university of groningen / faculty of mathematics and natural sciences / artificial intelligence

The Effect of pooling the Hidden Activation in a Patch-Based Image Classification System using Multi-Layer Perceptrons

Rogier de Ruijter,
Supervisor: Marco Wiering

Abstract: In this thesis a patch-based image classification system using a multi-layer perceptron (MLP) system is proposed. The patch-based MLP is trained on a set of randomly selected patches from the training images. This method is also used in a design where multiple patch-based MLPs are trained on specific regions of the images, this is called the scoped patch-based MLP system. Both the patch-based MLP and the scoped patch-based MLP system are used as a feature extractor for a classifier as well. The performance of these approaches are tested on the MNIST and CIFAR-10 datasets, where 99.43% and 71.63% classification accuracy are obtained, respectively.

1 Introduction

In this thesis, a technique for training a multi-layer perceptron (MLP) is proposed that is inspired by a theory in vision called ensemble perception, (Haberman and Whitney, 2012). This theory states that you do not identify objects by every detail, but rather by the general structure of objects from the same class. To extract the general structure from the classes of a dataset, we train on randomly selected patches from the training images. Before training, the patches are converted to feature vectors by a histogram of oriented gradients (HOG) descriptor (Dalal and Triggs, 2005). These features are used to train an MLP that classifies the images in a patch-based manner. Classifying in a patch based manner is performed by extracting all the patches from an image and classifying each patch separately. An MLP trained in this manner is called a patch-based MLP, and will be used in two different ways, as a classifier and as a feature extractor. The technique of training a patch-based MLP is similar to training an autoencoder on randomly selected unlabeled patches, with the difference that the training objectives of the two are different. The patch-based MLP learns more discriminative features of the dataset, since the actual classes are learned. Nevertheless, in a way we pre-train a set of convolutions in the form of the hidden units from the patch-based MLP. Therefore we are going to use the patch-based MLP and a pooling function, in a comparable way to a convolutional layer and a pooling layer in a convolutional neural network (CNN). The properties that are similar to the convolutional and pooling architecture in a CNN are: the way each convolution creates a feature matrix by sliding over the image, the pooling operation, and that a feature vector is constructed for a classifier. One of the main differences lies within the comprehensiveness of the convolutional and pooling architecture. In our approach there is one convolutional layer and the pooling operation used reduces each feature matrix to one value. This creates a feature vector, with the size of the hidden units from the patch-based MLP, that will be used by the classifier. In addition, the training of the convolutions and the classifier is split. With this design the error, which is calculated with a loss function, does not have to travel through a classifier before it can adjust the weights of a convolution. This results in a more accurate adjustment, compared to a CNN with one or multiple convolutional layers. This approach is used to determine whether the classification accuracy of the patch-based MLP can be improved. We also introduce the scoped patch-based MLP system. This is designed with a set of patch-based MLPs that are constructed to train on patches from only a specific region in the images. These regions are quadrants that capture, for example, the top-left corner of the images. In this

way a patch-based MLP learns what is important for classifying a particular class based on the information in its region. The scoped patch-based MLP system will also be used to construct features for a classifier.
The classification accuracy of the different patch-based MLP systems are tested on the MNIST, (LeCun, Bottou, Bengio, and Haffner, 2001) and CIFAR-10, (Krizhevsky, 2009) dataset. The effect of pooling the hidden activation of the patch-based MLP and the scoped patch-based MLP system will also be tested on these datasets.
The research question that follows from this is: does training a classifier on the pooled hidden activation improve the performance of the patch-based image classification system?
The layout of this thesis is as follows. In Section 2, the techniques within the Multi-layer Perceptron are explained. In section 3, the architecture is further elaborated upon. Section 4 describes an exploratory analysis. Section 5 presents the results and sections 6 and 7 show the discussion and the conclusion, respectively.

2 Multi-Layer Perceptron

In this section the techniques used in the MLP are outlined. Both the patch-based MLP and the classifier are three-layer MLPs that use feedforward and backpropagation to minimize the error with the cross-entropy function.

2.1 Feedforward

The feedforward of a three-layer MLP is a function $f : R^E \rightarrow R^L$, where the input vector x has size E and the output vector $f(x)$ has size L, so in matrix notation,

$$f(x) = G(b^{(2)} + W^{(2)}h(x))$$

Where, $h(x) = s(b^{(1)} + W^{(1)}x)$ are the hidden activations, $b^{(1)}$, $b^{(2)}$ are the bias vectors and $W^{(1)}$, $W^{(2)}$ are the weight matrices.
W is initialized as follows,

$$-0.1 \leq W(x, y) \leq 0.1$$

With $W(x, y)$ being an arbitrary value in the weight matrix.

The activation function is a ReLU, (R. H. R. Hahnloser and Seung, 2000).

$$s(z) = max(0, z)$$

The MLP uses a soft-max function on the output activations.

$$G(a)_i = e^{a_i - max(a)} / \sum_{j=1}^{L} e^{a_j - max(a)}$$

Where a_j are the output activations and $G(a)_i$ is the output probability of an output unit. The soft-max function used is a slight variation from the original, the maximum output activation found in a is subtracted from each output activation. This is done to cope with large values in the exponential function in C++ [1].

