Identifying Virtual World Induced Brain States using fMRI and Machine Learning

Andrew Floren[a,*], Bruce Naylor[a], David Ress[a]

[a] *The University of Texas at Austin, Austin, TX 78712 USA*

Abstract

abstract

Keywords: keyword1, keyword2

1. Introduction

In traditional fMRI experiments, investigators seek to identify relationships between the measured BOLD signal and a carefully designed stimulus in order to tease out the purpose of particular brain regions. Recently, a new trend has developed where researchers are instead looking to predict what stimulus was presented given the measured BOLD signal (**???**). While successful, most of these experiments involve presenting static images from a limited number classes such as faces and places. Then, the researchers try to classify which image or class of images was presented during each frame by analyzing the measured BOLD signal using machine learning classifiers. While this has proven to be a successful approach, it does not mimic the dynamic environment in which brains have evolved. Our goal is to analyze brain function with dynamically changing stimuli that portray real world experiences. Further, we were interested to see what information could be gleaned from the BOLD signal beyond object categories. We used virtual world technology to specify the stimuli in detail. Given our long-term interest in PTSD, we created a virtual town intended to suggest the kinds of real-world settings currently encountered by our military forces. Virtual character representing friendly forces and hostile combatants were presented

*Corresponding author.

Email address: afloren@utexas.edu (Andrew Floren)

in a virtual town. We then trained linear SVMs (support vector machines) and feed forward neural networks to predict the number of characters in each stimulation. [Bruce: this is most of your introduction from the abstract but feel free to rewrite the whole thing. I'll add in the literature review once you are happy with the introduction.]

2. Methods

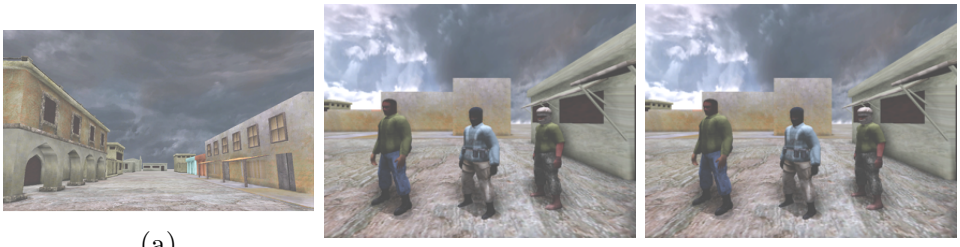
2.1. Subjects

Five male adults, with normal or corrected to normal vision, participated in the experiments. All subjects participated in three sessions, two sessions to collect machine-learning data, and one session to measure a high-resolution volume anatomy. Informed consent was obtained from all subjects.

2.2. Stimulus

We developed a virtual reality environment similar to many popular first person video games using the Unreal Engine 2 SDK (**?**). The stimuli is dynamically rendered and presented from the point of view of a camera moving through this virtual environment using a block design. First, the viewpoint moves for 15 seconds through the virtual environment with no characters present (Fig. 1a), then pauses for 15 seconds during which a group of characters fades into view (Fig. 1b). Each run consists of 12 alternations between moving through the virtual environment and character presentations. In each scanning session, 4 to 6 runs were collected.

The camera is constantly moving (even during the character presentation periods the camera slowly pans and rotates) and the characters are animated so that the presented scene is never static. The number and location of characters varies with each presentation in a quasi-random fashion. The number of characters presented varies from 1 to 6. A presentation with a particular number of characters appears twice in each run, however the order of these presentations was randomized. It should be noted that this random ordering was generated once and held constant between subjects. Additionally, even between character presentations with the same number of characters, the locations of those characters varies considerably; compare figures 1b and 1c. [I need another frame from the stimulus to illustrate this point. These figures are all from the original report but I'd like to generate new ones if I can get access to a machine with the stimulus.]



(a) (b) (c)

Figure 1: (a) An example frame from the stimulus while the camera is moving through the virtual envronment. (b) An example frame from the stimulus while characters are being presented. (c) Another example frame from the stimulus while characters are being presented. Note how the locations of the characters varies considerably between these two frames.

