Comparing EEG/ERP-like and fMRI-like Techniques for Reading Machine Thoughts

Fabio Massimo Zanzotto and Danilo Croce

University of Rome Tor Vergata
00133 Roma (Italy)
{zanzotto,croce}@info.uniroma2.it

Abstract. fMRI and ERP/EEG are two different sources for scanning the brain for building mind state decoders. fMRI produces accurate images but it is expensive and cumbersome. ERP/EEG is cheaper and potentially wearable but it gives more coarse-grain data. Recently the metaphor between machines and brains has been introduced in the context of mind state decoders: the "*readers for machines' thoughts*". This metaphor gives the possibility for comparing mind state decoder methods in a more controlled setting.
In this paper, we compare the fMRI and ERP/EEG in the context of building "*readers for machines' thoughts*". We want assess if the cheaper ERP/EEG can be competitive with fMRI models for building decoders for mind states. Experiments show that accuracy of "*readers*" based on ERP/EEG-like data are considerably lower than the one of those based on fMRI-like images.

1 Introduction

Mining brain related data for decoding mind states is a fascinating and active area of research. Two major and different sources of brain related data are considered in these studies: functional magnetic resonance imaging (fMRI) and electroencephalography for detecting event related potentials (ERP/EEG).

Brain images obtained with fMRI have been largely used in combination with machine learning for building decoders for mind states. fMRI and support vector machines [1] have been used to build decoders that determine if participants are looking to: (1) shoes or bottles [2]; (2) human faces or objects [3]. In [4], fMRI based mind state decoder have been successfully learnt and applied for determining if subjects were (1) looking at pictures or sentences, (2) reading an ambiguous or non-ambiguous sentence, and (3) looking at words describing food, people, buildings, etc. fMRI related information have been also used for building classifiers that can predict the perceived complexity of a simplified Sudoku schema [5]. Finally, corpus linguistics has been used to induce novel brain activation images for unobserved semantic categories from observed brain activation images [6].

Also ERP/EEG brain data have been used for the same purposes. ERP/EEG brain data are used to explore subjects performing super-ordinate categorization

between natural and artifact objects [7, 8]. Similarly to [6], corpus linguistic has been used to determine and classify ERP/EEG activation data for unseen semantic categories [9].

fMRI and ERP/EEG are then two alternative models for investigating and collecting data for the activities of human brains. Both these models have been used to induce mind state decoders from data. These are different models with different strengths and limits. In general, fMRI is used to extract detailed brain activation images representing snapshots of activation states [4, 6]. In some cases, time related series are extracted for specific voxels of the brain [2, 3, 5] giving the possibility of including state changes in the investigation. Yet, fMRI is still an expensive and cumbersome technique. ERP/EEG is instead relatively cheaper and possibly wearable but it gives more coarse-grained data. These data represent the electrical activation of specific spots in the brain (e.g., less than 100 spots in the 64-channel the Geodesic Sensor Net [10]). This technique naturally generates time series for the electric activation of specific voxels.

In this paper, we want to compare fMRI and ERP/EEG in order to assess if the cheaper ERP/EEG can be competitive with fMRI for building decoders for mind states. Using the metaphor and the method introduced in [11], we will compare these two models in a more controlled setting: we compare the two models in the context of building a "*reader for machines' thoughts*". We then propose an fMRI-like method and an ERP/EEG-like method to extract data from "*machine brains*" performing three different "*cognitive tasks*". We then compare the resultant "*readers for machines' thoughts*" based on fMRI-like and an ERP/EEG-like data. These more controlled setting gives a better possibility to compare these two methods. Experiments show that there is a consistent drop in accuracy of the detection of the "*cognitive tasks*" when using "*readers for machines' thoughts*" based on ERP/EEG-like data.

The rest of the paper is organized as follows. First, we describe the aim of the experiments in Section 2. We introduce the investigation method in Section 3. Then, we report the results of the experiments in Section 4. Finally, we draw some conclusions and we plan the future work in Section 5.

