## Introduction

Decoding cognitive states (refs) from fMRI opens up exciting new applications in training and therapy. However, many hurdles remain to be cleared before real-world applications are possible: training or therapy must be performed within the scanner, decoding techniques must be able to handle less structured and controlled stimuli to accommodate training and therapy programs, we must be able to decode across different sessions to reduce total time in the scanner, and we must be able to decode variables that are inherent to the subject rather than the stimuli.

How is it important?

Virtual reality extinction therapy and real-time neuro-feedback have both been shown to be effective at treating a variety of mental illnesses including PTSD and ADHD. Combining virtual reality therapies with real-time neuro-feedback could be even more effective.

Why not EEG?

Neuro-feedback is predominantly performed with EEG which is significantly less expensive and has much better temporal resolution and allows for a greater range of mobility so it would be easier to integrate with existing virtual reality therapies. However, the spatial resolution is significantly worse and the inverse problem of determining where in the brain relevant signals are coming from is ill-posed at best. With fMRI we can compare and potentially augment our resulting models with the large body of existing and publicly available fMRI experimental data. It is also much easier to translate interesting findings into computational neuroscience models. However, EEG still has a place in the proposed work. After a particular cognitive state decoder has been tuned in the MRI machine, we can perform simultaneous fMRI and EEG recordings to learn how to translate our fMRI model into an EEG model. This technique has already been pioneered by [ref].

Hos is it relevant to ECE?

The problem of decoding cognitive states is largely

Subjects must remain still inside a cramped and noisy machine during their training or therapy session for the best decoding results. This severely limits the types of programs were fMRI decoding is realistically applicable. A potential solution to this limitation is to use virtual reality environments for the stimuli. While motion and other somatosensory inputs are still restricted, virtual environments provide a more immersive experience that should cause the subject's neural response to be closer to that of the real world. Encouragingly, some therapies already utilize virtual reality such as PTSD extinction therapy (refs).

Most cognitive neuroscience experiments are focused on explanatory power at the expense of classification accuracy and stimuli complexity. This makes sense when the objective is answering a basic research question. However, our goal is to apply fMRI cognitive state decoding during training and therapy where classification accuracy is extremely important and the stimuli will likely be highly complex. Therefore, we propose the use of more complex multivariate decoding techniques including feed-forward neural-networks and time-dependent filtering. These techniques are more difficult to interpret in terms of significant correlation between stimuli and response but provide significantly improved decoding accuracy.

Unfortunately, fMRI data is relatively ill posed for advanced machine learning methods. The dimensionality of whole brain data is large compared to the number of samples we can collect. Furthermore, the samples are correlated in both space and time. Many data-agnostic approaches have been used to attempt to solve this problem such as PCA (refs) and ICA (refs). However, we want to use the structure of fMRI data and more specifically the brain to intelligently reduce the dimensionality of the data as much as possible before using something like PCA or ICA. We project the functional data on to extracted cortical surfaces by averaging through the gray matter normal to the surface. Only looking at signal from gray matter reduces the dimensionality of the data by a factor of #. Then we spatially smooth only along the surface thereby avoiding averaging our data with unwanted noise in the white matter and CSF. Furthermore, this also prevents averaging across sulcal/gyral boundaries where we no longer expect the signal to be spatially correlated. Finally, we down-sample on the surface based on our smoothing parameter to reduce the dimensionality again. In this way, we leverage our knowledge of the structure of the brain to reduce the dimensionality of the time series data.

It’s important to be able to train the classifier in one session, and then use that trained classifier in subsequent sessions. This allows us to reduce the necessary scanning time per subject and also opens up the possibility of a decoding algorithm that gets better with each successive scan. Ideally, we would also like to be able to user decoders trained on a large number of different subjects. The main difficulty to solving both of these problems is accurately registering the volumes between different sessions and subjects. We have found that spherical registration (ref) with surface based smoothing and down-sampling yields the best results for both cross-session and cross-subject registration in terms of decoding accuracy. This cross-session registration procedure also allows us to collect and utilize more training examples for machine learning which also helps with the disparity in dimension and training examples and yields improved classification accuracy.

It is important that we decode a variable inherent to the subject rather than a reflection of the stimuli. Decoding the presentation of a particular visual or audio stimuli are uninteresting for our purposes; we already know what stimuli we are presenting to the subject. In the work presented here, we propose to decode the subject’s performance at a difficult task. This is something we don’t already know when we present the stimuli. Furthermore, predicting a subject’s performance has obvious applications for training and therapy such as modulating difficulty to keep predicted performance on a specific trajectory.

## Background

Recent research has shown that fMRI is capable of decoding some cognitive states (Mitchell et al., 2004) such as the cognitive states associated with the perception of various types of objects (Cabral, Silveira, & Figueiredo, 2012; Shinkareva et al., 2008), what a person is saying and who is saying it (Formisano, De Martino, Bonte, & Goebel, 2008), and telling the truth or lying (Fan, Shen, & Davatzikos, 2006). The ability to decode cognitive states during training and therapy exercises could be invaluable for improving their efficacy. Virtual environments (VEs) are the most practical way to perform such exercises within the confines of an MRI scanner, and a number of virtual training and therapy environments already exist (Gerardi, Rothbaum, Ressler, Heekin, & Rizzo, 2008; Gonçalves, Pedrozo, Coutinho, Figueira, & Ventura, 2012). However, these exercises are far from the controlled stimuli used in most fMRI experiments. During such natural tasks, we expect a variety of complex interactions between many regions of the brain. A goal of the work reported in this paper is further development of computational analysis techniques that improve decoding accuracy of cognitive states in such an environment. Rather than focusing on a specific set of cognitive states, we look to develop a general approach to decoding task-relevant states with high accuracy. Additionally, the visual richness, the motion of the objects and viewer, and the real-time interaction with the virtual environment bring experiences to the subject much closer to those which shaped the evolution of our brains. It seems plausible then that using virtual environments can reveal how the brain functions under more realistic circumstances. Therefore, another goal of the work reported in this paper is to further development of analysis techniques that improve the interpretability of complex decoding algorithms for use in hypothesis driven experiments.

Using virtual environments in fMRI experiments have been explored by various researchers over the past decade. Early examples can be found in the work of (Spiers & Maguire, 2007b) and the resulting publications (Valente, De Martino, Esposito, Goebel, & Formisano, 2011) from the PBAIC 2007 competition (see <http://www.lrdc.pitt.edu/ebc/2007/competition.html>). In the case of Spiers & Maguire, a commercial taxi driving game was used as the stimulus, thereby leveraging many millions of dollars in development expense, but at the same time severely limiting control of the stimulus by the researchers. The game play was recorded during scanning, and afterwards the subject reviewed the video with a researcher and explained what they were thinking and doing at each point to assist in labeling the data. For the PBAIC 2007 competition, researchers constructed an interactive VE using the Source game engine. Subjects were given a relatively complex task to search for fruits, toy weapons, and characters with piercings, while avoiding contact with a dog. Subjects received compensation after the scan based on the score they received in the game. Similar to the work of Spiers & Maguire, the game play was recorded and participants rated their subjective mood along several axes, including arousal and valence, while reviewing the video. More recently, researchers have begun using VEs in more traditional controlled experimental protocols utilizing specially designed and far simpler VEs (Marsh et al., 2010; Mueller et al., 2012; Op de Beeck, Vermaercke, Woolley, & Wenderoth, 2013; Schindler & Bartels, 2013). While they use Ves, the stimuli are still highly controlled and far from natural.

For our experiments, we used a virtual environment specially developed for us by a professional game and simulation designer using a state-of-the-art game engine. The visual quality of the environment and the motion of characters and camera were comparable to what is found in PTSD exposure therapy systems. The visual quality is similar to the stimuli used in the work of Spiers & Maguire as well as the PBAIC 2007 competition. However, in those stimuli the induced cognitive states are not well balanced. Due to their interactive nature, the subjects may spend significantly more time in one state than another. This complicates the training and, in particular, the evaluation of decoding algorithms. In our stimulus, we have balanced the induced states at the cost of interactivity to provide better accuracy estimates of different decoding methods for comparison. On the other hand, the stimulus is considerably more realistic – and the subject’s state less controlled – than what is found in the recent neuroscientific investigations involving VEs (Marsh et al., 2010; Mueller et al., 2012; Op de Beeck, Vermaercke, Woolley, & Wenderoth, 2013; Schindler & Bartels, 2013). It was important to measure the performance of different decoding methods in this environment to gauge their potential for use with training and therapy exercises.

