Radboud University, Nijmegen

Bachelor Thesis

Artificial Intelligence

Data augmentation of a handwritten character dataset for a Convolutional Neural Network and integration into a Bayesian Linear Framework

Author:
Denise Klep
4210646

Supervisors:
Sanne Schoenmakers
Marcel van Gerven

June 3, 2016

1 Abstract

Convolutional neural networks are often used for image recognition. They have, for example, achieved the lowest error rate (0.23 percent) for the MNIST database up until now [6]. The use of receptive fields also makes them similar to how the human visual cortex works. The task at hand is to use convolutional neural networks to find optimizations for brainreading research [21]. The goal is to find these optimizations through using different preprocessing methods on the handwritten character dataset [27] and testing which method results in the highest classification accuracy for the convolutional neural networks. An optimization is achieved by using the most efficient data augmentation methods from the convolutional neural networks to preprocess the prior of the Bayesian linear framework.

2 Introduction

Brainreading is a technique in which a character which is shown to the test subject is successfully decoded again from their brain data [21]. It consists of an encoding and decoding step: Encoding provides a relation between the character features and the neural response, whereas decoding can reconstruct the stimulus again from the new neural response. Due to noise in the brain data, improvements can still be made to this process. A convolutional neural network was chosen to achieve this, due to its capabilities in image recognition. Jeroen Manders' research focused on finding the optimal parameters for optimization methods and different network structures. Leonieke van den Bulk's research focused on retrieving network filters using deconvolution and using the neural network features instead of pixels as input in the Bayesian linear framework. The neural network filters are used to approximate areas of the visual cortex, such as V2.

Another issue is that the used handwritten character dataset is small in size. Too few training samples can result in the features not being properly learned by the neural network, bringing about a poor classification accuracy. Data augmentation can counteract this by enlarging the training set, generating more training samples and simultaneously bringing more variation into the training set.

2.1 Neural networks

Neural networks are models which are inspired by the human brain, using connected neurons. These connections contain weights that are continually adjusted during the training phase of the neural network. This weight adjustment is similar to learning. An example of a neural network is a perceptron [19]. A single-layer perceptron consists of an input vector and a weight vector, which are then the input to an activation function to retrieve a binary output. As

such, a single-layer perceptron is comparable to one arti cial neuron. Figure 1 shows an example of a perceptron. The convergence of a perceptrons occurs through a process of rst initializing its weights and threshold, then for each training sample j the output of neuron i y_i is computed using the formula y_i = $f(\sum w_{ij} \quad x_j)$, where f(X) represents the activation function, w_{ij} represents the weight between input node j and neuron i, and x_j represents training sample j. Then according to this output, the weights are updated according to the following formula: $w \quad w + \eta(t_j \quad y_j)x_j$, where y_j represents the predicted output for training sample j, t_j represents the target for training sample j, and η represents the learning rate. Due to its use of labeled training data, there is knowledge of what the target of a training sample will be, and hence perceptrons utilize supervised learning.

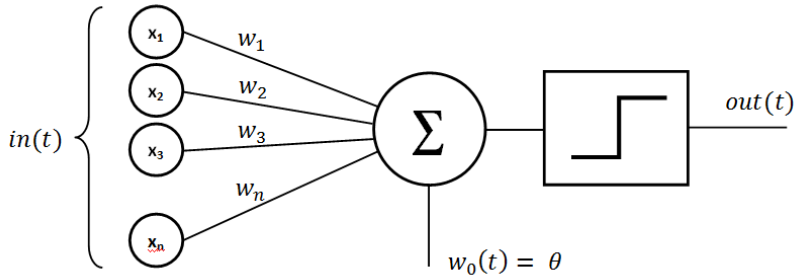


Figure 1: An example of a perceptron
(Source: Conner DiPaolo, 2015 [8]).

Multilayer perceptrons contain an input layer, one or more hidden layers and an output layer. Therefore, they are comparable to multiple arti cial neurons. They can classify non-linearly, utilizing a sigmoidal activation function [7]. They are fully connected, meaning each neuron from a layer is connected to each neuron of the next. Figure 2 shows an example of a multilayer perceptron. An MLP (multilayer perceptron) learns through backpropagation [20]. After the forward pass, which calculates the predicted output through the regular ow from input to output, the loss is calculated between the predicted ouput and the true output at the output layer, using δ_j = $t_j \quad o_j$. Then, through a backward pass the weight updates are calculated for each hidden or input neuron j, using δ_j = $o_j(1 \quad o_j) \quad \sum_{k2K} w_{kj} \quad \delta_k$, where the rst part is the derivative of the sigmoid activation function and the second part a weighted summing of the loss of the current neuron j and the next downstream neuron(s) k. The weights are then updated using the function $4w_{ji}$ = $\eta \quad \delta_j \quad x_{ji}$.

Certain optimizations can also be made to the neural network. Weight decay [13], dropout [25] and momentum [26] are main optimizations. Dropout is a technique used to counteract over tting by randomly dropping hidden units

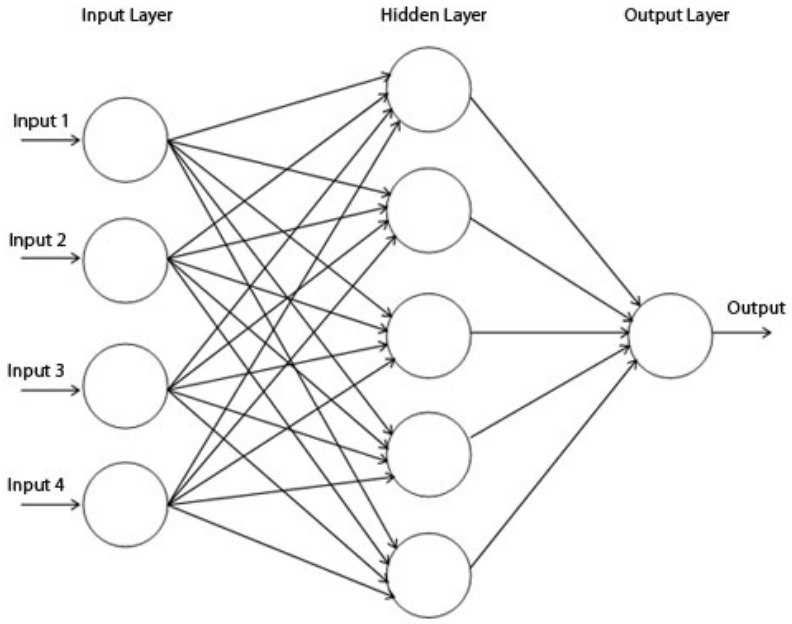


Figure 2: An example of an MLP (Source: Ciumac Sergiu, 2011 [24]).

from the training process. This amounts to the neural network not training with its full neuronal capacity, and continually training with di erent units. At the testing process, the full neural network is used to gain classi cation accuracy, without any dropout. Thus, dropout makes the training process more robust, because the network is trained in such a manner that the individual units do not grow as dependent of each other. Momentum is an optimization technique in which a portion of the previous weight update is added to the current one. This accelerates the gradient descent. Lastly, weight decay is an optimization method that pushes for a smaller weight vector, e ectively penalizing a larger weight vector. This constriction on the range of the weight vector decreases over tting.

2.2 Convolutional neural networks

A convolutional neural network is known to be e cient at image recognition. A convolutional neural network holds the current record for MNIST classi cation [6]. As such, the choice of network was a convolutional neural network. This choice was based on the fact that the used neural network was supposed to be as close to the visual cortex as possible. A convolutional neural network utilizes deep learning, and as such contains many hidden layers and multiple levels of feature abstractions.

