Identifying virtual-world stimulus information from fMRI data

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

When conducting research using fMRI to analyze how the brain responds to visual stimuli, static images have been the dominant choice. Such work has generally attempted to focus on a single aspect of the visual world, such as contrast, color, or object class, and create stimuli that isolate that aspect. While this has proven to be a successful approach, it does not mimic the dynamically changing environment in which the primate brain has evolved. It is likely that these richer, more dynamic stimuli will evoke a more complex network of brain responses. Here, we test the feasibility of extracting complex information from more natural stimuli by employing virtual worlds, similar to what is commonly used in video games, to generate our stimuli. Virtual worlds have two advantages over videos of the real world: 1) the stimulus designer has complete control over the stimuli 2) the subject can interact with the virtual world by making a sequence of decisions while in the scanner.

To analyze the brain response to our more complex stimuli, we employ two multi-voxel pattern recognition approaches that identify brain response patterns using a spatially distributed subset of stimulus-responsive voxels at each time point. This is in contrast to utlizing a General Linear Model (GLM), which treats each voxel independently as a time-series of responses over an extended period of time. Processing the entire brain over the course of a scanning session should allow for capturing the complex and anatomically distributed interplay of neural processing while permitting the stimulus to continually vary throughout the scan.

Multi-voxel pattern recognition applied to fMRI involves the use of statistical machine learning algorithms, such as artificial neural networks and support vector machines(**???**). When applying these algorithms, the stimulus typically is organized into multiple categories or classifications (e.g. houses, faces, tools etc.). The machine learning algorihms use the fMRI data to synthesize a model of the classes. These models map a small set of fMRI time points to a set of weights, one for each category. Each set of weights provides a prediction that the time sample corresponds to the brain response to stimuli from that category. A simple means of assessing the accuracy of the models is to compare the highest predicted class to the correct class for the stimuli being presented during that period of time.

The preceeding paragraph needs some expansion. Give a brief summary of each of the cited experiments and their results. Expand on the accuracy methods a bit and give references.

Our experiment was structured to identify those parts of the brain that encode the number of human-like characters realistically presented in a dynamic virtual world. Unlike previous experiments that examined the neural correlates of object counting or number evaluation (**?**), we used a passive-viewing stimulus paradigm and utilized machine-learning methods rather than a GLM analysis. We synthesized both artifical neural networks and support vector machines to classify each time point with respect to the number of characters. We also tested the temporal invariance of the classifers by examining their performance as a function of the separation in time between the data used to synthesize the classifier and the data used to test performance of the classifier. Classification accuracy was typically 3-4 times better than chance, over a range of subjects and machine-learning techniques. To permit anatomical evaluation of the results, the spatial distribution of classification-relevant brain activity was then obtained by "inverting" the classifiers using a sensitivity analysis, yielding a map that is computed very differently from GLM techniques.

2. Methods

2.1. Subjects

Five adult males, with normal or corrected-to-normal vision, participated in the experiments. All subjects participated in two 1-hour-duration MRI sessions with 36

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

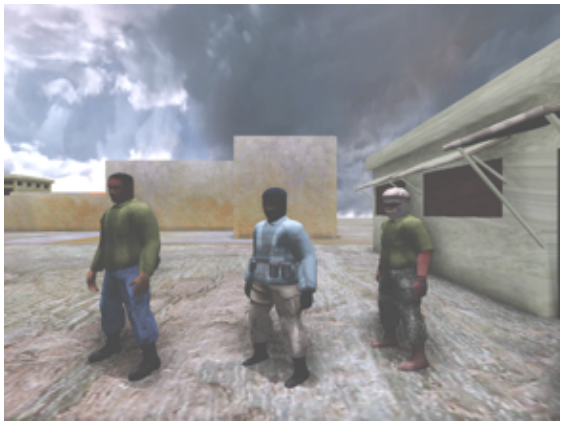
minutes of fMRI acquisiton. An additional session was performed to measure a high-resolution structural anatomy. Informed consent was obtained from all subjects.