Our goals were focused on exploring and improving methods of data analysis coupled with virtual environment stimulus design, rather than testing a specific neuroscience hypothesis. We aimed to extract the cognitive state of the subject associated with freely viewing a number of characters, rather than test the many possible perceptual mechanisms that encode this information in the human brain, such as object recognition, eye movements, or social group perception. Such decoding methods will be important for use of fMRI in clinical settings where it is useful to know the task-relevant cognitive state of the subject, but the neural mechanisms may not be well understood yet. We are, for example, interested in supporting work using virtual reality to treat PTSD due to combat, in which treatment exposes the subject to virtual stimuli that are highly suggestive of the physical situations that induced the trauma. Through carefully controlled use of VR, the patient is gradually desensitized over a period of weeks so that the likelihood of triggering of the trauma declines (Gerardi et al., 2008), as confirmed through fMRI measurements (Gonçalves et al., 2012). Our stimulus and experiments were developed with this in mind. In particular, we created a virtual town suggestive of the Middle East, and populated the town with a combination of U.S soldiers and foreign combatants (Figure 1).

Most neuroscience experiments analyze their data using hypothesis-based statistical techniques, such as the general linear model (GLM), Such methods can be very effective only when a distinct and testable hypothesis is available. However, the closer the stimuli get to realistic experiences, as offered using VE, the more difficult it becomes to isolate a tractable hypothesis. Moreover, it is likely that the more complex VE stimuli will evoke a more broadly distributed cortical response that includes both low-level sensory and higher-level associative regions. The treatment of each voxel independently by GLM cannot capture the structure of multi-voxel responses reflecting the coordinated activity these widely distributed brain regions. For all these reasons, we employ multi-voxel pattern analysis (MVPA) based on machine learning, an approach introduced in (Haxby et al., 2001).

We offer a new combination of methods to decode and analyze VE stimulus information from fMRI data. Most fMRI applications of machine learning have shown discrimination between distinct object categories (Haxby et al., 2001; Pereira, Mitchell, & Botvinick, 2009). More recently the relationship between multiple objects has been explored (Baeck, Wagemans, & Op de Beeck, 2013). Here we demonstrate that the cognitive state associated with object number rather than object classification can be decoded from fMRI data. Specifically, the cognitive state associated with viewing a number of animated characters, varying from 1—6 can be decoded in a dynamically changing virtual environment with accuracy from 58—93% (chance is 16.7%). Such high classification accuracy has important potential for real-time fMRI based therapies that adjust the stimulus in response to brain activity. In more recent work, we also show that it is possible to decode an internal cognitive state, in this case task performance with accuracy from 70–92% (chance is 33.3%). Being able to predict task performance is even more clearly relevant to the goal of modulating a stimulus. It could be used for example to keep the subject's performance at a specific target level with the goal of improving the efficacy of the training or therapy program.

To achieve this performance, we experimented with four machine learning algorithms. We were particularly interested in artificial neural networks (NN) and support vector machines (SVM). For completeness, we also tested a Gaussian naive Bayes classifier (GNB) (Duda & Hart, 1973), and k-nearest neighbor classifier (KNN). The SVM is the most commonly used machine-learning algorithm in MVPA analyses (Pereira et al., 2009). However, we found that NNs also produced very favorable results. Recent advances in NNs, such as deep learning (Hinton, Osindero, & Teh, 2006a) and convolutional networks, have been outperforming traditional SVMs in a variety of domains (Cireşan, Meier, Masci, & Schmidhuber, 2012). Before jumping to these advanced techniques, we wanted to explore the application of relatively simple feed-forward NNs on fMRI data, and we propose several methods for improving their classification performance.

MVPA classification performance can tell us to what degree the time-series data can be used to decode a target category, but we also want to know which voxels are encoding the desired stimulus information. The searchlight technique (Kriegeskorte, Goebel, & Bandettini, 2006) can be used in conjunction with any machine-learning algorithm to create a map, but it does not fully utilize the spatially distributed multivariate nature of the classifier. For SVMs, the absolute discriminative map (Formisano, De Martino, & Valente, 2008) has been used. However, the discriminative map is limited to SVM algorithms. We propose a new mapping method similar to the absolute discriminative map based on a technique called sensitivity analysis (Zurada, Malinowski, & Cloete, 1994). For an SVM, the method reduces to approximately the absolute discriminative map. However, the method is more general and has been adapted to the NN. The sensitivity analysis is used in several ways. First, we examine the use of the sensitivity results to train NNs on high-dimensional fMRI data with relatively few training examples. Second, we present a method for producing informative maps from trained networks based on sensitivity analysis. Finally, we develop a technique based on recursive feature elimination (Guyon, Weston, Barnhill, & Vapnik, 2002) to determine appropriate thresholds for these sensitivity maps. The recursive feature elimination technique also acts as a multivariate feature reduction technique that can improve decoding performance. A similar method was applied to SVMs in the work of (De Martino et al., 2008).

One of the long-term goals of this research is to create a tool that can assist in virtual reality training or therapy by providing measures of internal cognitive variables that are otherwise difficult to ascertain. In our previous study, we showed that it is possible to decode complex visual information from the pattern of BOLD activity in fMRI data while the subject views a virtual environment. This was an important first step to determine the feasibility of such a system, but visual information is clearly evident in the visual stimulus, and can be measured without fMRI.

Now, we are attempting to decode an internal cognitive variable: task difficulty. This is potentially a very useful measure as it would allow the difficulty of training or therapy tasks to be modulated to maximize subject attention even when the task lacks easily computed performance metrics. To accomplish this, we have designed a new stimulus, with substantially improved visual quality, upon which subjects perform a visual detection task of varying difficulty. Our previous study lacked an explicit task, and we believe this caused strong subject-to-subject and session-to-session variability. Therefore, we expect the addition of a task to help stabilize this variability in addition to giving us a far more relevant classification target.

Multi-variate pattern analysis (MVPA; Pereira, Mitchell, & Botvinick, 2009) is used to train classifiers that can estimate the difficulty of the task the subject is performing from the pattern of BOLD activation in the fMRI data, and the expected accuracy of the classifier on new data is estimated using statistical techniques. To gain some insight as to how the classifiers are accomplishing this task, we build sensitivity maps and project them onto the cortical surface for visualization. We have also begun experimenting with the application of intrinsic connectivity networks (ICNs; Laird et al., 2011) to help tease apart the contribution of various networks to the classifiers performance, and thereby provide a more semantic description of what’s going on.

## Completed Work

We started by establishing baseline decoding accuracy from 4 common machine learning algorithms in a virtual environment. We experimented with a variety of fMRI preprocessing techniques and feature selection methods, as well as established the necessary cross-validation restrictions to limit optimistic accuracy estimates due to temporal correlation in the signal. We then developed cognitive state space filtering techniques to improve the effective decoding accuracy by leveraging our knowledge of the temporal structure of the stimuli. We also developed a mapping technique to visualize what areas contributed to classification of cognitive state. This was useful for debugging as well as relating the results to existing neuroscience literature.

Five adult males, ages 24–57, with normal or corrected-to-normal vision, participated in the experiments. All subjects participated in two fMRI sessions and a third session to acquire a high-resolution structural anatomy. Informed consent was obtained from all subjects under a protocol approved by the University of Texas at Austin Institutional Review Board.

For designing our virtual environment, we used the Unreal Developer's Kit developed by Epic Games, Inc.. This development kit is available free of charge for non-commercial applications (<http://www.unrealengine.com/udk>) and uses the same rendering and game engine found in many current and popular video games. We created a virtual environment suggestive of a town in the Middle East (Figure 1).

|  |  |  |
| --- | --- | --- |
| (a) | (b) | (c) |

Figure 1. The stimulus in the experiment described in this paper employs a virtual environment and a blocked design where the view alternates between moving through the environment and viewing groups of animated characters. (a) An example frame from the stimulus where the camera is traveling through the virtual environment with no characters presented. (b) An example frame from the stimulus where five friendly characters are being presented. (c) An example frame from the stimulus where three hostile characters are being presented. Such stimuli allow studying how the brain responds in a more natural and complex environment.