2.2 Additional techniques

Below the normalization process can be seen.

$$x = \frac{x - min(\forall x_{train})}{max(\forall x_{train}) - min(\forall x_{train})}$$

With x, being any arbitrary input value in the system, and $\forall x_{train}$ being all input values in the training set.
Dropout, first introduced by Hinton, Srivastava, Krizhevsky, Sutskever, and Salakhutdinov (2012), is used where a system has the tendency to overfit. The formula is as follows,

$$h(x)_i = \begin{cases} h(x)_i \text{ if } p > 0.5 \\ 0 \text{ otherwise} \end{cases}$$

Where $h(x)_i$ is the hidden activation of a hidden unit. After dropout is applied to all the hidden units, the dropped hidden activation is spread out over the still active hidden units in the following manner,

$$h(x) = h(x) * \frac{h_{total}}{h_{active}}$$

Where h_{total} is the sum of the hidden activations before dropout and h_{active} is the sum of the hidden activations after dropout.
For speeding up the learning process momentum, Polyak (1964), is used,

$$W(t) = W(t) + \delta L(W(t-1))$$

[1] Further explanation can be found in appendix A.1

Where L is the loss function and $\delta = 0.9$. The last additional technique is a learning rate decay of 0.02 after each epoch.

3 Architecture

In this section the structure of the architecture will be explained. Firstly, the patch-based MLP system and the scoped patch-based MLP system are explained. Secondly, the classifier system is further elaborated upon.

3.1 Patch-based MLP system

The patch-based MLP system will be explained in the following order: the data transformation, the training phase, and the classification phase.

3.1.1 Data transformation

The data transformation is split into two parts, the patch extraction and the feature extraction.
Patch extraction is the process of extracting a sub-image from the original image. This sub-image is called a patch and will hold the same label as the original image. The amount of patches that one image contains (n_p) is a combination of the image width (i_w) and height (i_h), patch width (p_w) and height (p_h), and scanner stride (s). The following formula is used to calculate the amount of patches per image,

$$n_p = \Big(\frac{i_h - p_h}{s} + 1\Big) * \Big(\frac{i_w - p_w}{s} + 1\Big)$$

During experimentation we found that large patches with respect to the image give the best results. The reason for this is that the greater the patch, the more information about an image it contains and since we are constructing a patch-based classifier, the smaller the patch, the more challenging it will be to predict the particular class in this architecture. This is not always the case since a small patch can inspect a specific region that is very useful for classifying a particular class for example the ears of a cat. Nevertheless, with randomly selected small patches there will be a substantial amount of patches in the training set that are difficult to classify based on the information in them. The downside of using large patches is that the systems deal with high dimensional data, which results in large systems that take a long time to train. When the raw pixels of gray scaled patches are used the input size is as follows, $p_w \times p_h$, which generates a large set of inputs for an MLP. The solution for this problem is to convert the raw pixels into smaller and more descriptive features. This is done with a HOG descriptor (Dalal and Triggs, 2005).
Finding an optimal feature size is an interaction between four parameters, patch size, HOG cell size, HOG stride, and number of bins per cell. In an optimal scenario, a combination of settings is found that yields the most descriptive power and the smallest amount of inputs.
The HOG stride is always set equal to the size of the cell to create non overlapping cells.
The patches do not use padding, as a consequence, the usable patch width and height decrease by two pixels.

3.1.2 Training phase

In the training phase the training images are split into a training and validation set in a 90/10 fashion. For the training phase of the patch-based MLP an n amount of random patches are extracted from the training set. These randomly extracted patches are converted to features by a HOG descriptor and run through the patch-based MLP for a set amount of epochs. The validation set is used to test how well the system classifies a set of images if the training process would stop at that moment, this is performed every five epochs and will terminate the training process if the validation error is below 0.001. Validating the images works in the same manner as classifying an image, how this is done is explained in section 3.1.3.
The idea of taking random patches from the training images is inspired by the psychological phenomena in vision called, ensemble perception (Haberman and Whitney, 2012). The theory is that you do not identify objects by every detail of them, but rather by the general structure of objects from the same class. By training the system on random partial images we extract the statistical regularity of the classes in the training set. How well a particular set of patches works to train the patch-based MLP is dependent upon how well the test set variability is captured within the patches. It is therefore desirable to extract many patches from the training

set to get a proper statistical regularity and to average out the amount of luck involved in selecting a 'good' set of patches.
An exploratory analysis is performed to find the optimal settings for the patch-based MLP systems.

3.1.3 Classification phase

Classification works as follows: all patches are extracted from the to be classified image and converted to feature vectors by a HOG descriptor. Individually, each feature vector is feedforwarded through the patch-based MLP. After each feedforward the output probabilities are summed up to eventually choose the class with the highest probability as the winner.

