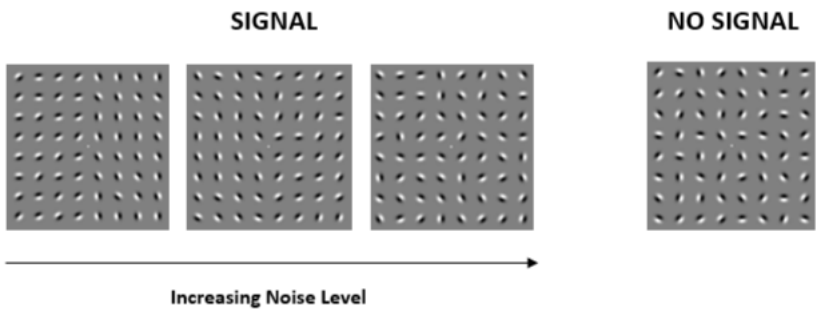
3. Methodology

3.1 Data Structure and Generation

Our datasets were comprised of 768x768 images each containing an 8 by 8 array of Gabor patches on a grey background. A MATLAB script built up these images by creating a Gabor patch in each of the 64 grid blocks by applying a ramp function at the input orientation angle and then applying a Gaussian blur grating. In the no signal case, a pure noise orientation array was passed into the function for creating Gabor patches, wheras in the signal case some noise was added into a pure signal image as can be seen below:



The amount of noise chosen to be added to a signal image was chosen by trial and error, with the maximum amount of noise chosen that the subject could just distinguish between the signal and no signal images accurately.

5,000 images were generated this way for each classification and labeled with their ground truth response - whether a signal or not was present. This would allow us to train a CNN against the ground truth and compare the structure of that CNN to one trained on user responses of if there was a signal present or not. User generated data was also collected through a MATLAB script, which flashed the test image quickly for 0.5s in between two gray images, then the user was asked to report whether they thought the signal was present or not in the image. The image was quickly flashed in order to ensure that only low-level V1 processing would have occured. If the test subject was able to look at the image for a longer period, other factors such as higher visual cortex layers or general thinking could have come into play, reducing the effectiveness of a model aimed at modelling V1 texture detection. These images were labelled with the user response, with the actual presence or not also recorded, allowing for a confusion matrix to be produced and to track false positives and negatives from both the user and our CNN.

3.1.1 CNN Structure

In order to implement our CNN, we used the TensorFlow library as it has better visualisation features than other deep learning proejcts like PyTorch. We also found that TensorFlow was generally chosen more often in the literature. Datasets were created using the image files generated in MATLAB, rescaled down to 256x256 in order to reduce training time as no information was lost at this level. At this early stage of model exploration, we used only 40% of the generated training data to speed up training time.

We initially intended to generate two CNNs - one trained upon the ground truth signal

presence data and one trained upon user detected presence of signal or not. This user generated data was obtained by one of our research members labeling the images as described above. However, we found that the same CNN architecture worked well for both the user generated and the groudn truth dataset. While we will eventually want to get a wider range of user data in order to reduce any biases, we expected that at this early stage creating training data from only one individual would not seriously affect the way we would need to architect our CNN model. We chose a 80%/20% training/validation split as reccomended in the TensorFlow API documentation.

3.1.2 Shallow CNN structures

We started investigating potential CNNs with a shallow architecture in order to simplify our network and speed up training. We started off with a network with two convolutional layers, then a fully connected dense layer connected to a single output neuron with sigmoidal activation. Convolutional layers used a 3x3 kernel with the ReLU activation function to introduce a nonlinearity. We ran our models for up to 20 epochs, however we observed during testing that losses and accuracies normally settled at around 8 to 10 epochs.

3.1.3 Structure modifications

We varied network hyperparameters such as the number of neurons of each layer, layer number, and kernel sizes in order to compare the effects on performance. One issue we ran into was that our model was overfitting, as around at 20 epochs training accuracy was exceeding 95% but validation accuracy had plateaud at 82%. Adding 20% dropout layer right beore the sigmoid clsasifier helped as in a dropout layer some of the connection weights are randomly set to zero to reduce overfitting.

We also used data augmentation by flipping our images horizontally and vertically. This helps to reduce overfitting by making sure our model is more generalised to a wider variety of training data, and effectively increases the size of the training set without requiring further user data to be collected.

We found that a similar architecture worked well for correctly identifying the presence of signal or noise against the ground truth or a subject classification. Our network, shown below gave us 94% validation accuracy on the subject dataset and 91% on the ground truth dataset.