2 Aim of the Experiments

The aim of the experiments is to compare fMRI and EEG/ERP for investigating human brains using the metaphor of "*reading machines' thoughts*". fMRI and EEG/ERP methods are extremely different in investigating human brain. Generally, the first takes snapshots of brain activations producing brain images. The second, instead, records the evolution of the electrical activity in specific spots of the brain. Thus, fMRI methods have high resolution but no time evolution while EEG/ERP methods report time evolution of brain activation in low resolution, i.e., the set of electric sensors is small. We want to compare these two alternative methods. EEG/ERP methods are less expensive and more practical than fMRI method. Then, we want to investigate whether the resolution reduction in EEG/ERP is balanced by the addition of the time dimension.

3 Method

For comparing the EEG/ERP and fMRI investigation methods within the metaphor of "*reading machines' thoughts*", we need to build two classifiers of machine activation images that detect the performed "*cognitive task*": the first is based on fMRI-like images and the second is based on EEG/ERP-like data.

In the rest of this section, first we describe the "*subjects*" of our study (i.e., the machines) (Sec. 3.1). Second, we introduce the three different "*cognitive tasks*" that the "*readers of machines' thoughts*" have to recognize (Sec. 3.2): this will produce a corpus of different observations that can be used for later experiments. We describe how we can extract fMRI-like (Sec. 3.3) and EEG/ERP-like (Sec. 3.4) activation images for the three different machines and we introduce the features that will be extracted from these images (Sec. 3.5). Finally, we shortly describe the machine learning methods used to build the classifiers implementing the "*readers of machines' thoughts*" (Sec. 3.6).

3.1 Machines

We selected three different kinds of machines for our comparative experiments. These machines are indeed different "*individuals*" as representing different levels of abstraction. We selected a physical machine, an interpreted machine, and a virtual machine. The second and the third run on the first machine. We hereafter describe these machines as if these were the "*subjects*" of our experiments.

Physical machine The physical machine we consider is a Von Neumann machine organized with a central processing unit and a memory. This machine runs a Linux operating system. "*Cognitive tasks*" are represented as programs of this machine working on particular input data. For our experiments, we use programs originally written in C language and compiled according to the hosting Linux machine.

We observe the *cognitive task* by looking at the process executing the program with a predefined input. Observing this process, we can see the compiled program, the ancillary libraries, the process control block, the heap area, the stack area, and the data area. This area does not include any other additional information.

Virtual machine A virtual machine is an abstract machine that runs over a physical machine. This machine is a software program that runs on a real hardware platform with an operating system. Yet, this machine has all the features of a real machine. First, it runs programs written in its particular language that is generally a bytecode. Programs written in upperlevel languages are then compiled (i.e., translated) in the particular machine language. As for any physical machine, the stream of bytecodes is a sequence of instructions: each instruction has one operation code and a possibly null sequence of operands. The operation code tells the actions that the machine has to undertake on possible operands.

A virtual machine has all the parts of a real machine: a computing part and a memory part. The memory part is organized according to the specific virtual machine.

For our experiments, we use a java virtual machine (JVM). This machine run java compiled programs. The memory of this machine is basically divided in four parts: the registers, the stack, the garbage-collected heap, and the method area. All these parts are virtual but implemented somewhere on top of the real machine. We will observe java virtual machines running only one compiled program. This is the observed "*cognitive activity*".

We monitor this virtual machine observing the memory of the process that hosts the java virtual machine running a compiled program. This memory is larger and more complex than the one of the process representing the "*cognitive activity*" of the physical machine (cf. Sec. 3.1). It contains: java virtual machine compiled in the language of the hosting physical machine, the java compiled code of the program, the state of the JVM registers, and the state of the memory of the java virtual machine (the stack, the heap, and the method area). All this information is not so clearly related to the performed "*cognitive process*", i.e., the algorithm.

Interpreted machine An interpreter is a machine that executes programs written in a high-level language. Likewise a virtual machine, this interpreted machine runs on a physical machine. The major difference with respect to virtual machines is that the program is not compiled in advance in a machine language but it is used in its original form. At running time, the interpreter translates high-level instructions into an intermediate form, which it then executes.

For our experiments, we use a PHP interpreter. This interpreter runs programs written in php. These latter are represented in the memory of the process. There is not a clear distinction between different types of memory.

As for the above virtual machine, we monitor the activity of the interpreter performing a "*cognitive task*" by observing its process while executing a specific program. We then observe: the interpreter compiled in the language of the hosting physical machine, the PHP program, and the state of the working memory.

