Handwritten Digit Recognition using SVM and kNN

Mohana Fathollahi, Thanh Binh Viedena Vu

December 12, 2021

Abstract

In this project, we study different factors that affect the accuracy of hand written recognition. For this purpose, the MNIST data set was selected. First, we represent presenting the importance of feature extraction. secondly we demonstrate the effect of kernelized approaches in SVM and KNN. Additionally, we evaluate Log and Power kernels as representative of conditionally positive definite kernels. At the end, we benchmark the used classifiers against each other.

1 Introduction

Nowadays, handwritten Digit Recognition has become widespread with the increased use of Machine Learning. This feature is especially important for Object Character Recognition (OCR). OCR is used for many different purposes, for example by the banks or the postal services that want to scan a customer's handwritten information. Processing forms that are not filled out digitally can hence be streamlined without having to prepare the data manually for further use. But even when filling forms out digitally, handwritten digit recognition allows reading text that was entered directly by the user through a touch screen.

For this report, we first summarized previous work that has been done on the this topic. Additionally, we briefly talk about papers that have been written about a hybrid of SVM and kNN as well as conditionally positive definite kernels.

These kernels are further explained in our theory section. Introducing all the kernels that were used in the scope of this project, we very briefly go over some of their characteristics. Furthermore, the theoretical background of two approaches of SVM and kNN are described.

This is continued by the description of our experiment. After presenting the data set, our results from applying SVM and kNN are demonstrated. Besides showing the difference in results due to HOG extraction, we compare the accuracy of classification using different kernels or metrics. In order to find the best accuracy, we tried to optimize certain parameters needed for the kernels or for kNN.

These results are discussed in the discussion section, where we also compare our own work to the work that has been done in previous literature. We try to conclude why our results differ from them and assess the methods we used.

Before coming to our final conclusion, we contemplate what could have been done additionally and how the work we have done can be continued in the future.

2 Previous work

Comparing SVM and kNN is not a completely new topic. Hence, there has been some previous work on it. In paper [5], SVM and kNN have been put against each other for handwritten recognition. For kNN, it was found that the highest accuracy comes from $k = 1$ being 97.3% whereas the SVM using polynomial kernel achieved the highest accuracy of 97.7%. The difference between these two is not very large, but when taking a look at SVM using rbf-kernel, we can see an accuracy of 98.2%. This finding was supported by [10], although the cost for memory space and computation need to be considered. In the end, it was concluded that the difference in accuracy is rather low with only 1% on their handwritten digit data set using SVM and kNN.

The two approaches have also been put against each other on different data sets. In a classification for anemia by [4], the difference in precision was more than 5% with the difference in accuracy being 10%. It was further found that kNN took almost twice as long as SVM. However, there are also data sets where kNN has better performance than SVM. [7] shows that for Page-blocks, kNN has a performance that is 8% higher than that of SVM. This supports the assumption that the best algorithm heavily depends on the data set and their features.

In order to further analyze both approaches, SVM and kNN were applied to a cursive handwriting data set in a hybrid way [15]. It could be observed that the improvement compared to regular kNN was 1.5%.

Additionally, there are different papers that introduced new kernels, for example in article [3] two new conditionally positive definite kernels have been introduced. These two kernels are the Log and Power kernel, whose performance in image recognition has been studied in the article.

3 Theory

In this section, we briefly explain SVM, KNN and the kernels that have been used in this project.

3.1 Support Vector Machines (SVM)

To classify handwritten images, SVM has been used because SVM is more effective in high dimensional spaces and it works well in unstructured and semi-structured data like text, images and trees.

The basic SVM classifier can draw a linear decision boundary to separate the different classes. It can be extended for non linear data sets that cannot be separated by a linear decision boundary with kernel functions.

The first problem that exists in handwritten classification is that handwritten pictures that are the same for human eyes, are different based on pixel distances. Therefore we should define useful and relevant features, because classification based on irrelevant features will not give us a good result. In this project, we used two invariant representations, Gradient and Histogram representation.

In HOG, an image is broken down to smaller regions, and for each region HOG can provide the edge direction and histogram. These histograms are created based on gradients and orientations of the pixel values.

3.2 k-Nearest Neighbors (kNN)

K-nearest neighbors classification is a common classification method requiring supervised data. This algorithm detects the k members that lie closest to the element that needs to be classified [11]. Based on the labels of these elements, the data will be classified to the majority class. Due to this majority class, an even value for k is avoided to not have ties.

