Bachelor thesis in Artificial Intelligence

Mapping of convolutional neural network activation maps on the visual cortex using a Bayesian linear framework

Thesis by

Leonieke van den Bulk

s4238516

Supervised by

Sanne Schoenmakers[1]

and

Marcel van Gerven[1]

[1] Donders Institute for Brain, Cognition and Behaviour

Department of Psychology and Artificial Intelligence

Radboud University Nijmegen

July 2016

Abstract

Decoding the brain is slowly starting to become a reality. Recently, new computational models have made it possible to make reconstructions of perceived images from BOLD responses in the visual cortex. A linear Gaussian framework was used by a previous study to decode functional magnetic resonance imaging data from subjects who were presented with images of handwritten characters. Here we expand this research by taking convolutional neural network activation maps instead of images of handwritten characters as the encoding and decoding stimuli as the different layers in the network appear to be good models for the different layers in the visual cortex. In contrast to former studies, a uniform prior is used to let the decoding be driven by the found encoding only. This approach results in good classification accuracies that become better as the convolutional layer becomes deeper and are more robust than previously found results.

1 Introduction

Decoding the brain has always been of great importance to cognitive neuroscience as it can give us new insights on how the brain works. Recent developments have shown that it is possible to make inferences from functional magnetic resonance imaging (fMRI) data about the meaning of certain brain activity instead of only analyzing on where which information is represented in the brain. Research has been mainly focused on the prediction of stimuli in the visual system, with early work like the prediction of objects (*Haxby et al. 2001*) or stimulus orientation (*Kamitani and Tong 2005*). More recent work has managed to make reconstructions of perceived images, which used observed brain data to reconstruct stimuli instead of choosing from a pre-made set of potential stimuli (*Thirion et al. 2006; Haynes and Rees 2006; Naselaris et al. 2009; Van Gerven et al. 2010 Horikawa et al. 2013*). This is a great breakthrough as it provides a new method for studying the inner workings of the brain.

To make these reconstructions, neural encoding and decoding are required. Neural encoding refers to the prediction of brain responses in certain neural populations represented by different stimuli or conditions, while neural decoding refers to the prediction of such stimuli from the measured brain activity (*Schoenmakers et al. 2013*). These predictions are made by learning a mapping between the stimuli and the brain responses. However, decoding models can also be learned by deriving the encoding distribution via Bayes' rule, this method lies at the basis of making reconstructions. The probability $P(s|r)$ of the stimulus s that is being reconstructed given a brain response r is expressed as the product of the probability that a brain response occurs given a certain stimulus $P(r|s)$ and the probability that the stimulus occurs independent of the brain activity $P(s)$ called the prior (*Friston et al. 2008; Naselaris et al. 2011*). The probability $P(r|s)$ is learned from the encoding model and the prior needs to be learned from the stimulus domain by looking at which stimuli occur more than others.

Most of the research on reconstructions has used images as the stimuli in the encoding and decoding process, but here we would like to present another option: the use of activation maps of convolutional neural networks. Whereas an image and a brain response are noticeably very different from each other, an activation map of a convolutional neural network is less so. In fact, neural networks were designed to simulate how the brain works and convolutional neural networks specifically mimic the brain's visual system (*LeCun et al. 1998*). One would expect that finding a good encoding model is easier with a stimulus more similar to the acquired brain data. Previous research by *Güçlü and van Gerven 2015* already showed that there is a gradient in the complexity of neural representations along the ventral stream. We therefore expect the lower layers of the neural network to map better onto visual area V1 and the higher layers to map better onto higher visual areas as V4.

To test this hypotheses we will build upon the work by *Schoenmakers et al. 2013* who were able to reconstruct handwritten characters by

means of a linear Gaussian approach. The same fMRI data and images of handwritten letters will be used. Each of the images will be propagated through a convolutional neural network to obtain the activation maps with which the encoding and decoding will be performed. Just like in Schoenmakers et al. a linear regression model is used as encoding model. The decoding model however will be slightly different. Whereas they used an unimodal prior containing only the stimulus categories, here we will use a uniform prior, which is the same for all possible categories. This allows the decoding to be driven by the found encoding only. Previous research using images and a uniform prior has resulted in accuracies of around 30% (*Schoenmakers et al. 2013*), but convolutional neural networks are the state-of-the-art when it comes to classification and contain more detailed information about features which could improve the reconstructions strongly.

The reconstructions of the activation maps will be propagated through a convolutional neural network in order to obtain classification accuracies. Furthermore, deconvolution will be used to visualize which features are represented in the brain. The classification results will be compared with the results from *Schoenmakers et al. 2014* where the prior is extended to include the entire alphabet. The use of convolutional neural network activation maps will also allow to compare which layers from the network and their corresponding features fit best on which visual areas of the brain, giving a bigger understanding about the workings of the visual cortex.

2 Convolutional neural networks

Artificial neural networks are inspired by the human central nervous system and are made up of artificial neurons called nodes. Nodes can take several inputs and produce a single output. Every input also has a weight which denotes the importance of that particular input. The neuron's output is determined by first calculating the sum of each input value times its weight and by adding this to the threshold value of the neuron called the bias. As a final step this is then passed through an activation function (e.g. the sigmoid function or the hyperbolic tangent) for the final output of the neuron.

These artificial neurons form different layers in the neural network. The first layer of neurons is called the input layer and each of the outputs from that layer are the input values for each of the neurons in the next layer of the network. This style of interconnected nodes continues until the last layer of the network called the output layer, which will output the answer to the problem we presented to the neural network.
Before a neural network can successfully output correct answers, the network first has to incorporate algorithms so it can learn. This is done via the backpropagation algorithm (for a detailed explanation of the algorithm, please see *Rumelhart et al. 1988*). Backpropagation changes the weights and biases of the network until the amount of errors the networks makes is minimized. This is called training the neural network. To know how many errors the network makes, we need a dataset to train on. This dataset contains samples of the problem we are trying to solve together with correct answers to those samples. A small portion of the dataset is held back so that after training it can be checked that the neural network can generalize to new samples of the problem and correctly assign the true label.

Standard neural networks do not take any spatial information of images into account. This is where convolutional neural networks (CNNs) come into play. CNNs are the state of the art when it comes to image recognition. The structure of these networks are based on mammalian visual cortices, found by *Hubel and Wiesel* in 1962. The cells in the visual cortex of those mammals respond to specific stimuli in their visual field, like certain orientations of objects, and are invariant to the stimuli's location.

Convolutional neural networks have three distinct characteristics that makes them different from standard neural networks: local receptive fields, shared weights and pooling (*LeCun et al. 1998*). The first two are incorporated in so called convolutional layers. Whereas in a normal neural

network the output would be fed to all the neurons of the next layer, a convolutional layer has a different approach. Each neuron of the next layer will be connected to a small region of neurons in the convolutional layer. These regions are called local receptive fields, and they become sensitive to one particular feature (e.g. a horizontal or vertical edge), because they share the same weights. The neuron in the next layer will receive a higher activation if the right feature is present within the receptive field. The activations from all the receptive fields on the input image combined is called a feature map. A convolutional layer consists of several feature maps, since we are not interested in only one feature, but a combination of many features.