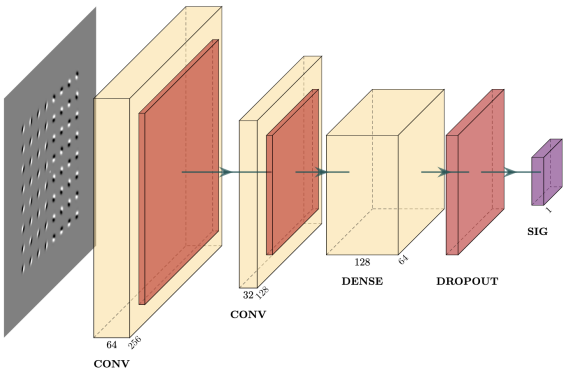


Figure 1 Shallow CNN architecture used to classify both user generated and ground truth data. Convolutional layers used a 3x3 kernel with ReLU activation, and the dropout was set to 20%

4. Results

Table 1 Subject classification

		Ground truth	
		No Signal	Signal
Prediction	No Signal	86.5%	13.5%
	Signal	15.8%	84.2%

Table 2 CNN classification

		Ground truth	
		No Signal	Signal
Prediction	No Signal	84.7%	15.3%
	Signal	17.2%	82.8%

Table 3 Confusion matrices for classification by the subject and classification from our CNN

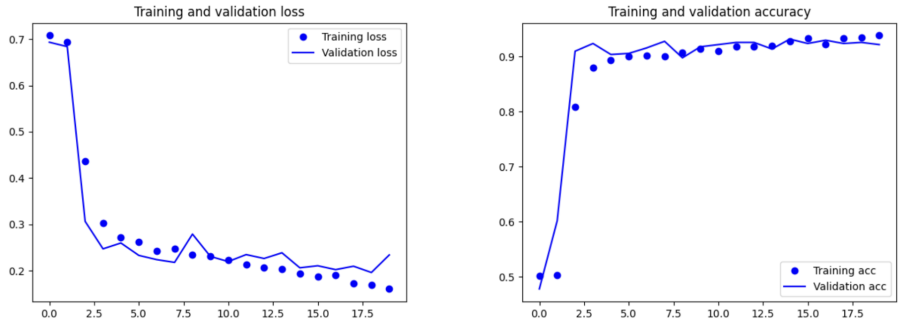


Figure 2 Training and validation losses and accuracy over epochs of our final model at this stage with dropout layers and data augmentation

Using the models trained on both sets of data, we produced accuracy over epoch graphs for our final model as in Figure 2 and confusion matrices detailing the false positive and negative rate on each dataset. There is still some minor overfitting as the training set accuracy slightly eclipses the validation dataset accuracy, but this was still substantially reduced from the overfitting level before adding a dropout layer comparing against the model without them in Figure 3.

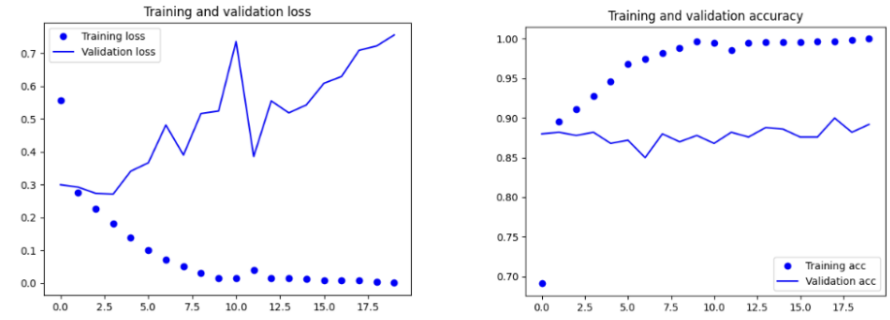


Figure 3 Training and validation losses and accuracy over epochs of our model without dropout or data augmentation, showing strong overfitting.

Our network was actually better at classifying data labelled by a subject than the ground truth data source. One reason this could be is that at the chosen noise added

to a signal could still completely scramble the image as a result of random chance. This puts a upper bound on the accuracy of our network as the signal has been erased by the noise. However, when a human subject classifies the same image it is likely that they would class it as having no signal, and as our CNN is meant to model human visual cortex behaviour it will also generally classify it as having no signal.

Our network still needs tinkering to investigate what improvements we could make to further increase our accuracy. We could experiment with a deeper CNN, and adjust hyperparameters or other forms of regularisation further. Running a parametric study is one way that we could identify what would be the best hyperparameters in order to improve our neural network performance. Additionally, we need to gather a proper training set form a variety of subjects in order to remove any bias that our single subject added to the test sample.

4.1 Next Steps

After further refining our network and improving our test set, our next steps would be to ensure that the classification accuracy is as close to the test subjects as possible, then trying to link our created neural network model to underlying biology of V1