2.2. Stimulus

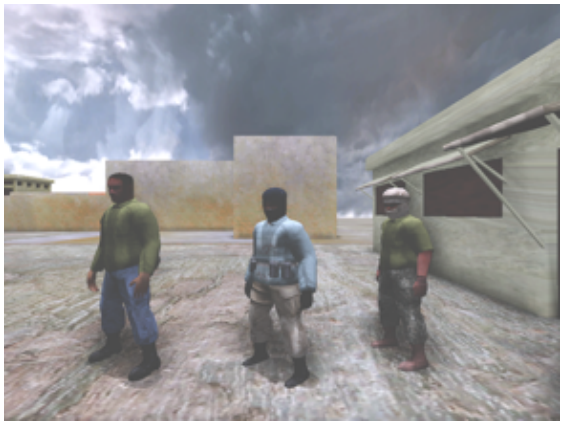
Given our long-term interest in PTSD, we created a virtual town intended to mirror the kinds of real-world settings encountered by our military forces (see Fig. 1). The graphics software we used was the Unreal Development Kit, a free-to-use game engine by Epic Games (**?**). Virtual characters representing friendly forces and hostile combatants were situated at a variety of locations thoughout the virtual town. An animation was synthezied in real-time, during which the camera moves at eye level throughout the town, stopping at locations where a varying number of characters, either friends or foes, where situated (see Fig. 2a and 2b). Both the camera and the characters were animated at all times (no static imagery), but we choose not to include subject interaction (passive viewing) for this first experiment using virtual worlds.

Because evaluation of entire fMRI volume datasets is compuationally intensive, we used a block design to enable selection of all voxels responsive to the character presentation. To provide a realistic scenario, blocks consisted of alternations of traveling through an unpopulated town for 15 sec, followed by 15-sec periods of character presentations. Thus, the camera moves for 15 seconds at a fairly rapid rate through the virtual environment with no characters present, then slows and hovers for 15 seconds around a group of animated characters while the camera contines to pan and rotate.

Scanning sessions included 5–6 six-min-duration runs, each containing 12 alternations between moving through the virtual environment and character presentations. The number of characters presented in each block varied 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. This random ordering was generated once and repeated for each subject. There were 4 locations in the town used for presenting characters, and a single run made 3 loops through the town. The order of presentation was chosed such that configurations with the same number of characters varied in location within the town; compare figures 2a and 2b. A single type of character (friends or foes) was presented in each run. Character type was randomly shuffled between runs.



(a)



(b)

Figure 2: (a) An example frame from the stimulus while characters are being presented. (b) 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. A high-order shim was performed to improve field homogeneity.

Figure 1: An example frame from the stimulus where the camera is travel through the virtual envronment with no charcters presented.

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) fast RF-spoiled GRASS (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-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. 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.

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

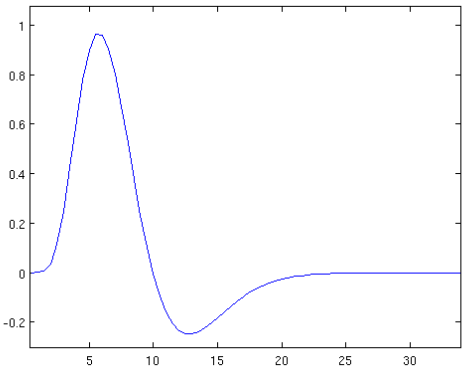
Where $\mathrm{SNR}(f)$ is the signal to noise ratio $\frac{|X(f)|}{|N(f)|}$.

In fMRI, $y(t)$ is the recorded BOLD signal, $h(t)$ is the hemodynamic response function, and $x(t)$ is the neural 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 $\mathrm{SNR}(f) = 1$ 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.

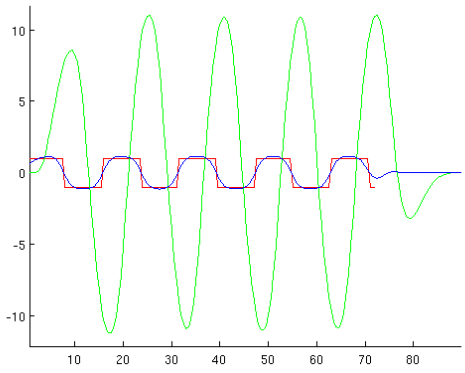
The high-resolution reference anatomies were segmented using the FreeSurfer analysis package (**?**) to create approximate masks of the gray matter in each subject, as well as a surface model useful for visualization of the results.