Our convolutional network consists of a combination of convolutional (ReLU) layers, pooling layers, fully connected layers and a loss layer at the very end.

2.2.1 Convolutional layers

Convolutional layers contain lters which have a receptive eld on the input image, similarly to the visual cortex. The visual cortex utilizes receptive elds. A V1 neuron contains a small receptive eld whereas for example a V4 neuron contains a much larger receptive eld. A V4 receptive eld is comprised of the smaller, lower-level receptive elds such as from V1 [11].

In Figure 3, one lter of ve by ve is used to obtain a feature map. A feature map is obtained by sliding the lter over, or convolving the lter with, the input image. The resulting feature map in Figure 3 is 24 by 24. When convolving a k by k lter with an l by l input image, the resulting feature map is $l \quad k + 1$ in size. Each convolutional layer can contain multiple lters and feature maps, as shown in Figure 4. This example contains three feature maps. Aside from its use of receptive elds, convolutional layers pose another advantage with regards to image recognition: An entire feature map uses the same weights and biases with respect to its input image. This hugely cuts back on used parameters, resulting in faster training.

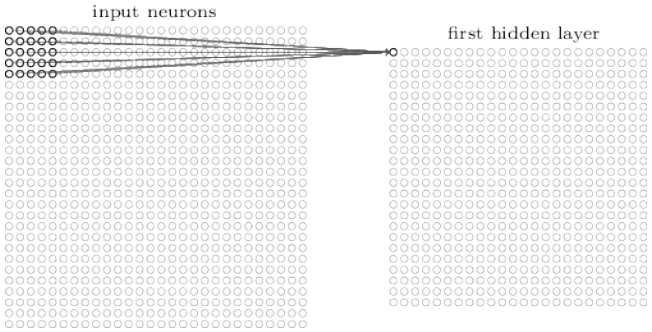


Figure 3: A visualisation of a feature map obtained by one ve by ve lter (Source: M.A. Nielsen, 2015 [18]).

2.2.2 Pooling layers

Max pooling, which is the pooling sort used here, takes the maximum value of non-overlapping sub-regions of the image, therefore sampling it down such that the image size is reduced (usually by half). As seen in Figure 5, a two by two input region is max-pooled, resulting in an image half the size. This is another advantage about using a convolutional neural network for image recognition,

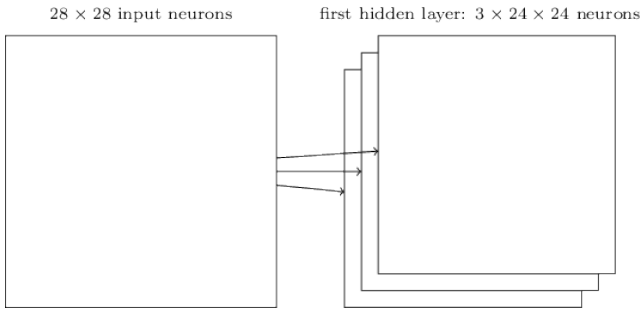


Figure 4: A visualisation of one convolutional layer (Source: M.A. Nielsen, 2015 [18]).

because yet again the amount of parameters is being decreased. Pooling is typically used after convolution.

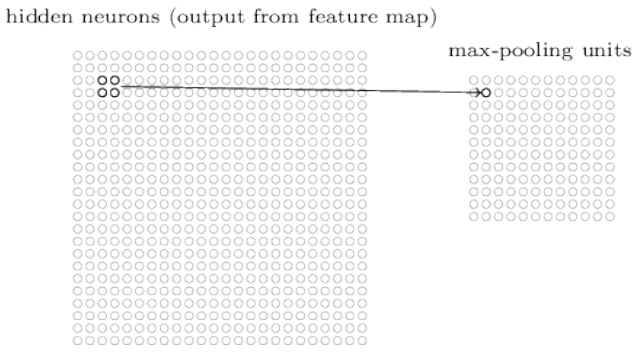


Figure 5: A visualisation of one pooling layer (Source: M.A. Nielsen, 2015 [18]).

2.2.3 ReLU layers

ReLU means rectified linear unit. A ReLU layer in the convolutional neural network applies the activation function $f(x) = max(0, x)$, with x being a neuron's input [17]. This activation function is also called the rectifier. By using ReLU, the deep neural network can train faster [12] compared to similar non-linear activation functions such as sigmoid or the hyperbolic tangent. ReLU is considered to be more biologically plausible for supervised deep neural network training than for example sigmoid activation function [9], which is desirable because for handwritten character image classification, we try to mimic the function of the human visual cortex as much as possible.

2.2.4 Fully connected layers & Loss layer

The fully-connected layers help in bringing the previous-level feature maps back to predicted labels. Spatial information that was preserved in the convolutional layers is lost in the fully connected layers. The input to the fully connected layers are the feature maps from the previous convolutional layer. Then, high-level reasoning over the feature maps is performed by the fully connected layers.

After that, the nal layer is the loss layer. The softmax function [3] handles multi-class classi cation through multinomial logistic regression. Softmax estimates the probabilities of training or test samples being a certain predicted label. The function for these estimated probabilities is: $P(y = ijx) = \frac{exp(x^T \ w_i)}{\sum_{k=1}^{K} exp(x^T \ w_k)}$, for i=1,...,K. The x represents the output of the last fully connected layer, which has to be labeled one of K class labels. In our case, there are 24 classes so K = 24. The denominator performs a normalization, such that the probabilities add to one. Using cross entropy, the loss over the estimated probabilities is calculated. In its most basic form, cross entropy is described by the function $H(y, y^0) = \sum_i y_i^0 \ log(y_i)$. The y is the predicted probability distribution, which was obtained with softmax. The y' is the true probability distribution. Figure 6 shows an entire convolutional neural network.

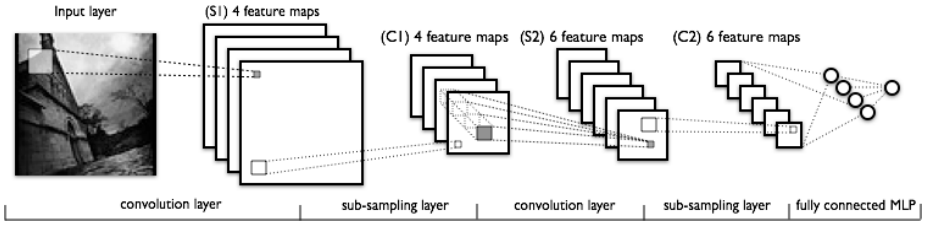


Figure 6: An entire convolutional neural network (Source: LISA lab, University of Montreal, 2008-2010 [14][16]).

2.3 The standard dataset

The handwritten character dataset by Van der Maaten (2009) [27] was used, containing grayscale images. The dataset used here contains 24 classes, which encompasses the entire alphabet except character 'X' and 'Q'. The issue that this dataset poses is that it is relatively small, especially when compared to a benchmark dataset such as MNIST [15], a handwritten digit dataset. Whereas MNIST consists of a training set of 60,000 samples and a test set of 10,000 samples for ten classes, the used handwritten character dataset consists of a total of 40,121 unbalanced samples for 24 classes. Due to the lack of training samples, data augmentation is performed over the training set to generate more training samples.