The nearest neighbors are determined using a distance metric. The most common metric is the euclidean distance which calculates the shortest distance between two points.

Let x and x' be two points in a d-dimensional space so that $x, x' \in \mathbb{R}^d$.

$$dist_{Euclidean}(x, x') = \sqrt{\sum_{i=1}^{n}(x_i - x'_i)^2}$$

However, in our experiment, we will also test out two additional metrics to find the metric that best fits our data. The metrics we selected are the Minkowski and Manhattan distance. The first one was used as it is the default metric given by the function we use. The Manhattan distance is another widely used metric.

$$dist_{Minkowski}(x, x') = \sqrt{|x_i - x'_i|^p}$$

$$dist_{Manhattan}(x, x') = |x_i - x'_i|^p$$

kNN uses a memory-based learning as the training algorithm remembers each training example to later apply the classification algorithm. Hence, no real training is necessary. However, the algorithm takes up quite some computation as it has to find the nearest distance out of the whole data set [8].

Additionally to the classic kNN, we also considered kernel kNN. Instead of using the distance metrics above, the following formula is used [14]:

$$dist := \|\phi(x) - \phi(x')\| = \sqrt{k(x, x) + k(x', x') + 2k(x, x')}$$

For this purpose, the linear and polynomial kernel were applied.

3.3 Kernels

In order to conduct our experiment, we will apply kernels for both the SVM as well as a kernel version of kNN. The kernels allow to learn non-linear decision boundary. Kernel functions can be written as a dot product.

$$k(x, x') = \langle \phi(x), \phi(x') \rangle$$

In this equation, ϕ is a function that maps x into a high dimension space. When there are more features, computation will be expensive, but now the Kernel trick will help us to operate in the original feature space with the dot product. In the following, two types of kernels that have been used in this study will be described.

3.3.1 Positive definite Kernels

One of the important things that should be checked in kernel function is positive definiteness. For example in SVM, positive definiteness should be checked because it will guarantee that the optimization problem is convex and the solution that we find is unique [12].

- Linear kernel:
  It is the most basic type of kernel, usually one dimensional and it is faster than other kernels.
  $$K(x, x') = sum(x, x')$$
- Polynomial kernel:
  It is a more general form of the linear kernel and is most popular in natural language processing.
  $$K(x, x') = (x^T x')^d$$
- RBF Kernel:
  It is usually chosen for non-linear data sets. When we do not have prior knowledge about data this kernel will give a proper separation.
  $$K(x, x') = exp(||x - x'||^2 \gamma) \quad \gamma = \frac{1}{2\sigma^2}$$

3.3.2 Conditionally positive definite Kernels

It has been proven that using conditionally positive definite Kernels for SVM is equivalent with associated positive definite kernels. The performance of these two kernels were compared with the RBF and Laplace kernel, and it was concluded that log kernel has a better performance, according to [3].

- Power Kernel:
  $$K(x, x') = exp(-\|x - x'\|^\beta)$$
- Log Kernel:
  This kernel is conditionally positive definite, It has been proven that conditionally positive definite kernels are suitable for SVM algorithm.
  $$K(x, x') = -log(1 + \|x - x'\|^\beta)$$

4 Experiments

4.1 Dataset

In order to conduct our experiment, we used the MNIST database handwritten digits [9]. This database contains 60000 samples in the training and 10000 samples in the testing set. We downloaded MNIST data set from [2] and used two functions to extract image and shape.

MNIST data set is images of digits from 0 to 9. each image has 28 rows and 28 columns. where the value of each pixel or entry in the matrix is between 0 and 255, gray scale image.

4.2 Applying SVM

4.2.1 Effect of feature descriptor

Before using SVM, HOG features for the different values of the pixels per cells were extracted [6]. Based on the table 4.2.1, with less pixels per cell, the number of features that is extracted increased and we will get a better accuracy.

Cells per block	pixels per cell	Feature dimension	Accuracy
without feature extraction		784	91.83%
(2,2)	(14,14)	36	89.42%
(1,1)	(8,8)	81	91.89%
(2,2)	(8,8)	144	94.77%
(2,2)	(4,4)	1296	98.23%

Table 1: Accuracy for different HOGs

Because of limited RAM in our PC, we used the first option that extracts 36 dimensions and we reduced the number of training from 60000 to 20000. In our next step, positive definite kernels and conditionally positive definite kernels have been compared.