Next to convolutional layers, a convolutional network also consists of pooling layers, which always come directly after a convolutional layer. Pooling layers decrease the amount of information from the output of a convolutional layer by reducing exact positional information to a smaller spatial representation. This is based on the notion that it is more important to know where certain features are relative to other features (*Nielsen 2015*). The pooling layer that is used most is called max pooling. With max pooling, each feature map is divided into rectangular areas and for each area the maximum value and its location are saved. The output of the layer is composed of only the maximum values, decreasing the amount of information significantly.

CNNs always end with one or several fully connected layers. These are layers with a standard neural network layout and act as the 'higher-level reasoning' layers with the role of classification (*Simard et al. 2003*). All spatial information will be lost, so fully connected layers cannot be followed with convolutional layers.

Because of these properties, CNNs are better for image recognition than standard neural networks for two reasons. The first is that there are significantly less parameters to be learned, which makes them faster to train. Secondly, because CNNs take the spatial structure of an image into account, they have less trouble detecting features across different positions in input images (*LeCun and Bengio 1995*).

3 Materials and Methods

3.1 Dataset

The stimuli dataset that was used during the acquisition of the brain data and for the training of the neural network consisted of handwritten characters in grayscale on a black background collected by *Van der Maaten 2009*. The database consists of 40.121 characters made by a total of 250 different writers. The characters that are included in the set are from the standard Latin alphabet and thus include letters A - Z in lower case as well in upper case, but for the research only the upper case letters were used. The size of each image is 56 x 56 pixels. To reduce the amount of classes slightly, two characters were excluded from the stimuli set to train the neural network with: the letters 'Q' and 'X' were chosen as they have a very low rate of use in our language. For the acquisition of the brain data even less character categories were used, because of the large amount of time it takes to acquire data per participant. Six letters were chosen: 'B', 'R', 'A', 'I', 'N' and 'S'.

3.2 Brain data

The acquisition of the fMRI data and its procedure are identical to those described in *Schoenmakers et al. 2013* and *Schoenmakers et al. 2014*. A short overview of the details described in those studies will be presented.

Brain data from visual areas V1 and V2 was acquired in a Siemens Trio 3T MRI scanner at the Donders Institute for Brain Cognition and Behaviour in Nijmegen, the Netherlands. Three healthy Dutch participants took part and were presented with 60 unique instances of each of the six letter categories. This was repeated once for a better estimate of the BOLD response, giving a total of 720 stimuli per participant. Participants were presented with a white square as their fixation point (and eye tracker was used to verify that the participants were fixating) and they had to respond with a button press when the square changed color to keep their attention focused on the experiment. The images of the letters were shown as flickering stimuli for one second, followed by three seconds of a black screen. The

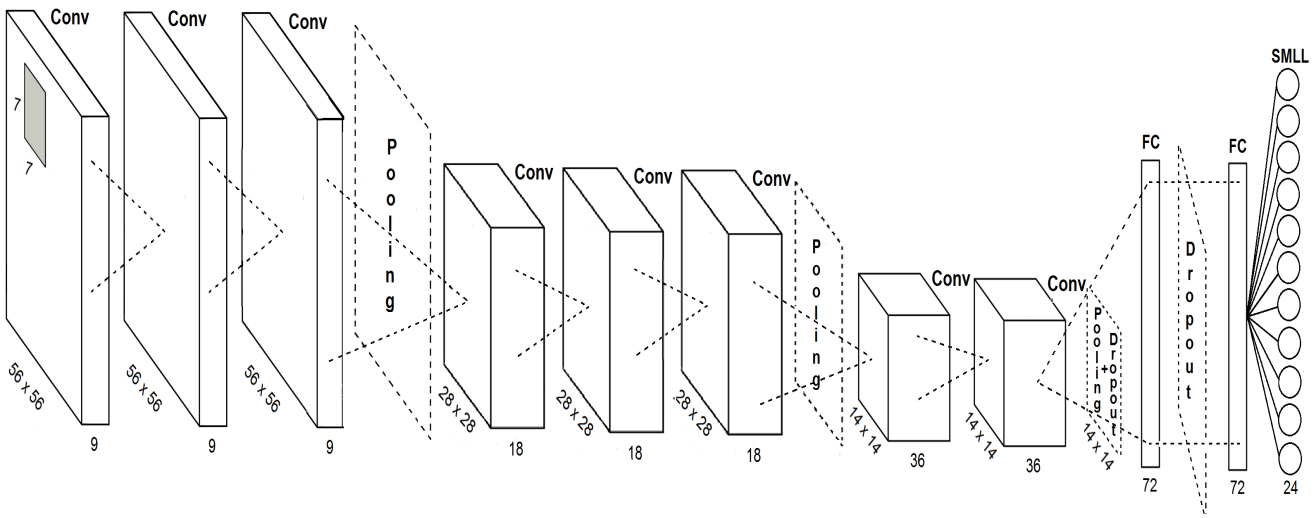

Figure 1: *The architecture of the used convolutional neural network where Conv stands for Convolutional layer, FC stands for Fully Connected layer and SMLL for Softmax Log-Loss layer. The dimensions stated in the layers indicate the feature map sizes.*

six characters were shown in pseudo-random order to prevent long repetitions of the same letter.

The functional images were collected with an EPI sequence using a 32 channel head coil (TR = 1.74 s, TE = 30 ms, GRAPPA acceleration factor 3, 83 flip angle, 30 slices in ascending order, voxel size 2 x 2 x 2 mm). To reconstruct the functional volumes, SPM8 software (Wellcome Department of Imaging Neuroscience, University College London) was used. Subsequently, a general linear model (GLM) was applied to compute the response of each voxel to a stimulus. The voxels in visual areas V1 and V2 were isolated using freesurfer software together with functional localizer data according to *Engel et al. 1997*.

3.3 Network architecture

To obtain the trained convolutional neural network needed to transform the character images to activation maps, the MatConvNet toolbox (*Vedaldi and Lenc 2015*) for the programming language Matlab was used. A special framework to make working with MatConvNet easier was made by *Manders 2015*, which enabled training multiple convolutional neural networks with different values for the parameters (e.g. architecture, amount of convolutional filters and the size of those respective filters) in order to determine which value or combination of values resulted in the highest accuracy.

An important consideration to take into account when setting the parameters was that a large number of convolutional layers in the network was preferred. The reason being that each of the convolutional layers will be evaluated on how well they map on V1 and V2. The more layers there are, the better can be seen which kind of features map best on which visual area, which should lead to a bigger understanding of the brain.

The architecture that was used consisted of eight convolutional and two fully connected layers. Convolutional layers three, six and eight used max pooling. The activation function that was used throughout the network was a leaky rectified linear function, which is a variation of the commonly used rectified linear function, because it resulted in slightly higher accuracies and better visualizations after deconvolution (see section 3.4.3). Dropout was applied to both the fully connected layers to improve accuracy (*Srivastava et al. 2014*). After layer ten a softmax log-loss function was used to transform the activation maps back into the letter class labels. The other settings were based on research by *Manders 2015*. The learning rate, momentum and weight decay were initialized to 0.001, 0.9 and 0.0005 respectively. The convolutional filter size was set to a size of 7x7 and the amount of first layer filters started with nine and doubled with each pooling layer. To obtain an even higher accuracy batch normalization was used during

training (*Ioffe and Szegedy 2015*).