3.2 Scoped patch-based MLP system

In this section the scoped patch-based MLP system is introduced. The idea is to expose a set of patch-based MLPs to specific regions of the images. This means that each image has to be divided into quadrants. The formula for calculating how many quadrants per image there are is as follows: $s \times s$. If s has the value 2 there are 4 quadrants, where each quadrant is equal in size. Each quadrant gets a patch-based MLP assigned to it that will train on patches only from that region. In practice, the only workable value for s is 2, because large patches are used which means there is too much overlap between adjacent squares if s is large than 2. If these values would be used we lose the idea of training on information from one region. A patch is assigned to a particular square if the middle of the patch falls into that square. It does not matter if parts of the patch overlap into adjacent squares, as long as the middle of the patch falls into the assigned square. This is displayed in figure 3.1. There is one special case, when a patch middle falls onto a separation line. When the middle is on the vertical separation line the patch is assigned to the right squares and if the middle is on the horizontal separation line the patch is assigned to the bottom squares.
In the scoped patch-based MLP system, the training phase is in essence the same as the not scoped patch-based MLP system, with the difference that when a random patch is extracted for the training images it is assigned to the training set of the patch-based MLP that covers the region where it came from. A similar process is performed for the validation set. The process works as follows, when a validation image is broken down into patches, each patch gets assigned to the validation set of the region were it came from. In this manner the validation error is a value that explains how well the patch-based MLP can classify the images in the validation set based on the patches from the corresponding square.
The classification phase is almost identical to the validation phase with the difference that the output probabilities of all the $s \times s$ patch-based MLPs are summed up linearly.

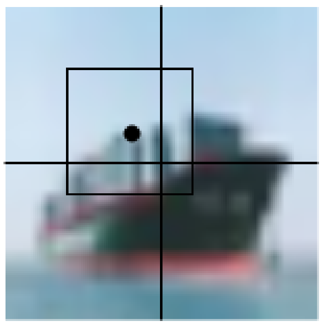
Figure 3.1: Visualization of a patch being extracted from an image in the scoped patch-based MLP system. The dot in the middle of the square represents the middle of the patch.

3.3 Classifier system

In this section the classifier system is explained. This is done in the following order: The construction process of the feature vectors with the patch-based MLP systems, and the training and classification process of the classifier.

3.3.1 Feature vector construction

The patch-based MLP system creates the features as follows. All the patches from one image are extracted. Each patch is converted to a feature vector by a HOG descriptor and feedforwarded through a patch-based MLP system. After a feedforward, each hidden activation is compared to its previous stored hidden activation via a pooling function. When all the feature vectors from an image have been run through the patch-based MLP, a new input feature

vector for the classifier is created.
This means that the amount of hidden units will determine the input units of the classifier. Therefore, if the classifier uses 1 patch-based MLP its input units will be equal to the amount of hidden units used in that patch-based MLP. When 4 scoped patch-based MLPs are used, each individual patch-based MLP will create a pooled hidden activation feature vector, which are appended together to create one feature vector, therefore the amount of input units from the classifier is the size of the four hidden unit feature vectors combined. The size of the training set is determined by the amount of training images in the dataset.
The pooling operations that are used during experimentation are min, average, max, and average+max pooling. Where with average+max pooling, the average and max features are combined into one feature vector.

3.3.2 Training and classification phase

To start off, the training images are converted into feature vectors in the manner described in section 3.3.1. Then the feature vectors are split into train/validation with a validation dataset of 10% from the original dataset. The training process consists of training the classifier for a set amount of epochs on the feature vectors in the training set. Every five epochs the system determines the validation error by classifying the validation set. If the validation error is below 0.001 the training process stops.
In the classification phase the test images are converted to feature vectors as described in section 3.3.1. Each feature vector is then classified to determine which class it belongs to.

4 Exploratory Analysis

Below is the motivation for which settings are used in the systems. For both system types there is one parameter that will be varied in a systematic manner. The patch-based MLP systems will vary the amount of random patches in the training set and the classifier systems will vary the amount of hidden units in the MLP. The other settings are found in a non-systematical manner. The systems that are going to be explored are:

I 1 patch-based MLP

II 4 scoped patch-based MLPs

III Training an MLP on the average pooled hidden activations of 1 patch-based MLP

IV Training an MLP on the average pooled hidden activations of 4 scoped patch-based MLPs

A few notes regarding the systems and the experiments:

I The patch-based MLPs that are used for the feature vector extraction process are *saved patch-based MLP systems*[2]. This is used to see the actual effect of the different hidden unit amounts.

II The experiments were run on an AMD Opteron(TM) Processor 6276, CPU 64bit, with code written in C++. The values in the MLPs are stored in floats.

III The results are one fold and training/testing is done on the true train/test set.

4.1 MNIST

MNIST is a handwritten digit dataset with 60.000 train images and 10.000 test images. All the images are 28x28 pixels and the digit itself is white with a black background. They are re-sized to 38x38 pixels.