2.3. MRI protocols

Imaging was performed an GE Signa Excite HD scanner using the product 8-channel head coil. We collected whole-brain image volumes using a custom GRAPPA-accelerated EPI sequence (**?**). 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 (Fig. 2). A high-order shim was performed to improve field homogeneity.

A set of T1-weighted structural images was obtained on the same prescription at the end of each machine-learning session using a three-dimensional (3D) RF-spoiled GRASS (SPGR) 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, SPGR sequence (minimum TE and TR, TI = 450 ms, 15 flip angle, isometric voxel size of 0.7 mm, 2 excitations, 28-minimum duration).

2.4. Preprocessing

Preprocessing of the fMRI data was performed using the mrVista software package (available for download at `http://vistalab.stanford.edu/`) as well as additional tools developed on the mrVista framework in our lab. The first 15 seconds seconds of data were discarded to reduce transient effects.

Figure 2: An example prescription from one of the subjects.

We then estimated in-scan motion using a robust scheme (**?**). Between-run motion was corrected using the same intensity-based scheme, this time applied to the temporal average intensity of the entire scan. The last run of the session was used as the reference. Additionally, we applied a Wiener filter deconvolution (**?**) using a generic difference-of-gamma HRF (**?**) to shift the peak response in time so that it is aligned with its associated stimulus.

Wiener filter deconvolution can be summarized as follows: Given a system

$$y(t) = h(t) * x(t) + n(t) \tag{1}$$

where $x(t)$ is the signal of interest, $h(t)$ is some blurring kernel, $n(t)$ is independent additive noise, and $y(t)$ is the recorded signal. We want to find the deconvolution kernel $g(t)$ such that

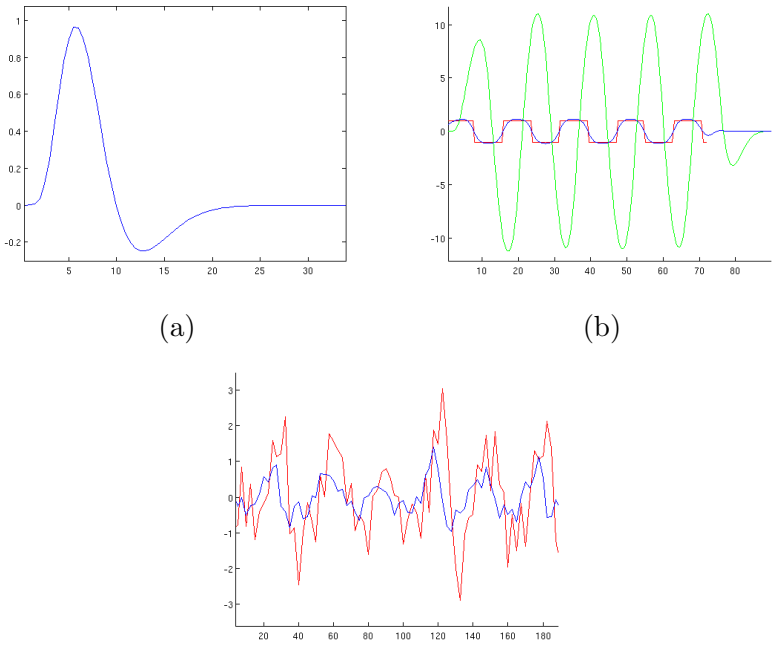
$$\hat{x}(t) = g(t) * y(t) \tag{2}$$

minimizes the mean squared error between $x(t)$ and $\hat{x}(t)$, or

$$\sum_t \left(\hat{x}(t) - x(t)\right)^2 \tag{3}$$

The solution for the optimal $g(t)$ is most easily expressed in the Fourier domain.

$$g(t) \xrightarrow{\mathcal{F}} \frac{H^*(f)}{|H(f)^2| + \mathrm{SNR}^{-1}(f)} \tag{4}$$



(a) (b)

(c)

Figure 3: (a) The difference-of-gamma HRF employed as $h(t)$. (b) A simple square wave in red. The same square wave after convolution with $h(t)$ in green, then after Wiener-filter deconvolution in blue. (c) An example voxel time series in red. The same time series after Wiener-filter deconvolution in blue.

Where SNR(f) is the signal to noise ratio $\frac{|X(f)|}{|N(f)|}$.