The network was trained on images of the previously mentioned *Van der Maaten*, but unfortunately not all images from that set could be included. The proportion of occurrences between the characters was not balanced. The letter J contained the least amount of instances with only 126 images, so in order to prevent biases towards letters when training, for every letter 126 random images were chosen and combined in a new dataset. This resulted in a set with 126 x 24 = 3024 images. Training was performed on a hundred images per letter, for both the validation and testset twelve instances per letter were used.

3.4 Implementation

3.4.1 Obtaining the feature maps

The trained convolutional neural network was used to obtain the feature maps from the eight different convolutional layers in the network. The 360 images of handwritten characters that were shown to the participants in the fMRI scanner during the brain data acquisition were fed to the network. None of these images were used to train the convolutional network on. After each convolutional layer the feature maps were saved for all images and split into two sets. A set of 288 images to learn the mapping between the feature maps and their corresponding brain response during encoding and a set of 72 images to train the decoding model with. For encoding and decoding, the stimuli data had to be represented as a vector, so the test and train set of all eight layers were transformed by concatenating each row of each feature map of each image into a vector.

3.4.2 Encoding model

As in the previous work by *Schoenmakers et al.* the encoding model used is a multiple-output linear regression model. The model uses the convolutional feature maps as stimulus with $\mathbf{s} = (s_1, s_2..., s_x)' \in \mathbb{R}^x$, where s_i denotes the activations values in the feature map, and the associated brain responses with $\mathbf{r} = (r_1, r_2..., r_y)' \in \mathbb{R}^y$, where r_i denotes the voxel responses.

In order to reconstruct the most probable feature map s from the brain response r, we require a forward model $P(r|s)$. This can be obtained with the encoding model using the following formula:

$$r = B's + \epsilon \tag{1}$$

where ϵ is zero-mean normally distributed noise and B the regression coefficients which are estimated using regularized linear regression according to *Güçlü and van Gerven 2014*.
The likelihood function $P(r|s)$ is then given by:

$$P(r|s) = \mathcal{N}(r; B's, \Sigma) \tag{2}$$

with regression coefficients $B = (b_1, b_2, ..., b_y)$ and covariance matrix $\Sigma = diag(\sigma_1^2, \sigma_2^2, ..., \sigma_y^2)$. Stratified 5-fold cross-validation is used to estimate b_i and σ_i^2. For a detailed description of this procedure, please see *Schoenmakers et al. 2014*.

3.4.3 Decoding model

With the decoding model we want to compute the maximum a posteriori (MAP) estimate of a reconstruction of a set of feature maps s given a brain response from the visual cortex r. Using Bayes' rule, we obtain:

$$P(s|r) = \frac{P(r|s) * P(s)}{P(r)} \tag{3}$$

with

$$P(r) = \int \mathrm{d}s \; P(r|s) * P(s) \tag{4}$$

We assume that the forward model is given by a regularized linear Gaussian model and that the prior is given by a multivariate Gaussian. Because we are interested in the MAP estimate, the denominator P(r) is not needed for optimization as it does not depend on s.

We are then left with $P(r|s)$ and $P(s)$, for which we need to solve the following formula:

$$\hat{s} = \arg\max_s \{P(r|s) * P(s)\} \tag{5}$$

Since the prior is uniform, $P(s)$ is proportional to 1, leaving just the forward model in the equation. The forward model function stated in the section 3.4.2 can be represented as a multivariate

Gaussian in its canonical form as follows:

$$P(r|s) \propto exp\Big(-\frac{1}{2} r'\Sigma^{-1} r + \big(B\Sigma^{-1} r\big)' s - \frac{1}{2} s' B\Sigma^{-1} B' s \Big) \quad (6)$$

It then follows that $P(s|r)$ is equal to Eq. (6), which when dropping the terms independent of s, comes down to a multivariate Gaussian with mean m = $QB\Sigma^{-1}r$ and covariance Q = $(B\Sigma^{-1}B')^{-1}$. This then yields:

$$\hat{s} = m = (B\Sigma^{-1}B')^{-1} B\Sigma^{-1} r \quad (7)$$

since the mean of a Gaussian distribution is equal to its mode.

3.4.4 Classification and deconvolution

After the reconstructions have been made using the decoding model, the data is interpreted in two ways: Classification by the trained convolutional neural network and deconvolution to visualize the features that were reconstructed.

Classification

In order to be able to classify the data, the vector that was obtained after decoding first needed to be transformed back into feature map space. For each convolutional layer in the network it is known how many feature maps there are, so for each layer the corresponding vector is split into the activations of the 72 training images and then into the right amount of feature maps per image. Because we have activations for eight different layers in the network, we have to insert the feature maps in different position in the neural network in order to be able to propagate them to the classification layer successfully. The reconstructed feature maps belonging to the i'th convolutional layer will first go through the activation function belonging to that layer, then possibly through a max pooling layer and afterwards propagated normally from layer i+1 until the softmax layer. Afterwards, the classification of the reconstruction is compared to both the actual class of the original testing image as the classification of that image by the neural network.

Deconvolution

Deconvolution was originally designed as a method for performing unsupervised learning (*Zeiler et al. 2011*), but has proven its worth as a visualization technique for feature activity in convolutional neural networks (*Zeiler and Fergus 2014*). A deconvolutional network is the opposite of a convolutional network, as it uses the same methods, but in reverse. This means that instead of mapping pixels to features like a CNN does, it maps features to pixels which result in images displaying the feature a certain feature map is most sensitive to (e.g. round or sharp edges).

A deconvnet is always attached to a convnet since certain information is needed before it can invert the layers. Every layer of the CNN is linked to the same type of layer in the deconvolutional network: each convolutional layer has a corresponding deconvolutional layer, each pooling layer a corresponding unpooling layer and each activation function is inverted. To visualize the features of the i'th convolutional layer, first the input image is propagated through the convolutional network up until the i'th layer. The highest activation per feature map is then selected and all the other activations in the map are set to zero, before the feature maps are passed on to the deconvolutional network in layer i. We then inverse every layer before layer i and go back through the deconvnet until pixel space is reached. When visualized, the output shows what the feature that the original feature map was most sensitive to looks like.

Deconvolutional layers use the same filters as their corresponding convolutional layers, but they are transposed, which comes down to flipping the filter both vertically and horizontally. It then applies the filters to the output of the activation function that belongs to the layer.

Unpooling layers reverse the action of pooling by taking each value in the feature map (which contains only the maximum values) and place them in a new feature map the size of the original pre-pooled feature map at the same locations as before pooling. The rest of the values are set to zero.

