Exploring the SVM-KNN with Vision Problems

Joon Hyuck Choi
The Johns Hopkins University
3400 N Charles Street
Baltimore, MD 21218, USA
jchoi100@jhu.edu

Joo Chang Lee
The Johns Hopkins University
3400 N Charles Street
Baltimore, MD 21218, USA
jlee381@jhu.edu

Abstract

The *k*-Nearest Neighbor (KNN) algorithm is a non-parametric method used for classification and regression. In a previous assignment, we implemented the KNN classifier and one of its variants, the distance weighted KNN. We felt the need to take another step from this assignment and explore another type of the KNN algorithm, which is more involved.

The KNN variant that we explore in this work is the SVM-KNN, a KNN algorithm that makes use of a kernel multiclass support vector machine (SVM) as a subprocedure. The nearest neighbor approach in visual recognition problems has proven to work well in the past. However, despite its benefits, the nearest neighbor approach may suffer from high variation due to finite sampling. The incorporation of an SVM can remedy this situation. The SVM-KNN algorithm attempts to use the standard KNN algorithm with a unanimous voting scheme and then turns to a multiclass SVM should there be a disagreement.

Using an SVM can be effective in the neighborhood of a small number of examples and classes [1]. We tested this algorithm and compared the performance with the standard KNN on two datasets: MNIST and USPS. We also tested for various values of K on both the KNN and SVM-KNN algorithms and different kernel functions on the SVM-KNN.

1 Introduction

Dynamics in college classrooms have changed drastically in the past decade. Just about ten years ago, rarely did students open their laptops in lecture to take notes−rarely did professors allow students to open their computers in class. Today, most students take notes using their laptops. Reasons range from the fact that many students these days have bad penmanship, they are used to typing on their computers, or electronic archives are easier to refer to in the future. However, some students still take notes using pen and paper because they can easily draw figures presented in class and get less distracted on their computers. Nevertheless, it is convenient to have electronic notes. As a result, we became interested in exploring optical character recognition (OCR). We chose to focus on one of the most basic problems in OCR: handwritten digit recognition.

2 Background

Performance on visual category recognition has improved by great amounts in the past decade. However, many algorithms are still far from reaching human level performance or are too slow to train. From positive results that past works in the literature have acheived in visual category recognition including the USPS zip code dataset by [5] and shape context based distance on the MNIST dataset [2], the authors of [1] concluded that exploring other approaches using the nearest neighbhor algorithm could yield fast and accurate results.

The motivations behind using the nearest neighbor approach in visual cateogry recognition are as follow. First, the nearest neighbor approach does not require the explicit construction of a feature space. Second, the nearest neighbor algorithm solves a multiclass classification naturally. Third, as the sam-

ple size grows large, the error rate of a nearest neighbor algorithm may decrease [1].

However, because we are given a limited amount of sample data, the KNN algorithm may suffer from high variation. The incorporation of an SVM mitigates this problem. First, the SVM allows for the use of various distance functions. Second, with the finite neighborhood passed by the KNN classifier as input, an SVM trains faster and performs multiclass classification more naturally than it would when given the entire dataset as input. Lastly, the authors of [1] note that the motivation also came from human psychophysics. Humans first perform quick coarse categorization and then perform more accurate and slower discrimination with time when performing visual recognition tasks. The KNN works as the preliminary pruning process, and the SVM performs the fine discrimination stage.

3 Datasets

We mainly used two datasets: MNIST and USPS, both of which are datasets of handwritten digits.

3.1 MNIST

MNIST [1] (Mixed National Institute of Standards and Technology database) is a database of handwritten digits commonly used for training image processing systems. The dataset was created by mixing and normalizing the samples from NIST's original datasets. The dataset contains 60,000 training images and 10,000 testing images. Each image is of size 28x28. Currently, the state of the art Convolutional Neural Network approach achieves an error rate of 0.21% on this dataset [6].