The stimulus was rendered in real-time from the point of view of a camera moving at eye level through the town, providing a first-person-perspective experience. Virtual characters representing friendly forces and hostile combatants were situated at four locations in the town. The camera would travel steadily on a predefined path from one part of the town to another over a 15 second interval during which no characters were visible. When one of the pre-determined locations was reached, characters would appear for 15 seconds during which the camera panned back-and-forth slowly while keeping all the characters in the field-of-view. The characters engaged in simple repetitive animated movement sequences. The number of characters presented at each of these locations varied from one to six, but the number and position of the characters did not vary over each 15-second period (Figure 1(b) and (c)). The 30-second block design of the stimulus was used for feature selection and for improving classifier performance (see below). The MVPA classification was always applied only to the periods of time when characters were presented. While the stimulus followed a 30-second block design, the subject always had the context of being present in the virtual town. This was done in order to preserve as much realism as possible by avoiding a discontinuity in context induced by using a blank screen for contrast.

A scanning session of a single subject entailed five to six “runs”, where each run was six minutes in duration. During a single run, every possible number of characters, from one to six, was presented twice, for a total of 12 presentations of characters (15 sec. each). The order of the number of characters was generated by a random permutation of 1–6 applied twice, once for the first half of the run and once for the second half. Since we placed characters at only four different locations in the town, the camera made 3 loops through the town in order to provide 12 presentations of characters. Finally, the type of character, soldiers or insurgents, was the same through a given run, and this character type was alternated between runs.

The subjects were not given any specific instructions other than to view the scene. The subjects were not military personnel, and they were not instructed to perceive specific characters as friendly or hostile. No background context was provided to bias the way the subjects perceived the environment.

Our VE stimulus was designed to reduce low-level visual differences between the group-number categories. First, the imagery was never static. Once again, achieving as much realism as possible motivated this decision. It is well known that the visual experience of looking at still-imagery is quite different from looking at moving imagery (Spiers & Maguire, 2007a). Second, the distance and viewpoint of the character groups was randomized from presentation-to-presentation to reduce low-level contrast differences between character-number categories. In addition, the subject performed free viewing of the scene, so exploratory eye movements should average away any local contrast correlations with character number. However, it is possible that the total contrast across the scene is correlated with character count, as this would not be affected by eye movements. Therefore, we calculated total scene contrast for each frame of the stimulus using the root-mean-square (RMS) contrast measure (Kukkonen, Rovamo, Tiippana, & Näsänen, 1993). We then fit a GLM to the result with character counts 0–6 as explanatory variables and performed a t-test on each of these variables to determine if a significant correlation existed between total scene contrast and character count (Friston et al., 1994). To determine what effect any such correlations, whether significant or not, had on the performance of the classifiers, we trained an SVM on the total contrast data and compared the performance with the fMRI data. Performance was estimated using 2-fold cross-validation. Since the total contrast is identical for each run there are only 2 possible folds that are unique and contain every character count.

Imaging was performed on a GE Signa Excite HD 3T scanner using the product eight-channel head coil. Whole-brain image volumes were collected using a custom GRAPPA EPI sequence (Griswold et al., 2002). Sequence parameters were g-factor = 2, TE = 25 ms, TR = 2.5 s, and 2.5-mm cubic voxels across a 200 mm field-of-view. The slice prescription included 40 slices oriented along the AC-PC axis. A high-order shim was performed before the start of the functional imaging to improve field homogeneity.

A set of T1-weighted structural images was obtained on the same prescription before the functional acquisition runs using a three-dimensional (3D) fast RF-spoiled gradient-echo (fSPGR) sequence. These anatomical images were then used to align the functional data to a structural 3D reference volume, which was acquired for each subject in a separate session. The structural reference volume was T1-weighted with good gray-white contrast and was acquired using a 3D inversion-prepared fSPGR sequence (minimum TE and TR, TI = 450 ms, 15° flip angle, isometric voxel size of 0.7 mm, 2 excitations, ~28-minute duration).

Preprocessing of the fMRI data was performed using the mrVista software package (available at <http://vistalab.stanford.edu/>), modified for use in our own lab. The first 15 seconds of data were discarded to reduce transient effects. Within-scan motion was then estimated using a robust intensity-based scheme (Nestares & Heeger, 2000). Between-run motion was corrected using the same scheme, this time applied to the temporal average intensity of the entire scan. The first run of the session was used as the reference. Because the goal is to learn associations between patterns of activation in the brain and stimulus presentation, it is important that the activation is temporally aligned with the stimulus. Therefore, a Wiener filter deconvolution (Poor, 1980) was applied using a generic difference-of-gamma hemodynamic response function (Glover, 1999) as the kernel to the recorded BOLD signal. Mostly, the deconvolution served to shift the peak response in time so that it was aligned with its associated stimulus, but it also provided some amount of noise reduction. The high-resolution reference anatomies were segmented using the Freesurfer image analysis suite (http://surfer.nmr.mgh.harvard.edu/) to create approximate parcellations of the gray matter in each subject, as well as a surface model for visualization of mapping results.

The performance of machine learning algorithms is generally defined to be the expected accuracy of the classifier on previously unseen examples (Bishop, 2006). In practice, this measure can only be estimated. A typical approach is to split the available examples into training and test sets. The classifier is first trained on the training set, and its performance on the test set is then taken as the estimate of classifier performance on future unseen data. The splitting process is performed multiple times to reduce the variance of the performance estimate. This procedure is known as k-fold cross-validation (Kohavi, 1995).

In order to generate p values for these performance estimates, the null distribution for the cross-validated performance was generated by randomly permuting the labels on the examples 2000 times and repeating the training and cross-validation procedure. That is, the distribution of performance estimates was generated under the assumption that the labels and data were independent. Using this distribution, p values were calculated for the performance estimates (Ojala & Garriga, 2010). The high-performance computing resources of the Texas Advanced Computing Center at The University of Texas at Austin were utilized to perform this computation.

Previous studies (Pereira et al., 2009) have raised issues with performance estimates that are optimistically biased due to temporal correlations between examples (time frames) that violate standard assumptions of independence between training and test sets. For fMRI, the hemodynamic response introduces temporal correlations on the order of 10 seconds, which raises the question: What is the relationship between performance estimates and temporal correlation? To address this question, we estimated classifier performance when classifying number of characters presented using four different methods for splitting the data between training and test sets. Frames where no characters were present were removed, leaving 72 frames per run. We grouped different numbers of the remaining consecutive frames into selection units: 1 frame (frame split), 6 frames (block split), 36 frames (half-run split), and 72 frames (run split). For each of these unit sizes, we formed training and test sets by randomly selecting individual units (without replacement) and estimated classifier performance using these sets. For the frame and block splits, classifier performance was estimated using ten-fold cross-validation. For the half-run and run splits, only eight- and four-fold cross-validation was used respectively, due to the limited number of runs per subject. Based on our results, we chose to utilize the block split for performance estimates, as it did not exhibit an optimistic bias and allowed us to use more folds in the cross-validation procedure, which reduces the variance of the performance estimates.

We used feature selection methods to remove uninformative voxels, thus improving both training time and performance of the machine-learning algorithms. This was particularly important for the NN, where training times can be quite long compared to the other methods. Common tools for feature selection in neuroimaging include anatomical region-of-interest (ROI) selection, principal component analysis (PCA; (Hotelling, 1933)), and univariate statistical tests. ROI selection is a powerful aide for hypothesis testing, but is much less useful for data exploration. PCA selects the orthogonal projections with the highest variance, which are generally dominated by physiological nuisance and is therefore not well suited for our purposes. Instead, we used ANOVA (Scheffe, 1959), which has been shown to be effective for feature selection in the context of MVPA (Norman, Polyn, Detre, & Haxby, 2006; Pereira et al., 2009). The idea behind ANOVA is to calculate the mean and variance for the set of samples in each class (e.g. number of characters), and then use these statistics to determine how different the distributions for each class are. We used ANOVA in one of two different ways: selecting voxels that differed significantly between with-character and without-character periods, or selecting voxels that differed significantly across classification targets (i.e. number of characters). To calculate significance, ANOVA estimates the probability that the means of two different samples are different. For comparison, we performed both task-activated feature selection and classification-target feature selection. We found classification-target feature selection yielded the best results on this dataset. Additionally, care must be taken to avoid optimistically biasing the accuracy estimates; voxel selection must be performed within each fold of a cross-validation procedure.