The inversion of an activation function is of course different for each function. Since solely rectified linear units were used in our model,

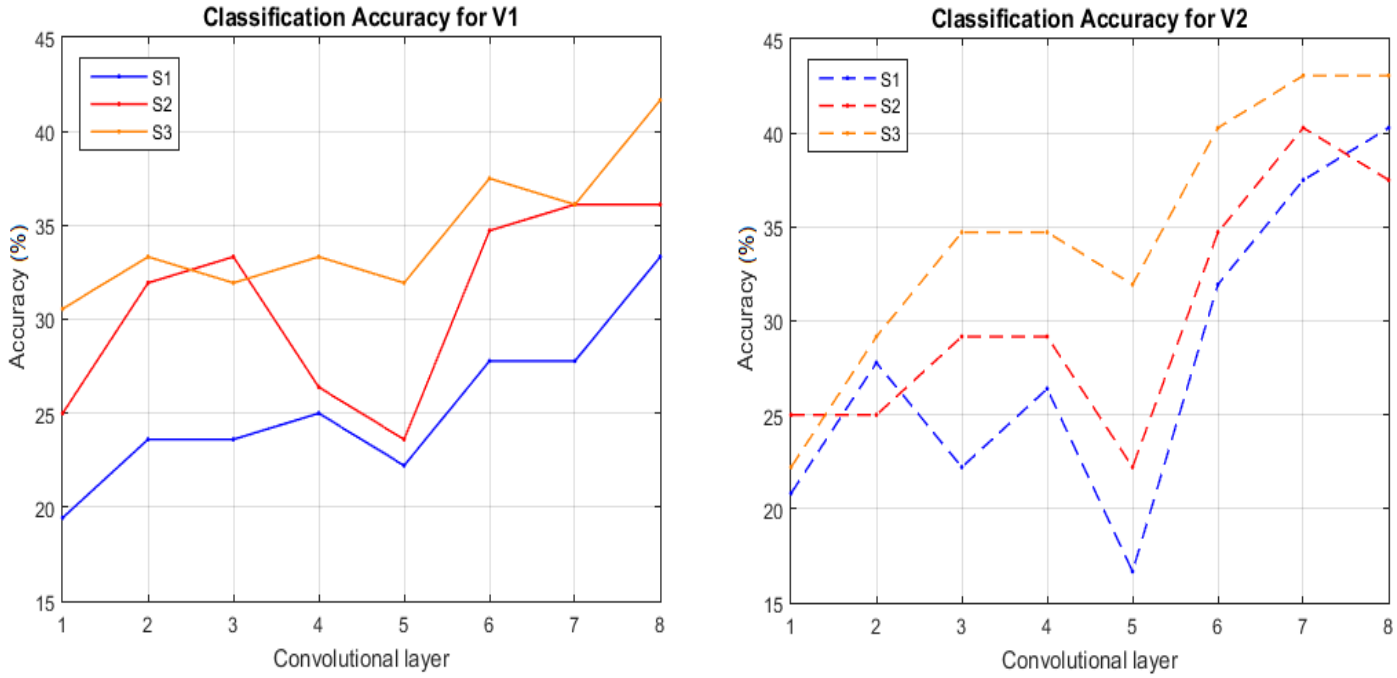

Figure 2: *Classification accuracies for all eight convolutional layers for both V1 (left) and V2 (right) on a testset of 72 images for three different subjects.*

only their inversion will be explained. Originally, *Zeiler and Fergus 2014* found that using a normal rectified unit as its own counterpart worked best to ensure that the feature maps only contained positive values. But *Springenberg et al. 2014* showed when using the guided backpropagation method, it resulted in much less noise, especially in the higher layers. This method still uses a rectified linear unit to set all values below zero to zero, but also remembers which values were changed to zero during the forward pass and set them to zero as well.

To visualize the features that were reconstructed during decoding in this study and compare the layers and their performance as best as possible, all the reconstructed feature maps are propagated through the neural network up until the last convolutional layer, so that the features visualized will form a complete letter. This way it can be seen what features of a letter are most important for identifying it.

As was the case for classification, after decoding the resulting vector first needs to be transformed back to feature space. Next the reconstructed feature maps get inserted in their corresponding layer in the neural network and are propagated forward up until the last layer. With only the highest activation active, it is then propagated backwards through the attached deconvolutional network. The feature maps are then normalized and saved as an image.

4 Results

To assess the performance of the used model the obtained reconstructions are evaluated by looking at the classification accuracies and by visualizing the features that were reconstructed.

Figure 2 displays the classification accuracies on the testset of 72 images for V1 (top) and V2 (bottom) for all eight convolutional layers for the three subjects. The accuracies for subject one lie between 19% and 33% in V1 and between 16% and 40% in V2, for subject two this is between 24% and 36% in V1 and 23% and 40% in V2 and subject three obtains accuracies between 31% and 41% in V1 and between 22% and 43% in V2. Accuracies are always highest for the last convolutional layer. At chance level we would expect to get only 4% correct.

Figure 3 and 4 illustrate the probabilities of the six character categories that were used in