3.2 USPS

The USPS (U.S. Postal Service) dataset constructed in [9] is a dataset of handwritten zipcodes with 7,291 training images and 2,007 testing images. The dataset was obtained from the scanning of handwritten digits from envelopes by the USPS. The images in this dataset have been deslanted and size normalized from the original images. Each image is 16x16 grayscale.

[1] http://yann.lecun.com/exdb/mnist/

4 SVM-KNN Algorithm

In this section, we describe the SVM-KNN algorithm proposed by [1]. Then we give short overviews of the SVM and KNN algorithms.

Algorithm 1 (L_2)
For input instance x,

(1) Compute distances of x to all training instances and pick K nearest neighbors.

(2) If the K neighbors have the same labels, x is labeled and terminate.
Else, compute the pairwise distances between the K neighbors.

(3) Convert the distance matrix to a kernel matrix and apply multiclass SVM.

(4) Use resulting classifier to label x.

For USPS, run the following algorithm as well.

Algorithm 2
For input instance x,

(1) Find $K_{sl} \approx 10K$ neighbors using L_2 distance.

(2) Compute the tangent distance function on the K_{sl} samples and pick the K nearest neighbors.

(3) Compute the pairwise tangent distance of the union of the K neighbors and x.

(4) Convert the pairwise distance matrix into a kernel matrix using the "kernel trick".

(5) Apply SVM on the kernel matrix and label x using the resulting classifier.

Choosing larger values for K_{sl} does not improve empirical results [1].

4.1 Support Vector Machine (SVM)

Support vector machines are supervised learning models used for classification and regression analysis. Given a set of training instances, each with a corresponding label, an SVM builds a machine learning model that assigns new instances to one category or the other. An SVM is a non-probabilistic classifier. It is a representation of the instances in space,

mapped so that the instances of different classes are divided by a gap that is as wide as it can be (max-margin principle). A new instance is mapped into the space and predicted to belong to a class based on the side of the boundary that it falls into. An SVM also allows for non-linear classification with the help of the kernel trick, which implicitly maps inputs into high-dimensional feature spaces.

4.2 *k*-Nearest Neighbors (KNN)

The k-Nearest Neighbhors Algorithm is a machine learning method used for classification and regression. The input consists of the K nearest training instances in the feature space. In a classification task, the output is a prediction of the class of the given input. The KNN algorithm decides on this prediction by using a majority vote of the input instance's neighbors. The input instance is assigned the label of the class most common among its K neighbors.

4.3 Implementation

We based our KNN code from our previous homework assignment. The code in the `predict` function needed the most modification. We needed to take into account the different ways that the USPS and MNIST datasets were encoded from the way the dataset files we were given in class (e.g. `bio.train`). Also, we changed the voting scheme to be unanimous, not majority. In `classify.py`, we needed to modify the `load_data` function to be able to correctly read and parse the USPS dataset so that it can be readily passed into our methods. Moreover, we needed to adjust the `main` function in this file to correctly handle the two datasets separately.

For the SVM portion of the code, we used `scikit-learn`'s `svm.SVC`[2] with a linear kernel as the kernel. For the tangent distance, we modified the RWTH-i6 C-implementation from an online source[3] to serve our purpose in Python, and applied kernel trick as defined in [1]. We used `scikit-learn`'s `SVC` with the RBF Kernel option to run experiments and compare results.

Our code can be viewed in our GitHub repository[4]. Run commands are listed in the text file `code/run_commands.txt`.

[2]http://scikit-learn.org/stable/modules/svm.html

[3]https://www-i6.informatik.rwth-aachen.de/ keysers/td/

[4]https://github.com/jchoi100/machine_learning_final_project

K	SVM-KNN	KNN
1	0.9691	0.9691
3	0.9721	**0.9705**
5	0.9741	0.9688
10	0.9764	0.9665
30	0.9815	0.9596
80	**0.9831**	0.9468
110	0.9826	N/A
150	0.9682	N/A

Table 1: SVM-KNN, KNN Accuracy on MNIST.