Using the time series from the voxels selected by the ANOVA process, we constructed classifiers of the following types: one-versus-one multi-class linear support-vector machine with C = 1 (Cortes & Vapnik, 1995; Weston & Watkins, 1999), feed-forward neural network with scaled conjugate gradient backpropagation training (Hagan & Menhaj, 1994; Hornik, Stinchcombe, & White, 1989; Møller, 1993), Gaussian naive Bayes classifier (GNB) (Duda & Hart, 1973), and k-nearest neighbor classifier (KNN) with k = 6 (Cover & Hart, 1967). The parameters for the SVM and KNN were determined by a grid-search (Hsu, Chang, & Lin, 2010) on a left-out dataset. That is, the parameters were obtained on data not used in the cross-validation procedure to estimate performance. The performance of each classifier was estimated for three different classification problems: whether characters were present, how many characters were present, and what type of characters was present. For the former classification, the full time series was utilized; for the latter two classifications, we used only the fMRI data obtained during the character-present periods. The number of examples for each label was always balanced.

Although the NN can potentially learn more complex classification functions than the other algorithms, it uses a stochastic training process and has many more free parameters. To overcome these issues, we performed model selection within each fold of the cross-validation procedure using a validation phase. First, part of the training data was held out as the validation set. We then performed a grid-search on the number of hidden nodes and selected the best value based on the network’s performance on the held-out validation set. Then, using this hidden-node value, we trained 20 more networks and again selected the best network based on performance on the held out validation set. This procedure reduced considerably the variance of the NNs cross-validated performance.

All classifiers return a label for an input, but not all classifiers return the probability that the label is correct. For example, the SVM can only return a label, whereas the GNB classifier and feed-forward NN can return the probability for all labels (Richard & Lippmann, 1991). Normally, one chooses the label with the maximum probability as the selected class while ignoring its value, but we explored the use of this probability information to improve classification accuracy. It is also useful to consider a heuristic, which we shall call confidence, which is correlated with the probability that the chosen label is correct. For the GNB classifier, the probability of the chosen label can be used directly. However, the output of the NN is only an approximation of the posterior probabilities. Therefore, the outputs are first normalized to sum to one across all labels, and then the output corresponding to the selected label is taken to be that label’s confidence. Since the SVM only returns a label, generating a measure of confidence is not as straightforward and we therefore elected to only measure the confidence of the NN. How well confidence correlates with the true probability depends on how well the NN has approximated the joint probability distribution after training. The true probability cannot be measured directly, but we can compare the average confidence with the average probability that a label is correct, that is, the estimated classifier accuracy. To see how well the NN is estimating the joint probability distribution, we averaged confidence across all frames in a session and plotted it against the session’s cross-validated performance. It is also worth noting that confidence is calculated from the output of a trained NN and an input example, but not the associated label. This means that confidence could potentially be used as an independent quality estimate if the neural network was trained on an independent dataset.

A common approach to boost classification accuracy is to average across frames in a stimulus block ( e.g. Pereira et al., 2009). We compared the use of individual frames as examples to the use of examples created by averaging across 15-second blocks, and found that the block-averaged examples produced better classifier performance. Block averaging exploits our prior knowledge about the temporal structure of the stimulus, but it is not the only alternative.

We explored three other approaches for exploiting this knowledge: block voting, confidence voting, and output averaging. Block voting can be applied to any machine-learning algorithm. In block voting, the classifier was trained using individual frames as input examples, but the classification of a block was chosen as the majority classification of all frames in that block – each frame in the block “votes” on the block classification. The block voting procedure can be interpreted as a median filter on the output of a classifier trained on individual frames,whereis the classification of the block andare the classifications of the individual frames in the block. Confidence voting requires an algorithm that returns a probability along with the label. Confidence voting was similar to block voting, but each frame’s vote was weighted by the probability of the chosen label on that frame, whereis the set of all classes,is the weight or confidence associated with frame, andis the indicator function for class. Output averaging requires an algorithm that returns a probability for each output class such as a NN. In output averaging, the probability values from the neural network were summed across the block and the label was selected to be the class with the greatest value,whereis the probability output of the neural network for classat frame.

We have extended NN sensitivity analysis to determine the spatial distribution of voxels that contribute to the classification of each class. The key idea is calculate the sensitivity (or derivative) of the neural network output (classes) with respect to each input (voxels). Let be the vector of outputs and be the vector of inputs. Then the sensitivity of output to input is defined by , which is the partial derivative of the output with respect to the input. Let be the weight matrix from the hidden layer to the output layer and be a single element of corresponding to the weight on the network edge connecting output k with hidden node j. Similarly, let be the weight matrix from the input layer to the hidden layer and be a single edge weight. Then the partial derivative can be expressed as , where is the total number of hidden units in that layer of the neural network, is the value of the derivative of the activation function at output , and is the value of the derivative of the activation function at hidden neuron . Finally, the entire sensitivity matrix can be expressed in matrix notation as , where and .

Since the activation functions are generally non-linear, the sensitivity matrix becomes a function , where is an input vector. However, the sensitivity matrix for a particular input vector can vary due to the stochastic nature of training neural networks. To compensate for this added variance, we trained 100 different nets and calculated the average sensitivity matrix across these samples.

We now have a sensitivity score for each voxel at all time points and for all output classes. However, we would like a measure of sensitivity only on voxels. Therefore, we calculated for each point in the time series, and then computed the RMS average sensitivity matrix across all input vectors as , where is the number of input vectors (time points). gives a sensitivity value for each voxel with respect to all outputs. We then calculated the maximum sensitivity of each voxel across all outputs, i.e. . This sensitivity was projected back into the volume anatomy to create a map of the relative incremental importance of each voxel's response to the classification decision.

In order to empirically determine a sensitivity threshold to eliminate irrelevant voxels, we propose an approach based on recursive feature elimination (RFE; (Guyon et al., 2002)) adapted to the feed-forward neural network. A similar approach was used by (Formisano, De Martino, & Valente, 2008) in conjunction with the weight vector of a regularized SVM. In RFE, a machine-learning algorithm is first trained on a full data set. Next, some ranking criterion is calculated for each input dimension. The dimension with the lowest rank is removed from the dataset (a fixed number or percentile of dimensions may be removed for speed reasons). Then, the machine-learning algorithm is retrained on the reduced dataset. This process can be repeated until all features have been removed. The performance of each subset can be calculated using a held-out test set to determine a good threshold to remove irrelevant voxels. We used the feed-forward neural network as our machine-learning algorithm, and the measure for our ranking criterion. For computational speed reasons as well as for inter-subject comparison, we used a fixed sensitivity threshold at each iteration to determine which features would be removed. This allowed us to bootstrap classifier performance on a held-out test set across all sessions to obtain 68% confidence intervals (Efron, 1979).

For a qualitative comparison, we created surface maps for the NN sensitivity analysis, GLM, and searchlight. These techniques cannot be used for a direct quantitative comparison because they present fundamentally different information. Similarly, the thresholds used for each map are not directly comparable. However, the thresholds have been selected based on standard practices for determining meaningful localization of function and information. For sensitivity analysis, recursive feature elimination was performed on each subject's volume sensitivity map until the bootstrapped classier performance fell significantly below (p = 0.05) the peak classifier performance. The resulting maps were projected onto their cortical surfaces and blurred along the surface using a 5 mm full-width half-maximum (FWHM) Gaussian kernel (voxel size is 2.5 mm). For GLM, a linear activation model was constructed using an explanatory variable for each character count. Processing of fMRI data was carried out using FEAT (FMRI Expert Analysis Tool) Version 5.98, part of FSL. Z (Gaussianized T/F) statistic images were thresholded using clusters determined by Z>2.3 and a (corrected) cluster significant threshold of P=0.05 (Worsley, 2001). For searchlight, we employed a 3x3x3 kernel and a linear SVM classifier using the PyMVPA toolkit (Hanke et al., 2009). The searchlight maps were thresholded at twice chance decoding accuracy (33%). These maps were then projected onto the Freesurfer generated surfaces for each subject. We attempted to use non-linear warping to create a group average, but we were not satisfied with the registration accuracy. In particular, there was a tendency to confuse activity on superior temporal areas with that on dorsal parietal regions. To average across subjects, therefore, we aggregated the maps across ten anatomical labels automatically generated by Freesurfer during surface construction. To account for variations in the total surface area covered by the different maps, we calculated percent coverage, the fractional area of the thresholded map contained within each surface label, and bootstrapping was used to calculate 68% confidence intervals for all three approaches and all ten surface labels.

