a student will pass an exam depending on the amount of time he spends studying. Thanks to its mechanism, this model can also produce an estimate of the probability for the prediction to be accurate.

As for SVM, LR can also include a regularization mechanism in order to penalize the complexity of the model (magnitude of $\mathbf{W}$ coefficients) compared to pure score results. The influence of the regularization parameter C is the same as for an SVM: a higher value tells the model to give more weight to the accuracy of the model on training data, hence less importance to regularization.

A big advantage of logistic regression is that it has a true multiclass strategy. Indeed, the model is trained to find the probabilities for a sample to belong to each class, which are constrained to sum to 1 as each sample only belongs to one class. This multinomial LR regression is also known as softmax regression or maximal entropy regression, and is very often used in natural language processing for example.

1.2.6 Data augmentation

In the ML field, it is not unusual to face datasets with a limited number of data samples. This is even more accurate in the biomedical field. To face this problem, data augmentation techniques have emerged. They consist of creating additional, artificial samples, from existing ones, in order to give larger datasets for the classifiers to train on, and hopefully obtain better results.

Many techniques of data augmentation exist, including complex neural networks, but let us will focus on one technique: Synthetic Minority Oversampling TEchnique (SMOTE), which was invented by Chawla et al. in 2002 (16). SMOTE is well known for its simplicity and therefore widely used, mostly for inflating an under-represented class, although it can also be used to inflate all classes.

The SMOTE algorithm works the following way (illustrated in figure 1.14):

1. Draw a sample from the class chosen to be augmented.
2. Identify the k nearest neighbours of this sample and select one of them randomly.
3. The algorithm then computes the vector v between the original sample and its selected neighbour.
4. The new sample is the addition of the selected sample and $a \cdot v$, a being a random number between 0 and 1.

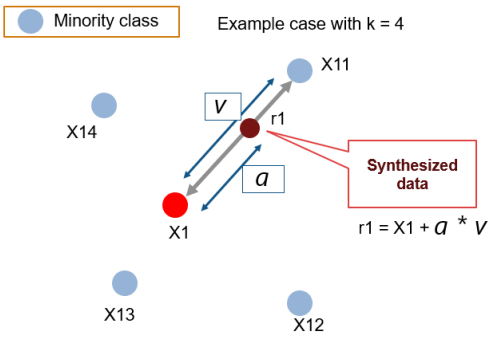

Figure 1.14: Visualization of the SMOTE algorithm (17)

When using SMOTE one can play with two parameters: k the number of nearest neighbours to be considered and the number of samples s wanted for each class in the end.

Chapter 2

Related work

The goal of this chapter is be to give an overview of what has already been successfully done in studies close to this thesis, and highlight similarities and differences with regard to this thesis.

2.1 Machine learning on fMRI brain data

The study conducted by Cox and Savoy in 2003 (18) is often considered as the first milestone for the application of ML techniques to fMRI data, so it is quite a recent field of research. In this study, they explored the possibility of multivoxel pattern analysis (MVPA) for the first time. Before that, there only was univariate analysis of each voxel independently. Furthermore, in fMRI data, with equivalent stimulus often comes equivalent data. It is thus crucial to have a more global point of view to hope to decode information. However in the case of this study, the subjects were shown images of 10 different kinds of objects (baskets, chairs, horses, etc.) which are quite easily distinguishable stimulus. This is why the researchers achieve accuracies above 80% (see fig. 2.1) which illustrates a pretty high score for fMRI decoding).
An interesting observation they made is that a linear SVM was more accurate than a polynomial SVM, which is surprising given that a polynom should better capture an unknown function than a simple linear regression. However, the score differences between these 2 types of SVMs were quite small compared to their difference with the other classifier they used (Linear Discriminant), which was way less accurate. Another observation that can be derived from their results is that the accuracy increases along with the amount of voxels used for the classifier. However, feature selection did not improve the score as it often does in Machine Learning in general.

6 years later, Pereira et al. released a study which goal was to make an overview

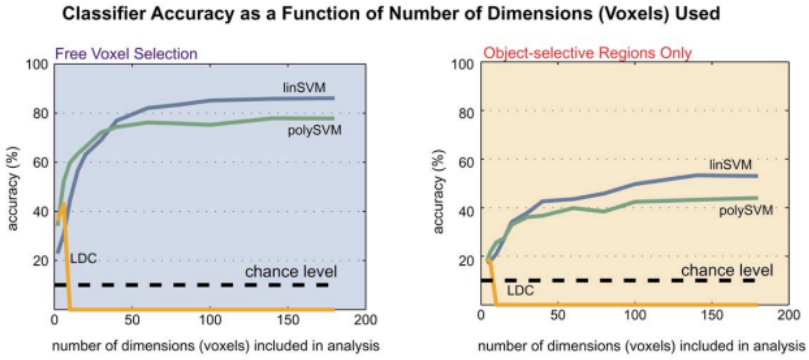

Figure 2.1: Results of the study of Cox and Savoy (source: (18))

of the application of ML to fMRI data, from a study case (19). This paper is definitely a must-read to quickly have an idea of the key points of this field.
For instance, they give an example on how feature selection can help, but remind that evaluating which voxels are the most relevant to keep should be done on the training test and not on the whole dataset, as it would result in overfitting. They also mention the possibility to perform dimensionality reduction, using Principal Component Analysis (PCA) for example. This allows to strongly reduce the number of features in a dataset and escape the curse of dimensionality, but this comes at the cost of losing information which could potentially have been useful. It is worth noting that such a technique does not take the classes of the data into account.
The study also emphasizes that wrapper methods such as RFE can be very effective. It then provides some hints about the choice of a relevant classifier. It argues that SVM and LR are the most powerful ones for fMRI classification. The first one is practical because its complexity is not constrained by the number of features (which can be very high in the case of fMRI) but instead by the number of samples in the data (which is often small in fMRI classification), as stated in section 1.2. As for LR, the main advantage is that it has its own multi-class classification mechanism, which is useful because one often wants to discriminate between (many) more than 2 classes in fMRI.
The study also explains the benefits of cross-validation and how to assess the results of a trained and tuned ML model, as already explained in 1.2.

Then, in their guide book of fMRI data analysis (20), Poldrack et al. took stock of what was generally working well for fMRI decoding. They are in line with the study of Pereira et al. (19) on many points. They additionally mentioned the choice of beta maps or t-maps, and they stated that there is no clear advantage for one type of maps, but *on average* slightly better results can be obtained by using t-maps instead of beta-maps.

2.2 Feature selection on fMRI voxels

Now that we've browsed through a selection of works already done on fMRI classification in general, we would like to focus on some more precise points that caught our attention. The first of them is feature selection, because there are very often more features than samples and this can very quickly become a problem if not treated properly. It is worth noting that the studies mentioned below use this technique to select features from the whole brain, and not from a predefined ROI.

In 2009 Jin et al. explored feature selection techniques to see their influence on the results they obtained (21). They used 2 filter methods: Fisher Criterion Score (FCS) and Relief-F. FCS provides a ranking of the features based on their capability to discriminate classes, without taking feature dependence into account. On the other hand, Relief-F uses a weighting approach, iterating through randomly selected data samples and adjusting the weight of each feature separately, depending on how it relates to the k nearest neighbors in the dataset. The last step is to remove features that have a weight lower than a certain threshold.
They also tried another approach using 2 wrapper methods proposed by Frölich et al. in 2002 (22): GAR2W2 and GAJH. these methods use a genetic algorithm in order to perform the best selection of features. They only differ in their criterion of selection.
As visible on figure 2.2, they obtained higher accuracies with their 2 wrapper methods, compared to the 2 filter methods.

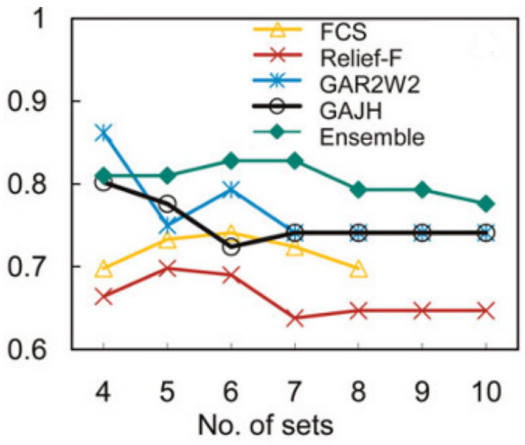

Figure 2.2: Results accuracies of the study of Jin et al. (21)

Then in 2017 Qureshi et al. used feature selection in the case of multi-class classification of attention deficit hyperactivity disorder (ADHD) subjects (23).

They used a hierarchical approach to select the optimal features to achieve the best accuracy, composed of 2 steps. The first one consists of a Least Absolute Shrinkage and Selection Operator (LASSO) regression that will select only a part of the features (optimizing an objective function with an error term and a regularization term). The selected features are then fed to an RFE algorithm (see section 1.2.4). The study does not say by how much this approach improved the results.

2.3 Extreme Learning Machine (ELM)

This Machine Learning model was developed by Huang, et al. (24). In the study conducted by Qureshi et al, (23) researchers performed classification on data of children with or without ADHD. After the feature selection step that is described above, comes the classification step. The goal of the classifier was to determine if the children had ADHD. They also did multi-class classification with subjects to detect different types of ADHD. One of the objectives of this study was to compare the performance of an ELM model to the performance of a SVM model. In this particular research the ELM outperformed the SVM.

This is not the first study to use an ELM classifier. Indeed, Another brain imaging study was conducted where the objective was to classify images of cocaine addicted people versus images of healthy people (25). In short this means the classifier has to predict different experimental conditions between subjects. In this paper, the researchers *"present a Computer Aided Diagnosis and image biomarker identification system for cocaine dependence, which selects relevant regions from a set of brain structural magnetic resonance images (sMRI)"*. They noticed that when the number of regions is high, the ELM classifier outperforms the other considered classifiers.

A third study compared classification with ELM and SVM models and came to the same conclusion that ELMs are a valid alternative to SVMs. This study performed binary classification to predict if a subject has a Mild Cognitive Impairment (MCI) or not. MCI being an intermediate step between dementia and normal brain functioning (26).

The ELM is a model that is often used as a Single-hidden Layer Feedforward Network (SLFN). The focus will only be on this specific instance of ELM from now on. Figure 2.3 illustrates what kind of networks ELMs are. When used for classification, the output function is defined as follows:

$$f_M(\boldsymbol{x}) = \sum_{i=1}^{M} tanh(\boldsymbol{a_i} \cdot \boldsymbol{x} + b_i) \cdot \boldsymbol{\beta_i} \tag{2.1}$$

Where M is the number of hidden nodes, β_i the output weight for hidden node i and a_i and b_i are the parameters for that same neuron. The particularity of this model is that the parameters of the hidden nodes ($\boldsymbol{a}$ and b) are not tuned. They are randomly assigned and never updated. The model then only updates $\boldsymbol{\beta}$. Because of this, it is rather fast to train such a model. The only hyperparameter of this model is M the number of neurons in the hidden layer. To sum it up, finding the optimal $\boldsymbol{\beta}$ actually corresponds to solving the following N linear equations with $j \in 1..N$.

$$\boldsymbol{T_j} = \sum_{i=1}^{M} tanh(\boldsymbol{a_i} \cdot \boldsymbol{x_j} + b_i) \cdot \boldsymbol{\beta_i} \tag{2.2}$$

Where $\boldsymbol{T_j}$ is the target vector for input vector $\boldsymbol{x_j}$.

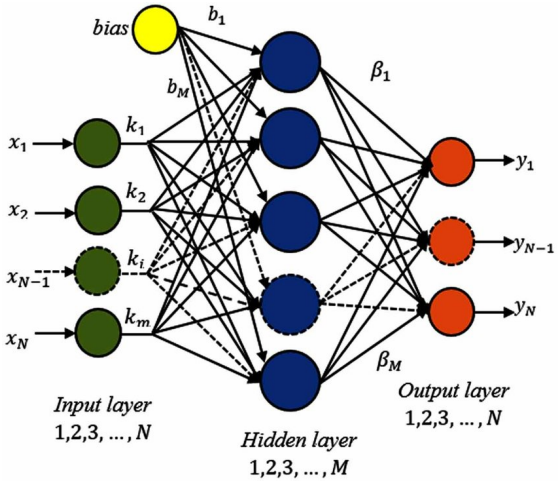

Figure 2.3: ELM network

2.4 Data augmentation

Generally speaking, data augmentation is used to solve ML problems where the number of data points belonging to a given class is very low. Data augmentation techniques then help generate new samples for the under represented class. These techniques can be pretty simple like SMOTE (see section 1.2.6) or they may be as complicated as a Generative Adversarial Network (GAN) (27).

In the fMRI field, the quantity of available data is often very low for a given problem. For this reason, the temptation to use data augmentation techniques is quite strong. The augmentation techniques most often used with fMRI data are GANs. The idea behind a GAN is to have a competition between two neural

networks. One of them is a generator and the other is a discriminator. While the generator produces new samples from the data it is fed, the discriminator tries to determine if a sample is a real sample or if it was generated by the other network. GANs are known for generating really realistic artificial images, hence why they were used in some fMRI research. The study carried out by Zhuang et al. in 2019 aimed at comparing different synthesis techniques such as GANs, Gaussian Mixture Models and the variational autoencoder in an fMRI decoding context (28). The paper claims the obtained *"results suggest that data augmentation via synthesis is a promising approach to address the limited availability of fMRI data"*. Another research introduces a new method to perform data augmentation on fMRI data (29). The focus of this study is on GANs, it indeed proposes several GANs architectures to generate new data, with a certain success.

Chapter 3

Dataset description

In this chapter we will describe the dataset used for this thesis, how and why it was acquired, and its specificities.

3.1 Data acquisition

Originally this dataset was acquired by Rezk et al. for a study to investigate shared representation of visual and auditory motion directions in the human middle-temporal cortex (hMT+/V5) (1). More precisely, they wanted to see if V5 was not only involved in visual motion direction, but also in auditory motion processing, and see if there was a shared representation between the 2 senses.

To this end they conducted 3 experiments on 24 subjects. The 2 first experiments allowed to localize 2 regions of interest (ROIs), because these are at variable locations depending on the subject.
The first experiment aimed at localizing V5/hMT+, a cortical region well known for processing visual motion. The subjects were exposed either to a static visual stimulus, or to a moving visual stimulus. By computing the contrast between the beta-maps obtained with the 2 types of stimulus, it is possible to see which brain regions have the highest activity specific to motion, which is presumably V5.
The second experiment was to localize the Planum Temporale (PT), a cortical region well known for processing auditory motion direction. The same principle as the first experiment was applied to extract the location of PT.
The purpose of the third and principal experiment was to expose the subjects to visual and auditory stimuli moving in 4 directions: up, down, left and right. There were 12 runs of the 8 possible stimulus (2 modalities x 4 directions).
Unfortunately, one subject had to be excluded from the study due to an excessive head motion during the experiments.

The data were acquired using a 4T MR scanner, with 3 x 3 x 3 mm voxel resolution . For each run, each of the 8 stimulus was presented to the subjects during 15 seconds.

3.2 Data preprocessing

For this study, The same preprocessing Matlab code was used as in Rezk's paper (1), but using different versions of Matlab (2016b vs 2012b) and SPM (30) (SPM8 vs SPM12).