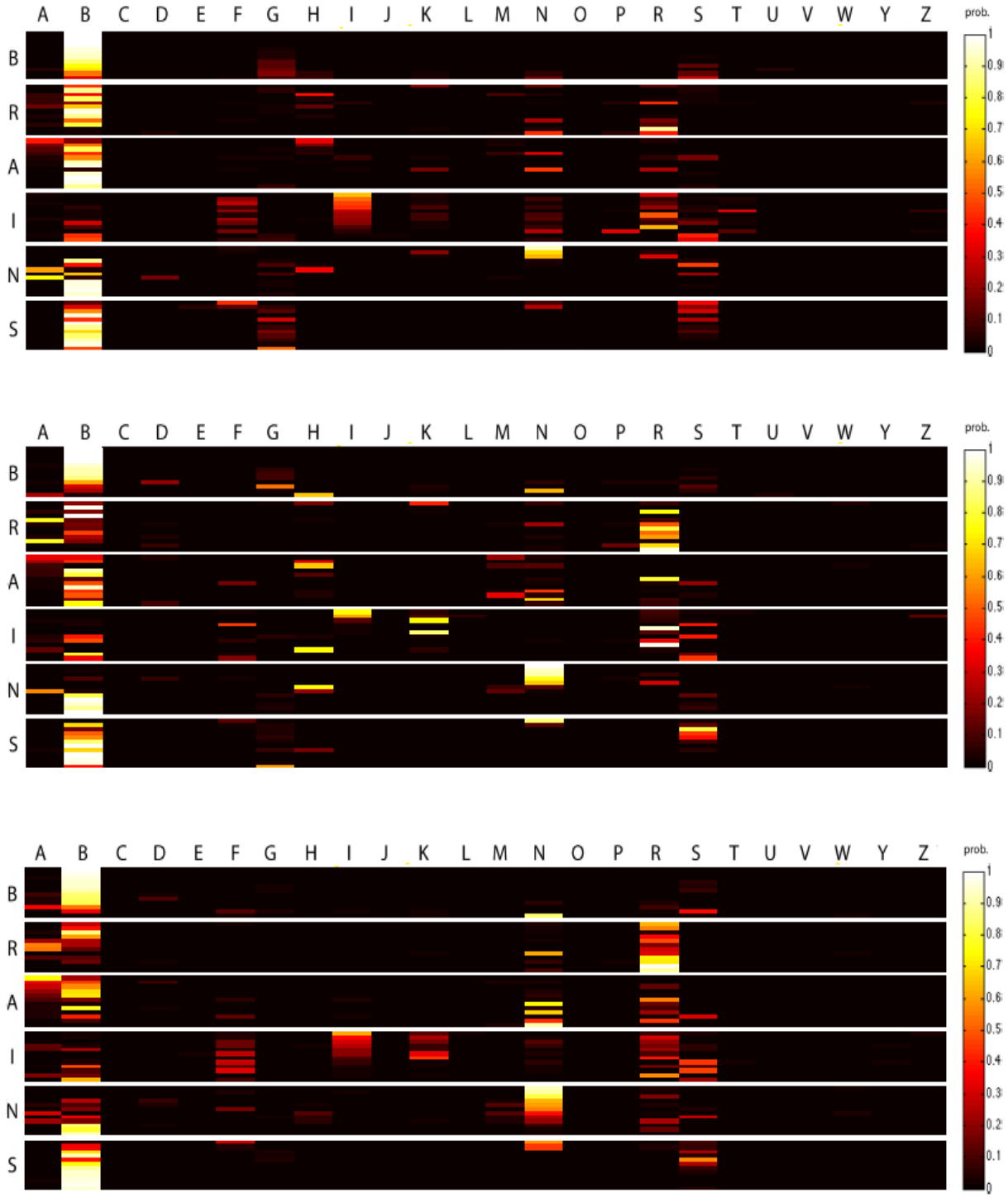

Figure 3: *Probability table for twelve instances of the six used character-categories belonging to each of the possible letters from convolutional layer 1 (top), 4 (middle) and 8 (bottom) in V1 from subject 3.*

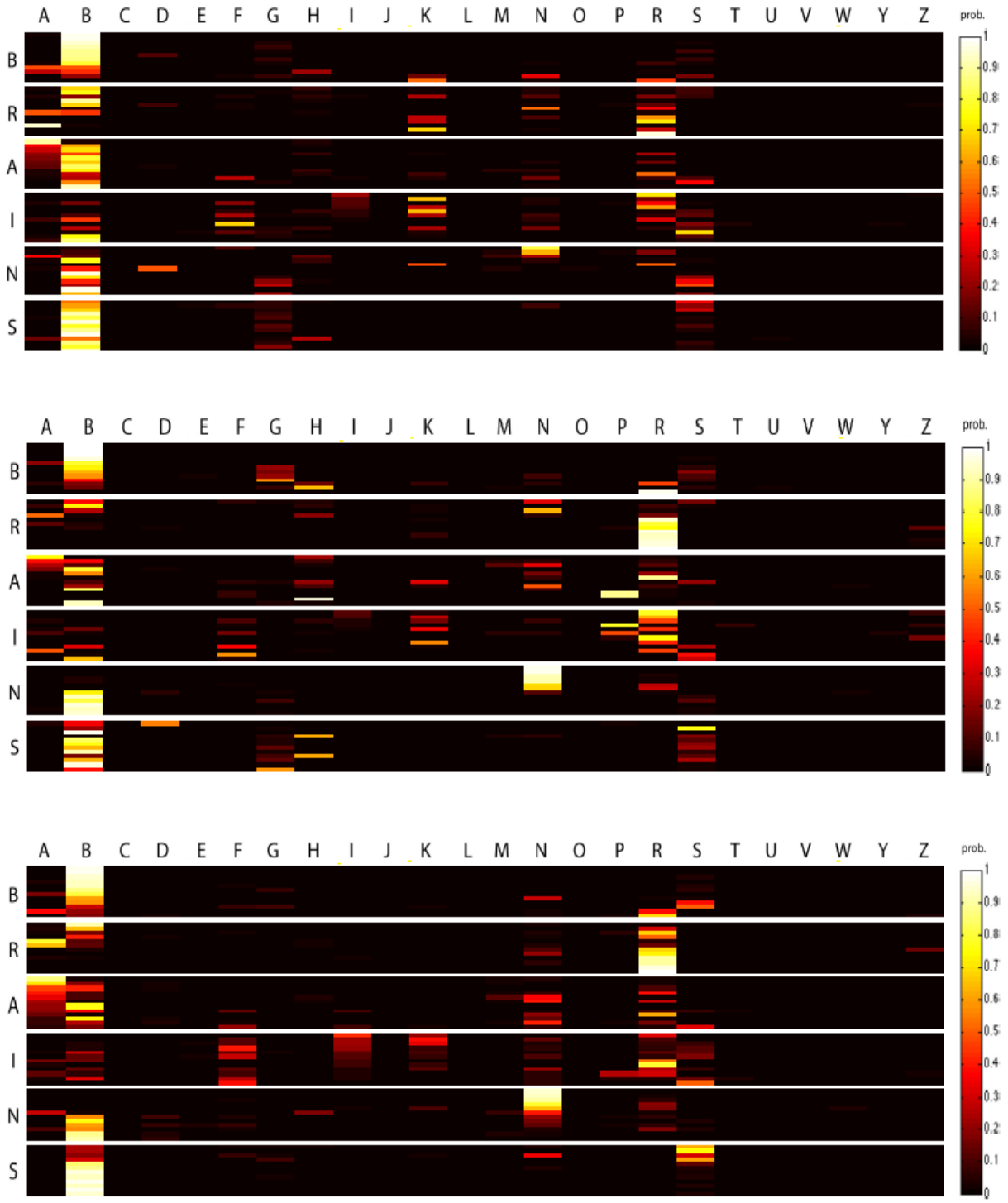

Figure 4: *Probability table for twelve instances of the six used character-categories belonging to each of the possible letters from convolutional layer 1 (top), 4 (middle) and 8 (bottom) in V2 from subject 3.*

the encoding and decoding process belonging to each of the possible categories. *Figure 3* displays this for V1 and *Figure 4* for V2 with convolutional layers one (top), four (middle) and eight (bottom) from S3 in both images. The instances have been sorted on the probability of the correct category. The letter B seems to be a confusing category for every letter, especially in the lower layers. The letter S seems to most affected by this.

Figure 5, 6, 7 and 8 depict the visualizations of the features made using deconvolution. In the first columns of the figures the original images are shown. *Figure 5* displays visualizations of the testset that are the result of passing the original images through a convolutional and deconvolutional network. No encoding or decoding was applied here, the visualizations were made for comparison purposes.

Figure 6 and 7 show correctly classified reconstructions for all convolutional layers from subject three in V1 and V2 respectively. After convolutional layer three the visualizations become less clear and more smoothened, because important information was missing for unpooling caused by the fact that the feature maps reconstructed are from deeper layers than the pooling layer. This means that the forward pass of the network started after the pooling layer, resulting in missing information for locations with values below the maximum. This was solved by filling the feature maps completely with the maximum values during unpooling.