3.2 Materials

We designed three different "*cognitive tasks*", i.e., three programs, for the three different machines. We selected three different classes of programs working on different data structures: vectors of numbers, strings, and binary trees. The three different algorithms are:

- a sorting algorithm performed by means of the quick sort algorithm
- a edit distance algorithm performed by means of the edit distance
- a tree visit algorithm performed as a dept-first search

We assume that these three are the classes of cognitive tasks we want to recognize. We want to perform the classification of the activation images at this level.

Different instances of these classes of algorithms are given by different associated data. An instance of the class of sorting algorithm is a sorting algorithm program executed with a specific input vector.

For each of the three different classes, we randomly prepared 40 input data: 40 vectors of 10^5 integers, 40 pairs of strings, and 40 binary trees. Each pair $(algorithm, data)$ is a different "*cognitive task*". For the three different machines (the physical machine, the virtual machine, and the interpreted machine), we then have 120 different "*cognitive task*" grouped in three classes. We run the 120 different "*cognitive tasks*" on the three machines obtaining our samples. We then captured the activation images of the different "*cognitive tasks*" performed on the three different machines.

3.3 fMRI-like activation capturing

To simulate the fMRI activation capturing, we snapshot machines performing the cognitive task at a given stage of execution. This is a more controlled setting with respect to what happens for real brain imaging by fMRI. We exactly know what we are taking as snapshots and the stage of the program that is executed.

Given a cognitive activity (i.e., a pair $(algorithm, data)$), the procedure for extracting images from this activity is then the following:

- running the process p representing the activity, i.e., the program and the related input data
- stopping the process p at given states or at given time intervals τ
- dumping the memory associated to the process $M(p)$
- given a fixed height image and the memory dump, read incrementally bytes of the memory dump, and fill the associated RGB pixel with the read values obtaining the image $I(p)$

This simple procedure can produce more images for each process related to a cognitive activity.

The process memory is transformed in an image using the following procedure. Let $M(p)$ be the memory dump of the process p. The memory dump is a sequence of bytes, i.e., $M(p) = [b_0, ..., b_m]$. Using this sequence of bytes we can produce an image $I(p)$ in Red-Green-Blue (RGB) coding. The image $I(p)$ is a bi-dimensional array of pixels $p_{i,j}$. Each pixel is three contiguous bytes of the memory image. Given the height h of the image, each RGB pixel has the following RGB values:

$$p_{i,j} = [b_{3(i+h\cdot j)}\ b_{3(i+h\cdot j)+1}\ b_{3(i+h\cdot j)+2}] \tag{1}$$

where the first byte $b_{3(i+h\cdot j)}$ is used for the red component, the second $b_{3(i+h\cdot j)+1}$ for the green one, and the third $b_{3(i+h\cdot j)+2}$ for the blue one.

The activation images we obtain are reported in Figure 1. The original images have a fixed height of 768 pixels. For displaying purposes, also the width of the images has been resized to a fixed value. Yet, this does not represent the reality as the virtual machine's and the interpreter's pictures are much bigger than the

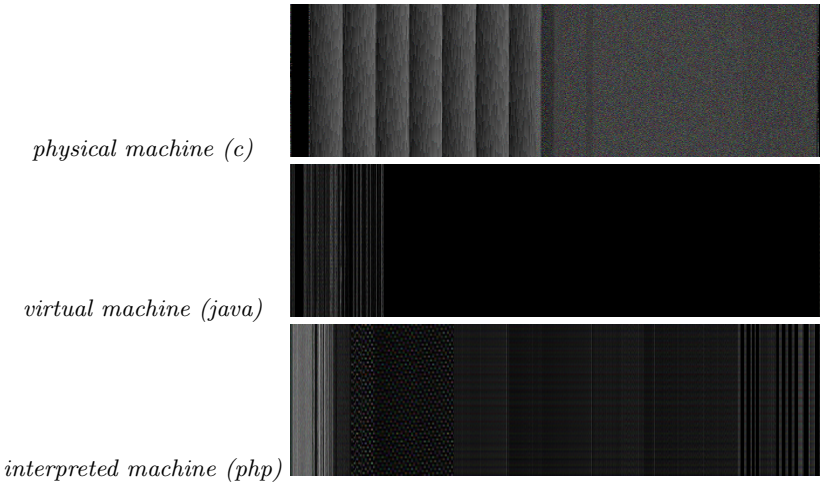

Fig. 1. Sorting process in the three different machines (fMRI-like snapshots)

physical machine's ones. These images represents how the memory of the three different machines react to the "*cognitive task*" of sorting a vector with the quick sort algorithm. These snapshots are taken in the middle of the computation. Looking at the *data area* of the "*cognitive process*" performed by the *physical machine*, we can observe that half of the vector forms a recurrent pattern while the other half has a completely random pattern.

