university of groningen / faculty of science and engineering

Weighting Patches of a Multilayer Perceptron for Image Classification

Bachelor's Project Thesis

Luis Knigge, s2560224, l.s.knigge@student.rug.nl, l.knigge@gmx.de
Supervisors: M.A.Wiering, E.Okafor

Abstract: Nowadays, image classification becomes increasingly indispensable in our modern day lives, with a wide range of applications such as in identifying handwritten postal codes or object recognition in computer vision. This thesis describes a method to improve image classification with patch-based multilayer perceptrons (MLP). A weight is applied to each patch that is produced by a second MLP. This MLP is also trained on patches, but it will learn the weight, which is the probability the MLP awarded to the correct class. For the test images the patches are extracted and fed into both MLPs which results in output vectors for the first, and weights for the second MLP. The weights are then applied to the output vectors which are then combined, using sum voting, to produce a classification. The purpose of weighting the patches is to improve the accuracy of the system. To test this, the MNIST (handwritten digits) and the CIFAR-10 (small images) datasets are used. We compare the accuracy rates with and without weighting as well as the certainties for these classifications. This study demonstrates that even with a simple MLP with one hidden layer, error rates lower than 1% on the MNIST dataset can be achieved. However it does not surpass the state-of-the-art techniques. The results also display that the new algorithm does not improve the image classification accuracy of the system on both datasets.

Keywords: Multilayer Perceptron, Machine Learning, Image Classification, MNIST, CIFAR-10

1 Introduction

Image classification is an important part of computer vision. From robots recognizing the objects around them [1] to classification of skin diseases [2], analyzing the contents of visual information has significant applications. In this thesis we are going to explore how neural networks, more specifically multilayer perceptrons (MLP), can perform this task. They belong to the group of supervised learners, which are trained by comparing the output of the system to a desired output. Testing will be done with the benchmark datasets MNIST [3] and CIFAR-10 [4]. They consist of small images of hand-written digits and a set of animals and vehicles, respectively.

The approach for the classification system for this project is based on the idea of splitting the image into smaller patches by sliding a window over the image with a fixed size and stride. This idea was first mentioned in Haralicks et al. paper from 1973 [5]. Later patch-based image classification was used in bag of word (or bag of features) models [6] [7] [8], where each patch would represent a word in the bag in for instance text interpretation. Each of these patches will be weighted with a weight which is produced by a second MLP (weight MLP). This MLP is trained to have an internal representation that links a patch to a weight, which is dependent on the certainty with which the first MLP (class probability MLP) would classify this patch correctly. A second MLP has to be used because the labels of the test images are not known to the MLP and therefore it cannot determine this certainty.

Similar to the idea discussed in this paper are so called boosting algorithms, where weak learners are forced to focus on the hard examples in the dataset, the examples with high error, giving them weights [9]. Schapire shows that the algorithm proposed increases performance while avoiding overfitting. The approach here is similar but uses only one cycle instead of multiple ones as proposed by Schapire.

The goal of this thesis is to research if the approach of giving each patch a weight depending on the certainty of similar patches in the train set can reduce the misclassifications of the network. For this reason the question we will answer is: can weighting patches improve the image classification

rate of a patch-based MLP?

2 System Design

In this section we will explain the basis for the system and its parts starting with the MLP followed by the general procedure and its parts.

2.1 Multilayer Perceptrons

An MLP is a simple feed-forward neural network that uses forward and backward propagation of information to learn internal representations [10]. In the forward pass the inputs are passed through the network to get a corresponding output. This output is then used in the backward propagation algorithm to adjust the weights and reduce the error. The internal representations are stored implicitly in the weights between the nodes therefore it is impossible to see how the systems links the input to the output.