4.1.1 System settings

For experiments on MNIST a patch size of 32x32, with a HOG descriptor that has a cell size of 5×5, and 9 bins per cell gives the best accuracy. This yields 324 input units.
For the patch-based MLP systems and the classifier systems the following settings are used: The learning rate is 0.01, the amount of epochs is 20, the scanner stride is 1, and the regularization technique dropout in combination with momentum is used.

[2]How a saved patch-based MLP is constructed can be found in section A.2.

4.1.2 Patch-based MLP systems

Below the exploratory analysis on MNIST for the patch-based MLP systems is displayed. Here the search is for the amount of random patches that give the highest classification accuracy. The following systems are explored:

I 1 patch-based MLP on MNIST.

II 4 scoped patch-based MLPs on MNIST.

With the following settings:

(a) 324 input units, 500 hidden units, 10 output units.

(b) 4 × (324 input units, 400 hidden units, 10 output units).

In the figure below the performance of the 1 patch-based MLP with different amounts of random patches is displayed.

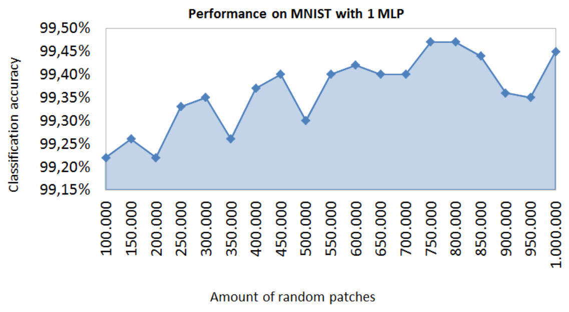

Figure 4.1: The classification accuracy versus different amounts of random patches in the training set, with respect to 1 patch-based MLP, performed on the MNIST images.

In figure 4.1, it can be seen that there is a positive general trend towards more random patches and a higher classification accuracy. This trend stagnates around the 750.000 random patches mark. The spikiness of the line is a consequence of the random patch selection. The amount of random patches that is chosen is 750.000. This system costs around 8 hours to train. Below the performance from the 4 scoped patch-based MLPs with different amounts of random patches is displayed.
In figure 4.2, a positive trend towards a higher classification accuracy with more randomly selected patches in the training set can be seen. In the same figure the maximum number of random patches is 2.000.000. This number is greater than the maximum number of random patches from the 1 patch-based MLP, because the patches have to be divided. What can be seen in this graph is that after the 1.000.000 mark the classification accuracies become fairly stable.
The amount of random patches that is chosen is 1.100.000. This system takes around 20 hours to train.

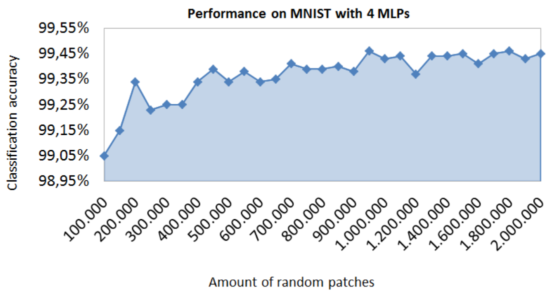

Figure 4.2: The classification accuracy versus different amounts of random patches in the training set, with respect to 4 scoped patch-based MLPs, performed on the MNIST images.

4.1.3 Pooling hidden activations

Below the exploratory analysis on MNIST of the classifier systems is displayed. For the classifier, an MLP, we try to find the amount of hidden units that yields the largest classification accuracy. The following classifier systems are used:

I An MLP using feature vectors constructed by 1 patch-based MLP on MNIST.

II An MLP using feature vectors constructed by 4 scoped patch-based MLPs on MNIST.

With these settings:

(a) 500 input units, x hidden units, 10 output units.

(b) 1600 (4 × 400) input units, x hidden units 10 output units.

Below, the classification accuracy of an MLP with different amounts of hidden units is displayed. This

MLP is trained upon feature vectors constructed by 1 patch-based MLP.

Hidden units	Accuracy test set(%)
100	99.45
200	**99.45**
300	99.45
400	99.44
500	99.43
600	99.40
700	99.41
800	99.39
900	99.44
1000	99.42

Table 4.1: The classification accuracy versus different amounts of hidden units, with respect to a classifier that uses feature vectors constructed by 1 patch-based MLP via average pooling, performed on the MNIST images. The saved patch-based MLP has a classification accuracy of 99.44%.

Below, the classification accuracy of an MLP with different amounts of hidden units is displayed. This MLP is trained upon feature vectors constructed by 4 scoped patch-based MLPs.

Hidden units	Accuracy test set(%)
100	99.39
200	99.39
300	99.39
400	99.39
500	**99.41**
600	99.35
700	99.37
800	99.40
900	99.40
1000	99.37

Table 4.2: The classification accuracy versus different amounts of hidden units, with respect to a classifier that uses feature vectors constructed by 4 scoped patch-based MLPs via average pooling, performed on the MNIST images. The saved patch-based MLPs have a classification accuracy of 99.45%.