The images in Figure 1 clearly represents the commonalities and the differences of the three machines. A common fact is that the sizes of *program code area* and the *data area* are extremely different. The *data area* is bigger than the *code area*. The differences are instead clear. The *virtual machine* allocates a big chunk of memory in advance even if this memory is not used. This is the black part of the related image. This behavior is explained as the virtual machine is completely represented in memory at the beginning of the process. Its virtual memory is completely allocated. The second difference is that the physical machine is the only one that has only the code area. In this case, the process representing the *cognitive activity* does not include a representation of the machine.

3.4 EEG/ERP-like activation capturing

We describe here a way to simulate EEG activation capturing. The major difference between fMRI and EEG/ERP methods for brain investigation is that the first produce a snapshot of the activation state of the brain while the second represent the electrical activation of the brain with respect to the time. Yet, fMRI images are more accurate and detailed than EEG/ERP activation plots. The resolution is extremely different: EEG/ERP tools observe a fixed number

of spots of the brain whereas fMRI captures virtually a picture of the activation of the brain.

The simulation for EEG/ERP that we propose here puts together the qualities of fMRI (i.e., high resolution) and EEG/ERP (i.e., capturing a time span). The idea is simple. We can build activation images by concatenating a sequence of snapshots taken for each "*cognitive activity*". This is possible as we can easily capture several activation images for a single cognitive activity. Resulting images represent the evolution of the zones of the memory with respect to the time. Even if time-related as EEG/ERP, these images have finer resolution with respect to this method. Common EEG/ERP tools capture around 100 channels of electric signals, e.g. the 64-channel Geodesic Sensor Net [10]. If we imagine to build images from the electric activation of these 64 channels, we can produce images with very scarce resolution, e.g., 1024×64. As each channel can be used to build a column of the image, the observed time can be partitioned in 1024 slots, and the color of the pixel is given by the mean electric activation of the channel in the specific time slot. Then, real EEG/ERP tools can produce some sort of "*activation images*". Simulated EEG/ERP images that we propose for "*machine thoughts' readers*" are valuable alternative to possible images that can be produced from real EEG/ERP investigation tools. Then, we can use these images to compare performance of "*machine thoughts' readers*" based on snapshots and on time series.

The process for capturing the EEG/ERP-like activation images is simple and it is based on the previous method. The procedure is the following:

- run the process p representing the activity, i.e., the program and the related input data
- stop the process p at each given time interval τ obtaining n memory dumps $M_0(p), \ldots, M_n(p)$
- for each memory dump $M_i(p)$, produce the activation image $I_i(p)$
- produce an image $\widehat{TI}(p)$ that is the concatenation of the images $I_0(p), \ldots, I_n(p)$
- resize the image $\widehat{TI}(p)$ to the related fixed size image $TI(p)$

Figure 2 represents these EEG-like images captured for the three different machines performing the sorting "*cognitive process*" based on the quick sort. The time series evolve from the top to the bottom of each image. For each machine, the first snapshot is taken before loading the target vector. For the physical machine, the space for the vector is allocated at the beginning of the process. We can then clearly distinguish the *code/machine area* and the *data area*.