2.4 Chosen preprocessing methods for data augmentation

Ciresan et al. (2012) [6] achieved the lowest error rate for MNIST [15] using a convolutional neural network, so it was useful to consider their choice of data augmentation methods. The methods that were useful in their research were scaling, translation and rotation. Horizontal shearing is a method also used as preprocessing method for MNIST classi cation [5]. Yann LeCun et al. (1998) [14] also showed a decreasing error rate when adding augmentations of original samples of handwritten digits. They used horizontal and vertical translation, horizontal shearing and scaling among other things, asserting the choice of data augmentation methods for this dataset. Adding noise to a neural network can help because of the increase in training samples it generates [4]. Brown et al. (2003) [4] found that, when they added uniform random noise to the training patterns, the classi cation accuracies obtained with an MLP were signi cantly better. The addition of noise has also been shown to reduce over tting [29]. Zur et al. (2009) [29] found improved classi cation when noise was added to breast ultrasounds.

Rotation, translation, scaling and shearing fall under a category of transformations called a ne transformations. An a ne transformation is a linear mapping which retains the mathematical structure (points, straight lines and planes) and parallelism. A ne transformations employ transformation matrices and matrix multiplications with coordinates to achieve augmentation of an original image.

Rotation is a method in which every point in the image is moved relative to at least one xed point in the image. In the case of the handwritten character dataset used in this research, the xed point was the center point of the image. Translation is a method in which every point in the image is moved a given amount of pixels horizontally and/or vertically. Scaling is a method in which every point in the image is multiplied with a certain scaling factor, which is determinant for how much the original image will shrink. At rst sight, shearing seems quite similar to rotation but it is a very di erent preprocessing method. Shearing creates a parallel plane of the original image, using a shearing vector to calculate the angle of the parallel plane. And lastly, by adding noise a random variation of colour values is added to the original image.

2.5 Bayesian linear framework

In the research done by Schoenmakers et al. (2013) [21], handwritten characters were shown to the test subjects and their brain responses were measured, with the goal of decoding visual space from the brain responses. The characters and their brain responses were encoded using a regularized linear regression as posed by Gucl and Van Gerven (2014) [10]. Using a prior, the brain responses could be decoded again to their corresponding character images using Gaussian mixture models. The derevation of this decoding technique can be found in

Schoenmakers et al. (2015) [22].

The optimization proposed for this multimodal prior [22] is to augment the prior using the data augmentation methods that proved most e cient when testing them on convolutional neural networks. The idea behind this is that, similarly to augmenting the training set for the convolutional neural networks, the prior contains more variance as well as more samples, and thus the decoded reconstructions might be more accurate, in the same manner the classi cation accuracy of the characters in the convolutional neural networks might be more accurate.

3 Methods

3.1 Network architecture

The data augmentation methods were tested on a small, basic network structure and a larger, more optimized network structure to compare how much in uence the preprocessing methods actually had under di erent circumstances. The smaller network structure was a *cpcp* , in which c stands for convolution layer, p for pooling layer and f for fully connected layer. This structure did not utilize techniques such as dropout [25], momentum [26] and weight decay [13]. The larger network was a *ccpccpccp* , which did utilize dropout [25], weight decay [13] and momentum [26].

The *ccpccpccp* network utilized a momentum of 0.9, a dropout of 0.5, a learning rate of 0.001, a weight decay of 0.0005. The *cpcp* network did not utilize momentum, dropout or weight decay, so these parameters were all set to 0. A learning rate of 0.01 was used. Both networks utilized a xed number of epochs, namely 20. The networks had a batch size of 24, and the number of batches adapted accordingly. Both also started with 9 lters of size 5, and lters increased exponentially. Lastly, both networks utilized a learning rate decay of 1. Jeroen Manders researched the optimal values for each of these parameters for each network structure in his paper.

3.2 Software

In order to expand the standard dataset with di erent data augmentation methods, a master script was created from which each method could separately be switched on or o . For the implementation of the data augmentation methods, an Image Processing MATLAB Toolbox was used [1]. MatConvNet [28] was used for training and testing the convolutional networks. The functionality of MatConvNet was con gured by Jeroen Manders. Software for the application and evaluation of the Bayesian linear framework was provided by Sanne Schoenmakers.

3.3 The dataset

A division of a training set, validation set and test set was chosen. When using only a training and test set, over tting can occur due to training on the same samples too much, learning features too speci c to be generalized to the test set. This can result in a lower classi cation accuracy on the test set with previously unseen samples. A validation set also contains separate, previously unseen samples. Over tting is regulated by monitoring the error over the validation set as well as the training set. When the error over the validation set starts to increase, over tting occurs. To solve this, the minimum objective is taken for the validation set and the network's weights and biases are saved at that point. The objective is the output of the loss layer, which calculates how much the predicted outcomes deviate from the true outcomes.

In the original handwritten character dataset [27], the proportions between the characters were not balanced. There was neither a balance in amount of samples per character nor in amount of characters per writer. Therefore, we chose to cut back the amount of samples for all the characters to the amount of the character containing the least samples. In the used dataset, this was the case for the letter 'J', containing only 126 samples. Due to wanting as many training samples as possible, a ratio of 100 training samples to thirteen test samples to thirteen validation samples was chosen. Thus, a training set of 2400 samples and a test and validation set of 312 samples remained. However, thirteen test samples per character was quiet scant. Therefore, the samples that remained unused after balancing the dataset were added to the test set, in order to obtain a larger test set with more variation.

It was not possible to also balance between the di erent writers of the characters [23]. There was no balance in the rst place in the used handwritten character dataset: Certain writers had not written certain characters at all, and other characters a lot.

3.4 Data augmentation methods

The data augmentation was done by creating new datasets for each variable of each data augmentation method. For example, a dataset was created for ten degree clockwise rotation, another one for twenty degree clockwise rotation, and so on. The handwritten character training set of 2400 samples was taken, and each of those 2400 training samples was augmented using one of the speci ed methods below. This resulted in 2400 new, unique images. These 2400 new training images could be combined into a new dataset together with the 2400 original training images, creating 4800 sample augmented training set.

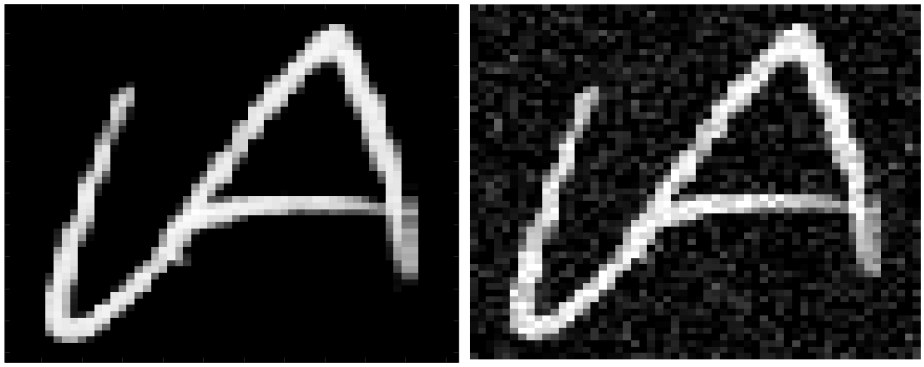
When comparing the results of the augmented dataset to a baseline, either the original training set of 2400 samples can be used, or a training set which has been modi ed to contain the same amount of samples as the augmented

dataset. This modi cation is done by adding the same original training samples again, such that when one has for example an augmented dataset containing 4800 samples, one has a control dataset containing original 4800 samples as well. This choice was made to rule out that an improved classi cation accuracy would only be due to an increased amount of training samples. The enlargement of the baseline from 2400 samples to 4800 samples or larger caused an increase of 0.6 to 1.0% in classi cation accuracy for *ccpccpccp* and an increase of 1.4 to 1.7% in classi cation accuracy for *cpcp* .