In table 4.1, it can be seen that the first three values in the table yield the same classification accuracy on the test set. 200 hidden units are chosen, because it also has the highest classification accuracy on the validation set. The creation of the feature vectors and training the classifier is accomplished in around 2 hours.
In table 4.2, it can be seen that using 500 hidden units gives the best result. Creating the feature vectors and training the classifier is achieved in around 17 hours.

4.2 CIFAR-10

CIFAR-10 is a dataset with 10 different real life colored objects. A few examples are horses, dogs, and ships. It consists of 50.000 training images and 10.000 test images. The images are 32 by 32 pixels. For experimentation they are resized to 38 by 38 and converted to gray scale.

4.2.1 System settings

For CIFAR-10 a smaller HOG cell size is used to catch the complex characteristics. Therefore, the cell size is set to 3×3. The patch size is set smaller to reduce the dimensionality and to not capture too much information in one patch. This results in a patch size of 26x26. Which after running it through a HOG descriptor results in a feature vector of 576 input values. For the patch-based MLP systems on CIFAR-10 the following settings are used: the learning rate is 0.01, epochs are set to 30, and the scanner stride is 1. When training the patch-based MLP systems on CIFAR-10, there is the problem that the training error does not reach a training error lower than 0.1. As a result of this, dropout is not used and momentum is. This decreases the training error to around 0.001. The settings for the classifier systems differs on one point, that momentum is not used and dropout is, because the system has a tendency to overfit.

4.2.2 Patch-based MLP

Below the exploratory analysis on CIFAR-10 for the patch-based MLP systems is displayed. Also here the search is for the amount of random patches that gives the highest classification accuracy. The following systems are explored:

I 1 patch-based MLP on CIFAR-10.

II 4 scoped patch-based MLPs on CIFAR-10.

With the following settings:

(a) 576 input units, 800 hidden units, 10 output units.

(b) 4× (576 input units, 500 hidden units, 10 output units).

In the figure below the performance of the 1 patch-based MLP with different amounts of random patches is displayed.

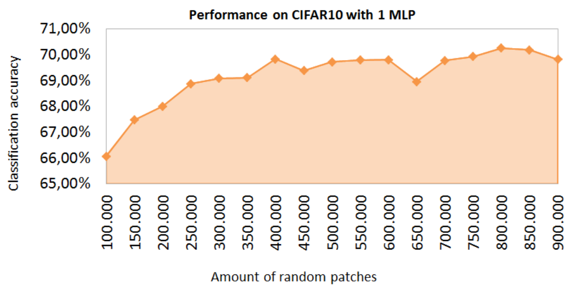

Figure 4.3: The classification accuracy versus different amounts of random patches in the training set, with respect to 1 patch-based MLP, performed on the CIFAR-10 images.

In figure 4.3, there is a slight general trend towards better classification accuracy with more random patches, but after 400.000 random patches there is no significant increase of the classification accuracy anymore. We looked at other possibilities to increase the test set classification score and found that increasing the amount of hidden units has a strong positive effect. Therefore, the amount of hidden units was later increased from 800 to 1500 hidden units. This large amount of hidden units in combination with 950.000 random patches is chosen for the final settings. Experiments on CIFAR-10 have the inclination to take a very long time to train. This system takes around 60 hours to train.
Below the performance from the 4 scoped patch-based MLPs with different amounts of random patches is displayed.

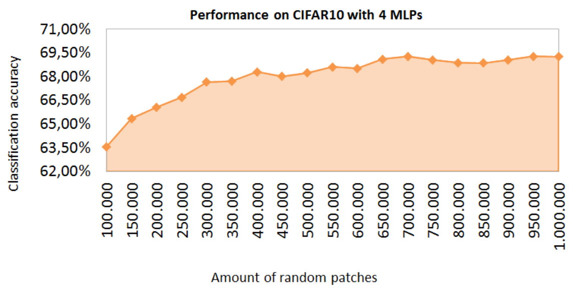

Figure 4.4: The classification accuracy versus different amounts of random patches in the training set, with respect to 4 patch-based MLPs, performed on the CIFAR-10 images

In figure 4.4, we see that there is a general trend towards the more random patches the higher the classification accuracy, which kind of flattens after 800.000 random patches. Also here we found that more hidden units gives a better result, therefore we increased the hidden units to 800 per system. The final settings will use a total of 1.000.000 random patches with 800 hidden units per system. Training this systems takes around 50 hours.

4.2.3 Pooling hidden activations

Below the exploratory analysis of the classifier systems is displayed on CIFAR-10. Also here we are looking for the optimal amount of hidden units. The following systems are explored:

I An MLP using feature vectors constructed by 1 patch-based MLP on CIFAR-10.

II An MLP using feature vectors constructed by 4 scoped patch-based MLPs on CIFAR-10.

With the following settings:

(a) 1500 input units, x hidden units, 10 output units.

(b) 3200 (4×800) input units, x hidden units, 10 output units.

Below, the classification accuracy of an MLP with different amounts of hidden units is displayed. This MLP is trained upon feature vectors constructed by 1 patch-based MLP.