The standard preprocessing pipeline was used (see section 1.1.4): slice time correction, alignment to the mean functional image with interpolation, coregistration of the functional images to the structural one and normalization to a standard template. For the 2 localizer experiments, images were also smoothed in order to reduce noise effects. For the motion direction experiment this was not recommended, as this would result in spatial information loss, which is required in order to perform multivariate analysis later on.
For some analyses, a resampling of the raw data was also performed (both functional and structural) to a 2 x 2 x 2 mm voxel resolution, which is the data Rezk used to obtain its results.

A GLM was run for each subject and each experiment, in order to obtain the beta estimates of each voxel, for the different runs and stimulus expositions. From these beta-maps, the corresponding t-maps were extracted by running a t-test for each voxel (univariate analysis).

3.3 Individual ROIs localization

From the 2 first experiments, it has been deduced where visual and auditory motion translated into higher contrast compared to static stimuli. From there it was possible to localize the ROIs needed for further analyses. The ROIs are then defined by creating a sphere with a radius of 10 mm centered at the activity peak within the area under study. Unfortunately, some of the ROIs (6 out of 92) were not localized with satisfactory confidence for some subjects, and were excluded from further analyses.

Unfortunately the location of these ROIs can be very different between individuals (especially for V5). It is therefore difficult to ensure that a peak found in a

certain region really belongs to the targeted ROI. To limit this effect, the location of each ROI was constrained by the global activity map across all subjects. This means the center of each subject's ROI has to be located inside a sphere of 12 mm of radius centered at the highest peak of that global activity map. The figure hereunder underlies this aspect of ROI definition. It shows the highest peak of the global activity map (□) as well as the 12mm sphere within which the highest activity peak is selected for this specific subject (□). The individual ROI is then defined from the latter.

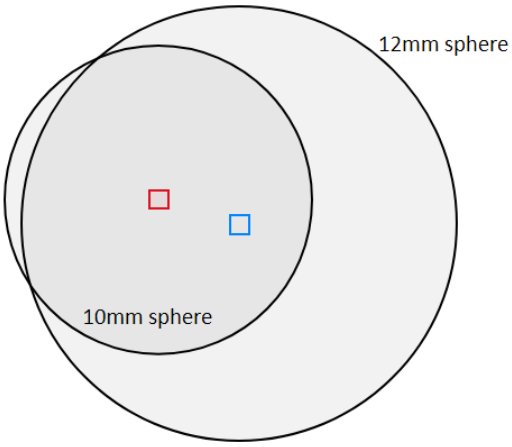

Figure 3.1: ROIs definition

In the end, for each subject, each ROI, and each hemisphere, a sphere with a radius of 10 mm is obtained. For each subject the following 4 ROIs are defined : left V5, right V5, left PT and right PT.

3.4 Specificity of these regions

Something that is very important when trying to apply Machine Learning on fMRI data is the underlying structure of the cortical regions. Indeed, as stressed by Norman et al. (31) in their study, the cortex of these somatosensory regions is organized in tiny columns, whose lengths are perpendicular to the surface of the cortex. Each column activates for a specific direction of the stimuli.
Unfortunately, these columns are much smaller than the voxel resolution one could hope to reach with fMRI. Therefore each voxel contains columns representing multiple different stimulus.

As illustrated on figure 3.2, these columns are (hopefully) not equally represented in each voxel. Some information on the stimulus direction can still be inferred from the activity of a voxel, but it is crucial to keep in mind that what is observed is a bias towards a certain direction, and not a clear expression of one specific direction. The differences in activity between the different directions will thus be very slight.

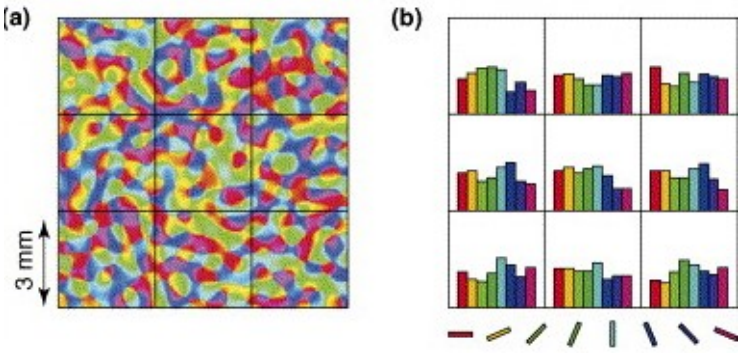

Figure 3.2: Column organization of the cortex (source: (31))

3.5 Machine Learning inputs

In the end, what we have to initiate the Machine Learning process are beta maps and t-maps, precisely 96 by subject (12 runs x 2 modalities x 4 directions). Each map is composed of more than 50 000 voxels, but they will be limited to 1 of the 4 ROIs, which corresponds to an amount of 171 voxels (with a 3 mm voxel resolution and a ROI sphere radius of 10 mm).
For each experiment we wish to perform, we will have a dataset made of 171 features or columns, and 96 samples or rows, except for analysis which involve only one modality, where there are 48 samples instead.

Chapter 4

Baseline pipeline

This chapter aims at describing our baseline Machine Learning pipeline to classify fMRI t-maps or beta-maps. This is the basis on top of which we modified parts in order to improve the results. The differences with regard to the thesis of Rezk, who used the same raw fMRI data (1), will also be detailed.

Just like Rezk, we have decided to use t-maps rather than beta-maps as they often hold better results (see section 2.1). We used a voxel resolution of 3 x 3 x 3 mm as it is the acquisition resolution. We decided to use a ROI radius of 10 mm, in order to have ROIs composed of 171 voxels each, as it is in the recommended range of voxels for a ROI (20).
As Rezk did, both within modality decoding and cross-modal decoding were performed. The first one corresponds to a situation where a classifier is trained on one modality (audition or vision) and test it on the same modality, in order to see how motion is encoded for this modality. The second aims at finding shared motion representation between modalities. Therefore, a classifier is trained on a certain modality, tested on the other, and vice-versa.
For both cases the processing pipelines used will be described in the below subsections. These are the same as Rezk used (1).

4.1 Within modality decoding

The whole pipeline of within modality decoding is summarized with the pseudo-code 1.

The process is divided into 2 parts: fit the model to compute the results based on the data, and then assessing their significance in order to estimate the probability of getting such decoding accuraries under the null hypothesis of chance level.

First, for each subject an analysis for V5 and PT was conducted, in each brain

Data: t-maps for each subject (and corresponding classes), ROIs for each subject

Result: performance evaluation of the chosen classifier for within modality decoding

```
for subject ← 1 ... 23 do
    apply masking to get the data in each ROI from the t-maps ;
    for ROI ← [left PT, right PT, left V5, right V5] do
        for modality ← [vision,audition] do
            for test run ← 1 ... 12 do        // cross-validation loop
                train runs ← 1 ... 12 without test run ;
                train data ← normalize_fit_transform(subject, ROI,
                  modality, train runs) ;
                test data ← normalize_transform(subject, ROI, modality,
                  test run) ;
                decoder ← SVM_fit(train data) ;
                predictions ← decoder.predict(test data) ;
                accuracy ← compute_accuracy(predictions) ;
            end
        end
    end
end
```

Algorithm 1: Within modality process pseudo-code

hemisphere separately. This was done for auditory motion and visual motion separately. In total, that means 8 analyses (2 modalities x 2 regions x 2 hemispheres) for each subject. For each analysis there is thus a dataset consisting of 48 samples x 171 features.
Then, in the most inner loop, cross-validation is applied using a leave-one-run-out (remember that 1 run consists of 4 samples, one for each direction). Before directly fitting a classifier, features are normalized, using standard scaling such as explained in section 1.2.3, to ensure that all features have the same influence at the beginning. After this, the classifier, in this case an SVM with a linear kernel and C parameter equal to 1 (see section 1.2.5 for the explanations of these hyper-parameters), is fitted on the training set, and used to compute accuracy from predictions on the test set. It is worth mentioning that we used it with a "one-vs-rest" strategy in order to perform multiclass classification.
Finally, accuracies on the 12 test runs are averaged to obtain a subject-level accuracy. These values are then averaged over the subjects to obtain group-level accuracies.

Now that results are obtained, it is necessary to assess their significance. One could say that if the results are far enough from the results one should obtain by chance (that is 25% in this case), the results can be considered as significant. But this is a mistake because "far enough" depends on the standard deviation of the results distribution. Of course results that are really far from chance level (more than 5% bigger let us say) will be significant, but properly assessing significance is key when the deviation from chance level is smaller (and it is often the case when using ML in the neuroscience domain). One could also argue that simple t-test would be enough. But again this assumes a normal distribution, which is not sure of being observed (especially with so few data). A method that makes less assumptions about the results distribution is thus needed.
To this end permutations and bootstrapping were used, as suggested by mcIntosh and Lobaugh (32). The procedure corresponds to pseudo-code 2 This is a technique that makes fewer assumptions about the results distribution. For each subject, 100 copies of them are created, for which the labels are randomly permuted inside each run. Doing so, 100 different copies of the original subject are artificially obtained. Afterwards, the analyses with each copy were performed as if it was a real subject. Then, 100 000 groups of 23 copies are created, by selecting for each subject one of its copies randomly. This allows to compute group results for each artificial group and ultimately gives a null group-level distribution.
Thanks to this distribution, a p-value estimation of the significance of the accuracy for each analysis finally obtained: p-value $= \frac{(1+\#(fakegroupswithhigheraccuracy))}{1+\#(fakegroupscreated)}$. As 4 ROIs are investigated, a Bonferroni correction (multiplication of the obtained

p-value estimation by 4) is introduced.
Unfortunately, this technique has a big drawback: it is computationally intensive, because it requires doing 100 times what was needed to get the original results. However, considering that the fMRI preprocessing takes much more time than this bootstrapping and permutations, using this procedure seems worth it.

Data: t-maps for each subject, ROIs for each subject, results with original labels
Result: significance evaluation of the obtained scores

```
for subject ← 1 ... 23 do
    for iter ← 1 ... 100 do
        for ROI ← [left PT, right PT, left V5, right V5] do
            for modality ← [vision,audition] do
                permute samples classes inside each run ;
                Do the above decoding with permutated sample classes ;
            end
        end
    end
end
for bootstrap ← 1 ... 100 000 do
    for subject ← 1 ... 23 do
        select randomly results of 1 of the 100 permutated datasets ;
    end
    group_results ← compute_group_results() ;
end
p-values ← compute_p-value_from_comparison(true results, permutations
  results) ;
return estimated p-values
```

Algorithm 2: Permutation and bootstrap pseudo-code

Results

The cross-validation average accuracies obtained with this baseline procedure are aggregated on figure 4.1. The individual points represent each subject's accuracy. The black caps represent the standard error of the mean, giving an insight of the confidence in the the results significance compared to chance level (gray horizontal

line). The white stars[1] at the bottom represent the significance level of each analysis compared to chance level, obtained via the permutations process explained in the previous section. As examples, there are below 2 histograms plots of the null group-level distribution for auditory and visual analyses in right PT, respectively on figure 4.2 and 4.3. In the first case, the analysis is highly significant (estimated p-value under 0.00001), in the second one no conclusion can be drawn (estimated p-value = 0.544).
As expected (see chapter 3), PT encodes information about auditory motion direction, and V5 encodes information about visual motion direction. It is also worth to notice, as Rezk found (1), that V5 encodes information about auditory motion direction. Conversely, PT does not seem to encode information about visual motion direction, which is also as expected. None of the below modifications or analyses led to an improvement that could make the results significant for vision in PT. We will therefore not talk about vision in PT any further in this thesis.
It is also interesting to note that there are no huge differences between hemispheres.

In order to see if there was some particular misclassifications patterns for the different analyses, the corresponding confusion matrices were plotted. Indeed one could for example expect that right is mostly confused with left and up with down. The confusion matrix for auditory motion decoding in left V5 is represented on figure 4.4. In this case, and this is actually true for all auditory analyses (both in V5 and PT), the confusion patterns are up-left and right-down. This is a bit surprising, but the percentages of predictions do not exceed 35% so it is not possible to deduce anything from that observation.
The matrix for visual motion decoding in right V5 on figure 4.5 shows a more prominent main diagonal, with more unstructured misclassifications, both of these remarks are also valid for left V5.

Comparison with thesis results

In his thesis, Rezk obtained the following results, that we were able to reproduce as he kindly gave us the t-maps, beta maps and ROIs he used. It is therefore possible to establish a comparison for within modality decoding results (figure 4.6).

The results are not the same, as we reached higher accuracies for all within modality decoding analyses. Regarding within modality auditory motion decoding in left V5, he did not get any significant result while we did. One will probably wonder where does this difference come from. Actually there are 2 differences. The

[1]Stars correspond to the level of significance with the following code: * = $p < 0.05$, ** = $p < 0.01$, *** = $p < 0.001$, **** = $p < 0.0001$

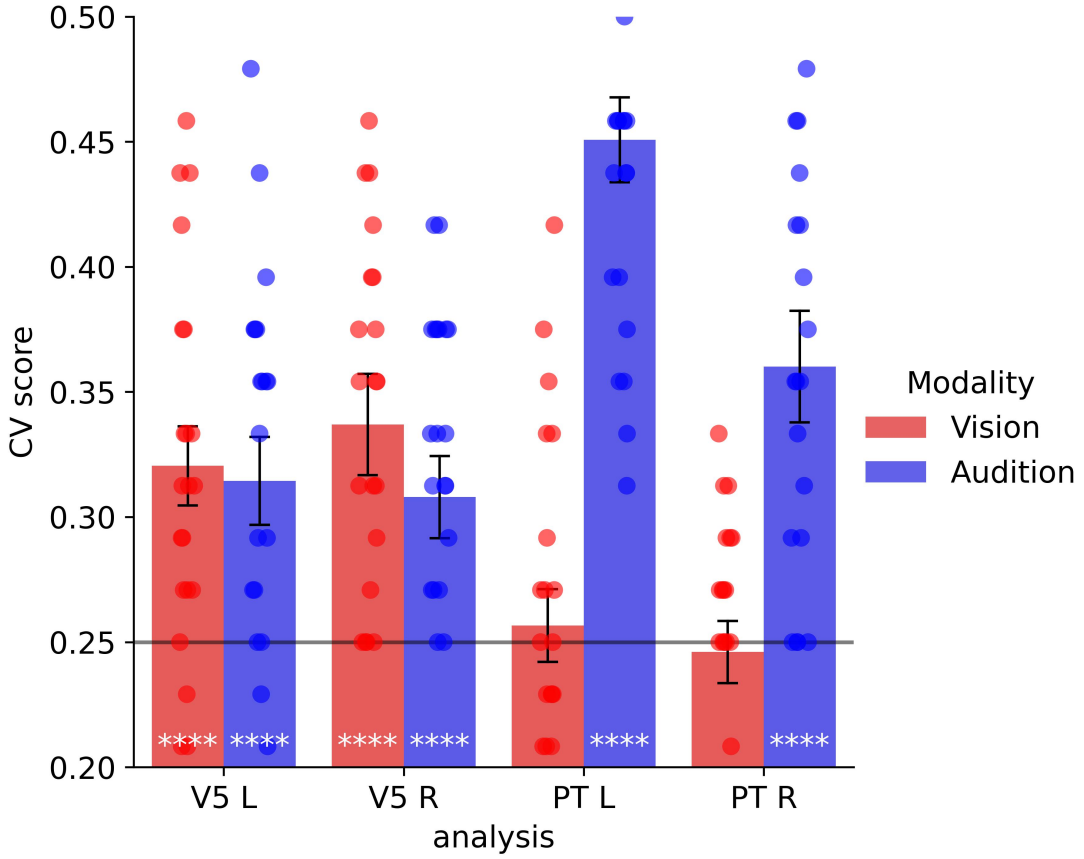

Figure 4.1: Within modality decoding accuracies with baseline procedure

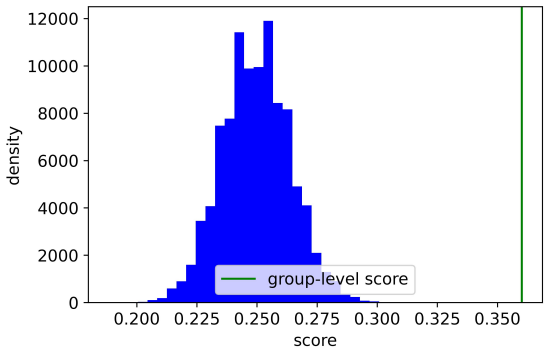

Figure 4.2: Null group-level distribution for auditory motion in right PT

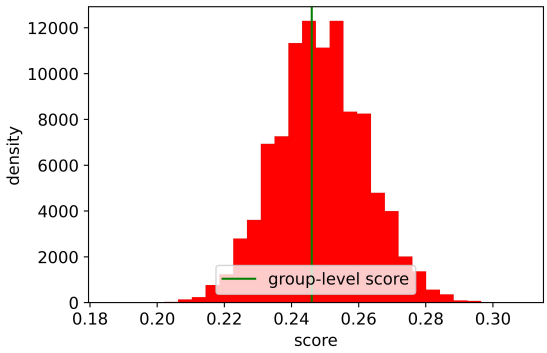

Figure 4.3: Null group-level distribution for visual motion in right PT

first one is that at the end of the fMRI pre-processing he resampled the data to obtain a 2 x 2 x 2 mm voxel resolution by interpolation and uses ROIs of 7 mm