K	SVM-KNN	KNN
1	0.9437	0.9437
3	0.9467	**0.9447**
5	0.9536	**0.9447**
10	**0.9576**	0.9357
30	0.9547	0.9118
80	0.9507	0.8764

Table 2: SVM-KNN, KNN results on USPS.

5 Experimental Results

We ran the standard KNN algorithm with $K = \{1, 3, 5, 10, 30, 80\}$ on both MNIST and USPS datasets using L_2 distance. We ran the SVM-KNN algorithm with $K = \{1, 3, 5, 10, 30, 80, 110, 150\}$ on the two datsets. We also randomly sampled 7,291 images from the MNIST dataset and ran SVM-KNN to evaluate the difference between MNIST and USPS. (Note that 7,291 is the number of training images in the USPS dataset.) We also ran experiments on the USPS dataset using the tangent distance kernel and MNIST dataset using the RBF kernel. Results and analysis follow in following sections.

5.1 KNN

We implemented and ran the standard KNN algorithm on both MNIST and USPS datasets on $K = \{1, 3, 5, 10, 30, 80\}$. Results are shown in Table 1 and Table 2.

5.2 SVM-KNN

We modified the KNN algorithm to use a unanimous voting scheme and integrated `scikit-learn`'s SVM package. For the first part of our experiments, we used a linear kernel and broke ties by choosing

the lower index. For the KNN portion of the algorithm, we used $K = \{1, 3, 5, 10, 30, 80, 110, 150\}$. Reasons for using larger K values (110 and 150) are elaborated in the next section. Results are shown in Table 1 and Table 2.

To compare MNIST and USPS, we randomly sampled the same number of training images from the MNIST dataset as the USPS dataset (7,291 images) and ran SVM-KNN on this sampled dataset. Results are shown in Table 3 and Figure 3.

5.3 Tangent Distance

Tangent distance is the minimal distance between the linear surfaces that best approximate the two non-linear manifolds from each image as defined in [10]. The distance in the implementation was calculated from seven transformations of the images, which includes shifting, deformation, rotation, and scaling. After the distance is calculated, it is used in the SVM by applying a kernel trick:

$$K(x, y) =< x, y >= \frac{1}{2}(d(x, 0)+d(y, 0)-d(x, y))$$

Unlike [1]'s implementation, the tangents were calculated from raw images without smoothening.

For the USPS dataset, we also ran **Algorithm 2** to observe the effect of finding more "accurate" neighbors, possibly with the same label as the test sample, through tangent distance matching.

5.4 RBF Kernel

Based on the approach taken by [7], we ran SVM-KNN using RBF kernel on the MNIST dataset.

The Radial Basis Function (RBF) Kernel is a kernel in the form of a Gaussian (radial basis) function form. The RBF kernel is defined as

$$K(x, y) = exp\left(-\frac{||x - y||^2}{2\sigma^2}\right)$$

where $\gamma = \frac{1}{2\sigma^2}$ sets the spread of the kernel.

We used `scikit-learn`'s `SVC` module with the `kernel="rbf"` option and ran experiments with $K = \{1, 3, 5, 10, 30, 80\}$. Results are shown in Table 3, plotted in Figure 3.

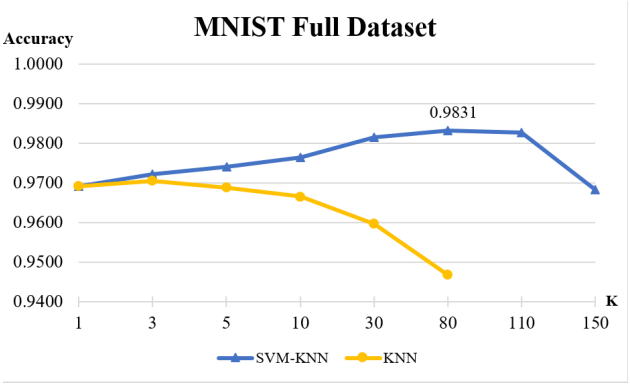

Figure 1: Plot of MNIST results.