2.1.1 Forward pass

The forward pass through the MLP calculates the activation of the nodes based on the weights between them, the activation of previous nodes, and the activation function. The activation a of a node i is calculated as follows:

$$a_i = \sum_j w_{ji} f(a_j) + b_i \tag{2.1}$$

Here w_{ji} is the weight from node j to i, f is the activation function and b_i is the bias for node i. The sum is over all nodes j connected to node i. The bias is implemented by adding an extra node to the layer. The more commonly used sigmoid function

$$S(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

where x is the sum of weighted inputs, is used in the weight MLP (see sec. 2.4).

$$R(x) = max(0, x) \tag{2.3}$$

Equation 2.3 shows the simpler rectifier which is used in the class MLP. Nodes using this activation function are called rectified linear units (short ReLU). It was first introduced by Hahnloser et al. [11]. This function is not as heavy computationally but gives similar results. We chose to use a different activation function for the class MLP, because it has far more connections from the hidden to the output layer. Instead of one output unit for the weight MLP the class probability MLP has an output size of the number of classes in the datasets.
A graphical illustration of both the sigmoid and the ReLU activation function is shown in Figure 2.1.

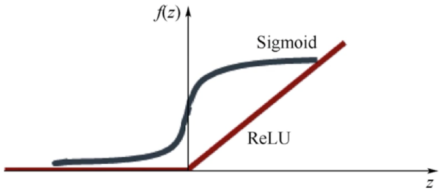


Figure 2.1: ReLU activation function and sigmoid function

2.1.2 Backward propagation

In this part of the process the error the system made in the forward pass is used to adjust the weights in the MLP with the help of the generalized delta rule [10].
To adjust the weights of the connections between the nodes the error e is propagated back through the network. The following equations (2.4 - 2.8) are adapted from Rumelhart's et al. paper [10]. In our system the error is defined as follows:

$$e_j(n) = d_j(n) - y_j(n) \tag{2.4}$$

where $e_j(n)$ is the error in output node j given by the n-th data example, $d_j(n)$ is the desired value, and $y_j(n)$ is the actual output of the output node. The purpose of the delta function is to reduce the summed square error

$$\varepsilon(n) = \frac{1}{2} \sum_j e_j^2(n) \tag{2.5}$$

of the system. The generalized delta rule has two parts which can be written in three equations. First we have the equation which gives us the change of the weights:

$$\Delta w_{ij}(n) = \eta \delta_j(n) y_i(n) \tag{2.6}$$

where $w_{ij}(n)$ is the weight between node i and node j, and η is the learning rate. The second part consists of two equations which guard how the error is passed back to the weight to be changed. If the weight is connected to an output unit the error signal is simply

$$\delta_j(n) = e_j(n) \tag{2.7}$$

for the input to hidden layer weights the error signal is

$$\delta_i(n) = \Big(\sum_k \delta_k(n) w_{ik}\Big) \frac{df_i(n)}{da_i(n)} \tag{2.8}$$

where $\frac{df_i(n)}{da_i(n)}$ is the derivative of the non-linear activation function with respect to the activation. This includes the bias as seen in Equation 2.1. In this manner the $\delta_j(n)$ values for deeper layers can be calculated as well, by recursivly calculating the δ-values.

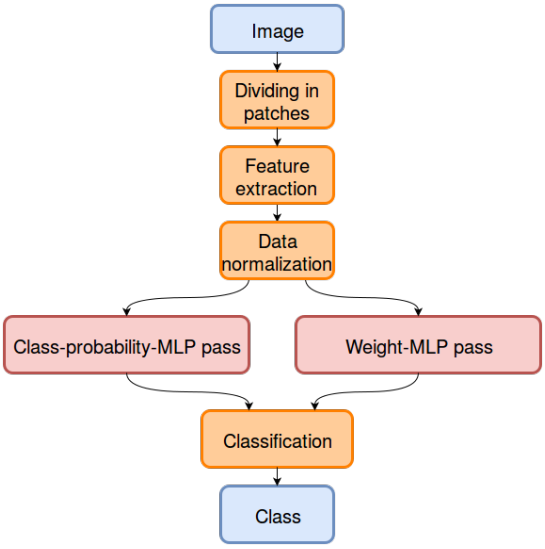


Figure 2.2: Architecture of the system doing classification

2.2 General Procedure

There are two different passes through the system. First we train the MLP then we test it by feeding it the test images. This is the actual classification step. For the training process there are random patches extracted from the train image set. These are used to train the weights between the layers of the MLP. They are translated into feature vectors using the HOG feature descriptor, which is explained in detail in section 2.5. The feature vector is then normalized and fed to the MLP. After the class probability MLP is trained, the weight MLP is trained. It has one output node whose desired output is the class probability awarded to the correct class passing through the first MLP.