Figure 8 shows reconstructions of features maps that were classified incorrectly. Notice that some visualizations look correct, but display the wrong character.

5 Discussion

A linear regression model was introduced that can decode brain data from the visual cortex. Convolutional neural network feature maps from eight convolutional layers were used as encoding and decoding stimuli. Results show that reconstructions can be made very accurately with performances of over 40% over a chance level of 4%. The classification accuracies show a clear trend upwards as the layer of the convolutional network becomes deeper with best results achieved at the eight convolutional layer. Layer 5 seems to be an exception to this rule, but it is unclear to why this is.

When comparing the results with the work from *Schoenmakers et al. 2014* where a multimodal prior was used with images as stimuli, it seems that the current approach results in more robust accuracies. In Schoenmakers et al. accuracies of 30.6, 16.7 and 45% were obtained for the three participants in V1, while here we obtained accuracies of 33.3, 36.1 and 41.7%.

An important observation from the results is that the letter B seems to be a confusing category for all the other letters, especially in the lower convolutional layers. While this seems very peculiar at first, it could have something to do with the fact that the mean of the letters 'B', 'R' 'A', 'I', 'N' and 'S' resembles the letter 'B'. The mean is used in encoding and decoding to calculate the z-scores and this could have a interfering effect on the reconstructions. Further research has to point out whether this is true and how to minimize this effect.

The visualizations of the reconstructions quite interestingly show that, when classified correctly, the neural network gives back reduced images of the original letters with only the most important feature standing out. When comparing these reconstructions with visualizations where the letters where only passed through the network without any influence from brain data, it becomes clear that apparently only the feature that makes a letter different from all other letters is important to the lower-level visual areas when it comes to letter classification.

Looking at the differences between brain areas V1 and V2, there does not seem to be a strong difference between the two, both in the classification accuracies and in the visualizations. Both V1 and V2 fit the best on the eight convolutional layer. One would have expected that V2 would fit best on a deeper layer than V1 as V2 has more complex receptive fields. The fact that this is not the case could have several reasons. It could be that activations in the brain and in a neural net-

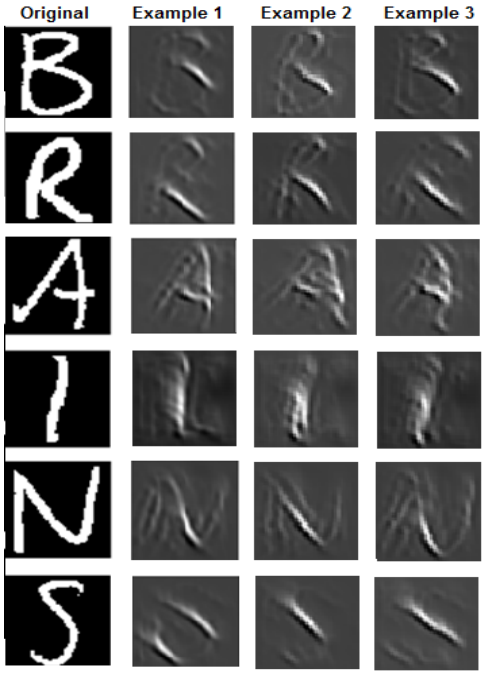

Figure 5: *Examples of visualizations from the testset passed through a convolutional and deconvolutional network.*

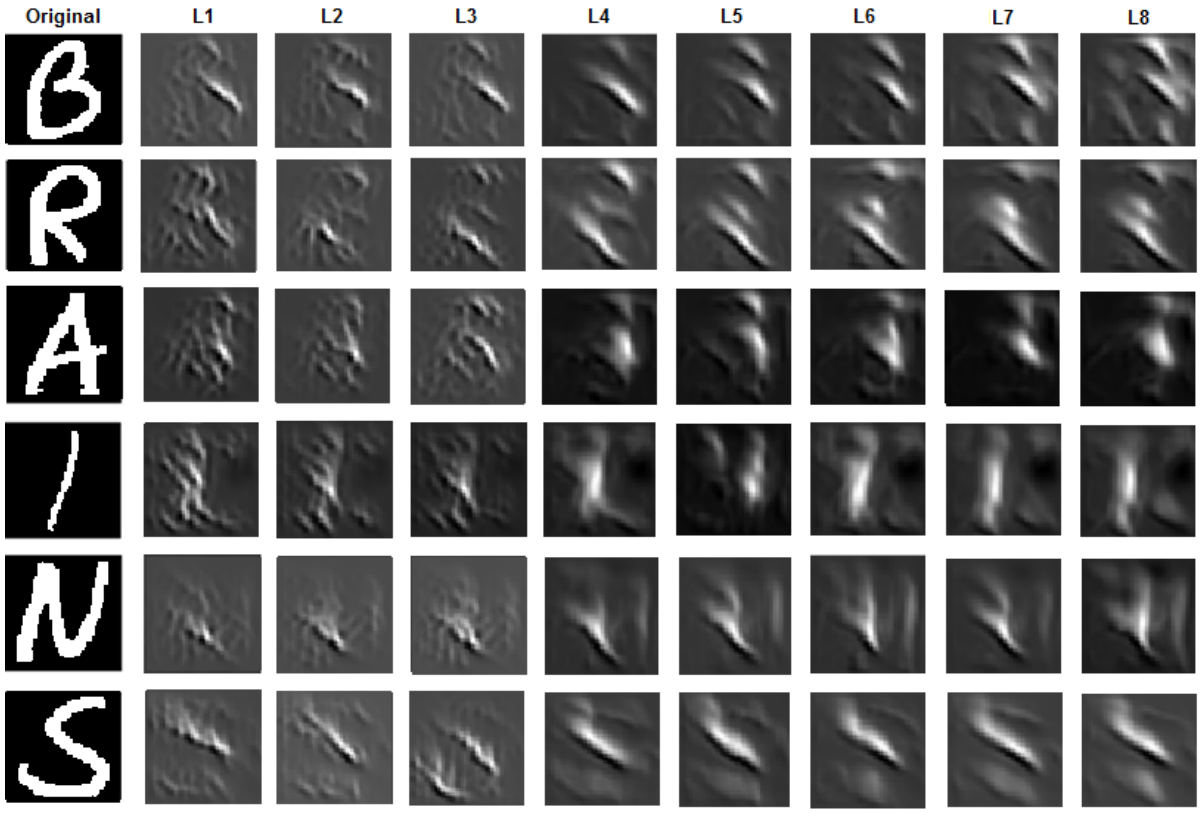

Figure 6: *Examples of visualizations of reconstructions from all eight convolutional layers with V1 from subject 3 that were correctly classified.*