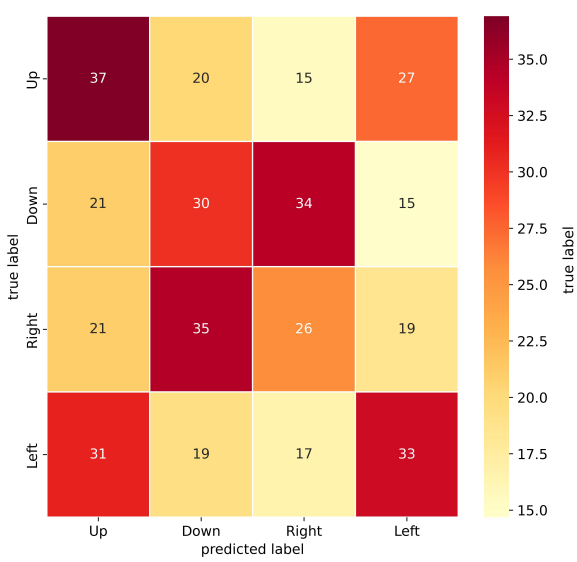

Figure 4.4: Confusion matrix for auditory motion decoding in left V5

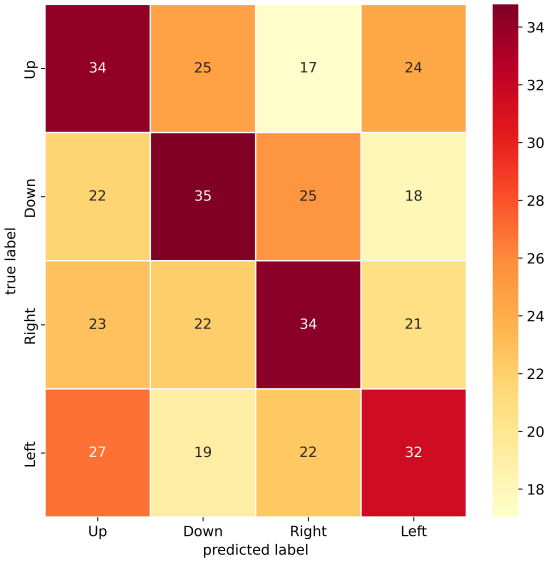

Figure 4.5: Confusion matrix for visual motion decoding in right V5

radius. However when we also preprocessed the data in order to resample them to a 2 x 2 x 2 mm voxel resolution and used ROIs of 7 mm radius, we still did not obtain the same results as Rezk, as illustrated on figure 4.7.
The second difference also comes from the fMRI pre-processing: even if we used the same source code, we did not use the same software version. Rezk used `Matlab`'s 2012b version and the `SPM8` version of the `SPM` package, while we used `Matlab` 2016b and `SPM12`.
Apart from this, the ML part of this baseline is the exact same as Rezk's. This is evidenced by the fact that we have been able to reproduce his results with our code, starting from the same t-maps and ROIs as him.

4.2 Cross-modal decoding

Again, the cross-modal decoding pipeline is summarized with the pseudo-code 3. Changes with regard to within modality decoding are highlighted in blue. The permutations and bootstrapping process is the same as for within modality decoding (pseudo-code 2)

The cross-modality analyses from Rezk's thesis were also replicated: that is, again for each region and each hemisphere, train a model on one modality, test on

Data: t-maps for each subject, ROIs for each subject
Result: performance evaluation of the chosen classifier for cross-modal decoding

```
for subject ← 1 ... 23 do
    apply masking to get each ROI from all the t-maps ;
    demean each voxels pattern ;
    for ROI ← [left PT, right PT, left V5, right V5] do
        for modality_order ← [[vision, audition], [audition, vision]] do
            for test run ← 1 ... 12 do        // cross-validation loop
                train runs ← 1 ... 12 without test run ;
                train data ← normalize_fit_transform(subject, ROI,
                 modality_order[1], train runs) ;
                test data ← normalize_transform(subject, ROI,
                 modality_order[2], test run) ;
                decoder ← SVM_fit(train data) ;
                predictions ← decoder.predict(test data) ;
                accuracy ← compute_accuracy(predictions) ;
            end
        end
    end
end
return averaged accuracies
```

Algorithm 3: Cross-modal process pseudo-code

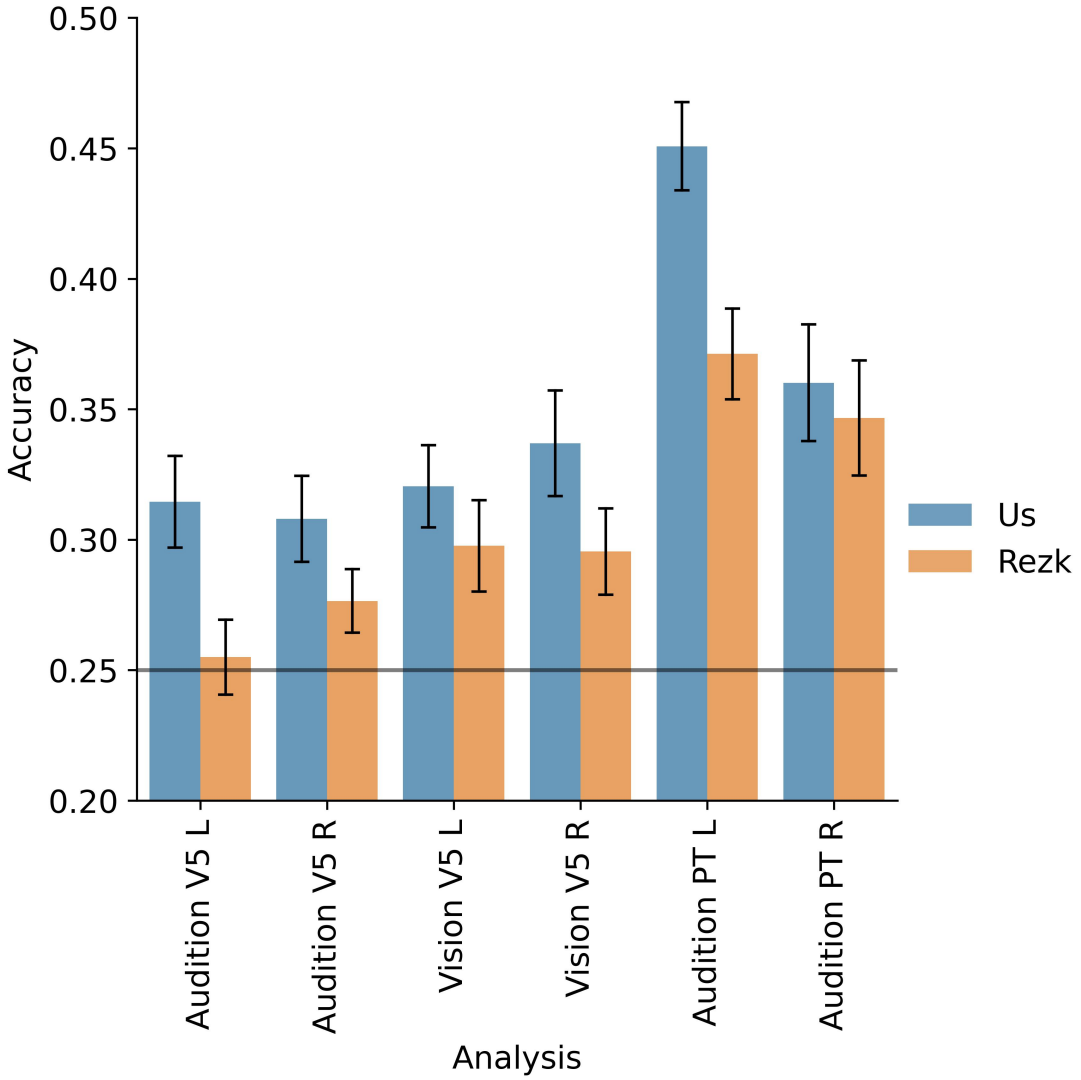

Figure 4.6: Comparison of within modality decoding results

the other modality, and vice-versa (averaging results of both). In total, that means 4 analyses (2 regions x 2 hemispheres) for each subject. For each analysis there is thus a dataset consisting of 96 samples x 171 features (i.e. twice the size of the dataset used for within modality decoding, as both modalities are combined).

Then, an additional step that is needed is a normalization across features. As illustrated on figure 4.8, in general the difference in voxel intensity between audition and vision is far greater than between each class within the same modality. If not treated this could make cross modal decoding very difficult. To remedy this, for

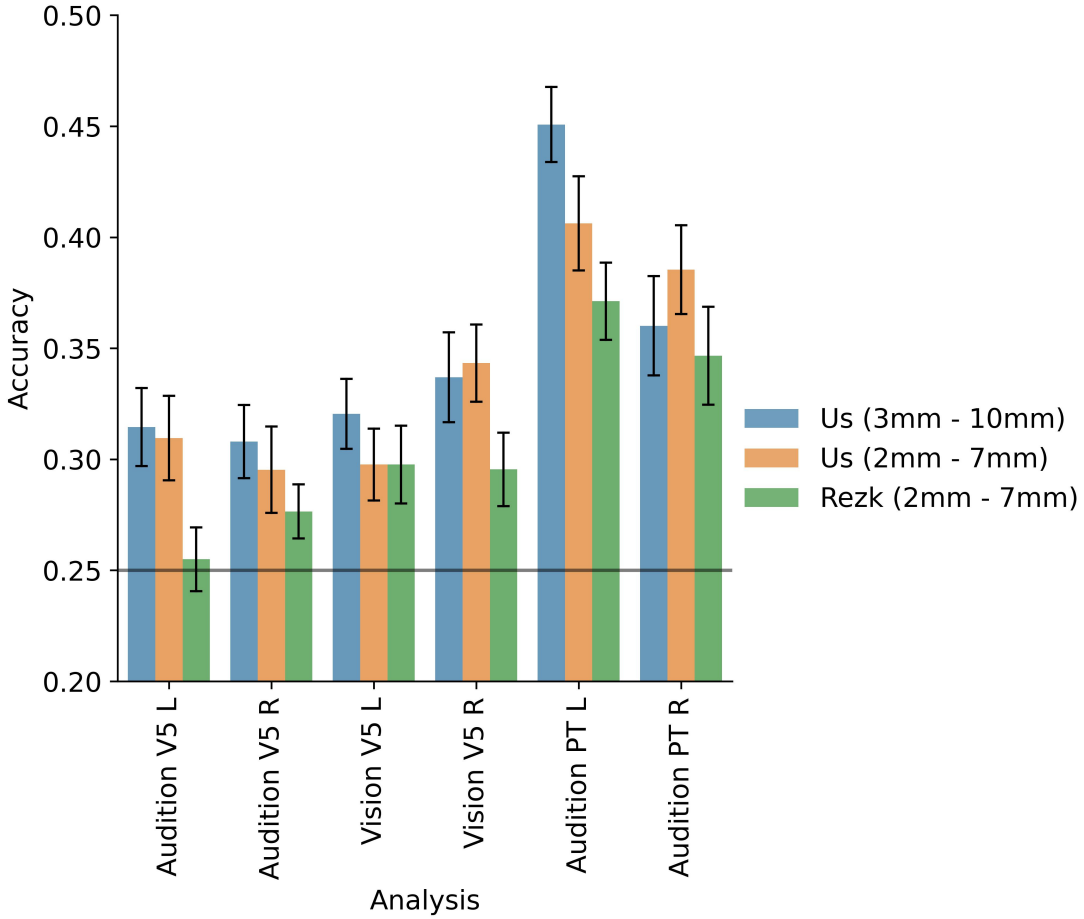

Figure 4.7: Comparison of within modality decoding results. Voxel resolution and ROI radius are detailed in the legend

each sample, its mean is substracted, making sure that their mean is equal and independent from the modality.

After this, the process is almost completely the same as for within modality decoding, with the exception of testing data (line 9) that is taken from the other modality dataset. The results of both directions are averaged.

Results

Concerning cross-modal decoding no significant decoding was observed, neither in PT nor in V5 (figure 4.9). Although this was expected for PT, there were no evidence for information encoding that was shared for visual and auditory motion

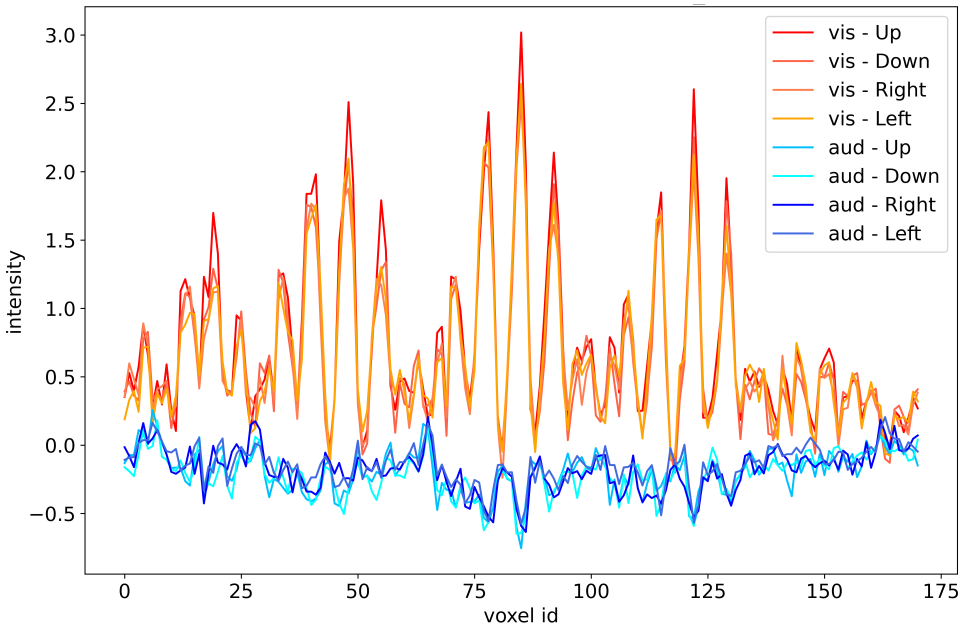

Figure 4.8: Average voxel intensities in right V5 for each class and modality

directions in V5.