We built classifiers for three separate cases: with/without characters, 1—6 characters, and soldiers vs. insurgents. Recall that the with/without characters case has a block structure of 15 sec. for each condition, and that both conditions contained images of the town. For this case, classification performance was excellent, with typical scores of 94%-97% for the NN. This high performance was not too surprising, as there were strong low-level visual image differences between these two conditions. In contrast, the third case of distinguishing between soldiers and insurgents did not produce classification performance well above chance. Consequently, we focused our analysis on the second case, character counting where we did not distinguish between soldiers and insurgents.

Averaged across all 10 sessions (five subjects with two sessions each), the cross-validated performance estimates of all four classifiers are significantly above chance, where chance is 1 out of 6 = 16.7% (Figure 2). The SVM had the best performance, followed by the feed-forward NN (without using our new output processing techniques; see below). The performance of all four independent classifiers being above chance increases confidence in the results, however the GNB and KNN classifiers will not be discussed further as their performance was significantly below the SVM and NN. There is considerable variation in performance between sessions for the same subject, as well as variation in average performance between subjects.

Figure 2. The estimated performance of all four classifiers averaged across all sessions. The performance of individual sessions are indicated by the symbols. Each subject performed two sessions and there are therefore two symbols per subject. The performances estimates were bootstrapped across sessions in order to obtain 68% confidence intervals. While the SVM had the best average performance, all four classifiers performed well above a chance performance of 16.7%.

It is also worth noting the computation time of these algorithms in practice. The average training time was 0.683 ms per example for the SVM, 121.299 ms per example for the NN, 0.073 ms per example for the GNB, and 0.044 ms per example for the KNN. The training time of the NN is this ~2 orders of magnitude slower than the SVM. Nevertheless, the full NN cross validation procedure still only took approximately 10 minutes per session. The average decoding time was 0.431 ms per example for the SVM, 0.197 ms per example for the NN, 0.172 ms per example for the GNB, and 0.466 ms per example for the KNN. Unlike training times, the NN is the second fastest at decoding.

We tested four different methods for exploiting the block structure of the stimulus to improve classification accuracy: input averaging, block vote, confidence vote, and output averaging. Since confidence vote and output averaging require an estimate for the probability of each output label, only the feed-forward NN was considered for this comparison. Both block voting methods, and output averaging improved session performance significantly over simple input averaging. The output averaging method had the greatest average improvement (Figure 3).

|  |  |
| --- | --- |
|  |  |

Figure 3. The chart on the left shows individual session accuracies for all four averaging methods. The sessions have been sorted by average performance for improved readability. The chance probability for all sessions is 16.7%. The chart on the right shows the impact of the individual aggregation methods calculated relative to the baseline score as 10 log(score/baseline) for each session. These relative accuracy scores (in dB) are averaged across all sessions and bootstrapped to obtain 68% confidence intervals.

mri:Neurometrics:manuscript:confidence-confusion.pdfFigure 4. The average confusion matrices for the feed-forward NN with output averaging across all subjects. The value in cell (i,j) of the matrix is the percent of examples from class i that were labeled as class j; values along the diagonal indicate correctly classified examples while the rest indicate incorrectly classified examples. The color of the cell indicates deviation from chance probability (16.7%); greener cells indicating values above chance, and redder cells indicating values below chance.

It is clear from Figure 3 that not all sessions performed equally well. Even for the same subject, session performance varied significantly. We found that the average confidence (i.e., the probability of the chosen label) returned by the NN was very significantly correlated (R2 = 0.98; negligible p) with the network’s cross-validated performance (Figure 5). The confidence measure was calculated without knowledge of the labels and thus provides a measure of the quality of the data being classified as well as an estimate for how well the NN has estimated the joint probability distribution. Figure 5. Cross-validated session accuracy plotted against average session confidence.

From the confusion matrix in Figure 4, we see that the classifier is best at detecting the presence of a single character. In fact, there are relatively few cases of confusion between one and two characters. Apparently, these two situations evoke very different responses in the brain. Also, note that the majority of the incorrect responses lay just off the main diagonal. These responses correspond to the classifier being wrong by a single character in its classification. 1 and 3 characters were classified with the highest accuracy. This is likely due in part to the correlation with total scene contrast. However, note that 2 characters were also classified with high accuracy and yet had the second lowest p value for contrast correlation. Sensitivity maps for individual subjects show a preponderance of classification sensitivity in lateral occipital areas, ventral early visual areas, and dorsal parietal lobe (Figure 6). Subjects also displayed small regions of high sensitivity in portions of temporal and frontal cortex. There is significant overlap between the sensitivity, GLM, and searchlight maps, but the sensitivity maps show greater contributions from anterior brain regions.

Figure 6. A qualitative comparison of sensitivity, GLM linear-response Z-statistic, and searchlight accuracy maps projected onto semi-inflated cortical surfaces for three different subjects. The maps are roughly similar across subjects and hemispheres, but substantial individual variations are evident.

The information contained in these maps are not equivalent, and neither are their associated thresholds. The thresholds were chosen based on accepted practice for their associated technique, but they are not statistically equivalent and should only be used for qualitative comparison. Sensitivity threshold values were determined using a recursive feature elimination approach (section Error: Reference source not found). GLM linear-response Z-statistic maps and searchlight accuracy maps are also presented for comparison. The sensitivity maps were thresholded using the recursive feature elimination technique described in the methods. The Z statistic images were thresholded using clusters determined by Z > 2.3 and a (corrected) cluster significant threshold of P=0.05 (Worsely 2001). The searchlight maps are thresholded at twice chance probability (33%). Figure 7 presents the performance of the NN and the fraction of voxels remaining after each iteration. Greater than half the voxels can be removed without significant loss of classification performance.

Figure 7. A plot of the feedforward neural network estimated performance and the fraction of voxels remaining at each iteration of the recursive feature elimination procedure. The fraction of voxels is calculated with respect to the 2000 voxels selected by ANOVA. The performance estimates and voxel counts were bootstrapped across sessions in order to obtain 68% confidence intervals.

There is substantial overlap between all three maps in lateral occipital and lingual cortex. Elsewhere the mapping methods show different patterns of response. For example, early visual cortex, roughly demarcated by the pericalcarine and cuneus labels, shows greatest classification sensitivity by the searchlight technique, intermediate response based on GLM, and relatively low information content based on our NN sensitivity metric. Interestingly, several temporal lobe regions show greater sensitivity based on the NN metric than either of the others. To determine if the sensitivity in these regions is meaningful, we estimated the performance of the NN on a subset of the original voxels constructed by taking all of the voxels considered significant by the sensitivity analysis and removing all those voxels considered significant by GLM. The cross-validated performance on this subset averaged across all sessions was 25% (with p < 0.05 for all sessions). While the performance dropped substantially, these voxels were still able to classify character count significantly above chance.

We set out with the practical goal of decoding the subject’s cognitive state associated with viewing a number of characters from time series of functional images. Using a combination of standard and novel machine-learning methods, we were able to extract this information with accuracy that varied from well above chance to nearly perfect (Figure 3), depending upon session and machine-learning algorithm. For the neural network results, we then presented a novel approach to relating the network’s decision-making sensitivity back to brain anatomy of the individual. These sensitivity maps suggest that a more widespread and diverse network of brain regions encoded the cognitive state, which is consistent with the complex nature of the VE stimulus.