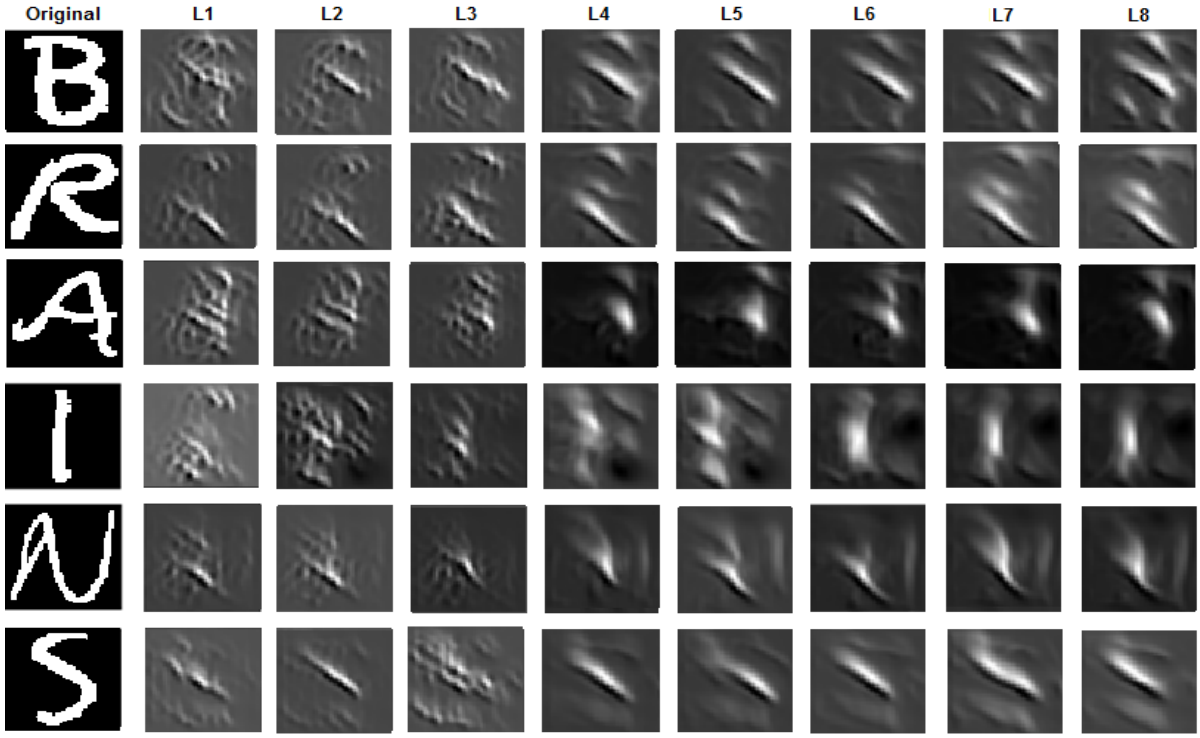

Figure 7: *Examples of visualizations of reconstructions from all eight convolutional layers with V2 from subject 3 that were correctly classified.*

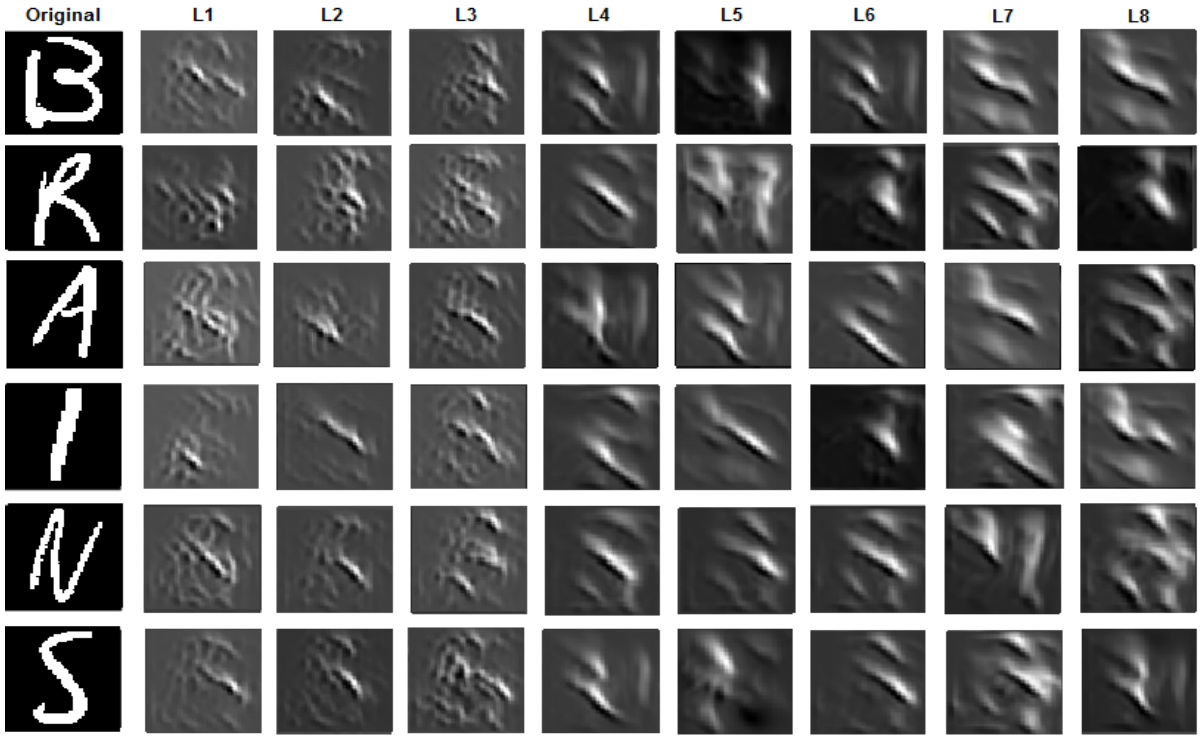

Figure 8: *Examples of visualizations of reconstructions from subject 3 that were incorrectly classified.*

work are not propagated in the same way, causing the feature maps to loose critical information when propagated through the network for classification, resulting in the highest accuracy for the feature maps that are propagated through the least amount of layers.
It could also be the case that because V1 and V2 have top-down influences from higher-order visual brain areas that the information present in V1 and V2 also contains high-level information instead of just low-level information as was previously thought (*Guo et al. 2007*). This would result in information that is a lot more complex than lower layers of a convolutional network represent, resulting in a best fit on the last layer of the network. The idea that V1 and V2 contain both low-level and high-level information is backed up by the fact that both areas have an early peak when looking at the classification accuracies. This could represent the low-level information. The peak for V1 is slightly earlier than the peak for V2 and we know indeed that the low-level information of V1 is less complex than the information in V2.

5.1 Further work

The current approach can be advanced further by testing various modifications that could lead to better classification and reconstruction performance. The first thing that comes to mind is to test other convolutional neural networks with different architectures to explore the effect of different parameters on the accuracy and the quality of the reconstructions. In the current study there was not enough time or computational power to test the effect of all parameters of the network very rigorously and it would especially be interesting to see what happens when the amount of convolutional layers is increased. Maybe the differences between what kind of information is processed in visual areas V1 and V2 will become more clear.
Secondly, it could be very beneficial to train the neural network on a bigger dataset of images as we now only had access to a small balanced set of 3024 letters. This is a very small amount when there are 24 categories to be learned.