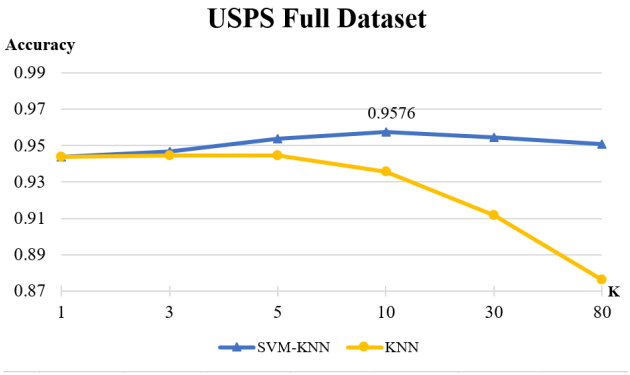

Figure 2: Plot of USPS results.

6 Discussion

6.1 KNN on MNIST and USPS

We can see from Figures 1 and 2 that as K increases, the accuracy decreases using the standard KNN algorithm. As K increases, the algorithm includes an increased number of irrelevant samples that are members of different classes in the neighborhood. Therefore, these irrelevant, incorrect neighbors disrupt the voting and cause incorrect predictions. Naturally, with more samples in the neighborhood that are further away from our input instance, we get more instances that are not in the same class as our input instance. We get lower accuracy as a result.

We perform slightly better on the MNIST dataset than on the USPS dataset. The authors in [1] attribute this to the fact that the USPS dataset is naturally harder. The human error rate on the USPS dataset is 2.5% according to [4].

K	MNIST 7291	RBF Kernel
1	0.9428	0.9691
3	0.9482	**0.9714**
5	0.9527	0.9691
10	0.9600	0.9673
30	0.9646	0.9595
80	0.9667	0.9496
110	**0.9668**	N/A
150	0.9652	N/A

Table 3: SVM-KNN Accuracy on MNIST 7291 and RBF.

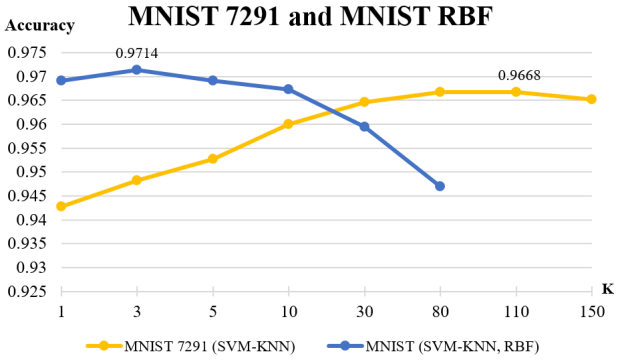

Figure 3: Plots of MNIST 7291 and MNIST RBF results.

6.2 SVM-KNN on MNIST and USPS

In running the SVM-KNN, we can run into three different cases: (1) The KNN algorithm reaches an incorrect unanimous vote, and the algorithm outputs that incorrect result; (2) The KNN algorithm reaches a correct unanimous vote, and the algorithm outputs that correct result; (3) The KNN algorithm does not agree unanimously and seeks help from the SVM. For the third case, there are four subcases based on the combinations of the KNN majority vote being correct or incorrect and SVM being correct or incorrect. Two of those cases ("KNN majority correct $\rightarrow$ SVM incorrect" and "KNN majority incorrect $\rightarrow$ SVM correct") are of interest to us. The case "KNN majority correct $\rightarrow$ SVM correct" is trivial, and the case "KNN majority incorrect $\rightarrow$ SVM incorrect" cannot be helped by the SVM-KNN algorithm. This is due to either the limits of this algorithm or the presence of illegible handwriting that caused the human error rates.