In fMRI, $y(t)$ is the recorded BOLD signal, $h(t)$ is the hemodynamic response function, and $x(t)$ is the neuronal population response. Calculating $g(t)$ requires estimates of the power spectral density of the signal of interest as well as the noise. However, the noise $n(t)$ corresponds not only to scanner noise but other nuisance factors such as pulse and respiration. This makes modeling the noise, and its power spectral density, very difficult. Therefore, we set SNR$(f) = 1$ for all frequencies f. Figure 3 illustrates the effects of this deconvolution on a simple square wave as well as an example voxel time series.

The high-resolution reference anatomies were segmented using the FreeSurfer

analysis package (**?**) to create approximate masks of the gray matter in each subject, as well as a surface model useful for visualization of the results.

2.5. Dimensionality reduction

We select a relevant subset of the volume using a harmonic power analysis based on the block design of our stimulus. One can show that the response of any linear system to a blocked alternation at frequency f will contain power only at f and its harmonics. Under a linear response assumption, we can therefore form an unbiased estimate of the response power by summing the power at these frequencies. Let $y(t)$ be the recorded discrete time series at some voxel. Then let $Y(f)$ be the discrete Fourier transform of $y(t)$. The fractional harmonic power of that time series is defined as:

$$P_h = \frac{\sum_{i=1}^{M} |Y(i \cdot N)|^2}{\sum_f |Y(f)|^2} \tag{5}$$

Where P_h is the fractional harmonic power, M is the number of harmonics, and N is the frequency of interest, in our case the period of the block alternations. Because the BOLD response has a predominantly low-pass temporal frequency response, we chose $M = 4$. Using P_h, we then selected a particular number N (range 1000-3000) voxels with the greatest power.

This harmonic-power selection was based on the alternation between characters present and characters absent, without regard to the number of characters presented. Therefore, we will only be presenting classifier accuracy estimates for character count, and not for the presence or absence of characters to avoid cross-contamination between dimension-reduction and classification criteria that would result in inflated classifier performance estimates (**?**).

2.6. Classification

Using the time series from the voxels selected by the harmonic power analysis, we trained a linear SVM and a feed forward neural network. These algorithms were trained and validated using a cross-fold approach (**?**). To maximize our temporal resolution, each frame (a point in the pre-processed time series) was treated as a separate data value, rather than of averaged across the block as has been done in previous work (**?**).

[Quick introduction to training, test, and validation sets will go here.]

Previous studies have discussed issues with optimistic performance estimates due to temporal correlations violating independence assumptions between training and test set examples (**?**). We were interested in more closely

Split	Mean Delay	Mean Minimum Delay
Frame	5776	26
Block	5883	215
Half-Run	6360	559
Run	6960	1300
Session	0	0

Table 1: Summary of training and test set split methods and the mean temporal delay between the training and test data.

examining the relationship between performance estimates and this temporal correlation. To accomplish this, we estimated classifier performance using 5 different methods for splitting the training and test examples in order to vary the average temporal delay between frames in the training and test sets. The first four methods deal only with frames from a particular session. In the first method, which we will refer to as the frame split, frames were independently drawn into the training and test sets. Although the draws were independent, there is no restriction to prevent adjacent frames being split between the training and test sets. In the second method, which we will refer to as the block split, sets of frames corresponding to the stimulus blocks were independently drawn into the training and test sets. Our stimulus design enables us to break runs into two sets that each contain a complete set of character-number presentations. Accordingly, in our third method, which we will refer to as the half-run split, sets of frames were independently drawn from these half-runs. Similarly, in our fourth method, termed run split, entire runs of frames were independently drawn. For the fifth method, termed session split, we used one entire session for training and a second session for test. This method is similar to measures of between-session classifier accuracy used previously (**?**). Table 1 summarizes these training and test set split methods and the mean temporal delay between the training and test data, which varies from $\sim$30 s (frame split) to several months (session split).

For the frame and block splits, the classifier performances were estimated using 10-fold cross-validation. That is, the dataset was randomly split into the training and test sets 10 times and a classifier is trained on each split. The classifier's performance is then estimated as the average of its performance across all 10 splits. For the half-run, run, and session splits, only 8, 4, and 2 unique splits are possible, respectively, due to the much smaller number

of runs and sessions per subject. Therefore, only 8-fold, 4-fold, and 2-fold cross-validation were employed for estimating classifier performance on these splits.