2.5. Dimensionality reduction

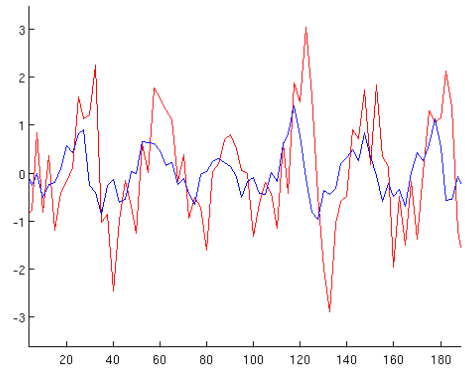
Each voxel can be thought of as corresponding to a separate dimension of a very high dimenional space. In our case, the number of voxels/dimensions was typically



(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.

80 x 80 x 40 = 256,000. An important first step performed for most all machine learning algorithms is to reduce the total amount of data fed into the algorithms. This not only speeds up the algorithm, but if done carefully, can improve the performance of the resulting classifiers.

In our case, we select voxels that are driven most strongly by the 30-sec duty cycle of our block design using a harmonic power analysis technique. 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 can accept* $M = 4$*. Using* P_h*, we then selected a particular number* N *(typically 2000) 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 (***?***). However, as a check we built classifiers to distinguish between time points with and without characters, and their performance was consistently ¿95*

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. Generally, the performance of machine learning algorithms is estimated by splitting the available data into a training set and a test set and then evaluating the performance of the algorithm on the test set after "training" the model using only the training set (***?***). The idea is that the algorithm's performance on the test set is an estimate of the classifier's performance on new data gathered at a different time. Neural network machine learning algorithms, further subdivide the data into training, validation, and test sets. The algorithm is trained on the training set and after each iteration is evaluated with the validation set. When the performance on validation set stops improving, the algorithm stops iterating; this procedure avoids so-called "over fitting" of the data. Performance of the resulting classifier is then evaluated on the test set in order to estimate its future performance (***?***).*

The above paragraph needs a couple of elementary references on machine-learning approaches.

*We estimated the performance of our algorithms using a cross-fold approach (**?**) where the data is split into "folds" that are used to form multiple training and test sets. To maximize our temporal resolution, each "frame" (time point) was treated as a separate distinct data value, rather than performing averaging across the 15-sec block as has been done in previous work (**?**). Since each frame is 2.5 sec., averaging would have replaced 15sec/2.5sec. = 6 samples with a single averaged sample, unnecessarily reducing the temporal resolution by a factor of 6.*

*Previous studies have discussed issues with optimistically biased performance estimates due to temporal correlations violating independence assumptions between training and test set samples (**?**). The slow speed of the fMRI hemodynamic response clearly introduces temporal correlations on time scales ¡15 sec. For our data, artificial correlations will occurs when the random sampling that divides the data into the different sets yields training and test samples from the same (15 sec.) block.*

*This raises a more general question: what is the relationship between performance estimates and temporal correlation? To explore this, we estimated classifier performance using 5 different methods for splitting the training and test examples in order to vary the average temporal distance between frames in the two sets. The first four methods deal only with frames from a particular session. In the first method, which we will refer to as "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.We expect this method to exhibit artificially high performance because of the aforementioned temporal correlation of the slow fMRI data. In the second method, which we will refer to as" block split", sets of frames corresponding to the 15 sec. blocks were independently drawn into the training and test sets. In our third method, we exploit the fact that each 6 min run cycles through all combinations twice, but in a different order the second time. Accordingly, sets of frames were independently drawn from these half-runs, refered to as "half-run split". In our fourth method, termed "run split", entire 6-min runs of frames were independently drawn. And finally, 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 in (**?**). Table 1 summarizes these training and test set split methods and the mean temporal distance between the training and test data, which varies from ∼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,