The work described in this paper expands the opportunities for utilizing virtual environments for scientific inquiry in cognitive neuroscience. The design of the stimulus provided a balance between realism and experimental control so that quantitative analysis of the fMRI data stream achieved a degree of confidence ranging from satisfactory (well above chance) to very high. Care was taken to preserve as much of a natural experience as possible. For example, we never exposed the subject to disruptions in the experience of being present in a virtual environment, yet the stimulus had an otherwise classic block design. And the synthesized video stream never showed static images at any time, which rarely occur under natural conditions. We also eschewed averaging data between different subjects in accordance with one of our goals: modeling individuals for therapies and learning regimens, including utilizing real-time fMRI.

Despite these seemingly greater challenges, we were able to achieve classifier performance that was significantly above chance with all four of the MVPA methods we tested. More importantly, for the two strongest methods, support-vector machines and artificial neural networks, the classifier performance was sometimes good enough to enable practical applications. This is especially impressive given that the cognitive states being discriminated were not based on differing object categories (e.g. houses, faces, tools, etc.) that often activate brain regions with limited anatomical overlap (Hanson, Matsuka, & Haxby, 2004), but rather were from a single object category, viz. combatants, and differed only in number of combatants.

We also discovered that the performance of classic feed-forward neural networks (NN), which have been somewhat neglected lately in favor of SVM, can be competitive with SVM on the data in this study. While the inherent properties of SVM make it well suited to sparse representations (small number of object categories vs. large number of voxels), neural networks provide a more general method that can (in principle) capture more subtle features given enough data. Moreover, NNs provide probability values that can be used to further improve classification performance. Looking to the future, building NNs using “deep learning” (Hinton, Osindero, & Teh, 2006b) has been shown repeatedly to outperform SVM on many types of data (Cireşan et al., 2012). Even greater classifier accuracy may be possible with such methods applied to VE data.

Classifier performance will be important for both on-line use of fMRI in brain-computer interfaces (BCI), such as PTSD therapy, as well as for off-line creation of brain maps using sensitivity analysis. The techniques block voting, confidence voting, and output averaging (see section 2.9), all improved performance over the baseline classifier performance as well as over input averaging. The concept of using the output of the classifier to ascribe confidence (see section 2.8) to each output could be very useful for differentiating the reliability of entire sessions. Similarly, any confidence measure could be quite valuable in BCI applications in which low confidence frames could be weighted by confidence to reduce their influence and/or dropped entirely from any on-line decision-making by the BCI software.

Classification sensitivity in early retinotopic visual areas and lateral-occipital areas suggests that retinotopic organization is important to decoding group size for our VE stimulus. Because LO combines object-selectivity with retinotopic specificity (Sayres & Grill-Spector, 2008), different group sizes could evoke complex but stereotypical patterns of responses in LO (and other retinotopically organized areas) as subjects visually interrogate the stimuli with a sequence of eye movements. Regions in the parietal cortex have been shown to be involved in mental arithmetic and magnitude judgment (Rickard et al., 2000) which may also play some role in decoding group size. More recent research suggests this region may even contain a topographic representation of numerosity (Harvey, Klein, Petridou, & Dumoulin, 2013). There is some debate as to whether this topographic map represents numerosity or sensory processing (Gebuis, Gevers, & Cohen Kadosh, 2013), but it would be useful for decoding group size regardless.

Integrating information from the whole brain improves decoding accuracy, but it makes interpreting functional localization problematic. From our sensitivity analysis, we see that regions associated with low-level vision, higher-level object-recognition, and potentially even cognitive representations of numerosity all contributed to decoding. However, the sensitivity analysis does not necessarily tell us how these regions contributed. Eye movements and other behavioral responses as the subjects visually interrogate the stimuli could induce reliable and complex patterns of activation in all of these areas. For example, our control analysis indicates that low-level contrast features may have partially, but not entirely contributed to decoding. Similarly, increased eye movements could create a higher variance of activation in retinotopic visual areas. If this behavior is reliable and consistent, the machine learning algorithms will learn to use that information to help decode the state. At this early stage, we did not collect eye tracking data during our experiments to evaluate to what extent this contributed to decoding. Eye-movement information is not obviously correlated with character count, but rather the cognitive evoked in the subject by the VE: being in a town and freely viewing a specific number of characters. It is this VE-specific state that we are interested in decoding. Such goal-driven decoding should be more useful for training and therapy exercises where the underlying neural mechanisms may not yet be well understood. However, neuroscientific studies looking to leverage VEs and sensitivity mapping for functional localization must still be careful to balance realism with control to avoid these kinds of confounds when interpreting their results.

The GLM produced Z-statistic maps indicate significant activation only in early ventral visual areas and lateral occipital regions. Searchlight produced results qualitatively similar to GLM, suggesting that the expansion from a single voxel with GLM to a 3x3x3 set of voxels in searchlight was not sufficient to capture potentially important long-range multi-voxel response patterns identified by the NN sensitivity analysis. Therefore, we conclude that extracting response patterns by performing classification on voxels selected from a spatially diverse collection of voxels captures potentially important brain information missed by both GLM and searchlight (Error: Reference source not found).

Note that the information contained in the maps is quite different, making them difficult to compare directly. The Z-statistic maps tell us how well individual voxels agree with a hypothetical model, the searchlight maps tells us how well small localized groups of voxels are able to decode the desired brain state, and the sensitivity maps tells us how much individual voxels contribute to a spatially-distributed decoding decision. We do not have a practical way to calculate p-values for individual voxels with the sensitivity analysis so care must be taken when interpreting the results. However, a qualitative comparison of the techniques is still useful. While we are unable to calculate the significance of individual voxels for our sensitivity analysis, the comparison shows that the resulting sensitivity maps highlight regions consistent with accepted mapping techniques where per voxel significance calculations are possible. This increases our confidence that the areas indicated by the sensitivity analysis, but not the other techniques, likely do contain information relevant for decoding the subject’s brain state and could merit further investigation.

In conclusion, it is possible to extract useful information from fMRI data obtained using a realistic virtual environment stimulus using machine-learning methods. Neural networks, supplemented by some averaging techniques, performed particularly well. The resulting classification data, moreover, can be mapped onto the brain using a novel form of sensitivity analysis. These methods open up new possibilities for the use of virtual environments in both neuroscience research and in clinical applications.

Six adult males, ages 24-57 (TODO: need to check how old Ethan was), with normal or corrected-to-normal vision, participated in the experiments. All subjects participated in two fMRI sessions and a third session to acquire a high-resolution structural anatomy. Informed consent was obtained from all subjects under a protocol approved by the University of Texas at Austin Institutional Review Board.

Given our long-term interest in PTSD, we created a virtual environment very similar to those used in current PTSD extinction therapy programs (see figure #). Subjects perform a weapon detection task at 6 different locations in this virtual town. Between detection tasks, the screen smoothly transitions to a new location over 5 seconds, then a cue is presented to the subject that indicates the difficulty of the upcoming block for 5 seconds. Then the subject performs the task at that location for 45 seconds. During the task, 20 characters move through subject’s view, and a random character will pull out a weapon every 2—4 s. Subjects press a button to indicate they detected the weapon and the character puts the weapon away. If after a certain amount of time the subject fails to notice the weapon, the character puts the weapon away. After the task, we have a control period that begins with another 5-second cue followed by 15 seconds of the characters moving around and pulling out flashlights rather than weapons. The subject is instructed not to respond during this period and any responses are considered incorrect. The expected difficulty of the stimuli is adjusted each time the subject moves to a new location. However, the difficulty settings and locations were balanced so that there would be no correlation between location and task difficulty. Furthermore, low-level contrast was held constant in real-time using a post-processing shader.

Screenshot from stimulus of 1 location

Screenshot from stimulus of another location, in this image a character is holding a weapon that is circled.