Classifier performance has been described by a number of metrics [refs]. Here, the performance of each classifier was characterized by its micro-averaged F-measure (**?**), which we calculate as follows. Analysis begins by calculating two simple accuracy metrics. The first is precision $= tp/(tp+fp)$, where tp is the number of true positives and fp is the number of false positives. The second metric is recall $= tp/(tp+fn)$, where fn is the number of false negatives. Intuitively, precision is the percentage of correctly classified examples out of all examples classified as a particular label, while recall is the percentage of correctly classified examples out of all examples that actually belong to a particular label. The F-measure for a single class is defined to be:

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{6}$$

The F-measure is a more robust measure of the performance of a classifier than either precision or recall alone. For example, if a binary classifier labeled everything as positive then the recall would be perfect but the precision would be at chance levels. On the other hand, if the binary classifier only labeled high-confidence examples as positive then precision would be high but recall would be low. The F-measure combines these two aspects of reliability.

These metrics can be generalized for multiple classes by summing true positive, false positive, and false negative counts across all classes.

$$\text{precision}_{avg} = \frac{\sum_i^M tp_i}{\sum_i^M (tp_i + fp_i)} \tag{7}$$

$$\text{recall}_{avg} = \frac{\sum_i^M tp_i}{\sum_i^M (tp_i + fn_i)} \tag{8}$$

$$F_{avg} = 2 \cdot \frac{\text{precision}_{avg} \cdot \text{recall}_{avg}}{\text{precision}_{avg} + \text{recall}_{avg}} \tag{9}$$

Where M is the number of classes. [You used M for harmonic numbers before; use some other symbol] This is known in the literature as the micro-averaged F-measure. It should be noted that in a symmetric multi-class scheme such as ours, the micro-averaged precision, recall, and F-measure will all be identical. Therefore, we will only be reporting a single accuracy

metric which we will refer to as the multi-class F-measure. [we could just as easily call this the multi-class precision/recall if we wanted to so long as we are consistent]

Another tool for examining classifier performance is the confusion matrix. If $\mathbf{C}$ is a confusion matrix, then the value of C_{ij} is equal to the number examples of class i that were classified as class j. Therefore, values along the diagonal of a confusion matrix correspond to correct classifications while other values correspond to incorrect classifications. The confusion matrix also simplifies calculating precision and recall for each class. The value of C_{ii} divided by the sum of all values along row i is the recall of the i^{th} class. Similarly, the value of C_{jj} divided by the sum of all values along column j is the precision of the j^{th} class. Finally, the micro-averaged F-measure can be calculated by dividing the sum along the diagonal, or the trace, by the sum of the entire matrix.

2.7. Sensitivity Analysis

Non-linear multi-variate machine-learning classifiers can tell us whether the time-series data from a subset of human brain voxels is discriminative with respect to the task being predicted. However, these results do not show which voxels in the large group were actually important for that discrimination. This is important for localizing functions in the brain. One existing technique is to train machine learning classifiers on small localized areas in the brain and use their performance as a measure of the strength of the function in question in that area. While this technique is effective for simple highly localized functions, the results are less clear when the function is sparsely distributed over the brain. No one region may contain enough information for accurate predictions.

To overcome this limitation, we have trained our classifiers on large regions of the brain and used sensitivity analysis to attempt to tease out the sparsely distributed voxels that are relevant for task discrimination. Specifically, we calculate the sensitivity, or magnitude of change, of the output of the classifier with respect to a change in each voxel. In feed forward neural networks, this problem has been well explored (**?**). Let $\mathbf{o}$ be the vector of outputs and $\mathbf{x}$ be the vector of inputs. Then the sensitivity of output k to input i is defined by:

$$S_{ki} = \frac{\delta o_k}{\delta x_i} \tag{10}$$

Or simply, the partial derivative of the output with respect to the input. If we let $\mathbf{w}$ be the weight matrix from the hidden layer to the output layer and $\mathbf{v}$ be the weight matrix from the input layer to the hidden layer then the partial derivative can be expressed as follows:

$$\frac{\delta o_k}{\delta x_i} = o'_k \sum_{j=1}^{J} w_{kj} y'_j v_{ji} \tag{11}$$