The behavior for the three machines is extremely different and, then, the resulting EEG/ERG-like activation images appear to be dissimilar. The memory occupation of the physical machine does not grow during the time. This does not happen for the other two machines where the space occupied by the vector grows as the computation goes forward. In the physical machine, the progress of the sorting of the vector is clearly represented by the image. Visual patterns representing ordered chunks of the vector grow during the computation. The

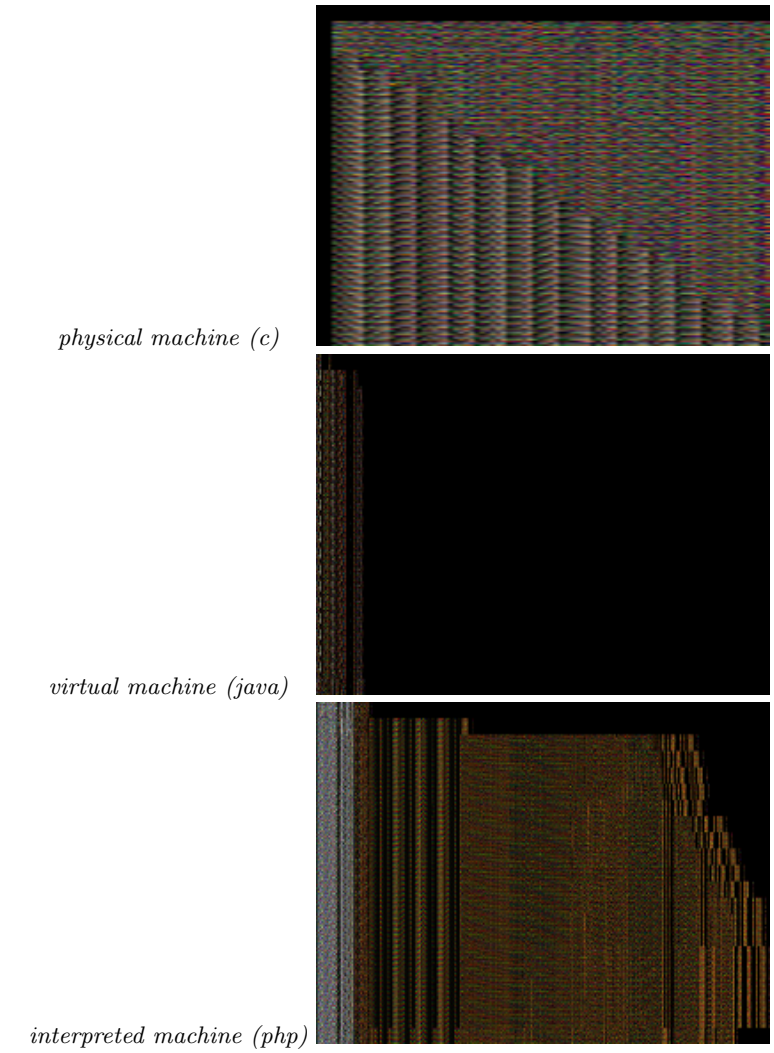

Fig. 2. Sorting process in the three different machines (EEG/ERP-like time series)

data space of the resulting image is then organized as two triangles: the upper unsorted triangle and the lower sorted triangle. In the virtual machine, the major part of the picture is represented by black pixel as it represents empty memory. In the interpreted machine instead, we can clearly see the growing occupation of the memory. This depends on the recursive nature of the quick sort algorithm that produces a stack of procedure calls where the vector is replicated.

In our experiments we considered time series of 21 snapshots for each pair $(algorithm, data)$. The final images have been resized to 1024×768.

3.5 Feature extraction from activation images

For using a classifier learner, we need to extract specific features from images. As in both cases, fMRI-like and EEG/ERP-like activation states, we have images, we can rely on the same method to extract features. This is a first step for comparing the fMRI-like and EEG/ERP-like activation states in a common ground.

We then used three major classes of features: chromatic, textures (OP - OGD) and transformation features (OGD), as described in [12]. Chromatic features express the color properties of the image. They determine, in particular, a n-dimensional vector representation of the 2D chromaticity histograms. Texture features emphasize the background properties and their composition. Texture feature extraction, in LTI-Lib [13], uses the steerability property of the oriented gaussian derivatives (OGD) to also generate rotation invariant feature vectors. Transformations are thus modeled by the OGD features. A more detailed discussion of the theoretical and methodological aspects behind each feature set are presented in [12].

3.6 Classifier learner for machine thought's readers

For finally building the classifiers of the "*cognitive task*" that the machine is performing, we used three alternative machine learning models. This is useful to see whether or not results are confirmed for any kind of classification method. We then used: a decision tree learner based [14], a simple Naive Bayes classifier (for more information see [15]), and, finally, an instance based learner (IBk) [16]. These machine learning methods have been used in the context of Weka [17].