For the classification or test phase, each image is split into patches and all these patches are fed into the classification MLP and the weighting MLP. From the first MLP we get an output vector with a length according to the amount of classes in the dataset, in this case it is 10 for both datasets. These are multiplied by the value the weight MLP produces. The final output vectors for all patches of the image are combined with sum voting. Now the class with the highest value is determined and the images can be classified (see Figure 2.2).

The general framework was provided by Groefsema [7] and Maas [8] which scans in the images, provides the training and test structure and presents the results in a readable form.

2.3 Patches

In our approach we split the image in smaller patches. The reason behind this is that it reduces the size of the feature vectors of an image which in turn allows for a reduction in size of the MLP. This also allows us to have a voting system and therefore we do not rely on one output of the MLP alone. When weighting the patches the system reduces the influence of patches with high error. See the next section for further information.

In comparison to using the whole image as the input at once another advantage of patches is that local features can be captured, which are representative of the class. For example the 'x'-like pattern in the middle of a handwritten '8' does not occur in any other digit, which could make it an identifier for this class [12].

The patches for test images are created by a sliding window of a fixed size and with a fixed stride over the image. In our experiments we used a stride which is smaller than the size of the patch. This leads to overlap between the patches and gives the advantage of not missing patterns in the image like the previously mentioned 'x'-shaped pattern in the

digit '8'. It could happen that the patch covers only half of the pattern which would lead to a different representation in the feature vector reducing the representativeness of the patch for the class.

2.4 Weighting patches

Weighting the patches is supposed to give the system an advantage over a normal neural network by giving the patches, that do not contain relevant information for the classification process, less influence on the final result. An example of this are patches that just contain parts of the sky. It is unknown if the picture contains an airplane or a bird from this patch and therefore the patch gets a low weight, but a patch containing the wing of this airplane, for example, has a high representativeness for the airplane so it will get a higher weight.
The weights of the patches are based on the class probability of the correct class similar patches had during the training phase of the class probability MLP. The certainties for the right class of an image are stored and then given to the second neural network where they are used as the desired outputs to train this MLP.

2.5 Local feature descriptor

The feature descriptor we use here is the Histogram of Oriented Gradients (HOG). Here the patch is divided into smaller cells where for each pixel the horizontal and vertical gradient is computed. These are then combined to get the direction and magnitude of the gradient. Now the gradients for a cell are put into a histogram of fixed size (number of bins) according to the orientation of the gradient. There is a histogram for each cell so if there are 9 cells in a patch and there are 16 bins for each histogram the feature vector dimension is 154 [8] [13]. The values are then standardized with the L2, or Euclidean normalization ($||x||_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$) to finalize the process of feature extraction.

2.6 Dropout

Dropout is a regularization method for neural networks. This procedure ensures that the neurons do not become over specialized on very specific features, but on the contrary are adaptive and can build a useful representation. This is done by turning nodes, and their connections, temporarily off. This happens in the training process and essentially means that a lot of different networks are trained with very few examples each. The nodes have a 50% chance of being turned off.
Srivastavas et al. [14] shows that dropout reduces error on the same datasets we are using here, MNIST and CIFAR-10.
We chose to only use dropout with the class MLP since it has more connections and therefore more weights.

2.7 Classification through neural network with weighting

To classify a new image the system has not seen before, the images are split into patches and run through the classification neural network as well as the weight neural network. The outputs of the first neural network are then weighted by the output of the second one. All of the resulting outputs are put together in a single classification vector with sum voting. The index of the largest value of this vector determines the classification of the image.