3.4.1 Addition of noise

White Gaussian noise was used to augment the images for the handwritten character database. This choice was supported by the paper of Zur et. al (2009) [29], in which the use of white Gaussian noise was shown to reduce over tting.

The function imnoise from the Matlab Image Processing Toolbox [1] was used for noise addition. This function provides a white Gaussian noise functionality. It takes its noise samples pseudo-randomly from the standard normal distribution, with an adjustable variance and a mean of 0. The values in the original image are rescaled in order to t between 0 and 1, which allows for superimposing the noise samples over the original image. The function imnoise achieves this by using the function b = a + $sqrt(variance)$ $randn(size(a))$ + $mean$. The white noise samples are independent and identically distributed. Figure 7b shows an A that is augmented with white Gaussian noise with a variance of 0.05, with the original image next to in in Figure 7a.



(a) An original A from the handwritten character dataset.

(b) An A augmented with white Gaussian noise, a variance of 0.05.

Figure 7: Noise example.

3.4.2 Rotation

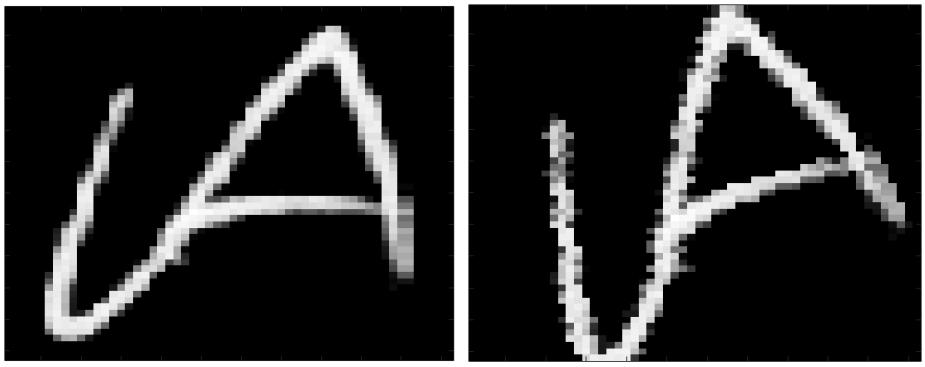
In the MATLAB Image Processing Toolbox [1], the function imrotate(im,angle) was used. A negative angle represents a clockwise rotation and a positive angle

represents a counterclockwise rotation. The following tranformation matrix multiplication is performed for each coordinate:

$$(x_j, y_j, 1) = (x_i, y_i, 1) \begin{pmatrix} cos(\theta) & sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

in which $(x_i, y_i, 1)$ represents the coordinate before the rotation transformation and $(x_j, y_j, 1)$ represents the coordinate after the rotation transformation. A positive angle θ represents a degree of counterclockwise rotations, whereas a negative angle θ represents a clockwise one. The new coordinate is therefore: $(x_j, y_j, 1) = (x_i \quad cos(\theta) \quad y_i \quad sin(\theta), x_i \quad sin(\theta) + y_i \quad cos(\theta), 1)$.

Figure 8b shows an example of a rotated image, against an original image in Figure 8a.



(a) An original A from the handwritten character dataset.

(b) An A augmented with twenty degree counterclockwise rotation.

Figure 8: Rotation example.

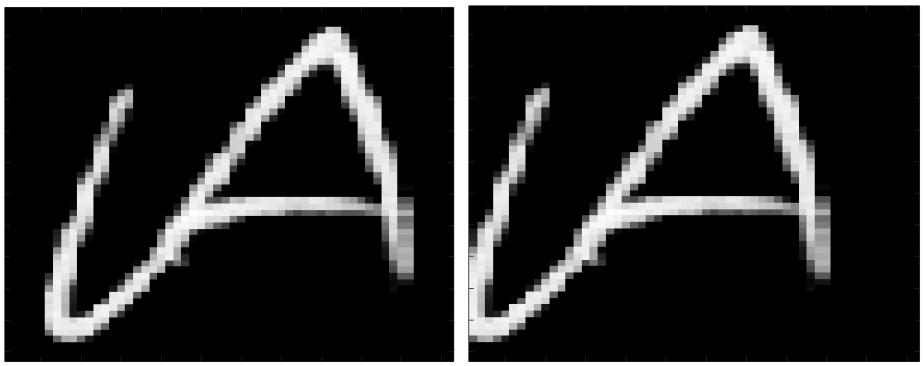
3.4.3 Translation

In the MATLAB Image Processing Toolbox [1], the function for translating is imtranslate. The handwritten character dataset used in this research consists of 2D images, so the translation vector consists of 2 elements. The first of these elements represents the amount of pixels the image is being translated horizontally, the second represents the amount of pixels the image is being translated vertically. The tranformation matrix multiplication is performed for each coordinate:

$$(x_j, y_j, 1) = (x_i, y_i, 1) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{pmatrix},$$

in which $(x_i, y_i, 1)$ represents the coordinate before the translation transformation and $(x_j, y_j, 1)$ represents the coordinate after the translation transformation. t_x and t_y stand for the translation vector for x and y coordinate, respectively. This results in $(x_j, y_j, 1) = (x_i + t_x, y_i + t_y, 1)$. Figure 9b shows a

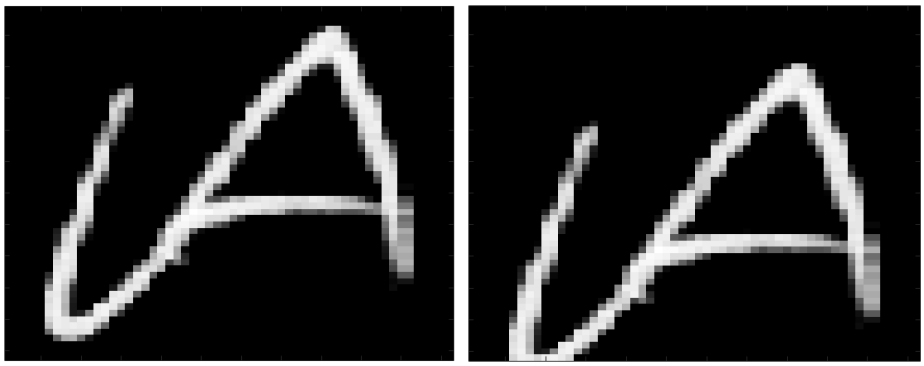
horizontally translated image, compared to the original in Figure 9a. A vertically translated image can be seen in Figure 10b.



(a) An original A from the handwritten character dataset.

(b) An A augmented with horizontal translation, 6 pixels leftward.

Figure 9: Horizontal translation example.



(a) An original A from the handwritten character dataset.

(b) An A augmented with vertical translation, 6 pixels downward.

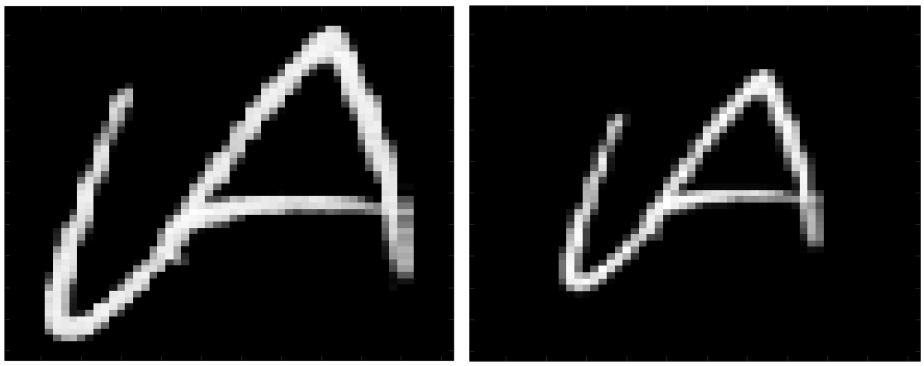
Figure 10: Vertical translation example.