The three models are different. Decision tree learners capture and select the best features for doing the classification. Naive bayes learners instead use a simple probabilistic model that considers all the features to be independent. The instance based learner defines a distance in the feature space, does not make any abstraction of the samples, and classifies new instances according to the distance of these new elements with respect to training samples. While the first model makes a sort of feature selection, the second and the third use all the features for taking the final decision.

4 Results

We are ready to compare fMRI-like and ERP/EEG-like "*machines' thought readers*", i.e., the classifiers. We then experimented with the set of feature vectors extracted from the images and the three different machine learning algorithms described in the previous section. We want to determine whether or not ERP/EEG-like methods can have similar performance with respect to fMRI-like methods. We performed two sets of experiments: a generic task and a one-machine-out task. In the generic task, we took all the images and we concealed the type of machine (i.e., physical, virtual, and interpreted). In the one-machine-out task, we learnt the classifiers on a kind of machine and we applied it to another kind. For example, we learnt the classifier on the physical machine and we applied it on the virtual machine. The second setting is extremely more complex than the first.

Table 1 reports the results for the *generic experiments*. The first row reports the accuracy of the classifiers when using 50% of the instance set as training and 50% of the instance set as testing. The accuracy of the classifier is the number of correctly classified instances with respect to the total number of decisions. The second row reports the accuracy on a 10-fold cross validation. The complete set of instances is used.

	fMRI-like			*ERP/EEG-like*		
	Decision Tree	*NaiveBayes*	*IBk*	*Decision Tree*	*NaiveBayes*	*IBk*
50% Train-50% Test	96.67	93.33	89.44	87.78	78.33	87.78
10-fold cross validation	96.67	92.78	90.83	88.06	79.44	88.61

Table 1. Accuracy of different learning algorithms for the fMRI-like and the ERP/EEG-like settings: Generic task

We can derive some facts. We can learn good classifiers for the generic task and decision tree learners derive the best classifiers. This implies that some feature is extremely more important than the others in taking the final decision. Yet, these experiments suggest that there is a consistent drop in performance by using ERP/EEG-like methods. This drop in performance is more important for decision tree learners and Naive Bayes methods (8-9% and 13-14%, respectively).

Table 2 reports the results of the one-machine-out experiments. This is a more complex task as we want to learn a classifier on some kinds of machines and apply this on other machines. We want to simulate the fact that brains may be similar but not exactly equal. In the third row (1 vs. 1 machine), we report the accuracy of classifiers learnt on a kind of machine and applied to another, e.g., the classifier is learnt on the physical machine and applied on the virtual machine. We have then 6 different pairs of experiments. We report here the average accuracy. The forth row instead reports the experiments where we

used two kinds of machines and we tested on the third, e.g., physical and virtual machines used as training and interpreted machine used as testing.

	fMRI-like			*ERP/EEG-like*		
	Decision Tree	*NaiveBayes*	*IBk*	*Decision Tree*	*NaiveBayes*	*IBk*
1 vs. 1 machine	46.81	33.33	35.83	29.31	33.33	35.42
2 vs. 1 machine	63.06	48.61	29.17	46.11	34.44	48.61

Table 2. Accuracy of different learning algorithms for the fMRI-like and the ERP/EEG-like settings: One-machine-out task

As the average accuracies show, this task is extremely more complex than the *generic task*. As the classifiers have to decide among three evenly distributed classes, the random baseline classifier obtains an accuracy of 33.33%. In two cases for the 1 vs. 1 setting for the Naive Bayes learning methods, the learnt classifiers cannot decide which cognitive task the machine are performing as these classifiers basically predict always one class. In one case for the 2 vs. 1 machine (IBk) and in one for the 1 vs. 1 machine (Decision Tree), we have a classifier that performs worse than a random classifier. We have two important observations. First, the 2 vs. 1 machine cases are generally better than 1 vs. 1 machine cases. Second, EEG/ERP methods are nearly always extremely worse than fMRI-like methods. There is only one case where this does not happen: the IBk classifiers in the 2 vs. 1 machine case. Yet, the accuracy obtained by the ERP/EEG-like IBk classifier is extremely lower than the best accuracy of the 2 vs. 1 machine obtained in the fMRI-like side, i.e., the fMRI-like decision tree classifier.