4.2.2 Positive definite kernel in SVM

In this study, the positive definite kernels that have been used are linear and RBF kernel. The steps for using linear kernel are trivial, but for RBF kernel we tuned the hyper parameters to find the optimal value for c and gamma. Three folds cross validations have been used to assess the performance of RBF for three values in gamma, 0.01, 0.1 and 1 and three values for c, 1,5 and 10. This has been shown in Figure 1.

One of the hyper-parameter in the grid search is C, C is the penalty of the error term in the training process. By increasing the value of C, the algorithm tries hard to reduce the miss-classified examples during the training phase. To achieve this, a complex decision boundary will be created to separate the classes. If the value of C is very high for the specific data set, it might even try to shape the boundary to cover all the noises as well. This is called over fitting. That's why in last two subplots we see that as move to the right in x-axis the distance between training and test accuracy increases. It means that the training accuracy improves and test accuracy degrades due to over fitting.

Another hyper-parameter is gamma, it is the inverse of the standard deviation of the RBF kernel. In the RBF kernel a Gaussian function is used as a similarity measure between two samples of the data set. A small gamma means a Gaussian with large variance. Therefore, two points can be considered similar

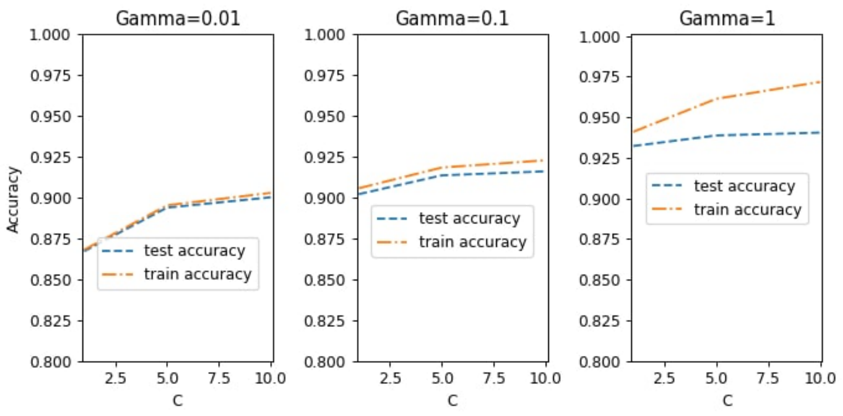


Figure 1: Compare accuracy for different value of c and gamma

even if they are far from each other. As a result, when Gamma is too small for the specific data set, the model cannot capture the complexity of the data. The decision boundary will behave similar to a simple linear model. We observe this effect in these subplots as well. At gamma=0.01 the model is so simple that both training and test accuracy are lower compared to second and third sub plots with higher gamma values. On the other hand, if gamma is too high, the third subplot. The Gaussian function gets so spiky wherever there is data and close to zero everywhere else. That's why in the third subplot, gamma=1, we observe over fitting even at small C values. It seems that the best accuracy without risk of overfiting is belong to gamma = 0.1 and c = 10.

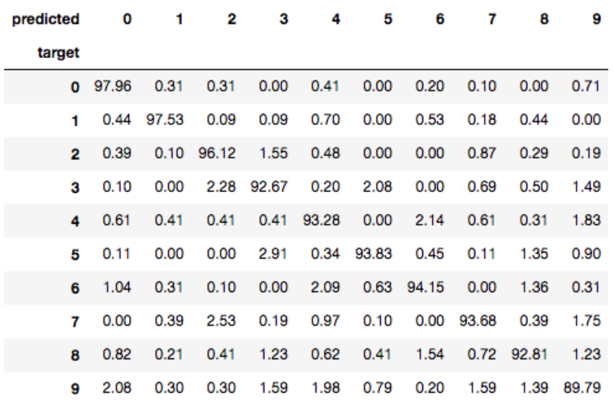
Based on Table 4.2.2, the best model is the RBF kernel with default parameters, $c = 1$ and $gamma = \frac{1}{f * X.var()}$, f is the number of features and x is the training features. The accuracy achieved in linear kernels is 91.65% which is much lower than the accuracy achieved by the non-linear kernel 94.23%. Therefore, it seems that the problem is highly non-linear in nature.