When taking a closer look at the confusion matrix for cross-modal decoding in left V5 on figure 4.10, it appears that the classifier predicts more 1 or 2 classes than the others. This also holds for cross-modal decoding in the other 3 ROIs.

Comparison with thesis results

In contrast to within modality decoding, results of cross-modal decoding were better in Rezk's thesis, especially in right V5 where he obtained significant results while we did not, as clearly visible on figure 4.11. These differences have the same source as those for within modality decoding: software versions.
It is important to note that, regarding cross-modal decoding, none of the below modifications or analyses led to an improvement that could make the results significant. It is the reason why we will not talk about cross-modal decoding any further in this thesis.

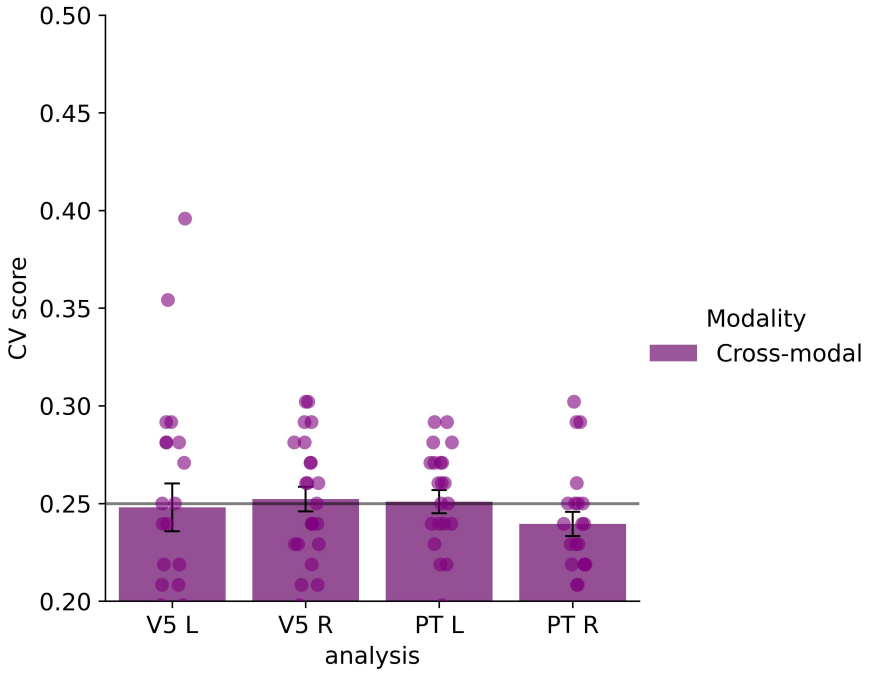

Figure 4.9: Cross-modal decoding accuracies with baseline procedure

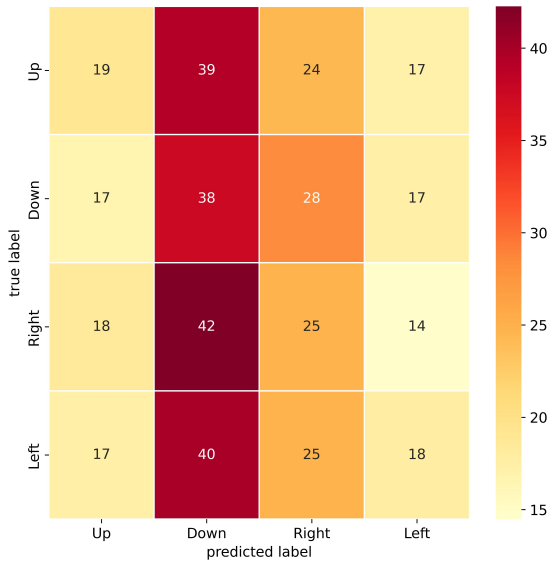

Figure 4.10: Confusion matrix for cross-modal decoding in left V5

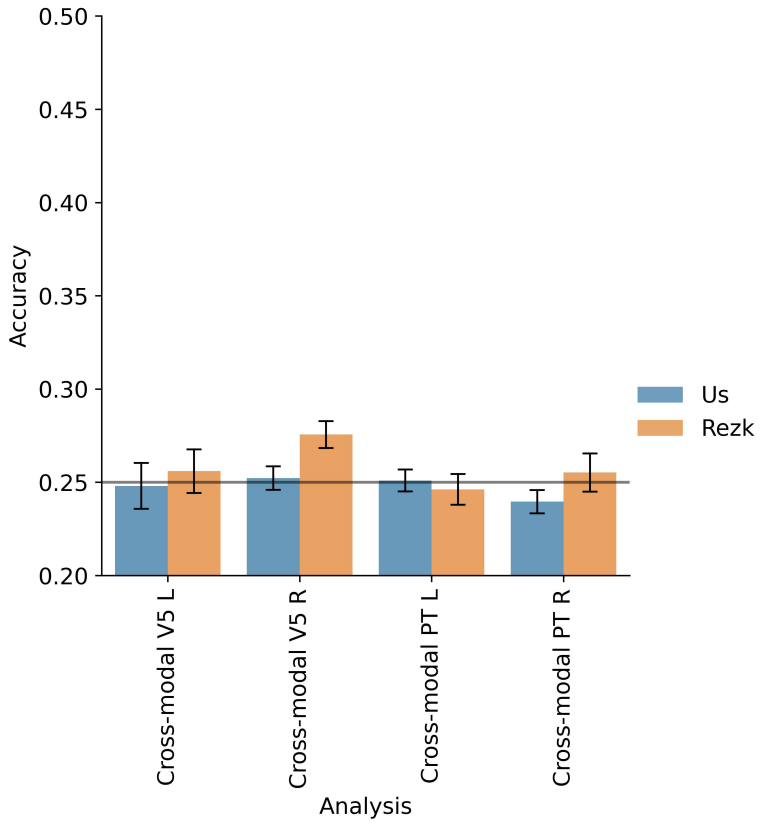

Figure 4.11: Comparison of cross-modal decoding results

Chapter 5

Experiments and results

This chapter aims at describing the experiments performed in order to improve the process of fMRI decoding. All modifications made to the above baseline pipeline will be presented, their motivations, as well as the obtained results and the according interpretations.
For each of these experiments, a four-way (factors are the technique used, the region, the hemisphere and the modality) repeated measures ANalysis Of VAriance (ANOVA) test, will evaluate if a modification improves within modality accuracies compared to the baseline results of the previous chapter. Also, the influence of the modification on the accuracy inter-subjects standard deviation will be inspected, as a lower standard deviation can be seen as an improvement and this standard deviation is often very high in the baseline results (around 10%).
On figures below, results are presented graphically, but all the numerical results are available in appendix A.

5.1 Beta-maps and t-maps: what to use

There are 2 outputs from the fMRI preprocessing: beta maps and t-maps (see section 1.1.5 for an explanation of the difference). As detailed in section 2.1, there is no evidence for one of the 2 types of maps to lead to better results than the other. It is the reason why we decided to compare both types, making them go through the exact same pipeline. Results are visible on figure 5.1.

Although there are variations in these results, they are not always in favor of the same map type. The p-value for evaluating the differences was only of $0.94 (F(1, 19) = 0.007)$. We therefore kept using t-maps for next experiments.

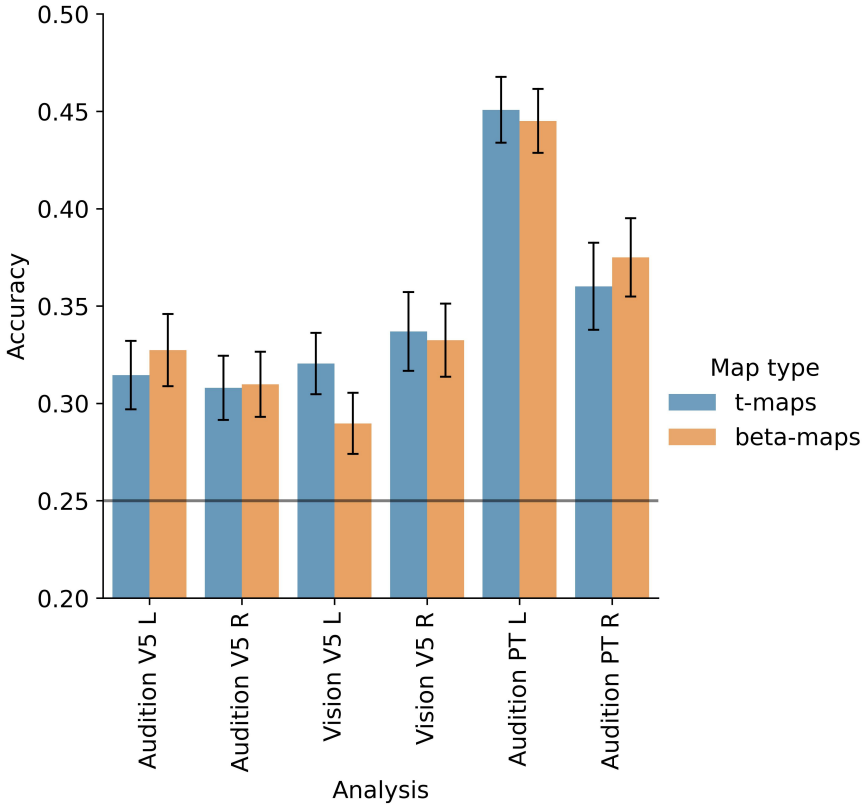

Figure 5.1: Comparison between beta maps and t-maps

5.2 Varying radius of ROIs and voxel resolution

As explained in section 3.3, ROIs are used to select the voxels on which the decoding will be performed. In this thesis, the ROIs used in most of the analyses are spheres of 10mm radius with voxels of 3 x 3 x 3 mm resolution. In such a sphere there are 171 voxels. In the analyses carried out by Rezk, the radius was set to 7 and the voxel resolution to voxels of 2 x 2 x 2 mm in size. These changes in voxel size and radius of the ROI influence the number of voxels considered and therefore the number of features for the Machine Learning model to be learned. In fact, as outlined in chapter 5, these changes led to significant accuracy differences. So the influence of these parameters was further explored. The following different values for the radius were used: 5, 6, 7, 8, 9 and 10 (millimeters). For the voxel resolution voxels of 3mm side length and 2mm side length were used. The plots presenting the obtained results for these analyses are presented in figures 5.2, 5.3, 5.4 and 5.5.

The values on the y axis here represent the mean accuracy across subjects.

The first four graphs illustrate the global trend is an increase in accuracy when increasing the radius of the ROI. This increase stops after reaching a plateau. This probably means that once the useful voxels are included in the ROI, adding even more is pointless.
It is also worth noting that the difference between the maximum and minimum accuracy in each of these graphs is not negligible.

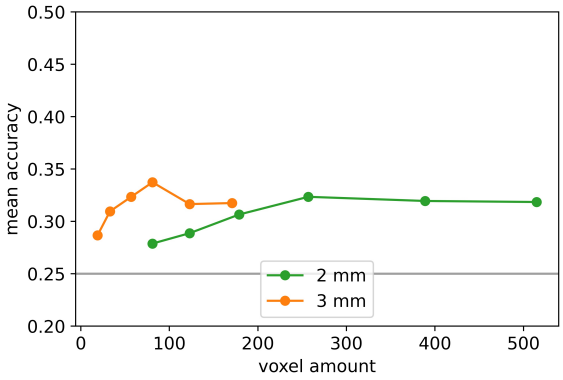

Figure 5.2: auditory motion decoding in left V5 depending on ROI radius

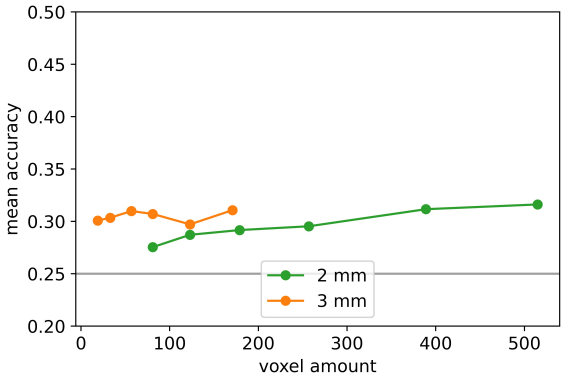

Figure 5.3: auditory motion decoding in right V5 depending on ROI radius

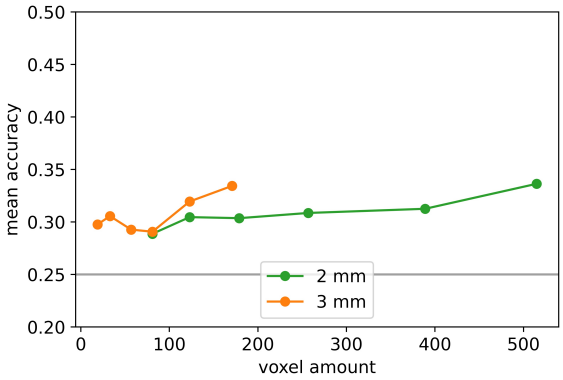

Figure 5.4: Visual motion decoding in left V5 depending on ROI radius

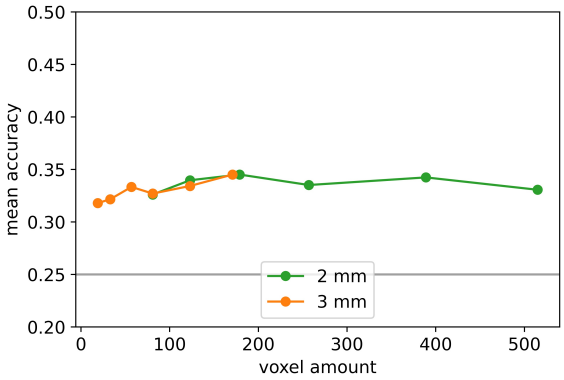

Figure 5.5: Visual motion decoding in right V5 depending on ROI radius

5.3 Different ML models

Now that the obvious parameters on the fMRI side of things was investigated, the same principle will be applied to the ML side. To this end, we decided to test

several ML models :

- SVM, which is the baseline and is often the basic classifier used in fMRI decoding, notably for its fast training with lots of features (see sections 1.2.5 and 2.1)
- LR, because of its inner multi-class strategy and its tendency to produce better results in fMRI decoding (see sections 1.2.5 and 2.1)
- KNN, as it is a very intuitive classifier and does not require a real training (see section 1.2.5)
- Perceptron, as it is one of the most basic and most used classifier (or at least its derivatives) in the ML field (see section 1.2.5)
- ELM, because of its good performances in previous studies in fMRI decoding (see section 2.3)

Hyper-parameters of these models have been previously tuned. The procedure to do so will be detailed in the next section.

The graph on figure 5.6 shows clear differences between the classifiers. KNN nearly always shows much worse results than the baseline SVM, the ANOVA test revealing it with a p-value of $0.0002(F(1, 19) = 21.260)$. The ELM and the Perceptron sometimes reach the same results as the SVM, but perform worse for other analyses. The ANOVA test confirms that both perform worse than the SVM, with p-values of respectively $0.041(F(1, 19) = 3.439)$ and $0.016(F(1, 19) = 7.042)$. Concerning LR, it seems to outperform all others classifiers. The ANOVA test indeed confirms with a p-value of $0.042(F(1, 19) = 4.742)$ that it outperforms the baseline.