Where J is the total number of hidden neurons, o'_k is the value of the derivative of the activation function at output k, and y'_j is the value of the derivative of the activation function at hidden neuron j. Finally, the entire sensitivity matrix can be expressed in matrix notation as:

$$\mathbf{S} = \mathbf{O}' \times \mathbf{W} \times \mathbf{Y}' \times \mathbf{V} \tag{12}$$

Where

$$\mathbf{O}' = diag(o'_1,\ o'_2,\ \cdots,\ o'_K) \tag{13}$$

$$\mathbf{Y}' = diag(y'_1,\ y'_2,\ \cdots,\ y'_K) \tag{14}$$

However, because the transfer functions are non-linear they can only be evaluated for specific input values. Therefore, we calculate the average sensitivity matrix across all input vectors.

$$\mathbf{S}_{avg} = \sqrt{\frac{\sum_{n=1}^{N} \left(\mathbf{S}^n\right)^2}{N}} \tag{15}$$

Where N is the number of input vectors. The magnitude is squared to avoid problems with positive and negative sensitivities canceling out when averaging. The average of the absolute value of sensitivities could also be employed. This still gives a sensitivity value for each voxel with respect to every output, whereas it is useful to have a measure of the sensitivity of a voxel with respect to any output. To calculate this number we simply take the maximum sensitivity of each voxel across all outputs.

$$\Phi_i = \max_{k=1\ldots K} S_{ki,\ avg} \tag{16}$$

This sensitivity can now be projected back into the volume anatomy space to create an activation map. Further, this sensitivity can also be used to reduce the dimensionality of the machine learning classifier by retraining the algorithms using only the voxels that had a sensitivity over some threshold.

	Frame		Block		Half-Run		Run		Session	
Subject	SVM	NN	SVM	NN	SVM	NN	SVM	NN	SVM	NN
A	0.71	0.51	0.48	0.46	0.51	0.39	0.51	0.39	0.37	0.30
B	0.57	0.38	0.39	0.25	0.45	0.33	0.48	0.27	0.30	0.30
C	0.64	0.45	0.44	0.37	0.50	0.38	0.48	0.35	0.22	0.23
D	0.79	0.78	0.60	0.63	0.61	0.61	0.62	0.45	0.51	0.48
E	0.61	0.53	0.47	0.46	0.48	0.48	0.47	0.47	0.26	0.24
Average	0.66	0.53	0.47	0.43	0.51	0.44	0.51	0.39	0.33	0.31

Table 2: The multi-class F-measures of the linear SVM and the feed forward neural network after cross-validation for all 5 subjects and all 5 training and test split methods.

3. Results

Table 2 contains the cross-validated F-measures of the linear SVM and feed forward neural networks for all 5 subjects and all 5 training and test split methods. There is some variation of classifier performance between subjects, but in all cases the performance is well above chance.

It is evident that as the average temporal distance between data in the training and test sets increases, the estimated performance of the classifiers decreases. This relationship is plotted in figure 4.

The average confusion matrix presented in figure 5 gives a more intuitive look at the performance of the classifier. From this matrix we can see that the classifiers are much better at detecting the presence of a single character than any other count. In fact, there are almost no cases of confusion between 1 and 2 characters. Apparently, these two situations evoke very different responses in the brain. The rest of the character counts are distinguished with relatively equal accuracy. However, there is a slight tendency to mis-classify the 6 character presentation. It should also be noted 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. For example, the machine learning algorithm classified a frame as containing 4 characters when it only contained 3 characters. The F-measure does not take the cardinality of the classes into account and considers mislabeling 1 character as 2 equivalent to mislabeling 1 character as 6 characters. A soft measure of accuracy that takes the cardinality of the classes into consider-