3 Experimental setup

3.1 Datasets

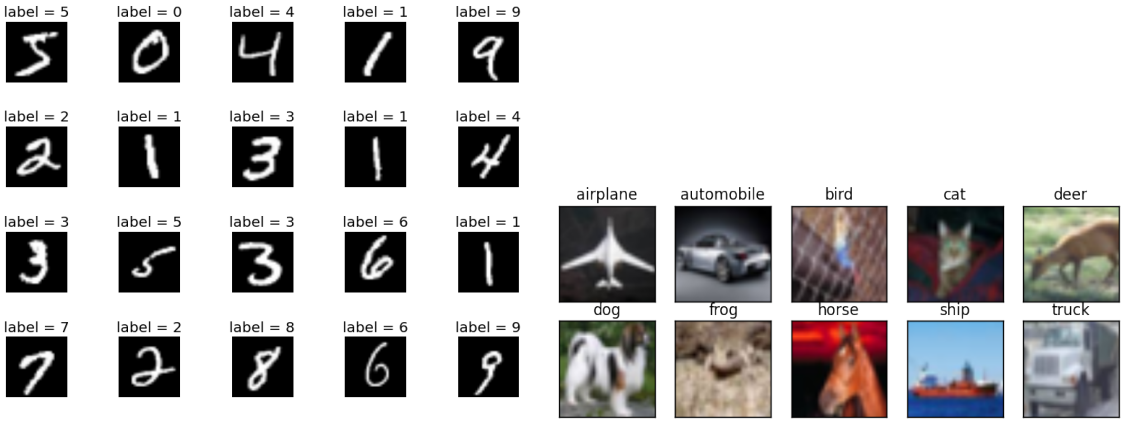
We used two datasets for this research, MNIST and CIFAR-10 which are described in the next two sections. Some examples of the datasets are shown in Figure 3.1.

3.1.1 MNIST

This dataset contains hand-written digits (0-9) in black and white. There are 60.000 train and 10.000 test images. For this dataset it is easier to do classification because the images of the same class are more alike than in CIFAR-10. The size of the images is 28×28 pixels. It is a modified version of the NIST database [3].

3.1.2 CIFAR-10

CIFAR-10 is a benchmark dataset for image classification containing 50.000 colored train and 10.000 test images, which are all labeled. There are 10



(a) MNIST (b) CIFAR-10

Figure 3.1: Example images for both datasets

classes ranging from vehicles to animals. The images contain these objects in different contexts, orientations and lighting. The size of each is 32×32 pixels [4].

3.2 Experiment design

The experiment is composed of a 10-fold Monte Carlo cross validation. For the MNIST dataset the images are re-scaled to 40×40 pixels using cubic interpolation. Patches have a size of 34×34 pixels and the HOG feature descriptor is set up to use 16 bins with a cell size of 4. We use no padding, which means that the effective size of the patch is 32×32. This leads to an input vector of size 1024 and we employ a total of 900.000 random patches for the training process. The training processes for both MLPs are 50 epochs long. The class probability MLP has 500 hidden units, while the weight MLP uses 300. The learning rate for the first MLP is 0.001 and for the second 0.02. The training for the second MLP is done on the validation set to learn a better function that generalizes over the dataset. If it would be trained on the training data that has already been learned by the first MLP the weights will be higher and more uniform since the errors on these are lower since the system has learned these already.
The parameters of the MLPs for the CIFAR-10 dataset are 36×36 pixels for the images also using the cubic interpolation and 26×26 pixels for the patches. The HOG descriptor is using 9 bins and the cell size is 3×3 pixels with no padding used. This leads to an input feature vector of size 576. The MLP has 1500 units in the hidden layer. The learning rate is 0.01 and in 30 epochs the class MLP learns its weights using 950.000 random patches. For the weighting MLP the parameters are 500 hidden units and a learning rate of 0.02. The input units are the same as for the class MLP since they are using the same parameters for the feature descriptor.