For further experiments we will then focus on SVM and LR because the first one is the baseline and the second one was identified as the best performing classifier.

5.4 Hyper-parameters tuning

Nearly all ML models have hyper-parameters, but they are often set to a default value in software implementations and not enough attention is paid to them in the fMRI field. There is therefore a possibility that correctly tuning them could enhance the results.
To this aim, a nested cross-validation scheme was used, such as explained in section

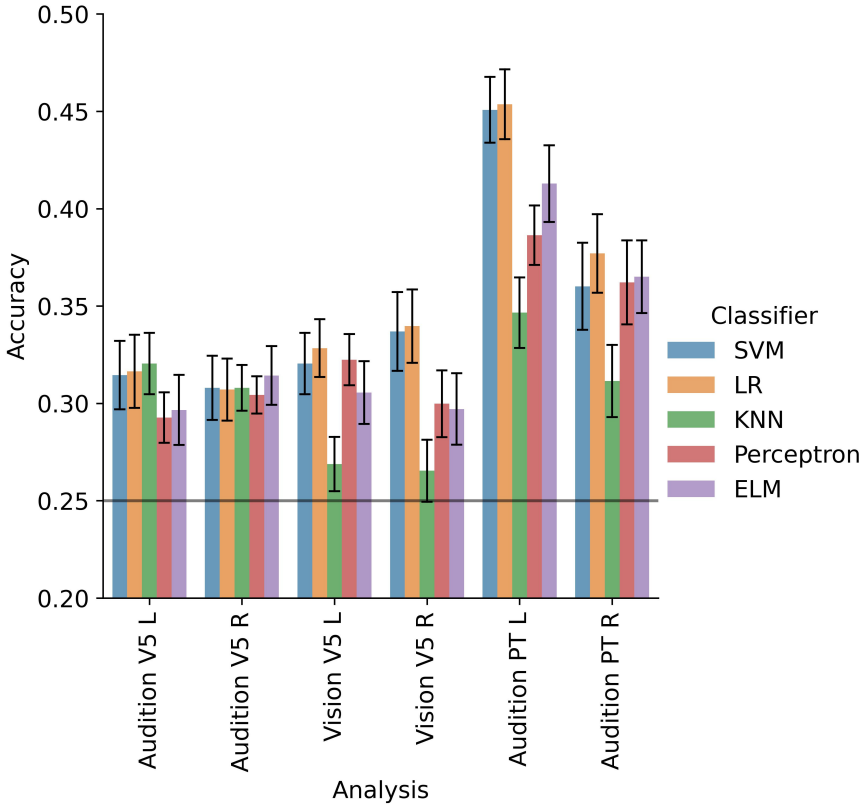

Figure 5.6: Comparison between ML models

1.2.2. Like the initial cross-validation, it is implemented in a leave-one-run-out way.

5.4.1 SVM

An SVM has many hyper-parameters. We will focus on two of them: the C regularization parameter and the type of kernel (see section 1.2.5). These are the ones that often influence the most the results(33). In neuroscience, it is common practice to set C to 1 and use a linear kernel (20), but this does not mean it is always the right choice.
Remember, this dataset offers a space with many dimensions, and less training samples than dimensions. Therefore, the algorithm will always find an hyper-plane perfectly separating all classes (with a one-vs-rest multi-class strategy). But this will not necessarily lead to the best margin. Therefore one could play with the C regularization parameter in order to introduce a tolerance. Remember that the higher the C, the less tolerance there is for misclassified samples (section 1.2.5).

Values of C between 10^{-3} and 10^{3} were used, in a logarithmic scale.
Moreover, the use of another kernel than the linear one could allow to capture more types of functions (33). In addition to the linear kernel, a Gaussian kernel (the most used when using SVMs in the ML field) also called RBF (Radial Basis Function) and a sigmoid kernel were therefore used. For these last two, the variance parameter (often called "gamma") was also tuned to make sure kernels are compared at their best capacities (see this document for further explanations of these concepts).

Validation scores for auditory motion decoding in left PT and visual motion decoding in right V5 are plotted on the figures below (figures 5.7 and 5.8 respectively). The graphs are very similar for other ROI-modality combinations.

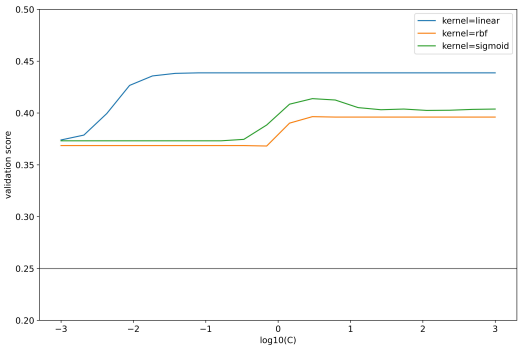

Figure 5.7: Validation score for auditory motion decoding in left PT using an SVM

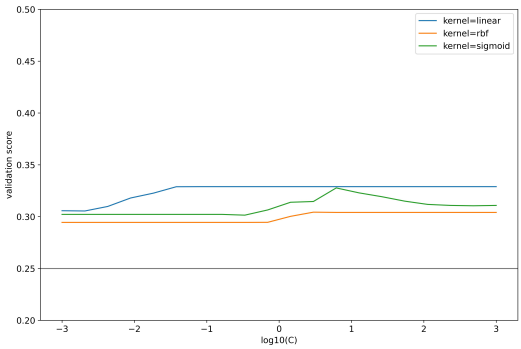

Figure 5.8: Validation score for visual motion decoding in right V5 using an SVM

The first striking observation is that the linear kernel outperforms the other kernels in most cases. So it is more interesting to use this simple kernel rather than a more complex one.
Then, it appears that with any kernel, the curves show 2 plateaus, the highest one corresponding to higher values of C. This implies that the best results are obtained when only a few tolerance towards misclassified samples is included. Results are better when the hyperplane perfectly separates the data.

The conclusion is quite obvious: the standard SVM parameters values used in fMRI decoding, namely a linear kernel and C equal to 1, actually are the optimal values for this dataset. Moreover, these plateaus do reflect the fact that the SVM is quite robust: the results do not drastically change when slightly modifying the C parameter.

5.4.2 LR

Like the SVM, LR has many hyper-parameters to tune. We decided to focus again on the C regularization parameter in order to see how the tolerance towards misclassified samples affects the results. Again, this is the parameter that is expected to influence the most the results.

Validation scores for visual motion decoding in V5 in both hemispheres are plotted on the figures 5.9 and 5.10 below. Validation score curves for other ROIs and modality look very similar to figure 5.9.

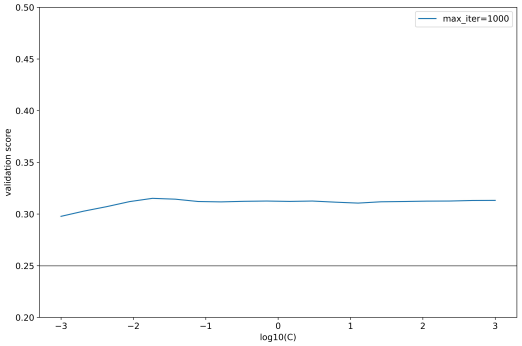

Figure 5.9: Validation score for visual motion decoding in left V5 using an LR

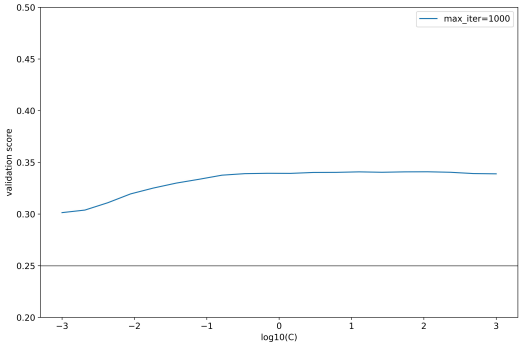

Figure 5.10: Validation score for visual motion decoding in right V5 using an LR

The C parameter do influence the results, but in a smoother way than for the SVM classifier. Indeed, there is no variation higher than 2% between the best and worst C values, except for one analysis (figure 5.10) where a higher C value definitely implies better results.

Nevertheless one can safely affirm that the LR classifier is robust to small changes of the regularization parameter and a C value around 1 leads to the best results in this case.

5.5 Feature selection

There are many different approaches to feature selection. The two approaches we used are filter methods and wrapper methods. As explained in section 2.2, these techniques were used in several fMRI researches. One of them used two

filter methods then experimented with two wrapper methods (21) based on genetic algorithms proposed by Frölich et al. in 2002 (22). Filter methods select features based on a criteria while wrapper methods recursively eliminate features that have the least influence on a classifier (see section 1.2.4). The feature selection step is always the one that comes right before classification in the pipeline.

5.5.1 Filter methods

When using filter methods one can play with two parameters: k, the number of features to select and *score_func*, the filter criteria to use. The function then selects the k best features according to the selected criteria. Two different criterion were used: *f_classif* and *mutual_info_classif*. For each criteria, the model is used with different values for k so as to see the influence of this parameter on the accuracy. This experiment begins with the first criteria, *f_classif*. This criteria is based on the F-statistic which is the variation between samples mean divided by the variation within samples. The values taken for k correspond to 20%, 40%, 60%, 80% and 100% of the number of voxels in the studied ROI. This number is equal to 171. The obtained mean accuracy and its inter-subject standard deviation are presented in figures 5.11 and 5.12 for the SVM classifier and in figures 5.13 and 5.14 for the LR classifier.

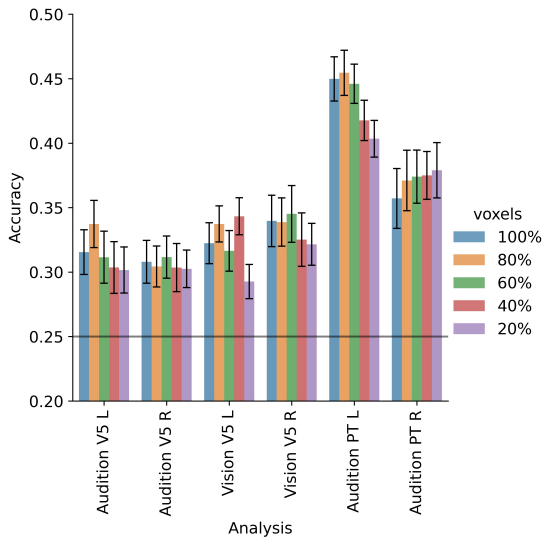

Figure 5.11: Mean accuracy depending on the percentage of voxels selected (*f_classif*) using SVM

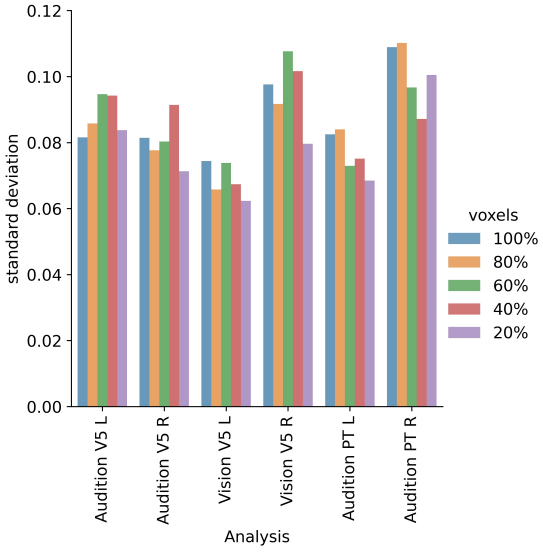

Figure 5.12: Accuracy standard deviation depending on the percentage of voxels selected (*f_classif*) using SVM

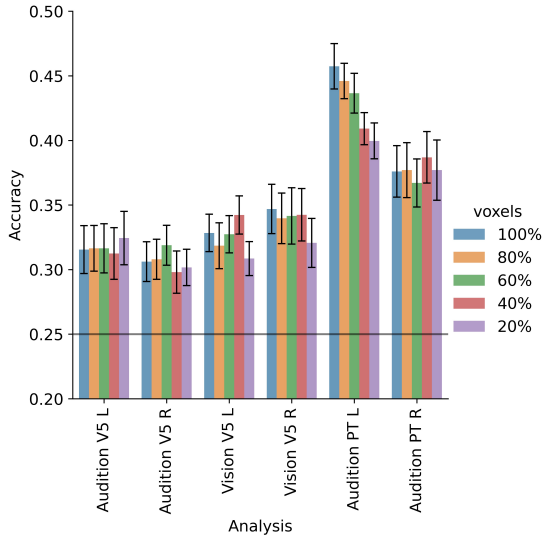

Figure 5.13: Mean accuracy depending on the percentage of voxels selected (*f_classif*) using LR

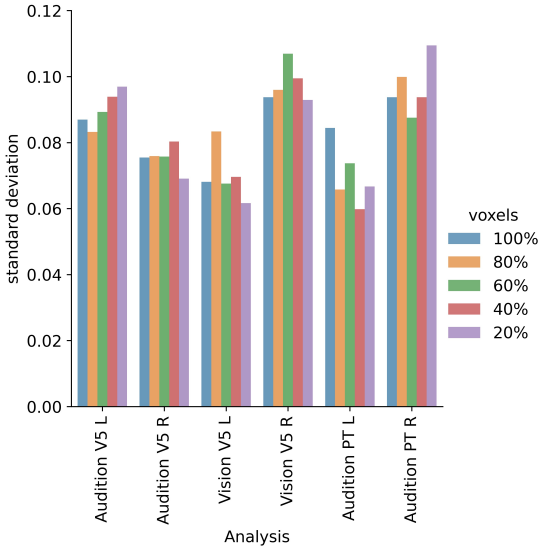

Figure 5.14: Accuracy standard deviation depending on the percentage of voxels selected (*f_classif*) using LR

When observing these results no recurring pattern appears. For example when 40% of the voxels ($k = 68$ voxels) are selected in figure 5.11, the model is the one that gives the best accuracy for vision in left V5 but also the one that gives the second worst accuracy for audition in left V5! The variance of the accuracy does not seem to follow a pattern either.

Now what if the criteria that selects the features to be kept for classification is different? The *f_classif* method helps capture linear dependency between variables, whereas "mutual information methods can capture any kind of statistical dependency" between features (see software documentation). Figures 5.15, 5.16, 5.17 and 5.18 present the results obtained with this second criteria.
The p-value obtained from the ANOVA test (which has then an additional factor in the percentage of selected voxels) when comparing results for the two criteria is of $0.39 (F(1, 19) = 0.774)$ for SVM and $0.44 (F(1, 19) = 0.624)$ for LR, meaning there is no significant difference in accuracy for these two experiments for both models. Additionally, the baseline results are often better than the results with feature selection when the *mutual_info_classif* criteria for the LR model is used. The p-value that indicates if feature selection itself has an influence on the result is of $0.0095 (F(4, 76) = 3.610)$ for the SVM classifier and is lower than $0.0001 (F(4, 76) = 7.726)$ for the LR classifier.

