Deep Architectures using the Bag of Words Model for Object and Handwritten Character Recognition

(Bachelor project)

Marc Groefsema, s2055236, m.groefsema.2@student.rug.nl,
M. A. Wiering*

June 30, 2016

Abstract

This thesis describes an image classification system, using feature extraction, the Bag of Visual Words model (BoW), a Deep Bag of Visual Words model (DBoW) and a Support Vector Machine (SVM) classifier. The performance of the system using the BoW model is compared to the performance using the DBoW model. The performances are assessed using 10-fold cross-validations on the MNIST and CIFAR-10 datasets. First different configurations are explored of both the BoW model and the DBoW model. Finally, both architectures are compared based on their best performances. Results show a lower performance by using the DBoW architecture compared to the performance using the BoW model.

1 Introduction

Lately 'Deep Learning' has become increasingly popular [1]. Often Convolutional Neural Networks (CNN) with many layers are used to learn complex high-level feature descriptors and classifiers [2][3]. A drawback with such methods is that features need to be learned using labeled data.

Another popular method for learning useful features is the *Bag of Visual Words* (BoW) paradigm [4]. Here unsupervised clustering algorithms are used to construct "prototype" features. Newly extracted features can afterwards be compared to these prototypes using a distance function, e.g the Euclidean distance. These distances are used to construct an activation pattern for the prototypes, using an activation function. This results in the BoW feature. The constructed BoW-features are used to train an L2-SVM classifier. A drawback of this is the $O(N^2)$ complexity of training the L2-SVM on N data points, hence it becomes computationally expensive to train such a system, given a lot of data.

In this research, we use a linear L2-SVM, trained with the primal objective. This method does not require a similarity matrix between all training examples, as the general used interpretation of the SVM does. This requires much less computation and results in an $O(N)$ complexity, allowing for larger datasets to train on.

In this thesis a multi-layer BoW, or *Deep Bag of Visual Words* (DBoW) is evaluated*. DBoW is evaluated on its parameters and compared to the BoW architecture in performance, using a linear L2-SVM as a classifier. Given the best parameters found for both architectures, the question is whether the use of a DBoW model yields a different performance compared to the usage of the BoW model.

The layout of this thesis is as follows: Section 2 describes related work. Section 3 describes the framework. Next, section 4 presents experiments and their analyses. Finally, the discussion and conclusions will be presented in the last section.

*University of Groningen, Department of Artificial Intelligence

*All source code used can be found at the following url: https://github.com/MarcGroef/CSVM/tree/MarcExp

2 Related Work

In [5], Coates et al. describe a BoW [4] architecture as a feature descriptor from raw pixels. These features are used for L2-SVM classification of images in NORB [6] and CIFAR-10 [7].

The patches are extracted in a convolutional manner, using a sliding window. The centroid activations are summed over in four equal sized quadrants, resulting in a $4 \times K$ sized feature for an image, where K is the number of centroids used in the BoW.

In [8], Coates describes a method for constructing a deep BoW architecture, using normalized centroid-kernels, on standardized and whitened local feature representations. Here a greedy method is used to link similar layer-n centroids to layer-n+1 centroids. The results indicate an increase in performance using a deep BoW-architecture.

In [9] Surinta et al., describe a similar architecture for classifying handwritten characters in Odia, MNIST and Bangla. In this architecture, an extra step is used, compared to the architecture described by Coates et al. Here patches are extracted in the same manner. But instead of using k-means on the raw pixels, a patch is first described using a *Histogram of Oriented Gradients* (HOG)[10]. From these HOG features, a Bag of Words is constructed, resulting in the so-called HOG-BOW architecture. An L2-SVM is used as a classifier on the HOG-BoW representations of the data.

3 Framework

Here the framework is described in the same consecutive steps as an image itself is processed and classified.

3.1 Patch Extraction

An image is first described as a collection of samples from this image. This is done by sampling smaller images, called patches, from the image, by using a sliding window, specified by a height, a width and a stride. Hence the dimensions of a patch are $H \times W \times C$, where H, W, C represents the height, width and the number of colour channels respectively.

3.2 Local Feature Extraction

Given a patch from an image, it can either be represented by its pixel-intensity values or using a feature extraction method.

In the experiments in this thesis the HOG descriptor is used. Here a patch is divided in subareas, called *cells*.