Method	Accuracy
SVM	89.24%
SVM-linear Kernel	91.65%
SVM-RBF Kernel/Default parameters	94.23%
SVM-RBF Kernel/Tuned parameters	91.45%

Table 2: Accuracy of different methods

To understand deeply how the selected model behaves in classifying data sets, a confusion matrix has been used. As we can see in Figure 2 the number of the diagonal axis shows the accuracy for that specific digit, and off diagonal value is the false positive percentage. This result shows that the model is able to have an almost equal performance for all the digits. Lowest accuracy is for digit '9', where the model is confused between '0' and '4'. Intuitively, this is because the semi circular pattern that exists in 9, 0, and 4.
The reason for this error to occur can be related to the features that were

extracted from HOG. If we used 4 pixels per cell and extracted 1296 features, our accuracy would be increased and the number of miss-classifications would decrease as well.



predicted	0	1	2	3	4	5	6	7	8	9
target										
0	97.96	0.31	0.31	0.00	0.41	0.00	0.20	0.10	0.00	0.71
1	0.44	97.53	0.09	0.09	0.70	0.00	0.53	0.18	0.44	0.00
2	0.39	0.10	96.12	1.55	0.48	0.00	0.00	0.87	0.29	0.19
3	0.10	0.00	2.28	92.67	0.20	2.08	0.00	0.69	0.50	1.49
4	0.61	0.41	0.41	0.41	93.28	0.00	2.14	0.61	0.31	1.83
5	0.11	0.00	0.00	2.91	0.34	93.83	0.45	0.11	1.35	0.90
6	1.04	0.31	0.10	0.00	2.09	0.63	94.15	0.00	1.36	0.31
7	0.00	0.39	2.53	0.19	0.97	0.10	0.00	93.68	0.39	1.75
8	0.82	0.21	0.41	1.23	0.62	0.41	1.54	0.72	92.81	1.23
9	2.08	0.30	0.30	1.59	1.98	0.79	0.20	1.59	1.39	89.79

Figure 2: Confusion Matrix

4.2.3 Conditionally positive definite kernel in SVM

Power and log kernel have been used as conditionally positive definite kernels in this study. Based on the formula for log and power kernel mentioned earlier, we should find a value for β. Different values for β are used, as we can see in table 4.2.3 by increasing β our accuracy decreased. The best performance belong to $\beta = 1$ in Log and Power kernel.

β	Method	Accuracy	Method	Accuracy
$\beta = 1$	Log Kernel	93.91%	Power Kernel	93.8%
$\beta = 2$	Log Kernel	93.24%	Power Kernel	91.7%

Table 3: Accuracy of different methods

4.3 Applying kNN

To classify our data using kNN, we decided to first find the metrics that give us the best results. The function `KNeighborsClassifier` from `sklearn.neighbors` uses k=5 as a default and calculates the distance using the Minkowski distance [13]. However, to compare different metrics and choose the optimal k, we apply the classifier for a range of k from 1 to 10. We limited `K_MAX` to 10 since a smaller k allows the algorithm to work faster, while also considering a somewhat larger k. Simultaneously, different metrics are applied with the Euclidean and Manhattan distance being the other two distance metrics we chose. After applying kNN for all the different values, we use the method `score` to evaluate the mean accuracy of our predictions on the test data [1].

Comparing the different plot in Figure 3, we can see that the accuracies for Minkowski and Euclidean distance are always the same. Furthermore, it can be observed that the highest accuracy is achieved with $k = 3$ and it decreases

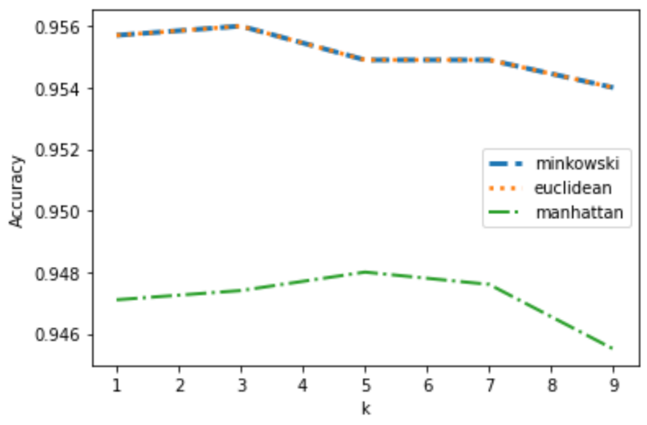


Figure 3: Accuracy for different distance metrics in kNN

afterwards. What we can see from the Manhatten distance is that even the best accuracy here is still below the worst accuracy we calculated for the other distances.