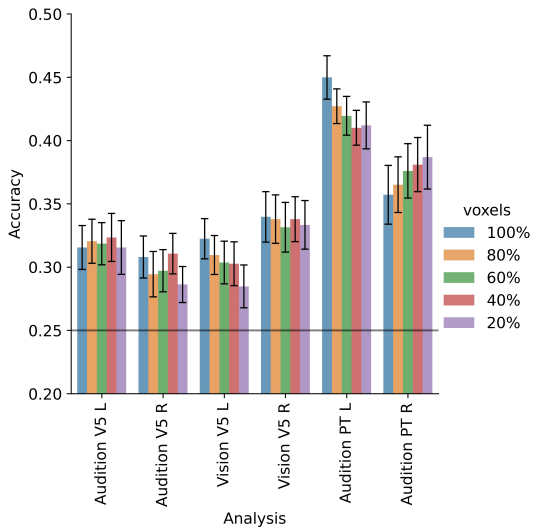

Figure 5.15: Mean accuracy depending on the percentage of voxels selected (*mutual_info_classif*) using SVM

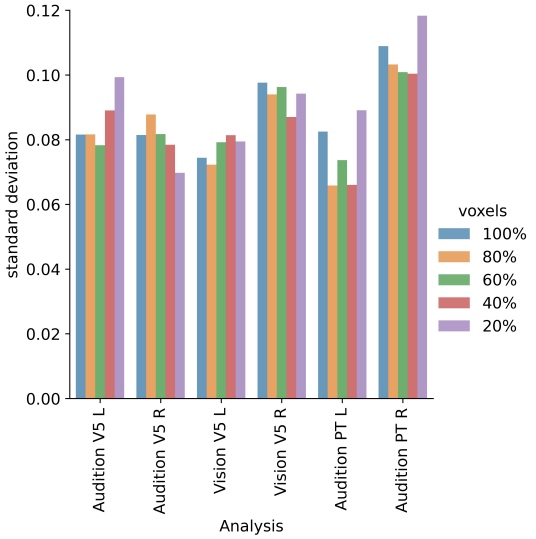

Figure 5.16: Accuracy standard deviation depending on the percentage of voxels selected (*mutual_info_classif*) using SVM

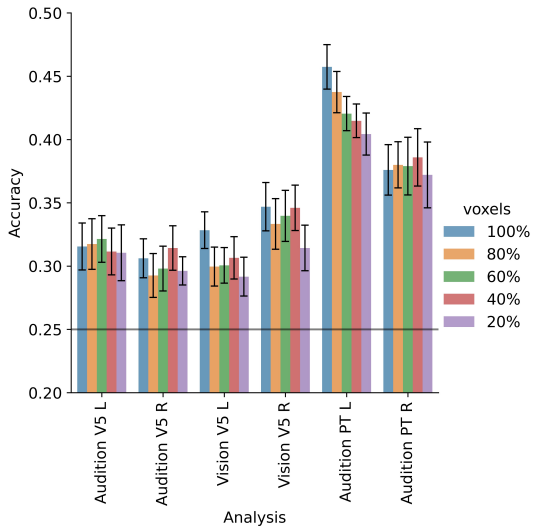

Figure 5.17: Mean accuracy depending on the percentage of voxels selected (*mutual_info_classif*) using LR

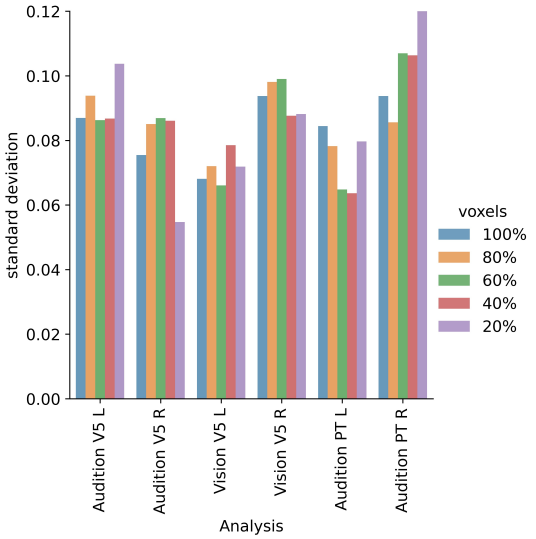

Figure 5.18: Accuracy standard deviation depending on the percentage of voxels selected (*mutual_info_classif*) using LR

5.5.2 Wrapper method

As suggested in multiple papers from the literature (chapter 2), a simple but yet effective way to improve results in fMRI decoding is using RFE (see section 1.2.4 for a memory refresh). We therefore decided to see how it could improve the results. As in the base case there 171 features, we decided to use RFE to eliminate 20%, 40%, 60%, and 80% of the features, eliminating 5 features at each iteration. It was used for both SVM and LR classifiers. For both cases, the same classifier was used (same type and same hyper-parameters) for the RFE and classification steps.

SVM

SVMs are known for being sensitive to RFE, also in fMRI decoding like in the study of De Martino et al. (34).
The figures below illustrate the influence of RFE on the average accuracy (figure 5.19) as well as the inter-subjects standard deviation (figure 5.20) when using an SVM-RFE to select the best features, before classifying with an SVM as well.

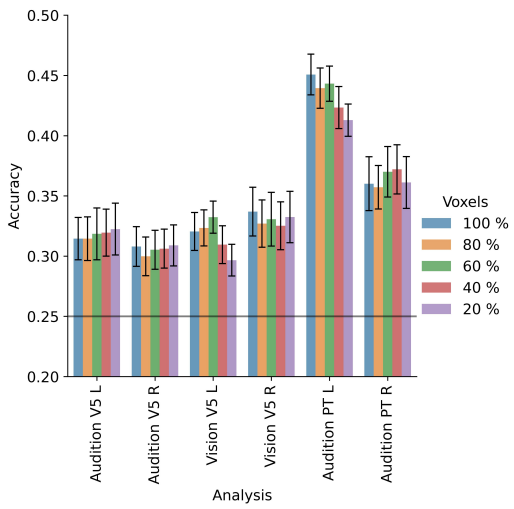

Figure 5.19: Mean accuracy using SVM-RFE with an SVM

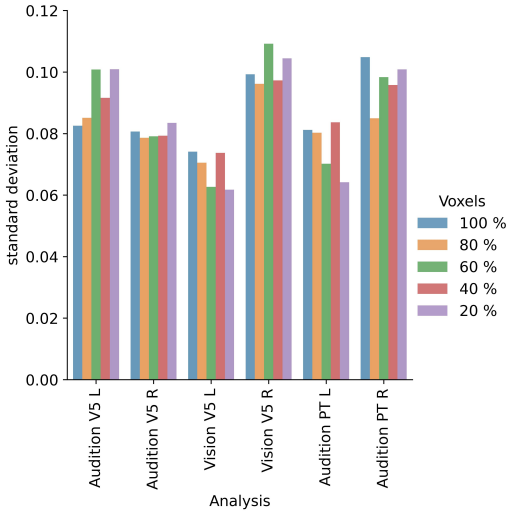

Figure 5.20: Accuracy standard deviation using SVM-RFE with an SVM

It appears that depending on the analysis RFE can improve or decrease accuracy. For example for auditory motion decoding in V5, the stronger the selection, the higher the accuracy. But for auditory motion decoding in left PT, it is the opposite

situation: the harder the selection, the lower the accuracy. And for other analyses the influence of RFE is not clearly defined. Moreover the ANOVA test gave a p-value of $0.49(F(4, 76) = 0.864)$ for the influence of the fraction of selection using the RFE in this case.
Regarding the inter-subjects accuracy standard deviation, no strong influence of RFE use can be derived.

LR

Less improvements with RFE were reported for LR. Nevertheless we performed it (that is using a LR-RFE before classifying with a LR as well) in order to see the influence on accuracy (figure 5.21) and inter-subjects standard deviation (figure 5.22).

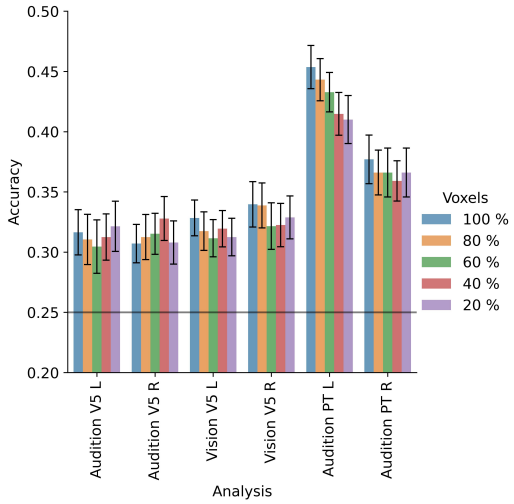

Figure 5.21: Mean accuracy using LR-RFE with a LR

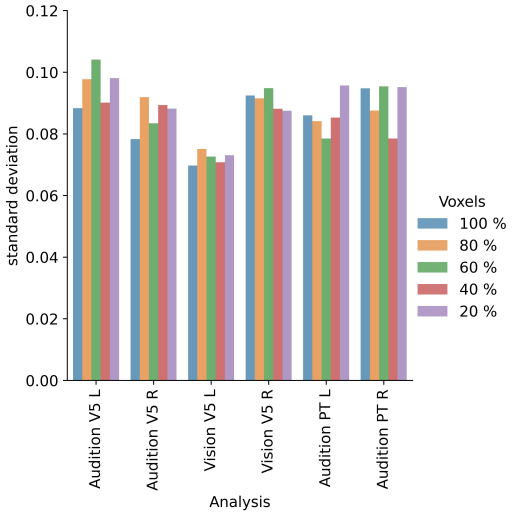

Figure 5.22: Accuracy standard deviation using LR-RFE with a LR

For the LR, RFE almost always decreases results. This is confirmed by the p-value of $0.034(F(4, 76) = 2.758)$ of the ANOVA test, meaning that the percentage of selection really influences the accuracy, unfortunately towards lower scores.
Regarding the inter-subjects accuracy standard deviation, again no a strong influence of RFE use can be derived.

5.6 Data augmentation

As stated in section 2.4, the most common data augmentation technique used in the fMRI field are GANs and other neural networks. However using these techniques is difficult, timely expensive and their convergence is not guaranteed. Actually, the convergence of such networks is still an open problem (35). Only a few studies used the SMOTE technique (see section 1.2.6) to augment an fMRI dataset, sometimes with success (36). it is therefore a motivation to see if it could have an impact on the results. In this case SMOTE was used to augment the number of samples in all classes as there are very few of them all. Indeed in the case the classifier does not have enough data to be correctly trained then creating new samples with combinations of the original data could help it learn a better model.

Remember there are 11 samples for each class to train the model. So if s is set equal to 15, the algorithm will generate 4 new samples. Three possibles values for k were used: 2, 3 and 5. For the parameter s, the following values were used: 15, 20, 30, 40 and 50. These values enable to visualise accuracy changes when k and s take small, medium and large values with respect to the size of the dataset. The results for all combinations of k and s are not shown on below figures as it makes for a messy visual.

SVM

The mean accuracy and inter-subjects accuracy standard deviation from this experiment are presented in figures 5.23 and 5.24 for an SVM classifier.

First, regarding inter-subject accuracy standard deviation, no strong influence of SMOTE is observed.
As regards mean accuracy, it appears that depending on the analysis, SMOTE sometimes really improves mean accuracy, but at the same time decreases accuracy for other analyses. Actually, the ANOVA test does not reveal an influence of the k parameter (p-value of 0.42, F(2, 38) = 0.893), but reveals an influence of the s parameter with a p-value lower than $0.0001 (F(5, 95) = 6.3238)$! This influence is not obvious on this graph, we therefore decided to make 2 additional graphs: one where results across the different k values are averaged (figure 5.25), and one where results across the different k values and across the different analyses are averaged (figure 5.26).

Unfortunately, there is a very heterogeneous influence of SMOTE on the accuracy depending on the analysis.
In conclusion SMOTE slightly improved the results when using an SVM classifier,

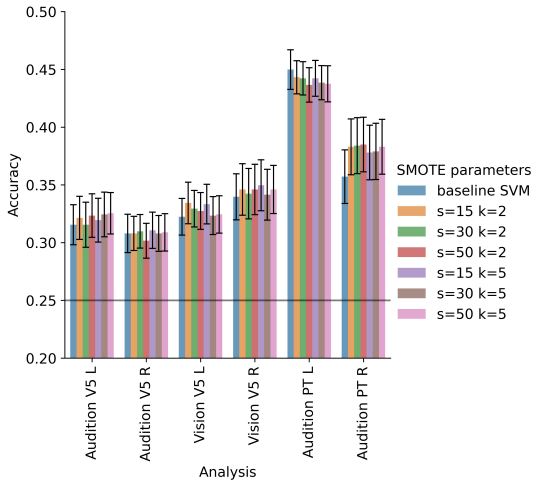

Figure 5.23: Mean accuracy depending on SMOTE's parameters, using SVM

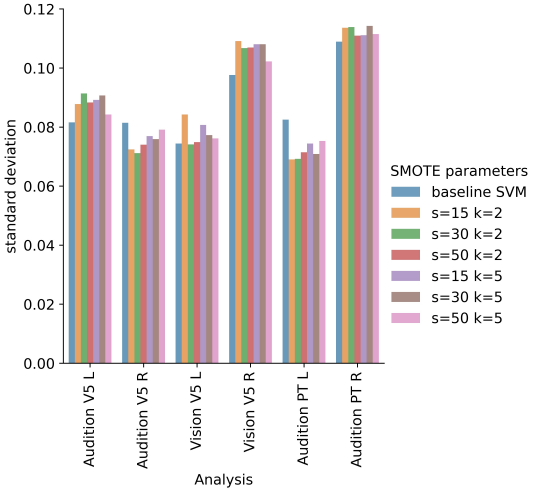

Figure 5.24: Accuracy standard deviation depending on SMOTE's parameters, using SVM

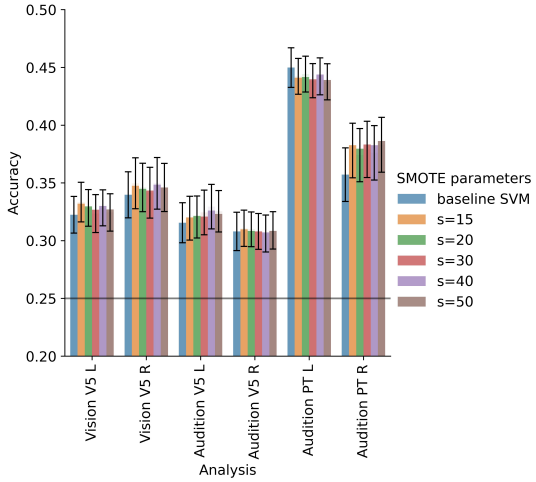

Figure 5.25: Mean accuracy averaging on k, using SMOTE with SVM

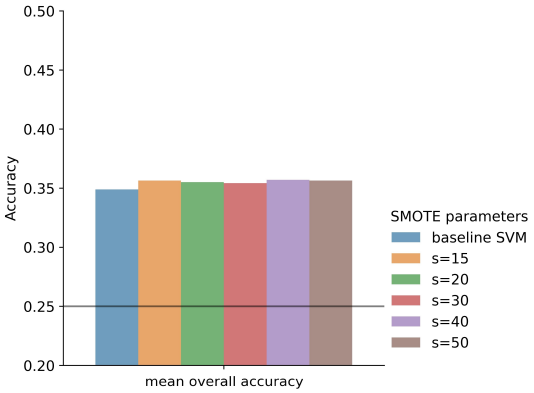

Figure 5.26: Mean accuracy averaging on k and analyses, using SMOTE with SVM

but not in a reliable and satisfactory way.

LR

Figures 5.27 and 5.28 represent results obtained with a LR classifier.