3.3 Measurements

Since both datasets are completely labeled we are measuring the accuracy of the single MLP and the weighted MLP by comparing the predicted label of the image to the actual label on the test set. This gives us an error rate. Time could also be measured but is secondary to the accuracy of the system.
We are also measuring the average certainty of the MLPs by looking at the output vectors and of the patches. The average certainty is defined as the sum of the values that are at the index of the right class of the final vote vector divided by the number of images. Since the outputs are converted into probabilities for each class we can see how certain the neural network was making its decisions on the class of the patch.

Weighting	MNIST		CIFAR-10	
	mean	SD	mean	SD
no	0.628	0.032	26.13	0.14
yes	0.64	0.055	26.26	0.15

Table 4.1: Error rates (in %) with their corresponding standard deviations for the MNIST and CIFAR-10 datasets

4 Results

In this section we will present the results from 10-fold Monte Carlo cross validation for the following scenarios: testing with and without weighting for the MNIST and the CIFAR-10 datasets. The important measures of performance are the error rates from the testing phase. Secondary we are looking at the corresponding certainties with which the images were classified.

4.1 MNIST

The results of our 10-fold Monte Carlo cross-validation show that with a mean of 0.628% the system not using weighted patches has a lower error rate than the system that uses weighting where the score is 0.64% on average as shown in Table 4.1. But even though the mean and median of the system without weighting are higher, the lowest error rate, 0.55% (obtained on a single test set), was produced by the MLP using weighted patches as seen in Figure 4.1(a). Furthermore Figure 4.1(a) shows that the range of the values of the error rates is inflated in both directions. We performed an unpaired two-tailed Student's t-test which showed that the difference in results were not significant (t-value = -0.59, p-value = 0.56). This could lead to the conclusion that weighting the patches is not a method to improve the prediction rate of a MLP.
Figure 4.2(a) and Figure 4.1(b) show that the certainties of the system using weights for the patches are higher than without the weights across the range of the error rates. The exceptions are the two values on the far end of the spectrum. Since these two have certainties so much lower than the rest of the experiments with weights they can be considered outliers. They will not be excluded from the results but it should be mentioned that they exist.

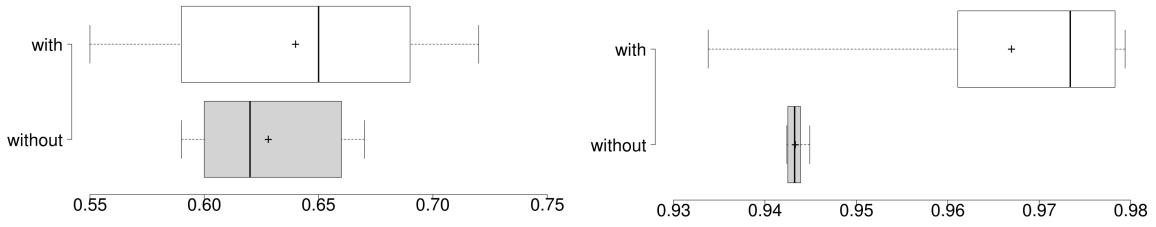
4.2 CIFAR-10

The results for the CIFAR-10 dataset show that weighting the patches did not perform as well as not doing so. In Table 4.1 you can see that the difference between using weights and not using them is 0.128%. Even though this difference is not significant with a unpaired two-tailed Student's t-test with a t-value of 1.99 and a p-value of 0.06 we can see in Figure 4.1(c) that the error rates are higher and the certainties are lower when using weighted patches (Figure 4.2(b) and Figure 4.1(d)).