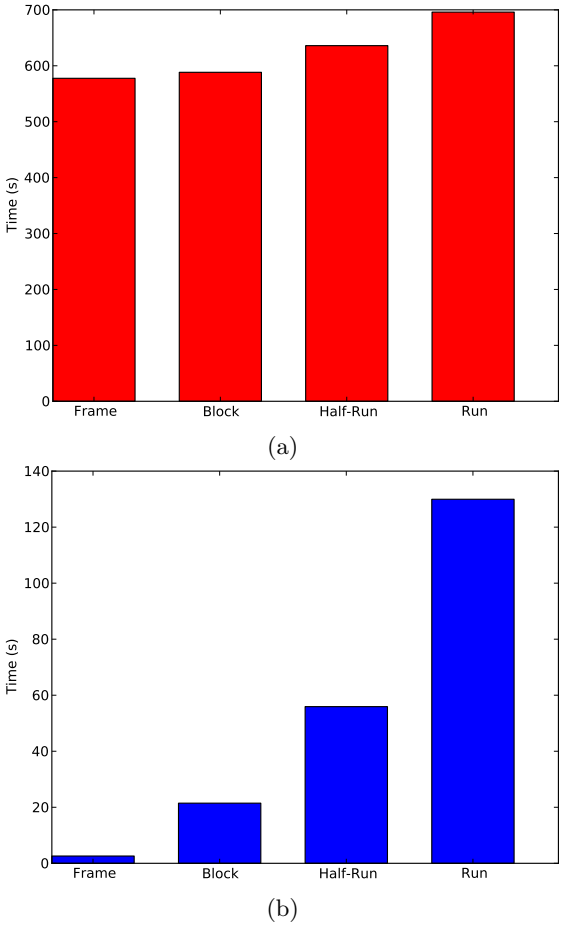


Figure 4: The (a) mean delay and (b) minimum mean delay between examples in the train and test set for four of the split methods. The session split method is excluded because both the mean and minimum delay are much larger than the other splits and the delays vary significantly between subjects.

Split	Mean Delay (s)	Mean Minimum Delay (s)
Frame	578	2.6
Block	588	21
Half-Run	636	56
Run	696	130

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

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.

We define the classifier's overall performance or accuracy to be the probability that it will correctly classify a previously unseen time-slice. As such, this measure of performance can only be estimated by dividing the number of correctly classified examples in the test set by the total number of examples in the test set. We can also discuss classifier performance with respect to a single class. Single class performance is traditionally measured along two axes: precision *and* recall. Precision *with respect to class c is the probability that a previously unseen data point was classified correctly given that it was classified as c.* Recall *with respect to class c is the probability that a previously unseen data point was classified correctly given that it is actually a member of c. Again, these measures can only be estimated from the test set.* precision $= tp/(tp + fp)$*, where tp is the number of true positives and fp is the number of false positives.* recall $= tp/(tp + fn)$*, where fn is the number of false negatives.*

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 estimating precision and recall for each class. The value of* C_{ii} *divided by the sum of all values along row i is the estimate for 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 estimate for the precision of the* j^{th} *class. Finally, overall classifier performance can be estimated 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 information 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 all stimulus-responsive regions of the brain and used a neural-network sensitivity analysis to display the spatially distributed set of voxels that are the most important for identifying the classes. Specifically, we calculate the sensitivity, or magnitude of change, of the output of the classifier with respect to a change in each voxel (***?***). 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:*

$$S_{ki} = \frac{\delta o_k}{\delta x_i} \tag{6}$$

which is 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:*

$$\frac{\delta o_k}{\delta x_i} = o'_k \sum_{j=1}^{J} w_{kj} y'_j v_{ji} \tag{7}$$

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 neuronj. Finally, the entire sensitivity matrix can be expressed in matrix notation as:*

$$\mathbf{S} = \mathbf{O}' \times \mathbf{W} \times \mathbf{Y}' \times \mathbf{V} \tag{8}$$

Where

$$\mathbf{O}' = diag(o'_1,\ o'_2,\ \cdots,\ o'_K) \tag{9}$$

$$\mathbf{Y}' = diag(y'_1,\ y'_2,\ \cdots,\ y'_K) \tag{10}$$

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.

$$\mathbf{S}_{avg} = \sqrt{\frac{\sum_{n=1}^{N} (\mathbf{S}^n)^2}{N}} \tag{11}$$

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.*

$$\Phi_i = \max_{k=1\ldots K} S_{ki,\ avg} \tag{12}$$

This sensitivity can now be projected back into the volume anatomy space to create a sensitivity map that reflects the relative incremental importance of each voxel's response to the classification decision. Further, we used this sensitivity analysis to further reduce the dimensionality of the neural network by retraining the algorithm using only the voxels that had a sensitivity over some threshold. This threshold was determined by iteratively retraining the neural network