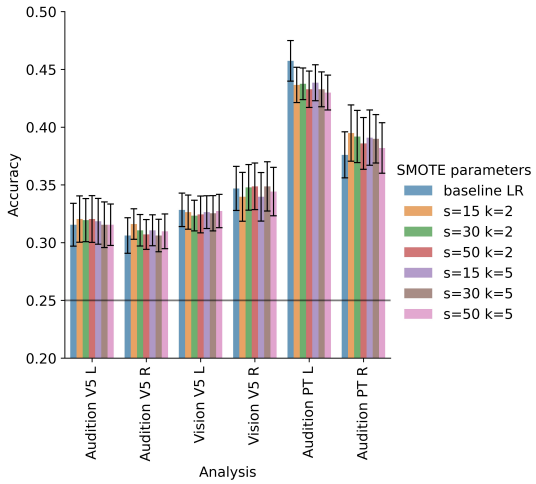

Figure 5.27: Mean accuracy depending on SMOTE's parameters using LR

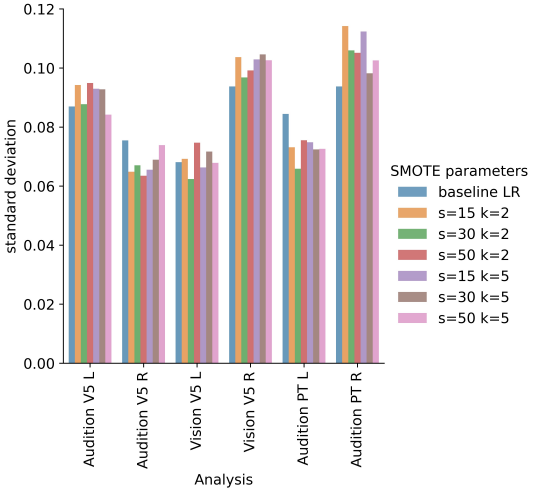

Figure 5.28: Accuracy standard deviation depending on SMOTE's parameters using LR

Again, depending on the analysis, SMOTE either improves or reduces accuracy. But in this case, the p-value for the ANOVA test for the influence of the k and s parameters are not even close to be significant: respectively $0.87(F(2, 38) = 0.868)$ and $0.57(F(5, 95) = 0.566)$. Actually, it appears that, regardless o the classifier used, the influence of SMOTE is really strong for 2 analyses: audition in left and right PT! But in the left case, it decreases the accuracy, and in the right case it improves it. It is therefore difficult to draw any conclusion from that observation. Regarding inter-subject accuracy standard deviation, again no strong influence of SMOTE is observed.

5.7 Average of 2 test sets

Up until now the nested cross-validation was performed with one run as test set and one run as validation set (Remember, each run contains 4 samples, one per class). This means training is done on 10 runs (= 40 samples). What has been proposed is to change the way testing is done. For this experiment it was decided to test on the average of 2 test sets (the samples from the 2 test sets were averaged pairwise, according to their class). Instead of 12 different test sets there are now $12 * 11/2 = 66$ test sets. The validation was still done with 1 run and therefore training is performed on 9 runs (= 36 samples). This idea emerged during a meeting with our supervisors. What we hope from this technique is an increase in accuracy. Indeed by testing on the mean of two test sets one can hope to reduce the influence

of noise in the data, while increasing the reliability of the performance measures. In the experiment the mean accuracy and its standard deviation were compared for two models, the SVM and LR classifiers.

The results for within modality decoding are presented in figures 5.29 and 5.30. A pattern emerges on both figures. There is an increase in accuracy with the proposed technique but at the cost of an increase of the inter-subjects standard deviation. The p-value that evaluates the significance of the increase in accuracy is equal to $0.0002(F(1,19) = 21.314)$ for the SVM classifier and to $0.0007(F(1,19) = 16.106)$ for the LR classifier. It can be inferred that there is a significant improvement when using such a technique !

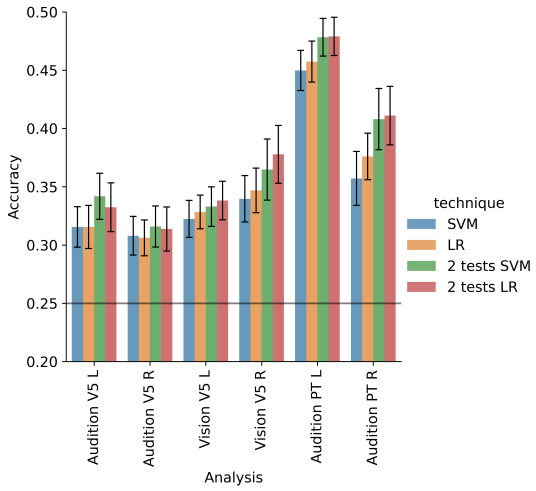

Figure 5.29: Mean accuracy when averaging 2 test sets

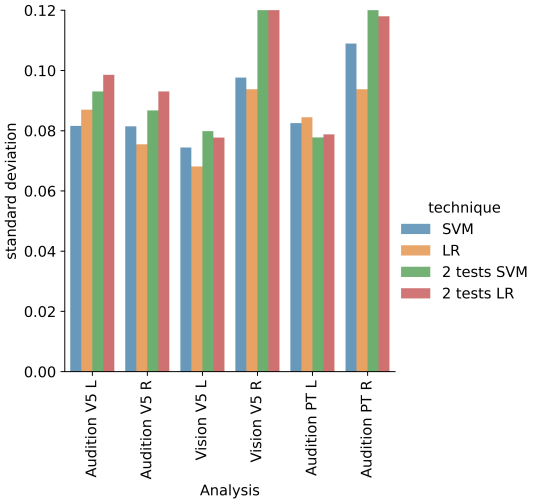

Figure 5.30: Accuracy standard deviation when averaging 2 test sets

Conclusion

This thesis aimed at exploring the application of Machine Learning on fMRI brain data. The introduction provided some context about the motivations of this thesis, and presented the question that occupied us during this thesis: Are there ML techniques that can reliably improve results in fMRI decoding, in the context of motion direction decoding in V5 and PT ?
Chapter 1 set up the background knowledge that was needed to understand this work, both on the fMRI and ML sides.
We gave an overview of state-of-the-art methods used in fMRI decoding in chapter 2. We have seen that, so far, different techniques are used and it does not seem that one technique is guaranteed to improve classification scores. Many approaches are still in competition in order to improve fMRI decoding. It is known that fMRI decoding is not an easy task, as performances are often (much) lower than for "traditional" Machine Learning problems.
The data used in this study and its particularities were described in chapter 3. It was pointed out that this dataset was quite limited, and actually will not allow to draw conclusions for general fMRI decoding cases. Nevertheless, it still represents a valid case study to analyse the influence of ML techniques on such data.
Chapter 4 presented the first ML pipeline used for this thesis, based on Rezk's study. This represents the baseline on top of which different techniques were added in order to improve the process. The initial results were presented, as well as differences with Rezk's work.
Finally, the chapter 5 explored the different techniques applied to the baseline pipeline, along with the reasons that lead to their implementation, the results they brought, and the according interpretations.

Now let us go back to the initial question: are there techniques that reliably improve fMRI decoding, in the context of motion direction in V5 and PT? First, it is safe to say that only a subset of the implemented techniques improved the results.
In the first place it is the choice of the classifier that is critical. For future similar studies, we definitely advise to use SVM or LR to classify fMRI data.

Then, hyper-parameters values have to be wisely chosen. As demonstrated, although the models were robust to small changes of these values, a poor value choice could cause a real drop in results. In particular, we advise to not include much tolerance towards misclassifications, as trying to improve generalization of the model is quite complicated with a small dataset and fMRI decoding dataset are often small.
Another technique that increased the accuracy was using averaged runs as test set. Doing so also increases the reliability of the performance measures by increasing the amount of test performance measures, but it is important to keep in mind that this technique lead to an increase in inter-subjects accuracy standard deviation.
The other experiments were less successful. First, at the border between fMRI and Machine Learning, It appears that there was no special advantage of choosing beta-maps rather than t-maps.
Then, it was highlighted that increasing the radius of the ROIs was useful up to a certain point, which was already met in the baseline pipeline, and the resampling of the voxel did not really improve the results.
More ML specific techniques, such as feature selection, did not improve the results either: both filter and wrapper methods did influence the results, but not in a consistent way.
The same conclusion was drown for data augmentation using SMOTE: sometimes it helps, sometimes it makes things worse.
In the end there are on one hand techniques that improved the results, and that we advise people to at least try if they are working on fMRI decoding. On the other hand, some techniques were not successful, and we would advise to only implement them if there is enough time for it.

However, all these conclusions have to be moderate, because this work comes with its limitations.
First, the mean accuracy and inter-subjects standard deviation changes observed were always small, there was nothing obvious. This is a direct consequence of the similarities between classes: the fMRI images are very similar for the different conditions. Improvements found in this thesis are really often on the edge between significant and not significant, hence it is not possible to affirm 100% that this or that technique is guaranteed to improve results.
Then, as previously said, the dataset used in this study consists of 23 subjects. This lowers its statistical power, and the conclusions might be different with a larger amount of subjects. Also, it would be better to work on different datasets at the same time, in order to avoid to make improvements that would be too specific to a certain experiment design or fMRI scanner. Unfortunately a major obstacle is that such experiments are very expensive. Even obtaining already existing datasets

is difficult and costly. It is likely that sharing data between neuroscientists could help reduce these costs. For the sake of confidentiality it is also possible to explore ML techniques such as federated learning so as not to transfer the data itself.
Moreover, there are more advanced techniques that were not investigated in this work, due to the limited time at our disposal. One could for example look at embedded methods to make a more relevant feature selection. Such techniques make use of classifiers performances to greedily select or eliminate features, but this comes at the cost of an increased computation time, as the amount of possible features subsets is exponential.
Finally, we limited ourselves to the analysis of mean accuracy and inter-subjects standard deviation, but there are many other ways to measure performances of a Machine Learning process, such as inspecting if a technique improves results for all subjects in the same way, or if the change is more messy.

At the time of finalising this thesis, we would like to highlight what are, in our sense, the most interesting future works that could be done in fMRI decoding using Machine Learning. We believe that the choice of an appropriate ML classifier is key to enhance results, and Machine Learning offers many possibilities that are worth exploring. We are also convinced that extending analyses to multiple and bigger datasets could help to draw more robust conclusions on the contribution of ML techniques. We believe the data augmentation path could prove interesting, as even big fMRI datasets do not contain lots of samples, and exploring more advanced techniques (such as GAN's if one could afford the time) is definitely an idea that is worth trying.

The use of Machine Learning in fMRI decoding is still recent and little explored, but fascinating and challenging! Without any doubt this field will expand in future, as there is plenty of information to discover from a biological point of view. It is therefore worth investigating Machine Learning techniques that could help facilitate these discoveries!

Bibliography

[1] M. Rezk, "Motion representation in hMT+/V5 : influence of cross-modal inputs and visual deprivation," Ph.D. dissertation, UCL - Université Catholique de Louvain, 2021. [Online]. Available: https://dial.uclouvain.be/pr/boreal/object/boreal:245796

[2] R. Dubner and S. Zeki, "Response properites and receptive fields of cells in an anatomically defined region of the superior temporal sulcus in the monkey." *Brain research*, 1971.

[3] "Principles of fMRI 1." [Online]. Available: https://fr.coursera.org/learn/functional-mri

[4] The physics of MRI and how we use it to reveal the mysteries of the mind. [Online]. Available: https://kids.frontiersin.org/articles/10.3389/frym.2019.00023

[5] Figure 7. hemodynamic response function (adapted from buechel c,... [Online]. Available: https://www.researchgate.net/figure/Hemodynamic-response-function-adapted-from-Buechel-C-Fsiston-K-Henson-R-Josephs-O_fig10_228686551

[6] R. A. Poldrack, "Region of interest analysis for fMRI," vol. 2, no. 1, pp. 67–70. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2555436/

[7] Figure 1. common fMRI paradigm designs: block and event-related. [Online]. Available: https://www.researchgate.net/figure/Common-fMRI-paradigm-designs-block-and-event-related_fig1_303820836

[8] "Bad nested CV example? It looks like each CV loop does the same exact split on the data · Issue #9470 · scikit-learn/scikit-learn." [Online]. Available: https://github.com/scikit-learn/scikit-learn/issues/9470

[9] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE transactions on systems, man, and cybernetics*, no. 4, pp. 580–585, 1985.

[10] B. Lantz, *Machine learning with R: expert techniques for predictive modeling*. Packt publishing ltd, 2019.

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

[12] S. SHARMA, "What the Hell is Perceptron?" Oct. 2019. [Online]. Available: https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53

[13] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, ser. COLT '92. New York, NY, USA: Association for Computing Machinery, 1992, pp. 144–152. [Online]. Available: https://doi.org/10.1145/130385.130401

[14] "Draka » A Brand of Prysmian Group." [Online]. Available: http://draka.se/jcss.aspx?iid=102463679&cid=28

[15] R. E. Wright, "Logistic regression," in *Reading and understanding multivariate statistics*. Washington, DC, US: American Psychological Association, 1995, pp. 217–244.

[16] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," vol. 16, pp. 321–357. [Online]. Available: https://www.jair.org/index.php/jair/article/view/10302

[17] M. Inoue, "Oversampling with SMOTE with its relative algorithms," original-date: 2019-12-02T03:01:52Z. [Online]. Available: https://github.com/minoue-xx/Oversampling-Imbalanced-Data

[18] D. D. Cox and R. L. Savoy, "Functional magnetic resonance imaging (fMRI) "brain reading": detecting and classifying distributed patterns of fMRI activity in human visual cortex," *NeuroImage*, vol. 19, no. 2, pp. 261–270, Jun. 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1053811903000491

[19] F. Pereira, T. Mitchell, and M. Botvinick, "Machine learning classifiers and fMRI: A tutorial overview," *NeuroImage*, vol. 45, no. 1, Supplement 1, pp. S199–S209, Mar. 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1053811908012263

[20] R. A. Poldrack, J. A. Mumford, and T. E. Nichols, *Handbook of Functional MRI Data Analysis*. Cambridge: Cambridge University Press, 2011. [Online]. Available: https://www.cambridge.org/core/books/handbook-of-functional-mri-data-analysis/8EDF966C65811FCCC306F7C916228529

[21] B. Jin, A. Strasburger, S. J. Laken, F. A. Kozel, K. A. Johnson, M. S. George, and X. Lu, "Feature selection for fMRI-based deception detection," *BMC Bioinformatics*, vol. 10, no. 9, p. S15, Sep. 2009. [Online]. Available: https://doi.org/10.1186/1471-2105-10-S9-S15

[22] H. Fröhlich, O. Chapelle, and B. Schölkopf, "Feature selection for support vector machines by means of genetic algorithms," in *Proceeding ICTAI*, vol. 3. Citeseer, 2002.

[23] M. N. I. Qureshi, J. Oh, B. Min, H. J. Jo, and B. Lee, "Multi-modal, Multi-measure, and Multi-class Discrimination of ADHD with Hierarchical Feature Extraction and Extreme Learning Machine Using Structural and Functional Brain MRI," *Frontiers in Human Neuroscience*, vol. 11, 2017. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnhum.2017.00157

[24] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," vol. 70, no. 1, pp. 489–501. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231206000385

[25] M. Termenon, M. Graña, A. Barrós-Loscertales, and C. Ávila, "Extreme learning machines for feature selection and classification of cocaine dependent patients on structural MRI data," vol. 38, no. 3, pp. 375–387. [Online]. Available: https://doi.org/10.1007/s11063-013-9277-x

[26] W. Zhang, H. Shen, Z. Ji, G. Meng, and B. Wang, "Identification of mild cognitive impairment using extreme learning machines model," in *Intelligent Computing Theories and Methodologies*, D.-S. Huang, K.-H. Jo, and A. Hussain, Eds. Springer International Publishing, pp. 589–600.