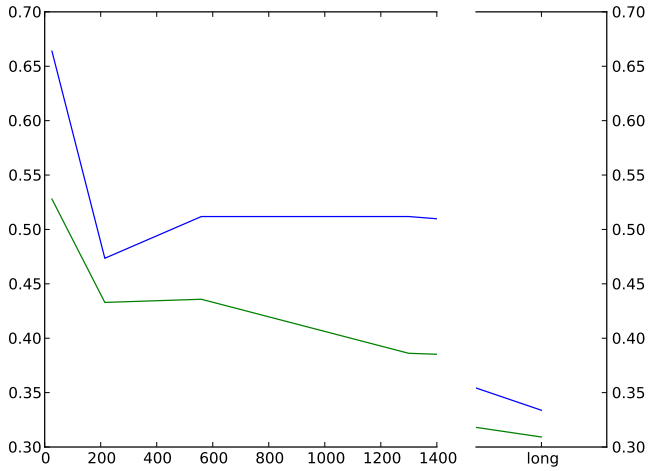


Figure 4: The average performance of the SVM and neural network classifiers plotted against the average temporal distance between data in the training and test sets.

ation may be useful but the results are harder to interpret. A confusion matrix gives us the advantages of the soft measure while providing an easily interpretable look into the performance of the classifiers.

The sensitivity analysis of the feed forward neural network projected back into the anatomy volume is presented in figure **??**. [David: We need to make come comments on this figure related to neuroscience/brain mapping. We should look at the data to make sure this is true, but since we are still employing the same harmonic analysis the activated regions should be very similar to the original report.] After aligning all of the subjects to the MNI template **??** and combining these activation maps we found that the majority of activation took place in [need to do this analysis still]. This combined activation map is presented in figure 6.

To better explore the relationship between sensitivity and classifier performance we plotted the performance of the classifier when trained on only a subset of the input voxels as determined by a minimum sensitivity cut off. Figure 7 shows this plot on top of the histogram of sensitivity values. Interestingly, the performance of the classifier is not significantly affected until a majority of the voxels have been removed. This would indicate that

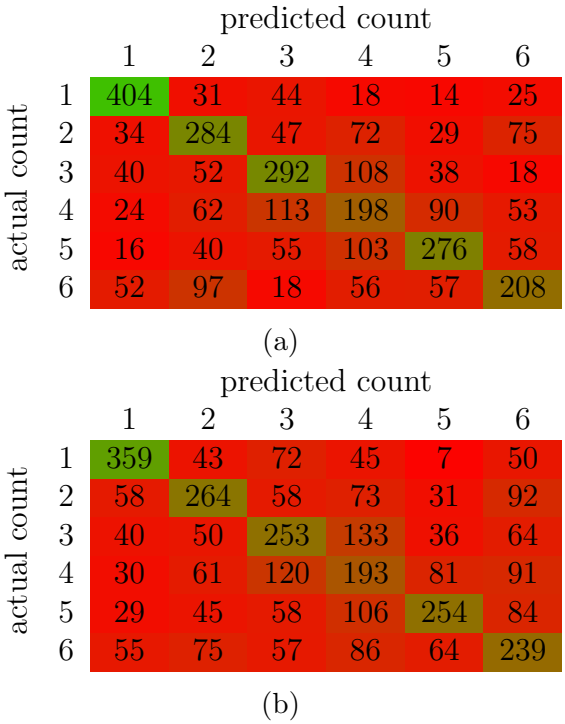
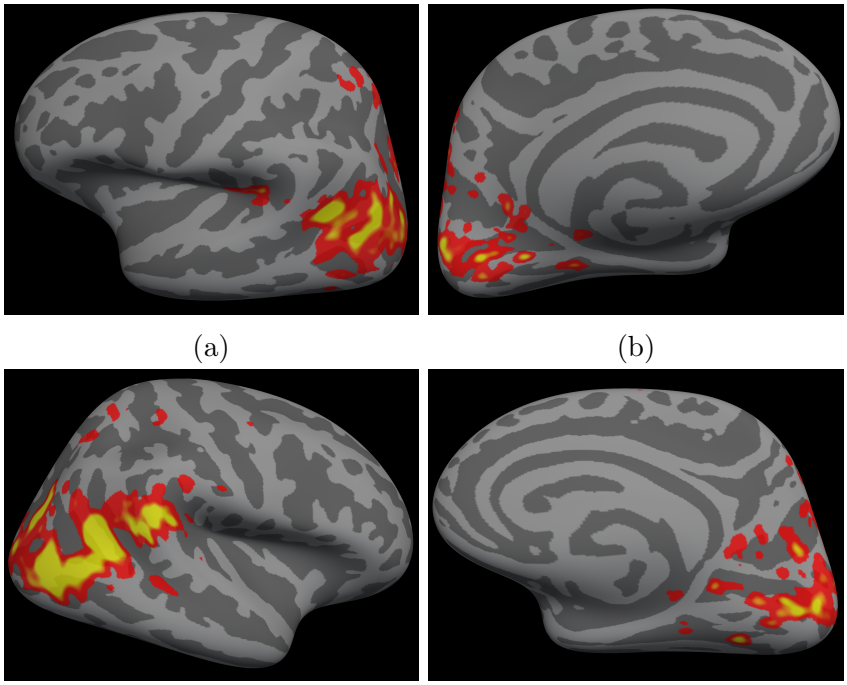