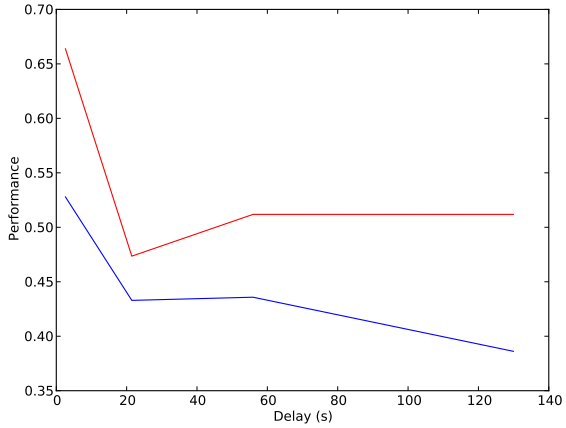


Figure 5: The average performance of the SVM and neural network classifiers plotted against the average temporal delay between data in the training and test sets.

with successively higher thresholds until the performance of the classifier began to degrade.

*To get a better idea where, on average, the neural network was finding useful information, we aligned each subject's sensitivity map to the MNI template brain (**?**) using FreeSurfer (**?**). After alignment, we averaged the sensitivity maps together, projected the results onto the cortical surface, and blurred along the surface using a 5 -mm full-width half-maximum (FWHM) Gaussian kernel.*

3. Results

Table 2 contains the cross-validated performance estimates 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 significantly ($p < 0.001$) above chance.

It is evident that, as the average temporal delay between data in the training and test sets increases, the estimated performance of the classifiers decreases. This relationship is plotted in figure 5.

The average confusion matrix presented in figure 6 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, except for a slightly larger 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 estimated performance does

predicted count

actual count	1	2	3	4	5	6
1	404	31	44	18	14	25
2	34	284	47	72	29	75
3	40	52	292	108	38	18
4	24	62	113	198	90	53
5	16	40	55	103	276	58
6	52	97	18	56	57	208

(a)

predicted count

actual count	1	2	3	4	5	6
1	359	43	72	45	7	50
2	58	264	58	73	31	92
3	40	50	253	133	36	64
4	30	61	120	193	81	91
5	29	45	58	106	254	84
6	55	75	57	86	64	239

(b)

Figure 6: The average confusion matrices for the (a) SVM and (b) NN across all subjects using the block split.

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 consideration 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.

*Sensitivity maps show a preponderance of brain activity in posterior parietal lobe, lateral occipital areas, and dorsal early extrastriate visual areas **??**. Individual subjects also displayed small regions of high sensitivity in idiosyncratic portions of temporal frontal cortex. After aligning all of the subjects to the MNI template **??** and combining these sensitivity maps, we find that classifier decisions were heavily dependent on activity in occipital and parietal lobes 7. Classifier sensitivity was most affect by, in order of importance, dorsal parietal lobe, cuneus, and lateral occipital regions **??**.*

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 8 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 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

	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 performance estimates of the linear SVM and the feedforward neural network after cross-validation for all 5 subjects and all 5 training and test split methods.

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.

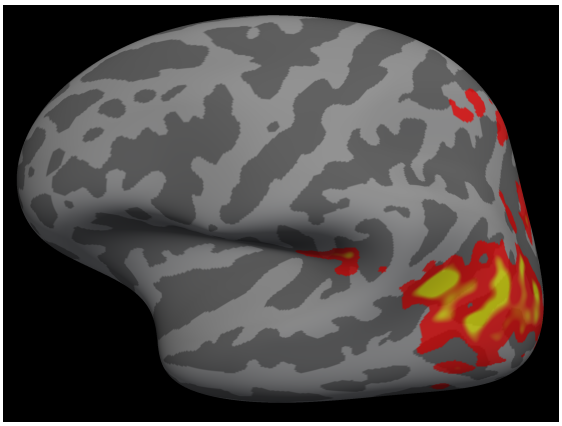
4. Discussion

Although both classifiers performed at well above chance levels for all subjects, the average performance on individual subjects varied significantly. These variations 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. The lack of a task during our stimulus makes these variations difficult to interpret since attention and cognitive state are not well controlled. In future experiments, we plan to introduce a difficult task for the subjects to perform and we expect to see the average performance increase and the variation between subjects decrease. However, the fact that the classifiers could still perform at well above chance levels without a focused task increases our confidence in the generalizability of the results of machine learning classifiers applied to fMRI data.