The duration that the character holds the weapon is dependent on the task difficulty and the actual visibility of the object. Since all of the characters are moving randomly and independently, the weapon frequently becomes partially or fully occluded during the search task. To account for this variability in difficulty, the duration that the weapon is visible is determined in real-time by its visibility. We calculate the total area of visible weapon in pixels and multiply this by the frame rate of the stimuli to arrive at the weapon’s total visibility in pixel-seconds. The character holds the weapon until the accumulated pixel-seconds exceed the threshold for the current difficulty setting. In addition to total visibility, varying the 3D depth of the character with the weapon also controlled task difficulty. The two variables are linked in that a weapon further away has a smaller projected area in pixels, so total visibility was calculated based on a constant factor from the distance.

Performance data was collected for each subject outside of the scanner. This data was used to individually tune the difficulty of the task to ensure that we would be able to collect an approximately uniform distribution of subject performance scores during the scanning session as well as to ensure that the subjects were not responding during the control period. Task performance was estimated as the average performance of the subject during a 30-second block while the distance of the weapon from the viewer and its total visibility were used to estimate difficulty. However, due to the semi-random movements of the characters, effective distance and visibility could vary considerably. Therefore, we collected extensive logging information during the stimuli that included the exact timing, positions, and visibility of all characters and weapons as well as the responses of the subject. Using this data, we fit sigmoid curves that allowed us to predict with reasonable accuracy the subject’s performance based on the difficulty setting of the stimuli (Figure #). During training periods outside of the scanner, the subject received audio feedback for successfully finding the weapon, missing a weapon, and indicating that they have seen a weapon while one is not present. However, during the scanning sessions there were no feedback cues.

Imaging was performed on a Sieman’s Skyra 3T scanner using the product 32-channel head coil. Structural reference volumes were T1-weighted with good gray-white contrast and acquired using a 3D inversion-prepared fSPGR sequence (minimum TE and TR, TI = 450 ms, 15deg flip angle, isometric voxel size of 0.7 mm, 2 excitions, ~28-min duration). fMRI scans were collected using a whole-brain GRAPPA EPI sequence with g-factor = 2, TE = 25 ms, TR = 2.5s, and 2-mm cubic voxels across 200mm field-of-view. The slice prescription included 60 slices oriented along the AC-PC axis. A high-order shim was performed before the start of the functional imaging to improve field homogeneity. A set of T1-weighted structural images were obtained on the same prescription as the functional acquisition runs in the same session directly before the functional scans were collected. Images were acquired using a three-dimensional (3D) fast RF-spoiled gradient-echo (fSPGR) sequence. These anatomical images were then used to align the functional data to the structural reference volume.

Automatic cortical segmentation and surface extraction was performed on the structural reference volume using FreeSurfer (ref). The cortical surfaces for each hemisphere were inflated into a sphere while minimizing metric distortion. These spherical surfaces are then registered to a spherical atlas first by coarsely aligning on large-scale folding patterns and then fine-tuned using small-scale curvature patterns (freesurfer sphere ref).

The inplane anatomical volumes were skull-stripped and normalized in the same fashion as the first stage of the automatic cortical segmentation and surface extraction. The processed inplane volumes were then affinely registered to the structural reference volumes using a method based on robust statistics to detect outliers and remove them from the registration (Highly Accurate Inverse Consistent Registration: A Robust Approach).

Simultaneous slice-timing and motion correction was performed on the functional scans (ref nipy.SpaceTimeRealign). Then, a rigid-body registration was performed between scans to align each frame to the first volume, i.e., the frame closest in time to the structural inplane (Improved Optimisation for the Robust and Accurate Linear Registration and Motion Correction of Brain Images).

The functional data was then approximately aligned to the structural reference volume using the previously calculated registration with the inplane anatomical data. A boundary based registration technique was then used to fine-tune the registration of the functional data to the structural reference volume (freesurfer bbregister).

Then, the functional data was projected onto the extracted cortical surfaces by averaging between white and pial surfaces along the surface normal (mri\_vol2surf). To minimize partial volume effects, values were only averaged between 20% and 80% of the distance between the white and pial surfaces along the normal.

Finally, the functional data was smoothed along the surface with a Gaussian filter and then projected on to an icosahedron with uniform spacing of vertices in the spherical template space (freesurfer mri\_surf2surf). We experimented with several different smoothing values and icosahedron order numbers (which determines the density of vertices on the sphere, i.e., the resolution of the data) to determine parameters that sufficiently reduce the dimensionality of the data while still retaining as much information as possible.

Machine learning was performed on the data at several points during the preprocessing pipeline: in the volume immediately after registration, in the volume after isotropic smoothing, on the surface, on the surface after surface-based smoothing, and finally on the down-sampled icosahedron surface. We employed this procedure as both a sanity check and a quality assurance measure. If classification accuracy is significantly degraded after a processing stage, that likely indicated an error or poor parameter choices in that stage. It also helps show how important various preprocessing steps are to ultimately boosting classification accuracy.

Before classification, performance values were binned into three categories: poor, average, and good. Boundaries between categories were chosen so as to make the number of examples in each category as similar as possible. Note that these boundaries were selected based on performance values collected during the training process outside of the scanner to avoid potentially biasing the estimated classification accuracy.

After binning, ANOVA feature-selection (ANOVA ref and sk-learn ref) was employed to select those voxels which had significantly different distributions between task periods and control periods. No knowledge of the subject's performance was used in this feature selection step. It only selected voxels that were consistently affected by the task itself and therefore were likely to contain useful information concerning the subject's performance. The top 3000 features were selected based on their ANOVA score. This number was selected based on experience from previous feature-selection experiments (neurometrics1).

Finally, classification accuracy was estimated using a 6-fold cross-validation procedure (cv ref). The 6 folds were selected based on the 6 task blocks in order to minimize temporal correlation between features from different blocks. The lack of independence in blocks can result in optimistic classification accuracy estimates.

The classifier used in this procedure was the SVM with C = 1 (support vector machine; SVM ref). The SVM algorithm is one of the most popular algorithms for doing machine learning on neuroscience data (example refs). This is due in large part to its simplicity and that it is well suited to problems where the dimensionality of the data exceeds the number of examples (nearly always true in neuroscience experiments). Although more complex algorithms can achieve better classification accuracies, we were interested in exploring a large parameter space and evaluating classification accuracy at several stages during preprocessing. We used a highly efficient implementation of the SVM (ref) to make this feasible.

## Future Work

We have preliminary evidence that suggests that combining classification outputs from each frame in a block to produce a single classification for the block significantly increases performance. The first combination scheme we tried was a simple voting mechanism, but we are exploring other more complex methods as well. We are also experimenting with training hierarchical SVMs and NNs on anatomic percellations. The hierarchy should help the machine learning algorithms exploit these inherent structures in the data. Due to their complexity, we are building a new computational framework to efficiently train and test these networks. We are also exploring the use of classification confidence measures. This is a machine learning topic that has received little attention, but may be very important in the training and therapy domain. Frequently when a machine learning algorithm misclassifies an input, it is because that input is either ambiguous or unlike any example the classifier was trained on. It would be helpful to have the classifier report some measure of confidence, particularly in the case of real-time feedback where the subject can be guided back to a state where the classifier is more reliable. Additionally, the trainer or therapist could ask the subject to self report only when the classifier reports low confidence. These data samples could then be used to retrain an even better classifier. We are performing a thorough investigation of the reliability of different confidence measures on a variety different machine learning algorithms to determine their effectiveness.

We are currently in the process of identifying additional internal cognitive variables to target. We are planning experiments where we encourage the subject to use different high-level strategies for completing tasks. In this way we would be measuring distinct neural states rather than single cognitive variables. We expect to see the different strategies reflected in the spatial distributions shown by the sensitivity maps and the ICNs contributing to classification. Another potential target is the subjective units of distress scale (SUDS). In the work of Gerardi, Rothbaum, Ressler, Heekin, & Rizzo (2008), the patient undergoing the virtual reality expose therapy is asked to report his current level of distress in SUDS every 10 seconds. Providing this same measure to the therapist based on patterns of neural activation without having to interrupt the immersion of the procedure would be extremely helpful. Although there are additional complications associated with measuring distress and anxiety, our results thus far indicate that such a system is within the realm of possibility.