Hidden units	Accuracy(%)
100	71.60
200	71.60
300	71.94
400	71.83
500	**71.99**
600	71.75
700	71.95
800	71.72
900	71.81
1000	71.99

Table 4.3: The classification accuracy versus different amounts of hidden units, with respect to an MLP that uses feature vectors constructed by 1 patch-based MLP via average pooling, performed on the CIFAR-10 images. The saved patch-based MLP has a classification accuracy of 71.69%.

Below, the classification accuracy of an MLP with different amounts of hidden units is displayed. This MLP is trained upon feature vectors constructed by 4 patch-based MLPs.

Hidden units	Accuracy(%)
100	70.52
200	70.31
300	**70.62**
400	70.51
500	70.24
600	70.58
700	70.53
800	70.40
900	70.52
1000	70.48

Table 4.4: The classification accuracy versus different amounts of hidden units, with respect to a classifier that uses feature vectors constructed by 4 scoped patch-based MLPs via average pooling, performed on the CIFAR-10 images. The saved patch-based MLPs have a classification accuracy of 70.63%.

In table 4.3, there are two hidden unit amounts that yield the same accuracy, 500 and 1000. The amount of hidden units that is chosen is 500, because it is the simplest system. The creation of the feature vectors and training the classifier takes around 15 hours. In table 4.4, it can be seen that using 300 hidden units yields the best classification accuracy. The creation of the feature vectors and training the classifier is performed in 18 hours.

4.3 Additional findings

During experimentation we found that adding max pooling and average pooling together into one feature vector gives results that are comparable to the single pooling types. Adding the two pooled feature vectors into one feature vector results in double the amount of input units. The hidden units are double according to the best hidden units found in their system. During the final experiments this pooling type is added.

5 Results

In this section the results are displayed. Firstly, the systems that are used to obtain these results are displayed. Secondly, the classification accuracies obtained will be presented. Within the tables in this section the abbreviations PB-MLP is used for patch-based MLP and SPB-MLP for scoped patch-based MLP.

5.1 Systems

In table 5.1, the systems that are used to obtain the results on MNIST are displayed. For the 1 patch-based MLP, 750.000 randomly selected patches are used for training and for the 4 scoped patch-based MLPs, 1.100.000 randomly selected patches are used. The other settings can be found in section 4.1.1.

Type	Input	Hidden	Output
1 PB-MLP	324	500	10
4 SPB-MLPs[3]	324	400	10
Classifier: 1 PB-MLP	500	200	10
Classifier: 4 SPB-MLPs	1600	500	10

Table 5.1: The MLP systems used to obtain the results on the MNIST dataset.

In table 5.2, the systems that are used for CIFAR-10 are displayed. The 1 patch-based MLP uses 950.000 random patches and the 4 scoped patch-based MLPs uses 1.000.000 random patches. The other settings can be found in section 4.2.1.

Type	Input	Hidden	Output
1 PB-MLP	576	1500	10
4 SPB-MLPs[3]	576	800	10
Classifier: 1 PB-MLP	1500	500	10
Classifier: 4 SPB-MLPs	3200	300	10

Table 5.2: The MLP systems used to obtain the results on the CIFAR-10 dataset.

5.2 Obtained results

All experiments that are displayed in table 5.3, 5.4, 5.5 are 10 fold cross validations.
In table 5.3, the classification accuracies are displayed of the patch-based MLP systems.

	MNIST	CIFAR-10
1 PB-MLP	**99.43 +/- 0.03**	**71.44 +/- 0.38**
4 SPB-MLPs	99.32 +/- 0.04	69.67 +/- 0.13

Table 5.3: Results obtained with 1 patch-based MLP and 4 scoped patch-based MLPs on the MNIST and CIFAR-10 dataset.

In table 5.4 and 5.5, the classification accuracies of the classifier trained on feature vectors created by the patch-based MLP systems are displayed.
The patch-based MLPs used in table 5.3 are saved and reused in the experiments displayed in table 5.4 and table 5.5. This is done to see the actual effect of pooling and to decrease the time needed to run the pooled experiments.

[3]Four of these patch-based MLPs are created that interacts as explained in section 3.2

[4]For this pooling type the input units and the hidden units are doubled according to their corresponding systems. In table 5.1 the corresponding systems for MNIST can be found and in table 5.2 the corresponding systems for CIFAR-10

MNIST	Classifier: 1 PB-MLP	Classifier: 4 SPB-MLPs
Min	94.08 +/- 0.27	99.28 +/-0.03
Average	99.36 +/- 0.05	**99.29 +/- 0.03**
Max	99.35 +/- 0.05	99.28 +/- 0.02
Av+Max[4]	**99.38 +/- 0.04**	99.29 +/- 0.03

Table 5.4: Results on the MNIST dataset obtained with eight separate MLPs trained upon either minimum, average, maximum, or average + maximum feature vectors extracted with 1 patch-based MLP or 4 scoped patch-based MLPs.