(1) A cell is first described by the gradient orientations of the pixels. For each pixel the gradient direction and magnitude are calculated.

(2) Next a histogram is built, where each bin represents a part of the domain of the orientations from $[0; \pi]$. If an orientation is in $[\pi; 2\pi]$, it is subtracted from 2π, causing to flip the orientation direction. If a pixel is in the domain of a bin, its magnitude is added to the histogram.

(3) The histograms of all cells, are appended, resulting in a one-dimensional HOG-vector. This vector is first normalized by the L2-norm, then it is standardized, by subtracting the mean from all elements and dividing the elements by the standard deviation within the vector.

3.3 The Bag of Words model

In the Bag of Words (BoW) model, a collection of random patch features are clustered using the k-means algorithm [11] to construct prototype patch-features. The learned prototype patch features, referred to as centroids, form a redundant basis for the patch feature space.

3.3.1 Image representation

Given an image, patches are extracted using a sliding window, defined by a size and a stride. An image is represented as a collection of these patches. Next, each patch is described using a local feature descriptor, e.g. using HOG as done by Surinta et al. [9]

The resulting feature vector from the local feature descriptor is then compared to all the centroids using a distance function. Using an activation function, this results in an activation of

these centroids. As an activation-function the Soft-Assignment function [5] is used:

$$Act(p, f) = max(0, \mu_d - d(p, f))$$

$$d(p, f) = \sum_{i}^{D} (p_i - f_i)^2$$

Where $d(p, f)$ is the squared Euclidean distance between patch p and centroid f, and μ_d is the mean squared Euclidean distance from patch p to all centroids in the BoW.

Given this method to map patch features into centroid activations, a method is needed to integrate centroid activations from all the patches of a particular image, to a single vector representation.

For this, BoW is used with quadrant pooling. The image is split-up into four equal sized quadrants. All centroid activations from a particular quadrant are pooled over by e.g. using element-wise addition of all activation-vectors.

With K centroids, this will finally result in a $4 \times K$ feature vector which is the representation of the entire image, and can be fed to a classifier for classification.

This 4 x K feature representation is standardized, by subtracting the mean, and dividing all elements by the standard deviation within the vector.

3.4 The Deep Bag of Words Model

As an extension to the BoW paradigm, a Deep Bag of Words (DBoW) model is evaluated. A similar architecture has previously been described by Coates [8].

Here the BoW paradigm is used iteratively, to construct centroids which represent higher-order features from an increasingly larger region of the image.

(1) First a regular BoW is constructed by clustering random patch features from random images. Given an image and the centroids, a feature map can be constructed for each centroid. Each entry on the feature map is a mapping from the corresponding patch representations to an activation, using an activation function. Hereafter a pooling layer is defined to be a smaller grid, where each entry is a pooling over a different local area of the feature map. If this pooling layer has the size 2×2 and no further BoW layers, then this would equal the quadrant pooling, as used with the standard BoW architecture.

(2) After the construction of the first BoW layer, it is possible to describe an image region using the values at the corresponding location in the pooling map of all centroids. Each location on the pooling layer hence equals an $1 \times K$ feature. This $1 \times K$ feature is standardized when drawn from the pooling layer.

Using the first layer of the DBoW, it is possible to collect features from the first pooling layer, to make a collection of random second order local features. Due to the pooling of the first layer, the second order local features represent a larger area of the image than a first order local feature.

This collection of second order local features can be used to generate new centroids using an algorithm as K-means clustering.

By comparing a pooling feature from the previous layer to the newly generated centroids, an activation function can be used to construct an activation feature vector. In this way the newly generated centroids can be used to make a second layer feature map of the first layer pooling layer. This second layer feature map is used to construct a second layer pooling map, which is again smaller than the second layer feature map.

(3) The method of constructing a BoW from the pooling layer of the previous BoW can be used to construct a Deep Bag of Words architecture with multiple layers.

(4) To construct a final feature vector from a final $N \times M$ pooling-layer, the entries of the pooling layer are appended into one vector, resulting in a $1 \times (N * M * K)$ vector, where K is the number of centroids used in the last layer.

Next, this feature vector is standardized, by subtracting the mean and dividing all elements by the standard deviation within the vector, after which it is given to the L2-SVM for classification.