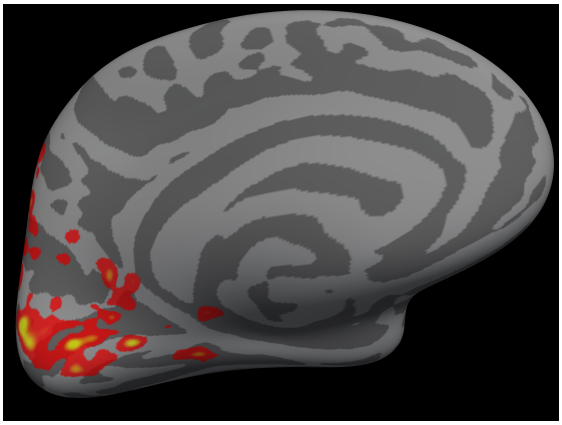
This could be a result of 6 characters being too much for the subject to consider individually. Interestingly, the confusion is not often between 5 and 6 characters, but rather between 1 or 2 and 6 characters. This would seem to indicate that when the number of characters grows too large, they are interpreted 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.

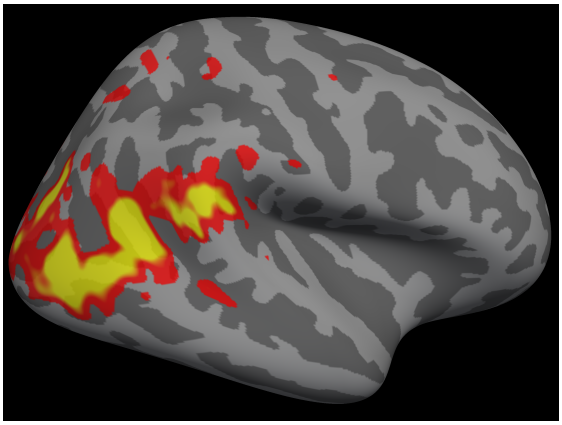
Earlier, we presented this new trend in brain sate classification as a departure from traditional fMRI experiments which seek to identify the purpose or function of particular brain regions. However, it is important to note that through sensitivity analysis these machine learning classifiers can be re-purposed 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.



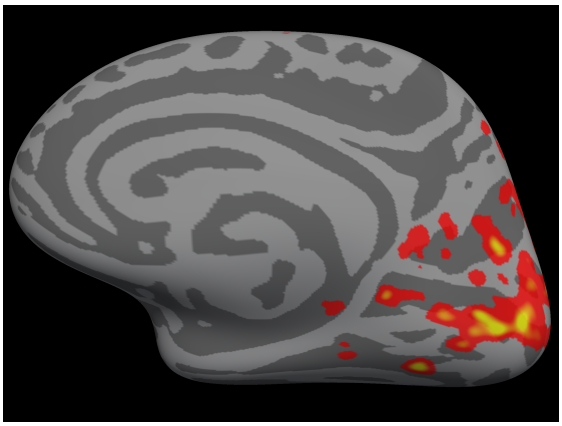
(a)



(b)



(c)



(d)

Figure 7: The sensitivity analysis agit veraged across all subjects mapped onto the slightly inflated MNI template brain.

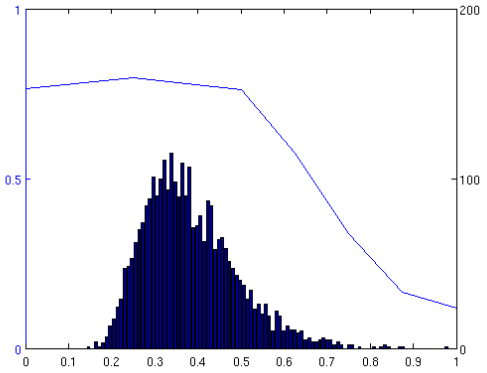


Figure 8: A histogram of the sensitivity analysis values and a plot of the feed forward neural network F_1 score when the inputs are pruned at a particular sensitivity value.