5 Discussion and conclusions

In this paper, we presented a comparison of two different methods for acquiring data from brain for building mind state decoders: fMRI and ERP/EEG. This comparison has been done using the metaphor between brains and machines. Experiments shows that there is a consistent drop in accuracy when using ERP/EEG-like models with respect to fMRI-like methods. These results are confirmed both in the simple case, i.e., the *generic task* and in the more complex case, i.e., the *one-machine-out task*. The drop in performance suggests that building mind state decoders using ERP/EEG will be extremely more complex than building these decoders using fMRI.

Acknowledgments

We would like to thank Marco Cesati for his precious advices on the organization of the memory in the Linux kernel and Paul Kantor for his insights on fMRI and EEG applied to the study of brain activation.

References

1. Vapnik, V.: The Nature of Statistical Learning Theory. Springer (1995)
2. Norman, K., Polyn, S., Detre, G., Haxby, J.: Beyond mind-reading: multi-voxel pattern analysis of fmri data. Trends in Cognitive Sciences **10**(9) (September 2006) 424–430
3. Haxby, J.V., Gobbini, M.I., Furey, M.L., Ishai, A., Schouten, J.L., Pietrini, P.: Distributed and overlapping representations of faces and objects in ventral temporal cortex. Science **293**(5539) (September 2001) 2425–2430
4. Mitchell, T.M., Hutchinson, R., Niculescu, R.S., Pereira, F., Wang, X., Just, M., Newman, S.: Learning to decode cognitive states from brain images. Mach. Learn. **57**(1-2) (2004) 145–175
5. Xiang, J., Chen, J., Zhou, H., Qin, Y., Li, K., Zhong, N.: Using svm to predict high-level cognition from fmri data: A case study of 4*4 sudoku solving. In Zhong, N., Li, K., Lu, S., Chen, L., eds.: Brain Informatics, International Conference, BI 2009, Beijing, China, October 22-24, 2009, Proceedings. Volume 5819 of Lecture Notes in Computer Science., Springer (2009) 171–181
6. Mitchell, T.M., Shinkareva, S.V., Carlson, A., Chang, K.M., Malave, V.L., Mason, R.A., Just, M.A.: Predicting human brain activity associated with the meanings of nouns. Science **320**(5880) (May 2008) 1191–1195
7. Kiefer, M.: Perceptual and semantic sources of category-specific effects: Event-related potentials during picture and word categorization. Memory & Cognition **29**(1) (2001) 100–116
8. Paz-Caballero, D., Cuetos, F., Dobarro, A.: Electrophysiological evidence for a natural/artifactual dissociation. Brain Research **1067**(1) (2006) 189 – 200
9. Murphy, B., Baroni, M., Poesio, M.: EEG responds to conceptual stimuli and corpus semantics. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, Singapore, Association for Computational Linguistics (August 2009) 619–627
10. Tucker, D.M.: Spatial sampling of head electrical fields: the geodesic sensor net. Electroencephalography and Clinical Neurophysiology **87**(3) (1993) 154 – 163
11. Zanzotto, F.M., Croce, D.: Reading what machines "*think*". In Zhong, N., Li, K., Lu, S., Chen, L., eds.: Brain Informatics, International Conference, BI 2009, Beijing, China, October 22-24, 2009, Proceedings. Volume 5819 of Lecture Notes in Computer Science., Springer (2009) 159–170
12. Alvarado, P., Doerfler, P., Wickel, J.: Axon^2 - a visual object recognition system for non-rigid objects. In: IASTED International Conference-Signal Processing, Pattern Recognition and Applications (SPPRA), Rhodes, IASTED (July 2001) 235–240
13. Alvarado, P., Doerfler, P.: LTI-Lib - A C++ Open Source Computer Vision Library. In Kraiss, K.F., ed.: Advanced Man-Machine Interaction. Fundamentals and Implementation. Springer Verlag, Dordrecht (2006) 399–421
14. Quinlan, J.: C4:5:programs for Machine Learning. Morgan Kaufmann, San Mateo (1993)
15. John, G.H., Langley, P.: Estimating continuous distributions in bayesian classifiers. (1995) 338–345
16. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Mach. Learn. **6**(1) (1991) 37–66
17. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, Chicago, IL (1999)