Differences between the DBoW described above, and the DBoW architecture described in [8], are the distance function and the way centroids from layer n are used to activate centroids of layer $n+1$.

(1) The distance function used by the architecture described here, is the Euclidean distance function. The architecture described in [8] uses a cosine distance, otherwise described as a dot product between the normalized vectors.

(2) The architecture described by Coates does not use all centroids from the previous layer to construct the current layer, where the architecture in this thesis does. In [8] *receptive field learning* is used to determine which centroids from the previous layer to use in the current layer.

3.5 SVM Classification

As a classifier, a Support Vector Machine using the L2-objective function (L2-SVM) is used. The label of the L2-SVM with the highest output will be considered as the classified label. The SVM uses the following objective function:

$$\min_{w,b} L = ||w||^2 + C * \sum_i \xi_i^2$$

and output function:

$$g(x_i) = w^T * x_i + b$$

Where

$$\xi_i = max(0, 1 - y_i * g(x_i))$$

Here $y_i = -1$ if y_i does not refer to the target label or $y_i = 1$ if it does refer to the target label.
At each iteration the weights to centroid j and the bias are updated with each example if $y_i * g(x_i) < 1$. This is done as follows:

$$\Delta\ w_j = -\lambda * (\frac{w_j}{C} - (y_i - g(x_i)) * x_i^j)$$

Where λ denotes the learning rate. At the end of each iteration, the biases are set to be the mean error $y_i - g(x_i)$ from all examples where $y_i * g(x_i) < 1$, if such examples occurred and to zero otherwise.

Here x_i denotes the centroid activations resulting from the (D)BoW. For a BoW architecture, this means the centroids from the BoW are used as a kernel, instead of using the feature representation of all other training examples. Hence the trainings complexity becomes linear with the amount of data, instead of quadratic. This allows for training on much larger datasets compared to e.g. a dual objective L2-SVM.

4 Experiments and Analysis

4.1 Datasets

The architectures are evaluated using the MNIST [12] and CIFAR-10 [7] datasets, as shown in figure 1. The experiments in this thesis are done by 10-fold cross-validations.

4.2 MNIST

MNIST is a widely used benchmark dataset, consisting of grey-scale images of 28×28 pixels. The dataset contains ten classes, representing handwritten digits from zero until nine, where each class has approximately 6000 images in the training set, and 1000 in the test set.

4.3 CIFAR-10

The *CIFAR-10* dataset is also a widely used benchmark dataset, consisting of 32×32 pixel colour images, with three color channels. The dataset contains ten classes, each represented by 5000 images in the training set, and 1000 images in the test set.

4.4 Preprocessing

As a preprocessing step for both MNIST and CIFAR-10, images are rescaled to 36×36 pixels, using bi-cubical interpolation on each channel of the image. This image size is chosen, since, with a patch size of 12×12 it allows for three pooling layers, with no overlap at the feature map, where each pooling map is half the size of a feature map.

4.5 Patch Extraction

Patches for both MNIST and CIFAR-10 are extracted using a sliding window of 12×12 pixels and a stride of a single pixel. To construct a (D)BoW,