We noticed empirically that (1) does not happen very often. It may occur occasionally when the sample image contains messy handwriting.

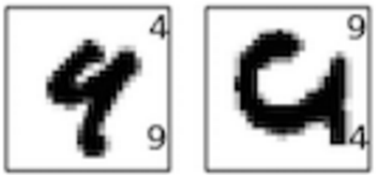

Figure 4: Examples of errors between "4" and "9".

Case (3) was where we saw interesting results. Previously in the KNN experiment, the KNN algorithm reached a majority vote on the wrong classification occasionally. However, for many of those cases, the SVM-KNN algorithm outputted the correct classification nonetheless. For example, for one particular instance, KNN outputted a prediction of "4" even though the correct label was "9" because the majority of the sample's neighbors were of class "4" (Figure 4). However, when this sample was run on SVM-KNN, the algorithm outputted "9". We conclude that this is due to the fact that SVM uses a max-margin principle. Therefore, although a particular instance may have many neighbors that are not of the same type, the *max-margin*-ed decision boundary learned by the SVM will correct this error.

However, we also contemplated about the case where the KNN would have been correct by majority vote, but the SVM outputted the wrong classification. We conclude that such cases may happen when our test instance is very close to a support vector on the other side of the decision boundary. Therefore, although the majority of its neighbors are of its own type, SVM may give a wrong prediction.

Furthermore, we noticed that the authors in [1] provided results for only a single K value for each of the experiments. In particular, they used $K = 80$ for SVM-KNN and $K = 3$ for KNN on MNIST, and $K = 10$ for SVM-KNN and $K = 3$ for KNN on USPS. We decided to run the standard KNN and SVM-KNN on various K values for both datasets. We first tested with K values of 1, 3, 5, 10, 30, 80 for both datasets. We were able to observe a pyramid shaped SVM-KNN accuracy plot for the USPS data with $K = 10$ serving as the peak (Figure 2). However, the SVM-KNN accuracy for the MNIST dataset required even larger values of K to observe this *pyramid* behavior. We extended the K testing

K	USPS (SVM-KNN)
1	0.9586
3	**0.9621**
5	0.9606
10	0.9601
30	0.

Table 4: SVM-KNN Accuracy using Tangent Distance.

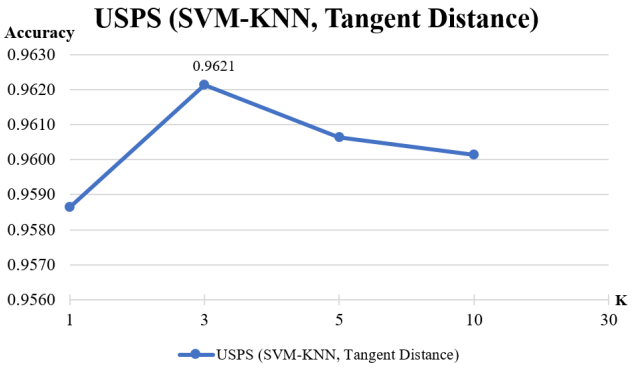

Figure 5: USPS SVM-KNN Tangent Distance results.

range to include $K = 110$ and $K = 150$ to observe declinding accuracy on the MNIST dataset.

We can observe from the SVM-KNN result plots in Figures 1, 2, and 3 that the SVM-KNN accuracy performance increases with larger values of K in the beginning and decreases after a peak at a certain K value. We conjecture that the reason the SVM-KNN improves at first is that the SVM can learn with more helpful samples (i.e. neighbors chosen by the KNN algorithm). However, with extremely large values of K, the KNN algorithm hands the SVM algorithm a neighborhood with many outliers from other labels that create noise and thus disrupt the SVM from correctly constructing the decision boundary. Therefore, we get higher error rates as K keeps on increasing beyond a certain threshold.