5 Conclusion

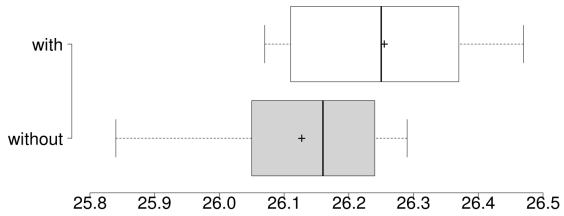
The results we got for our experiments on the MNIST dataset (0.628% and 0.64% error rate) are fairly comparable to state-of-the-art results for this dataset with a neural network. The best 2-layered network is from Simard et al. paper from 2003 [15] with an error rate of 0.7%, the best feedforward neural network is from Ciresan et al. (2010) [16]. It has a 6-layer architecture and produces an error of 0.35%. The overall best performance has been obtained with a convolutional multilayered neural net using droput [17] with an error rate of 0.21%.
The architectures for the last two papers are far more complex than the one from the paper from Simard which has an architecture closest to the one we were using in this paper.
For the CIFAR-10 dataset the results obtained with or without weighted patches are not comparable to the state-of-the-art results. The best results of a similar architecture were realized by Sato et al. [18] with an MLP using augmented data. They achieved an error rate of 14.06%. This is still not the golden standard which was accomplished by Graham in 2014 [19] with an error rate of just 3.47% with a technique called 'fractional max-pooling'.

The difference in certainties for the MNIST dataset, being 0.967 (without weighted patches: 0.943), and the CIFAR-10 dataset, being 0.348 (0.4863), indicates that the CIFAR-10 is a more complex dataset. This is also shown in the error rates. But the certainty shows that the system itself has difficulties differentiating between classes, so it is not just classifying more images wrong but also it is not as sure about the decisions.
In conclusion we can say that using weighted

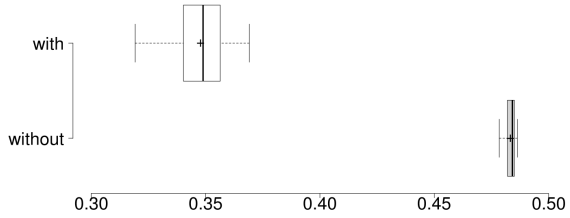


(a) MNIST error rates

(b) MNIST average certainties



(c) CIFAR-10 error rates



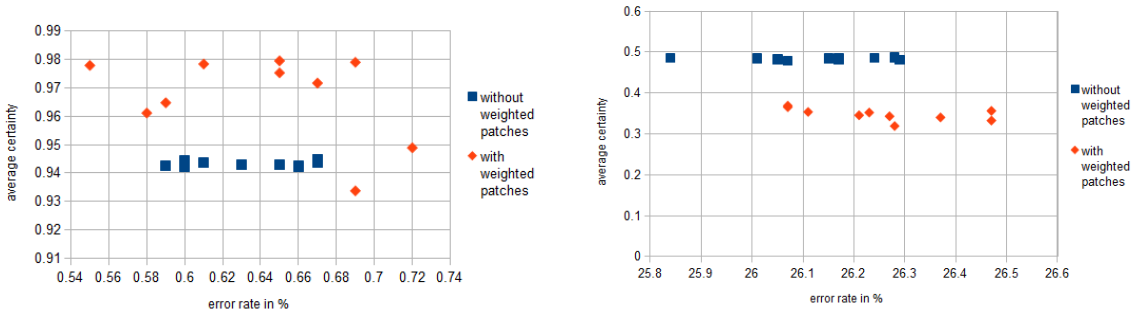
(d) CIFAR-10 average certainties

Figure 4.1: Error rates in % and average certainties with and without weighting for both datasets

patches does not increase the classification rate of the MLP.