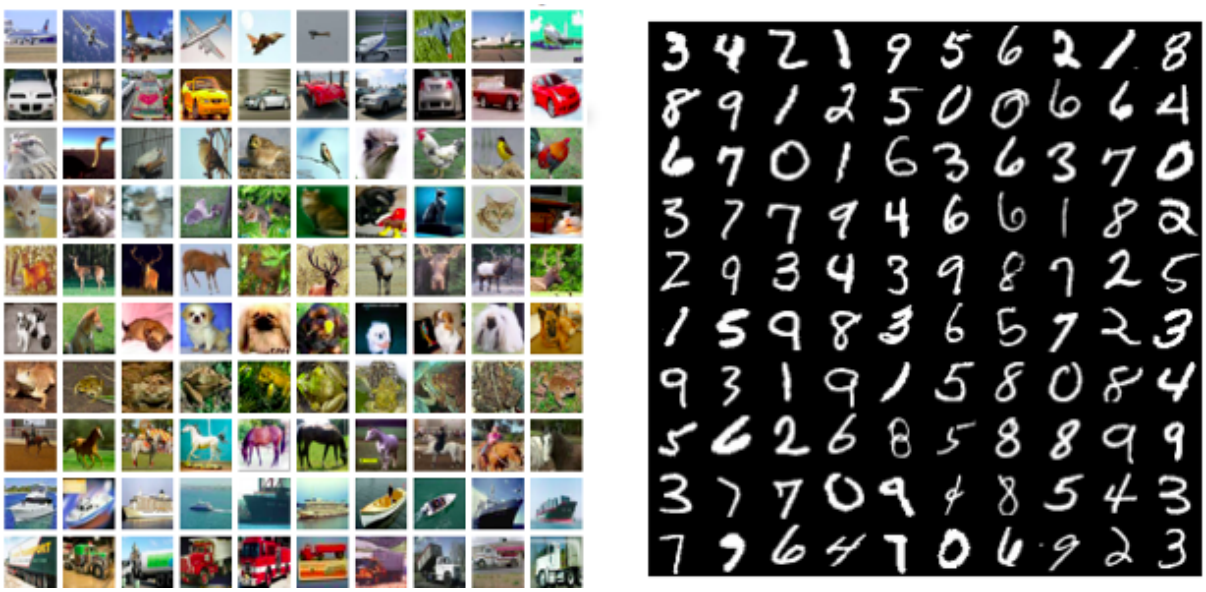

Figure 1: The CIFAR-10 dataset (left) and the MNIST dataset (right)

2×10^5 patches are extracted from random training images at random locations.

4.6 Feature Description

Each patch is described using a HOG-feature descriptor, using 4 6×6 cells with no overlap, with each 9 orientation bins, for each channel in the image. The HOG-feature from each channel are appended, resulting in a 1×36 feature vector using an MNIST image, and a 1×108 feature vector using a CIFAR-10 image.

4.7 Classifier Parameters

Within each dataset the same classifier parameters are used for all experiments.

- For the MNIST dataset, the following parameters are used. As a learning-rate, $\lambda = 2 \times 10^{-5}$ is used. As a value for the C parameter $C = 60$ is used. All the "weights" are initialized to be $w_{i,c} = 0$, for the weights from target node i to centroid c. During training, the classifier is trained for 500 iterations.

- For the CIFAR-10 dataset, the following parameters are used. As a learning-rate, $\lambda = 2 \times 10^{-7}$ is used. As a value for the C parameter $C = 1000$ is used. All the "weights" are initialized to be $w_{i,c} = 0$, for the weights from target node i to centroid c. During training, the classifier is trained for 1000 iterations.

4.8 Deep Bag of Words Configuration

To explore the performance using the Deep Bag of Words, three architectures are introduced, with one, two and three BoW-layers, called *Alpha*, *Beta* and *Gamma*, respectively. The final pooling layers of each architecture will be a 2×2 pooling layer, to keep the resulting feature vectors constant in size, and also to eliminate confounding SVM learning effects.

Feature maps and pooling maps, at depth n will be denoted as fm_n and pm_n respectively, where $n = 0$ refers to the lowest layer, and $n > 0$ refers to higher-order layers.

The number of centroids used for each layer will be kept constant and will be referred to as K.

4.8.1 The effect of the number of centroids

To evaluate a good number of centroids for each architecture, each architecture is tested using 200,

400 and 800 centroids. The best number of centroids for each architecture will be used to compare the final performances of the *Beta* and *Gamma* architecture to the best performance of the *Alpha* architecture.

4.8.2 The effect of pooling methods

For each architecture, the effect of sum-pooling and max-pooling will be evaluated. The best pooling method for each architecture will be used to compare the final performances of the *Beta* and *Gamma* architecture to the best performance of the *Alpha* architecture.

4.8.3 Alpha architecture

The first architecture, named *Alpha*, contains one BoW layer, hence equals a regular BoW model. Here fm_0 contains $24 \times 24 \times K$ entries and pm_0 contains $2 \times 2 \times K$ entries.

- Results using this architecture on MNIST, as shown in table 1, suggest an increasing performance as K is increased, for both max-pooling and sum-pooling. Comparing the highest results of max-pooling and sum-pooling suggest a higher performance using max-pooling ($t(13.33) = 6.02$, $p << 0.05$).
- Results using this architecture on CIFAR-10, as shown in table 2, suggest an increasing performance as K is increased for sum-pooling. When K is increased with max-pooling, results suggest more unstable performance for $K = 400$, after which the performance drops. Comparing the highest results of max-pooling and sum-pooling, the results suggest a higher performance using sum-pooling ($t(13.30) = 33.25$, $p << 0.05$).
- The optimal configuration within the explored parameter space for MNIST is using max-pooling with 800 centroids. For CIFAR-10 the best parameters found is using sum-pooling with 800 centroids.