For a better mapping we should not only look at improving the neural network side, the techniques to acquire the neuroimaging data could also be improved to obtain more precise readings of the brain. This can be done by doing longer recordings, better analysis and by improving the voxel selection methods.
Another suggestion could be to also collect data from the extrastriate cortex, such as V3 and V4, for a bigger understanding of the visual cortex and how low and high-level information is processed.

Furthermore would it be interesting to explore other domains than handwritten characters or even shift to other sensory modalities. An example of a different domain could for example be to take natural images. Although that is a lot more complex than just letters, convolutional neural networks have been proven to classify natural images with a very high accuracy (*Krizhevsky et al. 2012*). Interesting would be to see how performance is affected and if the differences between the visual areas becomes more distinct. It would also be possible to shift from the visual modality to another sensory modality such as auditory perception. This way we could not only classify what a person is seeing by just looking at the brain data, but also what he is hearing. In the future this type of classification of people's senses might for example help people with locked-in syndrome communicate with the outside world.

In this study deconvolution is only used to visualize the last layer of the convolutional neural network and create a complete letter, but it is possible to visualize every layer of the network. The first layer will represent very small features and with every layer deeper the features will become a bit bigger. This could be used to figure out which features and shapes are represented by which areas and voxels in the visual cortex to get a better representation of the inner workings of the brain.

5.2 Conclusion

Summarizing, results show that good classification accuracies can be obtained from mapping brain data on handwritten characters with a linear regression model using convolutional neural network feature maps as its stimulus. Visualiza-

tions of the reconstructions show that the most important features within each letter are sufficient for letter classification in the lower visual areas. Furthermore, there does not seem to be a clear difference between brain areas V1 and V2 with regards to their visualizations and classification accuracies. They both fit best on the deepest convolutional layer. Further research is needed to find out why this is.

References

Engel, S. A., Glover, G. H. and Wandell, B. A. (1997), 'Retinotopic organization in human visual cortex and the spatial precision of functional mri.', *Cerebral cortex* **7**(2), 181–192.

Friston, K., Chu, C., Mourão-Miranda, J., Hulme, O., Rees, G., Penny, W. and Ashburner, J. (2008), 'Bayesian decoding of brain images', *NeuroImage* **39**(1), 181–205.

Güçlü, U. and van Gerven, M. A. (2014), 'Unsupervised feature learning improves prediction of human brain activity in response to natural images', *PLoS Comput Biol* **10**(8), e1003724.

Güçlü, U. and van Gerven, M. A. (2015), 'Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream', *The Journal of Neuroscience* **35**(27), 10005–10014.

Guo, K., Robertson, R. G., Pulgarin, M., Nevado, A., Panzeri, S., Thiele, A. and Young, M. P. (2007), 'Spatio-temporal prediction and inference by v1 neurons', *European Journal of Neuroscience* **26**(4), 1045–1054.

Haxby, J. V., Gobbini, M. I., Furey, M. L., Ishai, A., Schouten, J. L. and Pietrini, P. (2001), 'Distributed and overlapping representations of faces and objects in ventral temporal cortex', *Science* **293**(5539), 2425–2430.

Haynes, J.-D. and Rees, G. (2006), 'Decoding mental states from brain activity in humans', *Nature Reviews Neuroscience* **7**(7), 523–534.

Horikawa, T., Tamaki, M., Miyawaki, Y. and Kamitani, Y. (2013), 'Neural decoding of visual imagery during sleep', *Science* **340**(6132), 639–642.

Hubel, D. H. and Wiesel, T. N. (1962), 'Receptive fields, binocular interaction and functional architecture in the cat's visual cortex', *The Journal of physiology* **160**(1), 106–154.

Ioffe, S. and Szegedy, C. (2015), 'Batch normalization: Accelerating deep network training by reducing internal covariate shift', *arXiv preprint arXiv:1502.03167* .

Kamitani, Y. and Tong, F. (2005), 'Decoding the visual and subjective contents of the human brain', *Nature neuroscience* **8**(5), 679–685.

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, *in* 'Advances in neural information processing systems', pp. 1097–1105.

LeCun, Y. and Bengio, Y. (1995), 'Convolutional networks for images, speech, and time series', *The handbook of brain theory and neural networks* **3361**(10), 1995.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998), 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE* **86**(11), 2278–2324.

Manders, J. (2015), 'Optimizing convolutional neural networks for fast training on a small dataset', *Radboud University Nijmegen* (Unpublished bachelor thesis).

Naselaris, T., Kay, K. N., Nishimoto, S. and Gallant, J. L. (2011), 'Encoding and decoding in fmri', *Neuroimage* **56**(2), 400–410.

Naselaris, T., Prenger, R. J., Kay, K. N., Oliver, M. and Gallant, J. L. (2009), 'Bayesian reconstruction of natural images from human brain activity', *Neuron* **63**(6), 902–915.

Nielsen, M. A. (2015), 'Neural networks and deep learning', *Determination Press* .

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1988), 'Learning representations by back-propagating errors', *Cognitive modeling* **5**(3), 1.

Schoenmakers, S., Barth, M., Heskes, T. and van Gerven, M. (2013), 'Linear reconstruction of perceived images from human brain activity', *NeuroImage* **83**, 951–961.

Schoenmakers, S., Güçlü, U., Heskes, T. and van Gerven, M. (2014), 'Gaussian mixture models and semantic gating improve reconstructions from human brain activity', *Frontiers in computational neuroscience* **8**.

Simard, P. Y., Steinkraus, D. and Platt, J. C. (2003), Best practices for convolutional neural networks applied to visual document analysis, *in* 'null', IEEE, p. 958.

Springenberg, J. T., Dosovitskiy, A., Brox, T. and Riedmiller, M. (2014), 'Striving for simplicity: The all convolutional net', *arXiv preprint arXiv:1412.6806* .

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014), 'Dropout: A simple way to prevent neural networks from overfitting', *The Journal of Machine Learning Research* **15**(1), 1929–1958.

Thirion, B., Duchesnay, E., Hubbard, E., Dubois, J., Poline, J.-B., Lebihan, D. and Dehaene, S. (2006), 'Inverse retinotopy: inferring the visual content of images from brain activation patterns', *Neuroimage* **33**(4), 1104–1116.

Van der Maaten, L. (2009), 'A new benchmark dataset for handwritten character recognition', *Tilburg University* pp. 2–5.

Van Gerven, M. A., De Lange, F. P. and Heskes, T. (2010), 'Neural decoding with hierarchical generative models', *Neural computation* **22**(12), 3127–3142.

Vedaldi, A. and Lenc, K. (2015), Matconvnet: Convolutional neural networks for matlab, *in* 'Proceedings of the 23rd Annual ACM Conference on Multimedia Conference', ACM, pp. 689–692.

Zeiler, M. D. and Fergus, R. (2014), Visualizing and understanding convolutional networks, *in* 'Computer vision–ECCV 2014', Springer, pp. 818–833.

Zeiler, M. D., Taylor, G. W. and Fergus, R. (2011), Adaptive deconvolutional networks for mid and high level feature learning, *in* 'Computer Vision (ICCV), 2011 IEEE International Conference on', IEEE, pp. 2018–2025.