#### Hidden Markov Model

Hidden Markov Models represent a system where there is a hidden or unobservable random variable that is linked to an observable variable. They allows us to calculate the probability distribution of the hidden state at some point in time given a sequence of observed variables. The model only requires two conditions on the hidden and observed variables. First, the dynamics of the hidden states obey the Markov property that the distribution of a hidden state at a particular point in time is conditionally independent given the value of the state at the previous point in time. Second, the observed variables must follow a similar property in that the distribution of an observed variable at a particular point in time must be conditionally independent given the value of the hidden state at that time. This model is particularly relevant for many engineering problems where we wish to estimate the value of some unknown variable given imperfect measurements but the dynamics of the variable are well known and can be mod- eled. There are a number of useful conditional probabilities to consider in this model and we will be examining three of them: the probability distribution of a state given all previous observations, the probability of a state given all previ- ous and future observations, and the probability of the entire sequence of states given the entire sequence of observed variables. We will also be introducing algorithms for calculating these probability distributions.

A Hidden Markov Model can be formally defined as the following. Let be a Markov chain on the discrete domain with transition probability matrix where . Let be a series of random variables on the discrete domain with observation probability matrix where and each is conditionally independent given .

Filtering is the problem of finding the probability distribution of a hidden state given all previous observations. Formally, we can define this distribution as

However, let us first consider how to calculate the full joint probability:

Note that and are defined directly by our model, and is the same as equation 2 at time . This implies some recursive algorithm exists for calculating the joint probability. To recover the conditional probability we calculate

Given observations , a probability transition matrix , an observation probability matrix , an initial state distribution , and , the filtered distribution of state can be determined using the forward algorithm [ref].

Smoothing is the problem of finding the probability distribution of a hidden state given all observations including those in the future. Unlike filtering, smoothing can not be performed in real-time as it requires future observations. Formally we can define this distribution as

where . Similar to filtering, let us begin by considering the full joint probability

Note that is the unnormalized output of the forward algorithm up to time . Therefore, we only need to solve

and are defined directly by our model, and is the same as equation 13 at time . Again, this implies some recursive algorithm exists for calculating the joint probability. To recover the conditional probability we calculate

Given observations , transition probability matrix , observation probability matrix , and initial state probability distribution , , and , the smoothed distribution of state can be determined using the forward-backward algorithm [ref]. This algorithm can be modified to efficiently calculate smoothed values for the entire sequence of hidden variables. Instead of performing the forward and backward recursive calculations to a specific time point, we can simply perform both operations on the entire sequence and store the results of each iteration in memory. Then to find the smoothed value for any particular time point, we multiply and normalize the results of the forward and backward procedure for that particular time.

Although smoothing calculates the probability distributions for all hidden variables given every observation, it cannot tell you what the most likely sequence of hidden variables was. Taking the sequence with the largest probability at each time step according to the smoothing algorithm will not yield this sequence, as these probabilities were not conditioned on the previous state. Therefore, we want to find

The Viterbi Algorithm is an efficient solution to the problem of finding the maximum likelihood sequence that was proposed by Andrew Viterbi in 1967 as a decoding algorithm for convolutional codes and has since been used in a variety of applications (Viterbi (1967)). Given observations , probability transition matrix , observation probability matrix , initial state distribution , and , we can determine the maximum probability sequence .

In my research, I am attempting to decode brain states from functional magnetic resonance imaging (fMRI) data. The level of neural activation in the brain is approximately captured by the blood-oxygen-level dependent (BOLD) signal. However, this signal is extremely noisy; the signal-to-noise ratio is generally on the order of 1. Therefore, we must integrate repeated measures to achieve good accuracy. Traditionally, this is done by averaging across blocks of time where the stimulus, and presumably the brain state, is constant. By leveraging a hidden Markov model, we can relax the constraints on the temporal structure of the stimulus and hidden brain states.

The true brain state is a massively high dimensional variable that is dependent on the state of all of the neurons and other signaling pathways in the brain. This state space is approximately bounded by where is the number of signaling pathways and is the number of neurons in the brain. The average adult human has around 86 billion neurons which makes working with the true state space intractable. Instead, we consider a much smaller state space defined by an indicator function on the true state. In our experiments, subjects are asked to make a choice or complete some task. Presumably, the choice they make or the way they complete the task is a function of the true brain state. Each choice then can be considered an indicator function on the true state space that is 1 when the brain’s state is such that it would choose that option. In this way, we can construct an arbitrary number of indicator functions to explore numerous neural hypotheses.

The BOLD signal is captured by the MRI machine every 2.5 seconds across the entire brain with a resolution of . The signal is produced by shifts in the level of oxygen concentration in the capillaries feeding neurons. As neurons spike more frequently, they consume more glucose and oxygen. To compensate for this increased metabolism, the brain increases blood flow specifically to the area of high activation. The increased blood flow results in increased oxygen concentration that can be measured by the MRI machine. The signal is correlated with neural activity but it is smeared out in time by the hemodynamic response function (HRF). This function is a measure of how the blood flow responds to activation. Unfortunately, this signal is generally quite weak compared to the noise in the system. Noise includes thermal and electronic noise as well as noise introduced by heart rate and respiration rate which also influence the oxygen concentration.

To accurately model the state transition probabilities we found that we needed to use a discrete number of previous states. The set of brain states we are interested in decoding is , where for our stimulus. The domain of the hidden states is , where is the number of memory states we use. This results in an extremely large but sparse state transition matrix . The domain of the observable sequence is where is the number of voxels we are measuring. We use a trained feed-forward neural network to approximate from the observed fMRI data (Richard and Lippmann (1991)). We can’t generate an observation probability matrix because our observable domain is continuous, but by fitting a distribution to we can calculate using Bayes’ rule. Fortunately, prior fMRI studies have shown that the distribution of fMRI data can be reasonably approximated by a multivariate Gaussian after whitening (Worsley (2001)).

Based on this model, we propose the following modified forward algorithm for fMRI data. Given observations , a probability transition matrix , an observation probability distribution , a function that approximates , an initial state distribution , and , the filtered distribution for the state can be determined using Algorithm 4. The performance of this algorithm was estimated using a cross-validation procedure (Kohavi (1995)) because the distribution and the function need to be constructed on a training set. We compared it’s performance to simple block averaging on previously collected fMRI data.

**Algorithm 4** Forward Algorithm

The stimulus is a realistic virtual environment that the subject views while inside the scanner. The subject’s view movies through a virtual town and periodically stops while some number of human characters move onto the screen. The characters stay on the screen for 15 seconds before the view moves on. The subject is not instructed to perform any task and simply views the scene passively.

Data was collected on 5 male subjects aged 25 to 57. For each subject, 2 sessions were collected.

The information we attempted to decode from the fMRI data was the number of characters visible on the screen. During the stimulus, the number of characters presented varied from 1 to 6 so we have 6 hidden states. However, we also considered the 5 previous states in our model which led to a total state space of 66 states. The observable was the fMRI data which was the BOLD signal on a 80 x 80 x 40 grid of voxels sampled every 2.5 seconds. Each 2.5 second period is called a frame and after removing the periods of moving through town we are left with 288 frames for each session.

For baseline comparison, a neural network was trained to estimate the number of characters presented in each frame. We measured the decoding accuracy of the network, or the percent of correctly estimated states. The estimated state is taken to be the state with the maximum posterior probability. The accuracy of the classifier was estimated using cross-validation to avoid any bias.

We also performed simple block averaging for comparison. During each 15 second period that characters are presented, the number of characters does not change. Therefore, we can average the fMRI data across these 15 second blocks to provide a better measurement. A second neural network was trained to estimate the number of characters presented in each block from the time- averaged fMRI data. The performance of the classifier was estimated using cross-validation.

The probability transition matrix was constructed such that the state would stay the same for 6 frames and then switch to another state with equal probability. Here, we are utilizing our knowledge that the number of characters stays the same for 15 seconds. This results in an extremely sparse transition matrix which we implemented using a sparse matrix class instead of forming a functional mapping. We estimated as a multivariate Gaussian which we fit to the fMRI data using a least squares approach. Similarly, we estimated using a trained neural network exactly as we did for the baseline. The modified forward algorithm was then used to generate the distribution for each frame. The state with the maximum posterior probability was selected as the estimated state for that frame. The accuracy of the model was estimated using cross-validation where both the training of the neural network and the estimation of the multivariate Gaussian were performed within each fold.