[27] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks." [Online]. Available: http://arxiv.org/abs/1406.2661

[28] P. Zhuang, A. G. Schwing, and O. Koyejo, "FMRI data augmentation via synthesis: 16th IEEE international symposium on biomedical imaging, ISBI 2019," pp. 1783–1787, publisher: IEEE Computer Society. [Online]. Available: http://www.scopus.com/inward/record.url?scp=85073901078&partnerID=8YFLogxK

[29] D. Li, C. Du, S. Wang, H. Wang, and H. He, "Multi-subject data augmentation for target subject semantic decoding with deep multi-view adversarial learning," vol. 547, pp. 1025–1044. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025520309221

[30] SPM - statistical parametric mapping. [Online]. Available: https://www.fil.ion.ucl.ac.uk/spm/

[31] K. A. Norman, S. M. Polyn, G. J. Detre, and J. V. Haxby, "Beyond mind-reading: multi-voxel pattern analysis of fMRI data," *Trends in Cognitive Sciences*, vol. 10, no. 9, pp. 424–430, Sep. 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364661306001847

[32] A. R. McIntosh and N. J. Lobaugh, "Partial least squares analysis of neuroimaging data: applications and advances," *Neuroimage*, vol. 23, pp. S250–S263, 2004.

[33] W. S. Noble, "What is a support vector machine?" *Nature Biotechnology*, vol. 24, no. 12, pp. 1565–1567, Dec. 2006, number: 12 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/nbt1206-1565

[34] F. De Martino, G. Valente, N. Staeren, J. Ashburner, R. Goebel, and E. Formisano, "Combining multivariate voxel selection and support vector machines for mapping and classification of fmri spatial patterns," *Neuroimage*, vol. 43, no. 1, pp. 44–58, 2008.

[35] L. Mescheder, A. Geiger, and S. Nowozin, "Which training methods for GANs do actually converge?" [Online]. Available: https://arxiv.org/abs/1801.04406v2

[36] T. Eslami and F. Saeed, "Auto-asd-network: a technique based on deep learning and support vector machines for diagnosing autism spectrum disorder using fmri data," in *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, 2019, pp. 646–651.

Appendix A

Group results for all experiments

A.1 Map types

Experiment	Modality	Region	Mean accuracy left hemisphere	Mean accuracy right hemisphere
t-maps = **BASELINE**	Vision	V5	0.32	0.337
	Audition	V5	0.314	0.308
		PT	0.451	0.36
beta-maps	Vision	V5	0.29	0.332
	Audition	V5	0.327	0.31
		PT	0.445	0.375

A.2 Varying radius of ROIs and voxel size

A.2.1 2x2x2mms voxels

Experiment	Modality	Region	Mean accuracy left hemisphere	Mean accuracy right hemisphere
radius = 5mm	Vision	V5	0.289	0.326
	Audition	V5	0.279	0.275
		PT	0.396	0.365
radius = 6mm	Vision	V5	0.305	0.34
	Audition	V5	0.289	0.287
		PT	0.412	0.389
radius = 7mm	Vision	V5	0.304	0.345
	Audition	V5	0.307	0.292
		PT	0.415	0.395
radius = 8mm	Vision	V5	0.309	0.335
	Audition	V5	0.323	0.295
		PT	0.429	0.402
radius = 9mm	Vision	V5	0.312	0.342
	Audition	V5	0.319	0.312
		PT	0.424	0.419
radius = 10mm	Vision	V5	0.336	0.331
	Audition	V5	0.318	0.316
		PT	0.444	0.417

A.2.2 3x3x3mms voxels

Experiment	Modality	Region	Mean accuracy left hemisphere	Mean accuracy right hemisphere
radius = 5mm	Vision	V5	0.298	0.318
	Audition	V5	0.287	0.301
		PT	0.391	0.325
radius = 6mm	Vision	V5	0.306	0.322
	Audition	V5	0.31	0.303
		PT	0.386	0.314
radius = 7mm	Vision	V5	0.293	0.333
	Audition	V5	0.323	0.31
		PT	0.414	0.365
radius = 8mm	Vision	V5	0.291	0.327
	Audition	V5	0.337	0.307
		PT	0.417	0.371
radius = 9mm	Vision	V5	0.319	0.334
	Audition	V5	0.316	0.297
		PT	0.441	0.369
radius = 10mm	Vision	V5	0.334	0.345
	Audition	V5	0.317	0.311
		PT	0.443	0.381

A.3 Different ML models

Experiment	Modality	Region	Mean accuracy left hemisphere	Mean accuracy right hemisphere
SVM	Vision	V5	0.32	0.337
	Audition	V5	0.314	0.308
		PT	0.451	0.36
LR	Vision	V5	0.328	0.34
	Audition	V5	0.316	0.307
		PT	0.454	0.377
KNN	Vision	V5	0.269	0.265
	Audition	V5	0.32	0.308
		PT	0.347	0.312
Perceptron	Vision	V5	0.322	0.3
	Audition	V5	0.293	0.304
		PT	0.386	0.362
ELM	Vision	V5	0.306	0.297
	Audition	V5	0.297	0.314
		PT	0.413	0.365

A.4 Feature selection

A.4.1 Filter methods

filter criteria : *f_classif*

Experiment	Modality	Region	Mean accuracy left hemisphere	Mean accuracy right hemisphere
SVM 100% of voxels	Vision	V5	0.322	0.34
	Audition	V5	0.315	0.308
		PT	0.45	0.357
SVM 80% of voxels	Vision	V5	0.337	0.339
	Audition	V5	0.337	0.304
		PT	0.455	0.371
SVM 60% of voxels	Vision	V5	0.316	0.345
	Audition	V5	0.312	0.312
		PT	0.446	0.374
SVM 40% of voxels	Vision	V5	0.343	0.325
	Audition	V5	0.304	0.303
		PT	0.418	0.375
SVM 20% of voxels	Vision	V5	0.293	0.322
	Audition	V5	0.302	0.303
		PT	0.403	0.379

Experiment	Modality	Region	Mean accuracy left hemisphere	Mean accuracy right hemisphere
LR 100% of voxels	Vision	V5	0.328	0.347
	Audition	V5	0.315	0.306
		PT	0.457	0.376
LR 80% of voxels	Vision	V5	0.318	0.34
	Audition	V5	0.316	0.308
		PT	0.446	0.377
LR 60% of voxels	Vision	V5	0.327	0.341
	Audition	V5	0.316	0.319
		PT	0.437	0.367
LR 40% of voxels	Vision	V5	0.342	0.342
	Audition	V5	0.313	0.298
		PT	0.409	0.387
LR 20% of voxels	Vision	V5	0.309	0.321
	Audition	V5	0.324	0.302
		PT	0.4	0.377

filter criteria : *mutual_info_classif*

Experiment	Modality	Region	Mean accuracy left hemisphere	Mean accuracy right hemisphere
SVM 100% of voxels	Vision	V5	0.322	0.34
	Audition	V5	0.315	0.308
		PT	0.45	0.357
SVM 80% of voxels	Vision	V5	0.31	0.338
	Audition	V5	0.32	0.294
		PT	0.427	0.365
SVM 60% of voxels	Vision	V5	0.304	0.332
	Audition	V5	0.318	0.297
		PT	0.42	0.376
SVM 40% of voxels	Vision	V5	0.303	0.338
	Audition	V5	0.323	0.311
		PT	0.41	0.381
SVM 20% of voxels	Vision	V5	0.285	0.333
	Audition	V5	0.315	0.286
		PT	0.412	0.387

Experiment	Modality	Region	Mean accuracy left hemisphere	Mean accuracy right hemisphere
LR 100% of voxels	Vision	V5	0.328	0.347
	Audition	V5	0.315	0.306
		PT	0.457	0.376
LR 80% of voxels	Vision	V5	0.3	0.333
	Audition	V5	0.317	0.293
		PT	0.438	0.38
LR 60% of voxels	Vision	V5	0.301	0.34
	Audition	V5	0.321	0.298
		PT	0.42	0.379
LR 40% of voxels	Vision	V5	0.307	0.346
	Audition	V5	0.312	0.314
		PT	0.415	0.386
LR 20% of voxels	Vision	V5	0.292	0.314
	Audition	V5	0.311	0.296
		PT	0.404	0.372

A.4.2 Wrapper method (RFE)

Experiment	Modality	Region	Mean accuracy left hemisphere	Mean accuracy right hemisphere
SVM 100% of voxels	Vision	V5	0.32	0.337
	Audition	V5	0.314	0.308
		PT	0.451	0.36
SVM 80% of voxels	Vision	V5	0.323	0.327
	Audition	V5	0.314	0.3
		PT	0.439	0.357
SVM 60% of voxels	Vision	V5	0.332	0.331
	Audition	V5	0.318	0.305
		PT	0.443	0.37
SVM 40% of voxels	Vision	V5	0.31	0.325
	Audition	V5	0.319	0.306
		PT	0.423	0.372
SVM 20% of voxels	Vision	V5	0.297	0.332
	Audition	V5	0.322	0.309
		PT	0.413	0.361

Experiment	Modality	Region	Mean accuracy left hemisphere	Mean accuracy right hemisphere
LR 100% of voxels	Vision	V5	0.328	0.34
	Audition	V5	0.316	0.307
		PT	0.454	0.377
LR 80% of voxels	Vision	V5	0.317	0.339
	Audition	V5	0.311	0.312
		PT	0.443	0.366
LR 60% of voxels	Vision	V5	0.312	0.322
	Audition	V5	0.305	0.315
		PT	0.433	0.366
LR 40% of voxels	Vision	V5	0.319	0.322
	Audition	V5	0.312	0.328
		PT	0.415	0.359
LR 20% of voxels	Vision	V5	0.312	0.329
	Audition	V5	0.321	0.308
		PT	0.41	0.366

A.5 Data augmentation (SMOTE)

A.5.1 k = 2

Experiment	Modality	Region	Mean accuracy left hemisphere	Mean accuracy right hemisphere
SVM no smote	Vision	V5	0.322	0.34
	Audition	V5	0.315	0.308
		PT	0.45	0.357
SVM s = 15	Vision	V5	0.334	0.346
	Audition	V5	0.321	0.308
		PT	0.443	0.383
SVM s = 20	Vision	V5	0.331	0.345
	Audition	V5	0.325	0.309
		PT	0.443	0.386
SVM s = 30	Vision	V5	0.329	0.342
	Audition	V5	0.315	0.31
		PT	0.442	0.384
SVM s = 40	Vision	V5	0.331	0.35
	Audition	V5	0.325	0.303
		PT	0.443	0.388
SVM s = 50	Vision	V5	0.327	0.346
	Audition	V5	0.323	0.302
		PT	0.437	0.385

Experiment	Modality	Region	Mean accuracy left hemisphere	Mean accuracy right hemisphere
LR no smote	Vision	V5	0.328	0.347
	Audition	V5	0.315	0.306
		PT	0.457	0.376
LR s = 15	Vision	V5	0.326	0.34
	Audition	V5	0.32	0.316
		PT	0.437	0.395
LR s = 20	Vision	V5	0.331	0.353
	Audition	V5	0.324	0.313
		PT	0.437	0.385
LR s = 30	Vision	V5	0.323	0.348
	Audition	V5	0.319	0.311
		PT	0.438	0.392
LR s = 40	Vision	V5	0.322	0.357
	Audition	V5	0.317	0.309
		PT	0.438	0.39
LR s = 50	Vision	V5	0.324	0.349
	Audition	V5	0.32	0.307
		PT	0.433	0.386

A.5.2 k = 3

Experiment	Modality	Region	Mean accuracy left hemisphere	Mean accuracy right hemisphere
SVM no smote	Vision	V5	0.322	0.34
	Audition	V5	0.315	0.308
		PT	0.45	0.357
SVM s = 15	Vision	V5	0.328	0.347
	Audition	V5	0.319	0.311
		PT	0.437	0.387
SVM s = 20	Vision	V5	0.329	0.343
	Audition	V5	0.318	0.306
		PT	0.438	0.379
SVM s = 30	Vision	V5	0.327	0.346
	Audition	V5	0.322	0.306
		PT	0.438	0.387
SVM s = 40	Vision	V5	0.33	0.346
	Audition	V5	0.323	0.312
		PT	0.446	0.384
SVM s = 50	Vision	V5	0.329	0.346
	Audition	V5	0.32	0.314
		PT	0.443	0.391

Experiment	Modality	Region	Mean accuracy left hemisphere	Mean accuracy right hemisphere
LR no smote	Vision	V5	0.328	0.347
	Audition	V5	0.315	0.306
		PT	0.457	0.376
LR s = 15	Vision	V5	0.325	0.354
	Audition	V5	0.315	0.312
		PT	0.441	0.395
LR s = 20	Vision	V5	0.325	0.345
	Audition	V5	0.321	0.307
		PT	0.446	0.383
LR s = 30	Vision	V5	0.33	0.36
	Audition	V5	0.316	0.306
		PT	0.434	0.38
LR s = 40	Vision	V5	0.323	0.346
	Audition	V5	0.314	0.307
		PT	0.438	0.377
LR s = 50	Vision	V5	0.33	0.346
	Audition	V5	0.314	0.309
		PT	0.441	0.379

A.5.3 k = 5

Experiment	Modality	Region	Mean accuracy left hemisphere	Mean accuracy right hemisphere
SVM no smote	Vision	V5	0.322	0.34
	Audition	V5	0.315	0.308
		PT	0.45	0.357
SVM s = 15	Vision	V5	0.333	0.35
	Audition	V5	0.319	0.311
		PT	0.442	0.378
SVM s = 20	Vision	V5	0.328	0.346
	Audition	V5	0.32	0.31
		PT	0.444	0.374
SVM s = 30	Vision	V5	0.323	0.341
	Audition	V5	0.324	0.308
		PT	0.438	0.379
SVM s = 40	Vision	V5	0.328	0.35
	Audition	V5	0.329	0.306
		PT	0.442	0.376
SVM s = 50	Vision	V5	0.324	0.346
	Audition	V5	0.325	0.309
		PT	0.437	0.383

Experiment	Modality	Region	Mean accuracy left hemisphere	Mean accuracy right hemisphere
LR no smote	Vision	V5	0.328	0.347
	Audition	V5	0.315	0.306
		PT	0.457	0.376
LR s = 15	Vision	V5	0.326	0.34
	Audition	V5	0.318	0.311
		PT	0.438	0.391
LR s = 20	Vision	V5	0.325	0.348
	Audition	V5	0.315	0.313
		PT	0.44	0.393
LR s = 30	Vision	V5	0.325	0.349
	Audition	V5	0.315	0.306
		PT	0.433	0.39
LR s = 40	Vision	V5	0.322	0.345
	Audition	V5	0.32	0.308
		PT	0.438	0.384
LR s = 50	Vision	V5	0.327	0.344
	Audition	V5	0.315	0.31
		PT	0.43	0.382

A.6 Average of 2 test sets

Experiment	Modality	Region	Mean accuracy left hemisphere	Mean accuracy right hemisphere
SVM baseline	Vision	V5	0.322	0.34
	Audition	V5	0.315	0.308
		PT	0.45	0.357
SVM 2 tests	Vision	V5	0.333	0.365
	Audition	V5	0.342	0.316
		PT	0.478	0.408
LR	Vision	V5	0.328	0.347
	Audition	V5	0.315	0.306
		PT	0.457	0.376
LR 2 tests	Vision	V5	0.338	0.378
	Audition	V5	0.332	0.314
		PT	0.479	0.411

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**