6.3 SVM-KNN on 7,291 Sampled MNIST

In order to be able to make qualitative comparisons of the MNIST and USPS datasets, we randomly sampled the same number of data points from the MNIST dataset as there are in the USPS dataset. (The MNIST dataset has 60,000 training images while the USPS dataset has only 7,291 training images.) Then, we ran the SVM-KNN algorithm on the reduced, randomly sampled MNIST dataset (call it *MNIST 7291*) and recorded the results. We tested for K values of 1, 3, 5, 10, 30, 80, 110, and 150.

We were able to observe that the MNIST 7291 dataset displayed the *pyramid*-ing behavior at $K = 80 \ldots 110$, which is fairly close to the *pyramid*-ing K value for the full MNIST dataset. Therefore, we were able to see that the size of the training dataset had a small effect in influencing the accuracy of the SVM-KNN on the entire datasets and also on the *pyramid*-ing behavior.

6.4 Tangent Distance SVM-KNN on USPS

On the USPS dataset, we also ran SVM-KNN, using **Algorithm 2** with a more "accurate" distance, the tangent distance. The experiments were performed with $K = \{1, 3, 5, 10\}$. Experiment with $K = 1$, which does not involve an SVM, was performed as well since the nearest neighbor from L_2 distance may differ from that of tangent distance. Results are shown in Table 4 and Figure 5.

Highest accuracy was achieved at $K = 3$ with an error rate of 3.79%, performing better than any of the experiments using L_2 distance, although lower than those of [1]. One possibility for the cause may be the smoothening of the images mentioned in [1].

6.5 RBF Kernel SVM-KNN on MNIST

The benefits of using the RBF kernel are as follow. First, it is a stationary kernel, meaning that the kernel is invariant to translation. For instance, the RBF kernel will output the same value for $K(x, y)$ as it would for $K(x + \lambda, y + \lambda)$, where λ is any constant valued vector with the same dimension as x and y. Second, the scaling by γ occurs by the same amount in all directions. In other words, it is isotropic. Third, the RBF kernel is infinitely smooth.

However, nonlinear kernels such as the RBF kernel can be slow since we need to evaluate kernel products for each support vector. This problem may be mitigated by the SVM-KNN algorithm since our SVM is given a subset of size K of the entire dataset, and we consider K to be small [1, 7].

According to the approach mentioned in [7], running an SVM (not SVM-KNN) using the RBF kernel yields better accuracy. However, as mentioned

by the authors of [7], this comes with a cost of extremely high computational expense, and we were not able to run the SVM on the entire dataset of 60,000 training and 10,000 test images in MNIST. We ran the SVM-KNN with RBF kernel instead. Results are shown in Table 3 and Figure 3.

7 Difficulties Faced

7.1 Experiment Run Time

The experiments we performed took extremely long to run. While KNN takes almost no training time, it takes very long during testing [7]. Had we used an external library's KNN algorithm (e.g. `OpenCV`, `scikit-learn`), the running time might have gone down. However, we used our own code since we needed to 1) change the voting scheme to be unanimous; 2) incorporate the SVM as a subprocedure. Moreover, because each test run took so long to complete, it was harder to find bugs in our program. And analysis of results from running our algorithm under a specific configuration on various K values took very long to produce complete output.

7.2 Analytical Difficulty

Some of the more "accurate" distance functions were beyond the scope of our project. Shape context required sampling points from the Canny edges, bipartite graph matching ($O(N^3)$), scaling of the image (e.g. 28x28 to 70x70 for MNIST [1]) and computing multiple distances, such as Appearance cost, Shape Context cost, and Transformation cost. Computing tangent distance on large data also caused memory issues, thus experiments with tangent distance on MNIST was not performed.