3.4.4 Scaling

The MATLAB Image Processing Toolbox [1] garnered a function for scaling as well, namely imresize. In order to perform this augmentation, the following tranformation matrix multiplication is performed for each coordinate:

$$(x_j, y_j, 1) = (x_i, y_i, 1) \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

in which $(x_i, y_i, 1)$ represents the coordinate before the scaling transformation and $(x_j, y_j, 1)$ represents the coordinate after the scaling transformation. s_x and s_y stand for the scaling factor for x and y coordinate, respectively. So $(x_j, y_j, 1) = (x_i \quad s_x, y_i \quad s_y, 1)$. Figure 11b shows a scaled image, compared to its original in Figure 11a. The A is scaled 0.7 the size of the original.



(a) An original A from the handwritten character dataset.

(b) An A augmented with scaling, with a factor of 0.7.

Figure 11: Scaling example.

3.4.5 Shearing

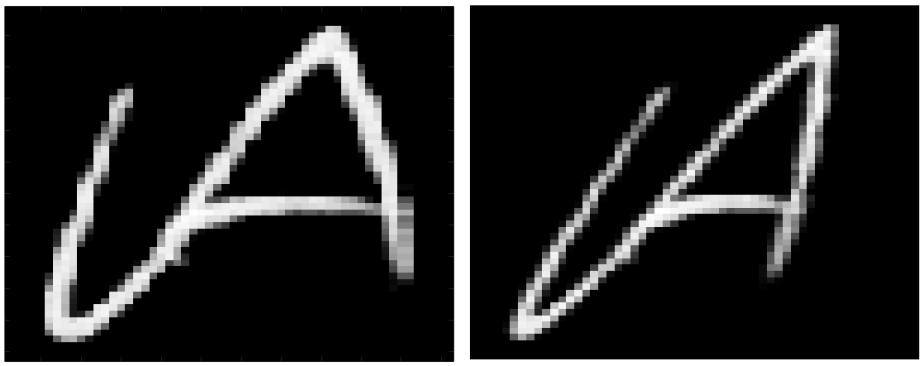
From the MATLAB Image Processing Toolbox [1], the function imwarp was used to apply shearing. Each coordinate (x,y) is mapped to (x+my,y), and this is done by the following matrix multiplication for each coordinate $(x, y, 1)$:

$$(x_j, y_j, 1) = (x_i, y_i, 1) \begin{pmatrix} 1 & 0 & 0 \\ m & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

in which $(x_i, y_i, 1)$ represents the coordinate before the shearing transformation and $(x_j, y_j, 1)$ represents the coordinate after the shearing transformation, the matrix represents the transformation matrix with m being the shearing factor. As such each $(x_j, y_j, 1) = (x_i + m \quad y_i, y_i, 1)$. Figure 12b shows a sheared image, compared to its original in Figure 12a.

3.5 Bayesian linear framework adapted prior

In order to optimize the bayesian linear framework used by Schoenmakers et al. (2015) [22], the data augmentation methods that proved fruitful in the convolutional neural network tests were used to augment the multimodal prior. The prior is multimodal because it consists of, in this case, 6 letter categories, namely B, R, A, I, N and S. Each category consisted of 700 unique prior images, not previously used in the encoding step. Similarly to the augmenting of the training set for the convolutional neural networks, 700 augmented prior images



(a) An original A from the handwritten character dataset.

(b) An A augmented with shearing, with a factor of -0.5 (clockwise).

Figure 12: Shearing example.

were added to the 700 original prior images, creating a new prior consisting of 1400 images. This augmented prior was then integrated into the decoding step. The prior containing only the 700 original images and no augmented images was used as a baseline.

3.6 Experimental setup

In this research, a data augmentation method is only called e cient when its e ect holds on both the cpcp and the ccpccpccp .

Training sets were augmented with rotation, translation, scaling, shearing and noise addition. To train and test rotation, the chosen parameters were: Degrees from ve to 30, in steps of ve, both clockwise and counterclockwise. We decided to distinguish between horizontal and vertical translation. The chosen amounts of translation were three, six and nine pixels horizontally and three and six pixels vertically. After rotation and translation, the resulting images were cropped in order to t the same image space as the original images. With larger amounts of rotation (for example thirty degrees), the outer layer of a character could be partially cropped o . However, this did not hurt the results nor did it decrease the recognizability of the character. The same was the case for larger amounts of translation (for example nine pixels).

Characters were only scaled smaller and padding was added around the character to retain the same image size as the original images. Characters were scaled down from 0.9 to 0.2 times their original size. Characters were not scaled up because this resulted in having to augment the validation and test set as well. This was the case because, when the training set is, for example, 61x61 pixels, the validation and test set are still the original size, namely 56x56. This posed

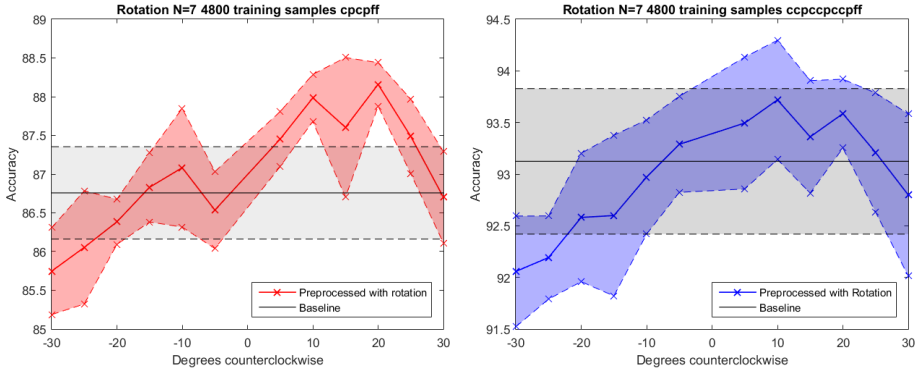
a problem because the image database that the convolutional neural network in MatConvNet accepts is supposed to be equal in size for the training, validation and test set. Shearing factors of 0.2, 0.5 and 0.7 were chosen in both directions, meaning clockwise and counterclockwise.

The results of the Bayesian linear framework are measured as the correlation between the original shown character images and the from brain data reconstructed images.

4 Results

4.1 Results on the convolutional neural networks

4.1.1 Rotation



(a) Rotation results on the small CNN. (b) Rotation results on the large CNN.

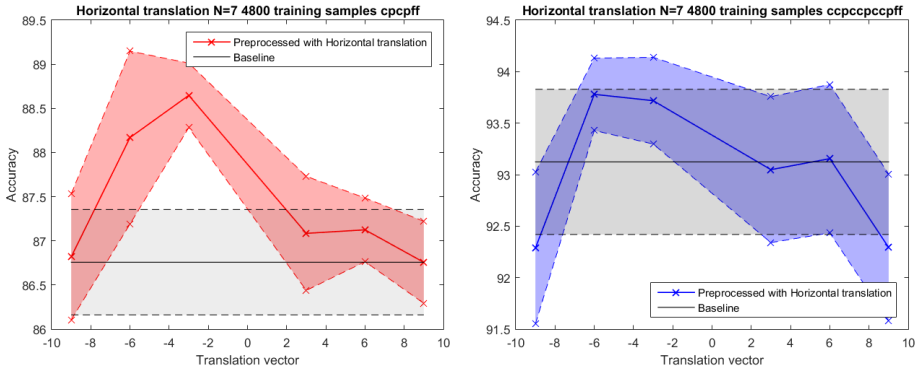
Figure 13: Rotation results

As can be seen in Figure 13a as well as Figure 13b, there is a clear trend toward counterclockwise rotation, especially at ten and twenty degrees, for both *cpcp* as well as *ccpccpccp* . Clockwise rotation does not improve classi cation accuracy for the handwritten character dataset.