CIFAR-10	Classifier: 1 PB-MLP	Classifier: 4 SPB-MLPs
Min	12.99 +/- 0.80	55.78 +/- 0.60
Average	**71.63 +/- 0.40**	**69.64 +/- 0.14**
Max	68.20 +/- 0.31	67.95 +/- 0.33
Av+Max[4]	68.66 +/- 0.50	68.92 +/- 0.24

Table 5.5: Results on the CIFAR-10 dataset obtained with eight separate MLPs trained upon either minimum, average, maximum, or average + maximum feature vectors extracted with 1 patch-based MLP or 4 patch-based MLPs.

6 Discussion

In figure 6.1, four images that are classified wrongly by 1 patch-based MLP on MNIST are displayed.
These digits have one similar property, they are out of shape. Besides that there is not much wrong with them. A possible solution for MNIST can be to make the system more robust towards spatial change by deforming the visible digit part in the patch. This is done image wise in

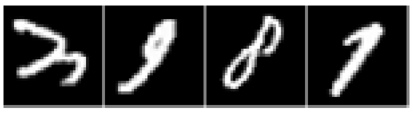

Figure 6.1: From left to right, the actual label $\rightarrow$ the classification , $(3 \rightarrow 2)$,$(9 \rightarrow 8)$,$(8 \rightarrow 1)$,$(1 \rightarrow 7)$

(Cireşan, Meier, Gambardella, and Schmidhuber, 2010), where at the beginning of each epoch the digits would be deformed. It will most likely increase the robustness of the system, as it has also been shown that distortion works on MNIST in (Cireşan et al., 2010), (Cireşan, Meier, Gambardella, and Schmidhuber, 2011), and (Max Jaderberg and Kavukcuoglu, 2015). Deformation of patches could also be used on the CIFAR-10 experiments, nevertheless this will require a more complex 'transformer'. The transformer proposed in Max Jaderberg and Kavukcuoglu (2015) is an option since it can be implemented in a 'standard neural network architecture' and has shown positive results on a similar RGB dataset, the bird dataset CUB-200-2011 (Wah, Branson, Welinder, Perona, and Belongie, 2011).

At the end of our research, when all the final experiments were run we thought of a possible improvement to train the classifier system. The idea is to add the images from the validation set to the training set. This way 90% of the images are images that are also used during the patch-based training and 10% are unseen images. A rerun of the final experiments for MNIST using the classifier system with 1 patch-based MLP shows improvements. The accuracies over 10 runs with training on the validation set is 99.42%, and without 99.36%. This shows potential, where an optimal hyper parameter for the split between the training and validation for the patch-based MLP might result in an even higher score. If there was more time we would find the optimal hyper parameter and re run the experiments where the patch-based MLP systems are used to create feature vectors.

As seen in table 4.3 in the exploratory analysis, when the classifier has 300 hidden units and 1.000 hidden units the same classification is obtained. As shown in, Cireşan et al. (2010), using an MLP with multiple hidden layers achieves the highest classification accuracy. In this system the amount of hidden units is sequentially lowered per layer, starting from a large amount of hidden units. Researching the effect of multiple hidden layers in the classifier can be done in future research.

Many of the systems used during experimentation take a lot of time to train. For example, training a classifier system that uses 1 patch-based MLP on CIFAR-10 costs around 82 hours to train. One of the reasons for this is that the system runs on a CPU. This takes around 50 to 100 times longer, than the now commonly used in image recognition research, GPUs (Cireşan, Meier, and Schmidhuber, 2012, 28-30). Creating a GPU implementation of the framework could reduce the run time of the system described above, to 1 hours and 21 minutes, if a speed up of 75 times is achieved.

A handicap for experiments done on CIFAR-10 is that the images are converted to gray scale. It is not possible in this implementation of the system to train on the RGB images, that would create $576*3 = 1.728$ input values per patch, with the current HOG settings. This would just take too much time. A possible idea is to train one patch-based MLP per color. As preliminary results of this system has shown, that using the colors can improve the results. The classification accuracy obtained is 73.9%.

Min pooling did not give a classification accuracy that was close to any other pooling type used, with the lowest scores in combination with 1 patch-based MLP and a much better score with 4 scoped patch-based MLPs. The bad scores can mostly be explained by the use of the ReLU activation function. A ReLU returns zero if its input is a negative number and returns its input linearly if it is a positive number. This is a problem if a lot of patches are fed through the patch-based MLP. Only if a hidden unit stays active during the whole pooling operation a non zero value will be stored with min-pooling. Therefore, with 1 patch-based MLP on CIFAR-10 with 144 patches per image it is rare that a hidden unit will not go to zero at least once. This results in a large chance of having an input vector with only zeros. With the 4 scoped patch-based MLPs, min pooling has more of a chance because less patches are fed through these patch-based MLPs, which give it more of a chance to store a useful pattern. With another activation function, for example, sigmoid or inverse tangent, min pooling would have a fairer shot at competing with average and max pooling. Another option could be to

not save the lowest, but the second lowest value.
Training MLPs in general is a game between all the parameters in the system. For example, if the classifier system with 1 patch-based MLP is used there are 28 parameters that can be set. The optimal combination between all these parameters is difficult to determine, because a lot of settings influence each other. For example, if the amount of random patches is increased the optimal settings for the learning rate, hidden units, and amount of epochs changes. Therefore a set of parameters can never completely be ruled out. This is a problem that will always be there with training any parameter-based learning algorithm. In this research we coped with this problem by performing a large exploratory analysis. Around 1.400 experiments were conducted.