8 Conclusion

In this project, we explored the SVM-KNN algorithm on several datasets and compared the results with the KNN algorithm. We experimented with the SVM-KNN and KNN on MNIST and USPS datasets of handwritten digits on various K values and sample sizes. We conclude that the standard KNN algorithm performs worse with increasing K while the SVM-KNN shows a *pyramid*-ing behavior with increasing K. The incorporation of SVM into the KNN algorithm as a subpattern improves the speed and accuracy in visual object recognition tasks for handwritten digit images. Future work needs to explore more on this algorithm's effectiveness on more complex image datasets such as the Caltech 101 [3], Caltech 256 [8], and ImageNet [5].

9 Comparison to Proposal

9.1 Must Achieve

- Create SVM-KNN by using our KNN modified from homework and scikit-learn SVM. **(Done)**
- Write kernel trick function mentioned. **(Done)**
- Parse MNIST and USPS datasets to fit as input to our SVM-KNN impelmentation. **(Done)**
- Run SVM-KNN and KNN on MNIST and USPS using L_2 distance function. Try different K values for SVM-KNN and KNN. **(Done)**
- Compare & analyze experiment results. **(Done)**

9.2 Expected to Achieve

- Use shape context instead of L_2 to compute distance matrix to feed to Kernel SVM when running SVM-KNN on MNIST. **(Not Done)**
- Use tangent distance instead of L_2 to compute distance matrix to feed to Kernel SVM in running SVM-KNN on USPS. **(Done)**

9.3 Would Like to Achieve

- Run SVM-KNN on Caltech 101. **(Not Done)**

  (The implementation was more involved to be completed in the time we had remaining. This portion of the work and more has been proposed as future extensions of this project.)

9.4 Extra Work Not Mentioned in Proposal

- Experiment with RBF kernel on SVM-KNN with MNIST dataset. Refer to [7]. Use various K values when experimenting. **(Done)**
- Experiment with more various values of K than originally planned. This helped us gain a better intuition on the effectiveness and limits of the SVM-KNN algorithm. **(Done)**

[5] http://image-net.org/index

References

[1] Hao Zhang, Alexander C. Berg, Michael Maire, Jitendra Malik. 2006. *SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition*. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

[2] Serge Belongie, Jitendra Malik, Jan Puzicah. 2002. *Shape Matching and Object Recognition Using Shape Contexts*. IEEE Trans. Pattern Anal. Mach. Intell.

[3] L. Fei-Fei, R. Fergus and P. Perona. 2004. *SVM-KNN: One-Shot Learning of Object Categories*. 2004 IEEE Trans. Pattern Recognition and Machine Intelligence. In press.

[4] Jane Bromley and Eduard Sackinger. 1991. *Neural-network and K-nearest-neighbor Classifiers*. Technical Report 11359-910819-16TM, AT&T.

[5] Patrice Simard, Yann LeCun, John Denker. 1993. *Efficient Pattern Recognition Using a New Transformation Distance*. In NIPS, pages 50-58, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[6] Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, Rob Fergus. 2013. *Regularization of Neural Networks using DropConnect*. International Conference on Machine Learning 2013.

[7] Subhransu Maji, Jitendra Malik. 2009. *Fast and Accurate Digit Classification*. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-159.

[8] Gregory Griffin, Alex Holub, Pietro Perona. 2007. *Caltech-256 Object Category Dataset*. (Unpublished).

[9] Y LeCun, B Boser, JS Denker, D Henderson, RE Howard, W Hubbard, LD Jackel. 1990. *Handwritten Digit Recognition with a Back-Propagation Network*. Advances in neural information processing systems 2, NIPS 1989.

[10] Patrice Simard, Yann LeCun, John Denker, Bernard Victorri. 1998. *Transformation Invariance in Pattern Recognition, Tangent Distance and Tangent Propagation*. Orr, G. and Muller K. (Eds), Neural Networks: Tricks of the trade, Springer.