4.1.2 Translation

Figure 14a as well as Figure 14b show a clear trend toward a leftward translation, especially at three or six pixels, for both *cpcp* as well as *ccpccpccp* . Rightward translation does not decrease nor increase classi cation accuracy for the handwritten character dataset.

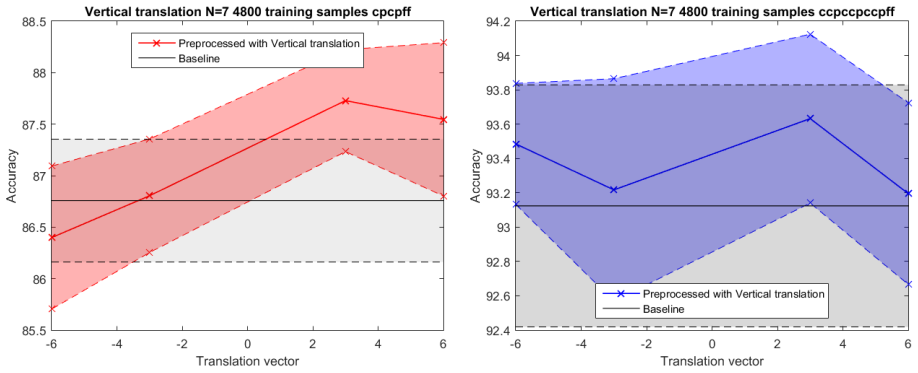
Figure 15a as well as Figure 15b show a slight trend toward three pixels downward translation, for both *cpcp* as well as *ccpccpccp* . Upward translation performs similarly to the baseline.



(a) Horizontal translation results on the small CNN.

(b) Horizontal translation results on the large CNN.

Figure 14: Horizontal translation results



(a) Vertical translation results on the small CNN.

(b) Vertical translation results on the large CNN.

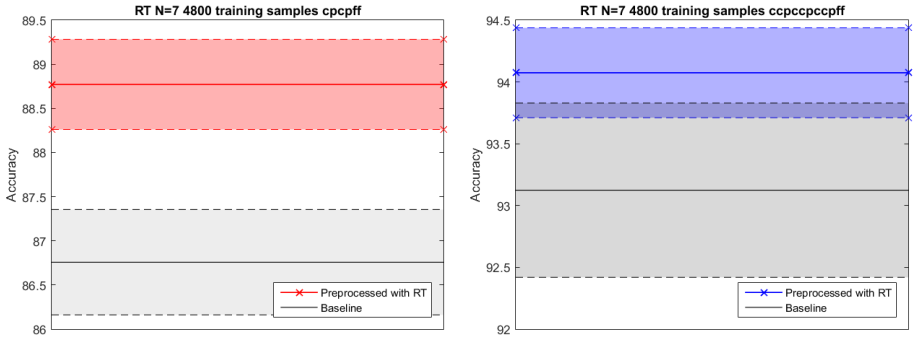
Figure 15: Vertical translation results

4.1.3 Rotation & Translation

This augmented dataset combined rotation and translation, by rst rotating original images ten degrees counterclockwise and then translating them six pixels to the left. As can be seen in Figure 16a, this resulted in a signi cant increase in classi cation accuracy for *cpcp* . As for *ccpccpccp* , there was an increase in classi cation accuracy though not signi cant, as can be seen in Figure 16b.

4.1.4 Rotation, Translation, Rotation & Translation

This augmented dataset was preprocessed with one part ten degrees counter-clockwise rotation, one part six pixels leftward translation and one part ten

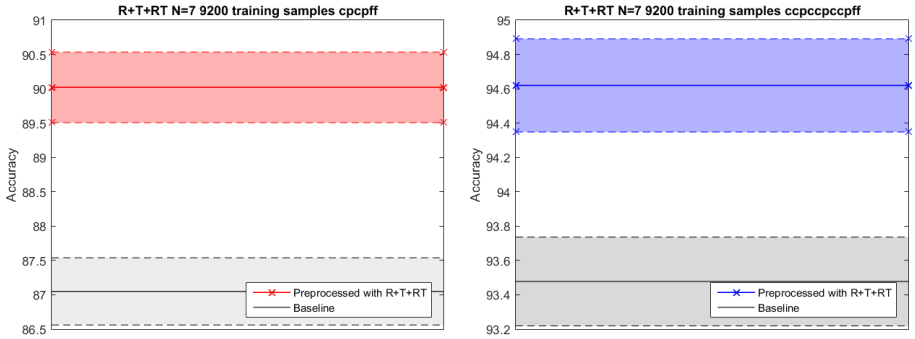


(a) Rotation & translation results on the small CNN.

(b) Rotation & translation results on the large CNN.

Figure 16: Rotation & translation results

degrees counterlockwise rotation and six pixels leftward translation combined, resulting in a four times enlarged augmented dataset. Figure 17a as well as Figure 17b show a signi cant increase in classi cation accuracy, for both *cpcp* as well as *ccpccpccp* .



(a) Rotation, translation, rotation & translation results on the small CNN.

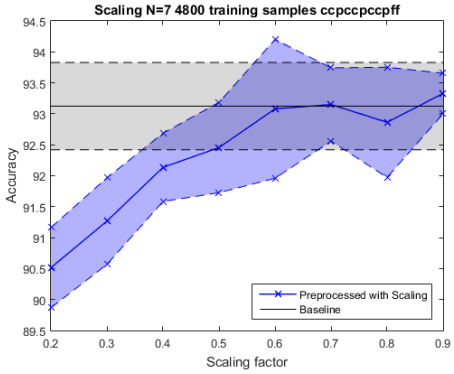
(b) Rotation, translation, rotation & translation results on the large CNN.

Figure 17: Rotation, translation, rotation & translation results

4.1.5 Other methods

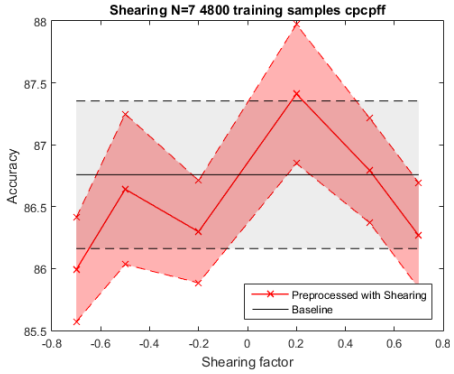
Shearing peaked at shearing factor 0.2 in the *cpcp* . However, this did not hold for the *ccpccpccp* , as there was a slight peak at -0.2. Neither peaks surpassed con dence intervals of the baseline. See Figure 19a and 19b.

(a) Scaling results on the small CNN.

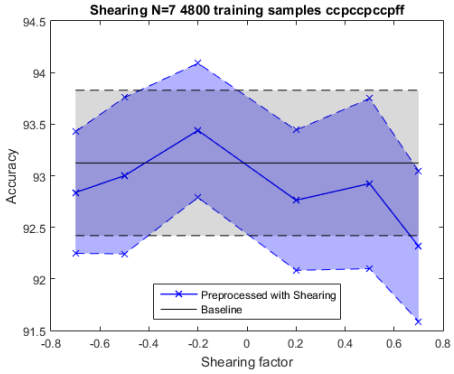


(b) Scaling results on the large CNN.