6 Discussion

In this thesis we proposed an idea to improve the accuracy of a MLP, which uses patches, by giving them a weight. The results showed that there was no improvement. A particularity for the MNIST dataset while doing experiments for parameter fine tuning was that with a learning rate slightly higher than we used in the final experiments the weights in the first MLP would explode to the point that they were overflowing the *double* variables. This caused the outputs to turn out N/A values.
With more time we might be able to find better parameters for the weight MLP to improve the results especially for the CIFAR-10 dataset.
More ideas to improve on the MLP image classification system could be for example to only allow patches that have a high certainty for the classification to take part in the voting for the final classification. This would eliminate the patches with low weights completely and could increase the performance of the system. Another possibility is to use more regularization methods similar to dropout. There is a promising method, called 'dropconnect' [20] [17], which will set weights to be zero and therefore dropping a connection between nodes in the training process but the node can potentially still be activated. That is the difference to 'dropout'. Another method worth exploring is called 'momentum' [10] It could potentially increase the learning speed of the MLP since at the moment it takes around seven days to complete the training of the MLPs for the CIFAR-10 dataset. Here the weight change of the previous epoch also influences the current weight change. Essentially it means that the weight change goes into a similar direction as it was in the epoch before.
As shown in the paper of Sato et al. [18] and Graham [19] data augmentation can lead to lower error rates. This could be a possibility to improve the MLP we are using in this paper. A slight change in the architecture could yield a similar effect like adding more layers [16] or changing the network type from a feed forward MLP to a convolutional neural net. Since the system as it is of now takes a long time to run using GPUs is a good idea.
To finalize this thesis, it can be said that there is still some research to be done on this topic since there are so many ways that it could still be improved.



(a) MNIST

(b) CIFAR-10

Figure 4.2: Error values vs average certainties of both datasets

References

[1] C. Siagian and L. Itti, "Rapid biologically-inspired scene classification using features shared with visual attention," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, pp. 300–312, Feb. 2007.

[2] H. K. Al-Mohair, J. Mohamad Saleh, and S. A. Suandi, "Hybrid human skin detection using neural network and k-means clustering technique," *Appl. Soft Comput.*, vol. 33, pp. 337–347, Aug. 2015.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–324, 1998.

[4] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Master's Thesis, University of Toronto*, 2009.

[5] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-3, no. 6, pp. 610–21, 1973.

[6] E. Nowak, F. Jurie, and B. Triggs, "Sampling strategies for bag-of-features image classification," in *9th European Conference on Computer Vision (ECCV 06)*, pp. 490–503, Springer, 3954, 2006, Austria.

[7] M. Groefsema, "Deep architectures using the bag of words model for object and handwritten character recognition," *Bachelor's Thesis, University of Groningen*, 2016.

[8] J. L. Maas, "The dual codebook: Combining bags of visual words in image classification," *Bachelor's Thesis, University of Groningen*, 2016.

[9] R. E. Schapire, "A brief introduction to boosting," *Proceedings of the 16th international joint conference on Artificial intelligence*, vol. 2, pp. 1401–6, 1999.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel distributed processing: explorations in the microstructure of cognition*, vol. 1, pp. 318–62, MIT Press Cambridge, 1986.

[11] R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature*, vol. 405, no. 6789, pp. 947–51, 2000.

[12] C. Xu, T. Wang, J. Gao, S. Cao, W. Tao, and F. Liu, "An ordered-patch-based image classification approach on the image grassmannian manifold," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 4, pp. 728–37, 2014.

[13] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893 vol. 1, 2005.

[14] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–58, 2014.

[15] P. Y. Simard, D. Steinkraus, and J. Platt, "Best practices for convolutional neural networks applied to visual document analysis," Institute of Electrical and Electronics Engineers, Inc., 2003.

[16] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep big simple neural nets excel on handwritten digit recognition," *CoRR*, vol. abs/1003.0358, 2010.

[17] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)* (S. Dasgupta and D. Mcallester, eds.), vol. 28, pp. 1058–1066, JMLR Workshop and Conference Proceedings, May 2013.

[18] I. Sato, H. Nishimura, and K. Yokoi, "APAC: augmented pattern classification with neural networks," *CoRR*, vol. abs/1505.03229, 2015.

[19] B. Graham, "Fractional max-pooling," *CoRR*, vol. abs/1412.6071, 2014.

[20] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 1058–1066, 2013.