4.8.4 Beta architecture

The second architecture, named *Beta*, contains two BoW layers. At the bottom layer, fm_0 contains $24 \times 24 \times K$ entries and pm_0 contains $12 \times 12 \times K$ entries. On the second layer fm_1 contains $12 \times 12 \times K$ entries and pm_1 contains $2 \times 2 \times K$ entries.

Table 1: Results on MNIST using the Alpha architecture with standard error (% correct)

MNIST	K = 200	K = 400	K = 800
Sum	98.72 ± 0.03	98.85 ± 0.05	99.12 ± 0.02
Max	98.96 ± 0.05	98.97 ± 0.03	99.26 ± 0.04

Table 2: Results on CIFAR-10 using the Alpha architecture with standard error (% correct)

CIFAR-10	K = 200	K = 400	K = 800
Sum	54.99 ± 0.25	58.11 ± 0.49	60.33 ± 0.25
Max	51.04 ± 0.49	50.49 ± 5.65	33.11 ± 1.19

- Results using this architecture on MNIST, as shown in table 3, show an increase in performance when the number of centroids is increased, for both sum-pooling and max-pooling. Comparing the highest results of max-pooling and sum-pooling suggest no significant difference between the highest scores ($t(15.49) = 0.47$, $p = 0.65$).
- Results using this architecture on CIFAR-10, as shown in table 4, show an increase in performance when the number of centroids is increased, using sum-pooling. Using max-pooling, the results suggest a performance drop when the number of centroids is increased. Comparing the highest results of max-pooling and sum-pooling, the results suggest a higher performance using sum-pooling ($t(2.52) = 9.12$, $p = 0.03$).
- The optimal configuration for CIFAR-10, within the explored parameter-space, is using sum-pooling with 800 centroids. For MNIST the optimal configuration is using 800 centroids. The choice between max-pooling or sum-pooling doesn't show a difference in performance.

Table 3: Results on MNIST using the Beta architecture with standard error (% correct)

MNIST	K = 200	K = 400	K = 800
Sum	98.51 ± 0.04	98.69 ± 0.08	98.93 ± 0.04
Max	98.60 ± 0.01	98.68 ± 0.09	98.91 ± 0.07

Table 4: Results on CIFAR-10 using the Beta architecture with standard error (% correct)

CIFAR-10	K = 200	K = 400	K = 800
Sum	49.12 ± 0.21	53.59 ± 0.26	56.90 ± 0.28
Max	51.08 ± 2.52	52.48 ± 3.43	49.79 ± 0.46

4.8.5 Gamma architecture

The third architecture, named *Gamma*, contains three BoW layers. At the bottom layer, fm_0 contains $24 \times 24 \times K$ entries and pm_0 contains $12 \times 12 \times K$ entries. At the second layer fm_1 contains $12 \times 12 \times K$ entries and pm_1 contains $6 \times 6 \times K$ entries. At the third layer fm_2 contains $6 \times 6 \times K$ entries and pm_2 contains $2 \times 2 \times K$ entries.

- Results using this architecture on MNIST, as shown in table 5, show an increase in performance as the number of centroids is increased, for both the max-pooling and sum-pooling. Comparing the highest results of max-pooling and sum-pooling suggest no significant difference between the highest scores ($t(10.79) = 1.66$, $p = 0.13$).
- Results using this architecture on CIFAR-10, as shown in table 6, suggest an increase of performance, by increasing the number of centroids, using sum-pooling. Using max-pooling, the results also suggest an increase in performance as the number of centroids is increased from $K = 200$ to $K = 400$. Though when the number of centroids is increased to $K = 800$ the results show a decreased performance. Comparing the highest results of max-pooling and sum-pooling, the results suggest a higher performance using sum-pooling ($t(17.71) = 9.66$, $p << 0.05$).