Figure 18: Scaling results



(a) Shearing results on the small CNN.



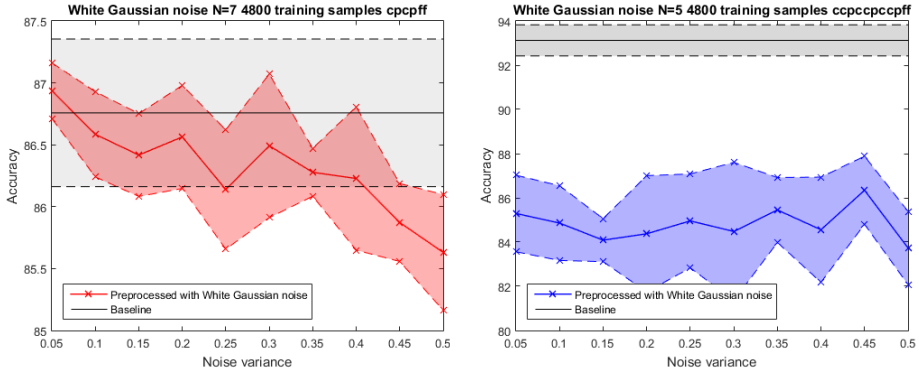
(b) Shearing results on the large CNN.

Figure 19: Shearing results

Noise addition performed equally unsuccessfully in *cpcp* as well as *ccpccpccp* . It performed particularly badly in *ccpccpccp* , with classi cation accuracies far below the baseline con dence intervals. See Figure 18a and 18b.
Lastly, scaling contained one slight peak in classi cation accuracy for factor 0.8 for the *cpcp* , but this did not hold for *ccpccpccp* . Again, none of the peaks surpassed the con dence intervals of the baseline. See Figure 20a and 20b.

4.2 Results on the linear Bayesian framework

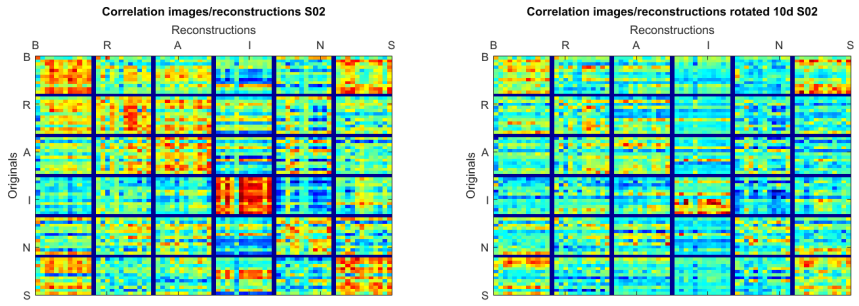
In the case of Figure 21b, the prior was augmented with ten degrees counter-clockwise rotation. The correlations lowered compared to the correlations using the original prior, see Figure 21a. Correlations from the augmented prior are lower overall, but aside from that, there are no relatively higher correlations



(a) White Gaussian noise results on the small CNN.

(b) White Gaussian noise results on the large CNN.

Figure 20: White Gaussian noise results



(a) Correlation between original images and reconstructed images for test subject 02 using the original prior. A

(b) Correlation between original images and reconstructed images for test subject 02 using a prior augmented with 10 degree counterclockwise rotation. A

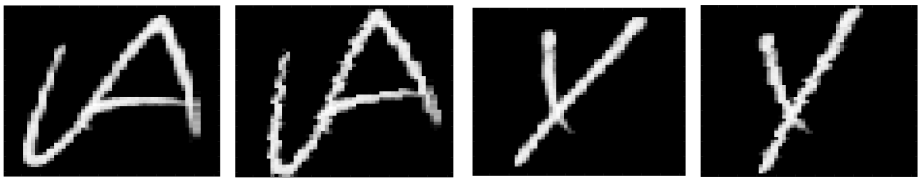
Figure 21: Bayesian linear framework results

at the diagonal. So there is no higher correlation between reconstructions and their corresponding original characters. As can be seen in Figure 21b, only the character I seems to have higher correlation between reconstructions and original characters. the reconstructions of the character B seem to correlate equally much with the original images of character S, if not more. The same goes for reconstructions of S with respect to original images of character B, respectively.

This pattern in correlation was seen regardless of the data augmentation method used. Rotation, translation and a combination of both was used to augment the prior, with ten degrees counterclockwise and six pixels leftward

respectively.

5 Discussion



(a) Original A (b) Ten degree counterclockwise rotated A (c) Original Y (d) Ten degree counterclockwise rotated Y

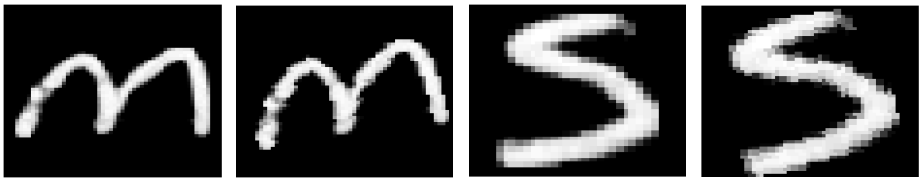
Figure 22: Deskewing of v-shapes

For the handwritten character dataset used [27], the best data augmentation can be performed by enlarging the balanced training set four times. This is done by augmenting the originals using rotation ten degrees counterclockwise, augmenting the originals using translation of six pixels to the left and augmenting the originals using rotation ten degrees counterclockwise and using translation of six pixels to the left. This is a signi cant e ect for this handwritten character dataset due to the augmented dataset falling completely above the 95% con dence intervals of the baseline.

As for the rotation results, when sampling the original handwritten character dataset, it was observed that some characters were slightly angled in a rightward fashion. Additionally, when sampling the rotated handwritten character dataset, it was observed that rightward orientation experienced by some of the characters was corrected. Van der Maaten (2009) [27] proposed deskewing might improve the performance of the handwritten character dataset, which is an explanation for these rotation results. Figure 23 shows an I and an L that

(a) Original I (b) Ten degree counterclockwise rotated I (c) Original L (d) Ten degree counterclockwise rotated L

Figure 23: Vertically deskewing



(a) Original M (b) Ten degree counterclockwise rotated M (c) Original S (d) Ten degree counterclockwise rotated S

Figure 24: Deskewing of bows

were originally sloped to the right but were corrected to stand more upright.

However, not all characters had a rightward orientation that needed to be corrected. The characters that already were upright and not skewed could also bene t from a counterclockwise rotation, because features that were not as prominent in the original character image, are observed more prominently in the counterclockwise rotated image. A di erent kind of deskewing happens in this case, because only certain features (such as a bow or a v-shape) are being deskewed instead of the character image as a whole. Figure 22 shows the deskewing of v-shapes for an A and a Y. Figure 24b shows a deskewing of the bows in a vertical sense, whereas Figure 24d shows a deskewing of the bows in a horizontal sense.

Perhaps making use of skew detection methods such as proposed by Alginahi (2010) [2] would be a good option for e ciently nding the (partial) skew of a handwritten character dataset. He proposed to use Hough transform analyses and orientation sensitive feature analyses, among other things.