7 Conclusion

In this section the answer to the research question is given and the effect of using the scoped patch-based MLP system is discussed. The research question formulated in the introduction is as follows: does training a classifier on the pooled hidden activation improve the performance of the patch-based image classification system?
What can be seen in table 5.3 and 5.4, is that for MNIST there is no system for which there is a positive effect of pooling the hidden activation. For CIFAR-10 there is one system that benefits from pooling the hidden activation, which is the classifier trained upon features created with 1 patch-based MLP in combination with average pooling (table 5.5). Therefore we can say that there was no general positive effect of training a classifier on the pooled hidden activations.
That brings us to the effect of the scoped patch-based MLP system. The scope system did not improve the 1 patch-based MLP in either the solo classification or in the pooling process. There are multiple explanations for this. One is that the large patches that are assigned to the squares overlap too much with the adjacent squares and with that you lose the idea of learning information from a specific region, because the patches in the patch-based MLP cover almost the size of the original image. It could still be possible that this helps the classification accuracy. The image gets classified from different angles, which in turn could help with spatial robustness, but this is not the case. Another reason could be that it is too difficult to classify the image just from the patches in, for example, the top right corner. As can be seen in all the output files from the scoped experiments, the validation error per patch-based MLP is always higher than when there is no scoping involved.
The architecture does not have a positive effect on the classification accuracy, only for 1 out of 16 variations of the pooling architecture there was a positive increase of 0.19%. It is good that this improvement is found on the dataset that is most complex and where there is most room for improvement. If the improvements that are proposed in the discussion are implemented, there is potential for this system. For MNIST, already high classification accuracies are obtained by classifying it with the patch-based MLP. The nice characteristic of this system is that it is relatively small which means that it can be trained fast. Adding spatial deformation of the patches into the equation will increase the training time, but most likely help the system obtain an even higher score, potentially coming close to or overtaking the current highest classification accuracy of 99.79% (Graham, 2014).

References

D. Cireşan, U. Meier, and J. Schmidhuber. Multi-column Deep Neural Networks for Image Classification. *ArXiv e-prints*, February 2012.

Dan C. Cireşan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. Handwritten digit recognition with a committee of deep neural net on GPUs. *CoRR*, March 2011.

DC Cireşan, U. Meier, LM Gambardella, and J. Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural computation*, December 2010.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In Carlo Tomasi Cordelia Schmid, Stefano Soatto, editor, *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE Computer Society, 2005.

Benjamin Graham. Fractional max-pooling. *CoRR*, December 2014.

Jason Haberman and David Whitney. Ensemble perception: Summarizing the scene and broadening the limits of visual processing. In Jeremy M. Wolfe and Lynn C. Robertson, editors, *From Perception to Consciousness: Searching with Anne Treisman*. Oxford University Press, 2012.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, July 2012.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Computer Science Department, University of Toronto, 2009.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, November 2001.

Andrew Zisserman Max Jaderberg, Karen Simonyan and Koray Kavukcuoglu. Spatial transformer networks. *CoRR*, June 2015.

B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 1964.

M. A. Mahowald R. J. Douglas R. H. R. Hahnloser, R. Sarpeshkar and H. S. Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, April 2000.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, 2011.

A Appendix

Below a more comprehensive explanation for the modified soft-max and the process of saving a patch-based MLP can be found.

A.1 Explanation soft-max

This is the explanation of why the slight variation within the soft-max function is used. Our function is as follows:

$$G(a)_i = e^{a_i - max(a)} / \sum_{j=1}^{L} e^{a_j - max(a)}$$

Each output activation is subtracted by the maximum output activation found in that particular output vector. Resulting in each output activation being smaller or equal to zero. This normalization is applied to solve an issue C++ has with large values in its e^x function. The problem is that if x becomes larger than 772.0, it will put out infinity, resulting in the system to produces $-nan$ as its training error. By applying this normalization the largest output activation is reduced to zero and the smallest to 0 minus the maximum output activation. For example, if the maximum output activation is 800 and the smallest output activation is 0.5. After normalization the minimum value is -799.5. If this is the input of the e^x function in C++ it will just return zero. Which is desirable with relatively low output activations.

A.2 Explanation saved patch-based MLP

After the training phase a patch-based MLP system can be saved. This is useful if you want to compare the effect of different pooling techniques with the feature extractor process producing the same features. When saving a patch-based MLP the following parameters are saved:

- The weight matrices
- Bias vectors
- Training images used
- Testing images used
- Minimum and maximum value used for normalizing the HOG features.