Table 5: Results on MNIST using the Gamma architecture with standard error (% correct)

MNIST	K = 200	K = 400	K = 800
Sum	97.77 ± 0.09	98.41 ± 0.03	98.55 ± 0.07
Max	97.90 ± 0.02	98.14 ± 0.05	98.49 ± 0.02

4.9 Comparing the Architectures

Returning to the main question of whether there is a difference in performance using a DBoW architecture compared to a regular BoW architecture, the best performances of the Beta and Gamma architectures are compared to the best performance of the Alpha architecture.

Table 6: Results on CIFAR-10 using the Gamma architecture with standard error (% correct)

CIFAR-10	K = 200	K = 400	K = 800
Sum	43.61 ± 0.25	47.77 ± 0.14	51.53 ± 0.35
Max	47.22 ± 0.32	49.24 ± 0.31	46.06 ± 1.96

- The best mean performances measured with the Alpha architecture are 99.26% and 60.33% correct for MNIST and CIFAR-10 respectively. The best mean performances measured with the Beta architecture are 56.90% correct for CIFAR-10 and 98.93%, 98.91% for MNIST, using sum-pooling and max-pooling respectively. The best mean performances measured with the Gamma architecture are 51.33% correct for CIFAR-10, and 98.55%, 98.49% for MNIST using sum-pooling and max-pooling respectively.
- A two-sided t-test shows the performances of the Beta architecture are worse than the performances with the Alpha architecture ($t(17.92) = 10.68$, $p << 0.05$), ($t(14.90) = 8.62$, $p << 0.05$) for MNIST, using sum-pooling and max-pooling with the beta architecture respectively, and ($t(17.77) = 18.13$, $p << 0.05$) for CIFAR-10.
- A two-sided t-test shows the performances of the Gamma architecture are worse than the performances with the Alpha architecture ($t(14.69) = 17.31$, $p << 0.05$), ($t(13.71) = 32.40$, $p << 0.05$) for MNIST, using sum-pooling and max-pooling with the gamma architecture respectively, and ($t(16.17) = 40.23$, $p << 0.05$) for CIFAR-10.

5 Discussion

In general the performance seems to decline when more layers are added to the system. If we compare this to the Deep Bag of Words model as described by [8], it is noted that the performance of the architecture used there improves with the addition of layers to the BoW model. A difference

between the architecture described here, and the architecture described in [8], is their usage of receptive field learning.

Using receptive field learning not all centroid activations from a previous layer influence the centroid activations at the current layer. This selective choice might be essential for deep architectures. Deep CNN kernels can learn to ignore certain inputs from a previous layer, by setting the respective weights to a low value. The kernels in this thesis are learned using K-means and all centroid activations from a previous layer to contribute equally.

Another difference between the architectures are the distance functions used. The architecture in [8] uses a cosine distance, where the architecture in this thesis uses a squared Euclidean distance.

For almost all architectures the best performances are achieved using sum-pooling, except for MNIST using the architecture. With sum-pooling the performances seem to increase as the number of centroids increase. For the results on the MNIST dataset the same effect is observed with max-pooling. Though when max-pooling is used with the CIFAR-10 data-set, a performance drop is observed when the number of centroids are increased from $K = 400$ to $K = 800$.

6 Conclusion

The Deep Bag of Words architecture as described, shows to perform worse than the regular Bag of Words architecture. Methods for linking pooled centroid activations to a next BoW layer and different distance functions might be suitable for further research.

References

[1] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[3] G. E. H. Y. LeCun, Y. Bengio, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[4] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints.," *European Conference on Computer Vision(ECCV)*, vol. 8, pp. 1–22, 2004.

[5] A. Coates, H. Lee, and A. Y. Ng, "An analysis of single-layer networks in unsupervised feature learning," *Journal of Machine Learning Research*, vol. 15, pp. 215–223, 2011.

[6] Y. LeCun, F. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting.," *The IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

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

[8] A. Coates, "Demystifying unsupervised feature learning," Master's thesis, Stanford University, California, 2012.

[9] O. Surinta, T. K. Karaaba, L. R. B. Schomaker, and M. A. Wiering, "Recognizing handwritten characters with local descriptors and bags of visual words," *Engineering Applications of Artificial Intelligence*, vol. 45, pp. 405–414, 2015.

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

[11] J. MacQueen, "Some methods for classification and analysis of multivariate observations," *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297, 1967.

[12] Y. LeCun and C. Cortes, "The MNIST dataset," 1998.