Figure 5: The average confusion matrices for the (a) SVM and (b) NN across all subjects using the block split.



(a) (b)

(c) (d)

Figure 6: The sensitivity analysis averaged across all subjects mapped onto the slightly inflated MNI template brain.

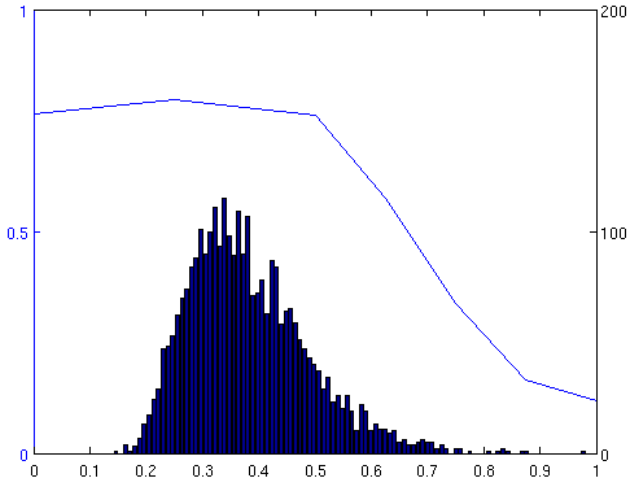


Figure 7: A histogram of the sensitivity analysis values and a plot of the feed forward neural network F_1 score when the inputs are pruned at a particular sensitivity value.

either only a small number of voxels are relevant for classification or that the information is highly redundant between voxels. In actuality, it is likely a combination of the two. The harmonic analysis will select some voxels that only appeared to covary with the stimulus presentation by chance. These voxels will be assigned very low sensitivity values by the trained classifiers. However, other voxels may actually covary with the stimulus but their patterns of activation may not be highly discriminative with respect to character count. These voxels will also be assigned low sensitivity values by the trained classifiers. Therefore, high sensitivity is sufficient but not necessary for the localization of a function in a particular region.

4. Discussion

These variations between subjects could be the result of differences in age, general cognitive state, or simply how much attention the subject was paying to the stimulus during the scan.

This could be a result of 6 characters being too much for the subject to consider individually. Interestingly, the confusion is not often between 5 and 6 characters, but rather between 1 or 2 and 6 characters. This would seem

to indicate that when the number of characters grows too large, they are interpreted as a single unit.

The reported results are well above chance, indicating there is useful information about character number in the BOLD signal. The sensitivity analysis indicates that no single region of the brain is responsible for counting characters, and there is not a simple linear relationship between magnitude of activation and cardinality. Rather, it is a complex pattern of distributed activation requiring machine learning methods to capture the stimulus-response relationship.

Earlier, we presented this new trend in brain sate classification as a departure from traditional fMRI experiments which seek to identify the purpose or function of particular brain regions. However, it is important to note that through sensitivity analysis these machine learning classifiers can be re-purposed for just that goal. If a region of the brain is highly important for accurately predicting the presence of a particular stimulus then it logically follows that that region must somehow be involved in the processing of that stimulus. Furthermore, multi-voxel non-linear machine learning classifiers can potentially identify much more complex interactions between brain regions than the simple GLM.