As for the remainder of the data augmentation methods: Shearing, white Gaussian noise addition and scaling proved not to be successful methods for the used dataset. For shearing, the peak at 0.2 for *cpcp* was considered quite logical because this was in the same counterclockwise direction that gave positive results for rotation. However, the counterclockwise peak did not hold for *ccpccpccp* . Noise does not perform well on either network, but it performs particularly badly in *ccpccpccp* . Possible explanations are that noise generally does not improve classi cation accuracies for convolutional neural networks, or that this particular dataset does not bene t from white Gaussian noise. After all, Zur et al. 2009 [29] only found such bene cial results for an MLP. It could be the case that scaling up would be more successful for this handwritten character dataset, but there was no room for scaling up in this research.

As for the Bayesian linear framework: When using the original prior, also used in the brainreading research [21], the correlation shows to have most e ect on the diagonal. This is a desirable outcome, because then the character reconstructions from the brain are similar to the same original characters, e.g. a B

correlates strongly with the same original image of B or another image of a B. As can be seen, the correlation between originals and reconstructions using the original prior is highest in the character I. It is not clear why the data augmentation methods resulting from the experiment on the convolutional neural networks do not generate the same positive results for the Bayesian linear framework. A possible explanation could be that the combination of preprocessing methods that proved optimal for the used handwritten character dataset on the convolutional neural network is not similarly optimal for the prior in the Bayesian linear framework. Another possible explanation could be that expanding the prior with augmented samples creates noise for the Bayesian linear framework instead of useful variation. A nal possible explanation could be that the training and test set for the Bayesian linear framework closely resemble each other. The convolutional network, on the other hand, utilizes a validation set which might include samples very di erent from the ones in the training set, therefore promoting robust training. Given more time, it would be interesting to further explore this issue.

However, in the research done by Leonieke van Bulk, results from a convolutional neural network are successfully used as an input in the Bayesian linear framework. This shows that the results from a convolutional neural network and a Bayesian linear framework can be successfully used together.

Furthermore, there are some suggestions for future augmentation of the handwritten character dataset. A di erent composition of the new, augmented datasets could be considered. Instead of creating a dataset for one speci c parameter per data augmentation method, for example 10 degrees counterclockwise for rotation, a dataset could be composed of multiple parameters per data augmentation method. This might especially come in handy when taking into account the results gathered from this research. For example, the results for rotation stated that ten and twenty degrees counterclockwise were most e cient in improving classi cation accuracy for this handwritten character dataset. As of now, the used combinations were all comprised of the most e cient parameters of each data augmentation method, but another method would be to take the top two or top three most e cient parameters of the same data augmentation method. On top of that, combinations could be made between data augmentation methods and their top two or top three parameters. As such, there are many possibilities for e cient dataset enlargement, but there was not enough time in this research to review the many possible combinations.

More enlargement of the augmented datasets could also be given by taking the parameters around the optimal parameter for a data augmentation method. For example, for the most e cient parameter for rotation, ten degrees counterclockwise, a new dataset could be comprised using nine, ten and eleven degrees counterclockwise. This would already result in a four times enlarged dataset. However, it is not known what the results for nine and eleven degrees would be, as testing for rotation was only conducted in steps of ve in this research.

Another thing that could be tested, given the luxury of more time, was a chance to try augmenting the validation set as well. It would be interesting to explore whether a validation set with more samples and variation could reduce over tting more e ciently.

Another thing that could be tried on the used handwritten character dataset is a continuation of testing with vertical translation. Due to time constraints, horizontal translation was picked for further combined testing on the convolutional neural networks due to it performing slightly better in pilot testing. A downward translation of 3 pixels could be used for this further testing..

A last thing Van der Maaten [27] suggested, was to augment the dataset by means of blurring. He believed this would improve classi cation, similarly to rotation.

Conclusively, the handwritten character dataset by Van der Maaten [27] was most successfully enlarged using rotation and translation and a combination of the two. Additionally, the classi cation accuracy improved using this combination of techniques. The various preprocessing methods had no positive e ect on the Bayesian linear framework. Additional research would be needed to explore this issue.

References

[1] *MATLAB and Image Processing Toolbox R2015a*. The MathWorks Inc., Natick, Massachusetts, 2015.

[2] Y. Alginahi. Preprocessing techniques in character recognition, 2010.

[3] C.M. Bishop. Pattern recognition and machine learning. *Springer*, 2006.

[4] W.M. Brown, T.D. Gedeon, and D.I. Groves. Use of noise to augment training data: A neural network method of mineral-potential mapping in regions of limited known deposit examples. *Natural Resources Research*, 12(2), June 2003.

[5] D. Ciresan, U. Meier, J. Masci, L.M. Gambardella, and J. Schmidhuber. High-performance neural networks for visual object classi cation. *CoRR*, abs/1102.0183, 2011.

[6] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classi cation. *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3642 { 3649, 2012.

[7] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control Signals and Systems*, 2:303 { 314, 1989.

[8] Conner DiPaolo. *On-line Machine Learning in Go (and so much more)*. Retrieved from https://github.com/cdipaolo/goml/tree/master/perceptron, obtained in 2016, 2015.

[9] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. 2011.

[10] U. Güçlü and M.A.J. van Gerven. Unsupervised feature learning improves prediction of human brain activity in response to natural images. *PLoS Computational Biology*, 10(8), 2014.

[11] D.H. Hubel and T.N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J Physiol.*, 160:106 { 154, 1962.

[12] A. Krizhevsky, I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 1:1097 { 1105, 2012.

[13] A. Krogh and J.A. Hertz. A simple weight decay can improve generalization. *Advances in Neural Information Processing Systems*, 4:950 { 957, 1992.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278 { 604, November 1998.

[15] Y. LeCun, C. Cortes, and C.J.C. Burges. *THE MNIST DATABASE of handwritten digits*. 1998.

[16] University of Montreal LISA labs. *Convolutional Neural Networks (LeNet)*. Retrieved from http://deeplearning.net/tutorial/lenet.html, obtained in 2016.

[17] V. Nair and G. Hinton. Rectified linear units improve restricted boltzmann machines. 2010.

[18] M.A. Nielsen. *Neural Networks and Deep Learning*. Retrieved from http://neuralnetworksanddeeplearning.com/, obtained in 2016, 2015.

[19] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386 { 408, 1958.

[20] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1: Foundations, 1986.

[21] S. Schoenmakers, M. Barth, T. Heskes, and M.A.J. van Gerven. Linear reconstruction of perceived images from human brain activity. *Neuroimage*, 83:951 { 961, 2013.

[22] S. Schoenmakers, U. Güçlü, M.A.J. van Gerven, and T. Heskes. Gaussian mixture models and semantic gating improve reconstructions from human brain activity. *Frontiers in Computational Neuroscience*, 8(173), 2015.

[23] L.R.B. Schomaker and L.G. Vuurpijl. Forensic writer identication: A benchmark data set and a comparison of two systems. Technical report, Nijmegen Institute for Cognition and Information, University of Nijmegen, 2000.

[24] Ciumac Sergiu. *Financial Predictor via Neural Network*. Retrieved from http://www.codeproject.com/Articles/175777/Financial-predictor-via-neural-network, obtained in 2016, 2011.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from over tting. *Journal of Machine Learning Research*, 15:1929 { 1958, 2014.

[26] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. *JMLR:WCP*, 28, 2013.

[27] L.J.P. van der Maaten. A new benchmark dataset for handwritten character recognition. *TiCC TR*, 2009 { 002, 2009.

[28] A. Vedaldi and K. Lenc. *MatConvNet { Convolutional Neural Networks for MATLAB*. Proceeding of the ACM Int. Conf. on Multimedia, 2015.

[29] R.M. Zur, Y. Jiang, L.L. Presce, and K. Drukker. Noise injection for training arti cial neural networks: A comparison with weight decay and early stopping. *Med. Phys.*, 36(10), 2009.