Selecting the highest mean accuracy out of all the scores we collected, we are able to find our optimal values with the Minkowski distance and $k = 1$. Now using these values, we have an accuracy of 95.57%.

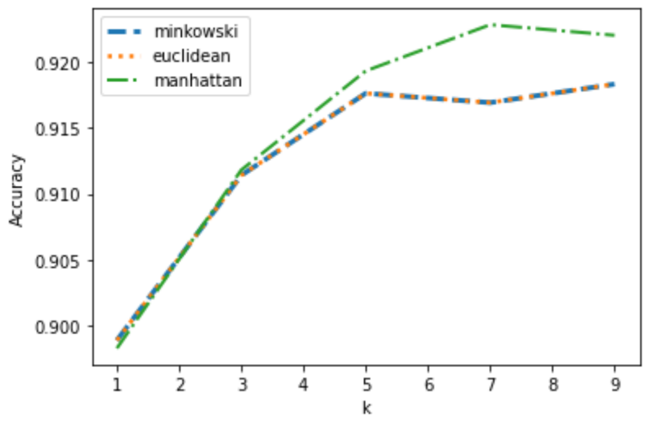


Figure 4: Accuracy for different metrics in kNN using HOG features

However, when applied to the data extracted using HOG in Figure 4, we can see that a high k gives better accuracy than smaller ones unlike using the original data. It could lead to the assumption that an even higher k than $k > 10$ would give a result comparable to what we had using the original data. Interestingly, the Manhattan distance gives us the best accuracy of 88.92% and spikes at $k = 7$.

Afterwards, we compare this value to the accuracy score we got from the kernelized kNN. Before looking at the results, it should be kept in mind that we reduced the number of samples to 5000 here since the algorithm takes a lot of time to execute. As seen in 5, the accuracy for the linear kernel in kNN

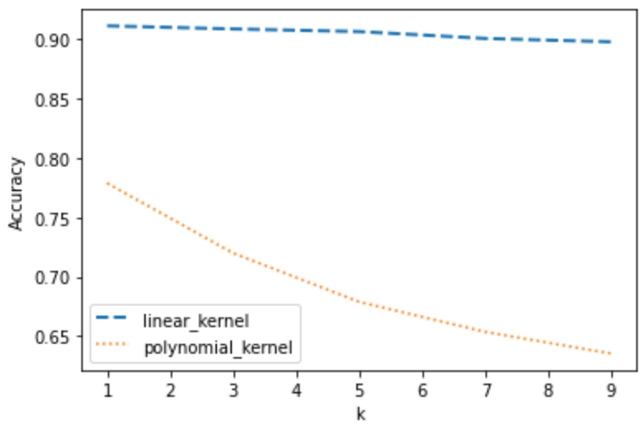


Figure 5: Accuracy for different kernels in kNN

does not change much despite changing k and stays around 90%. However, the polynomial kernel perform the worst out of all the different options we tried so far with its highest accuracy being only 77.86%. This accuracy drops further with the increase of k by almost 14%.

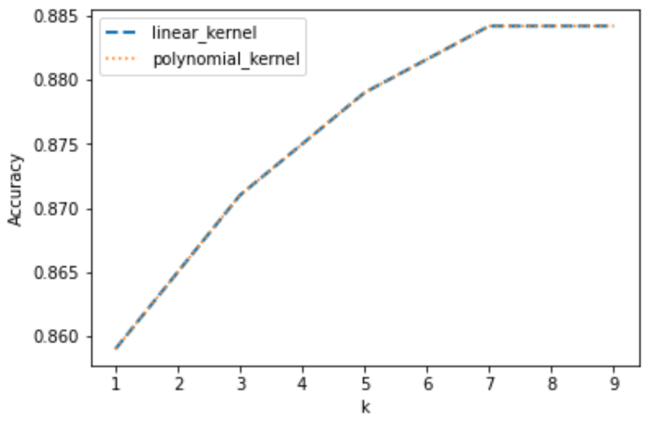


Figure 6: Accuracy for different kernels in kNN using HOG features

Interestingly, when using the HOG features instead of the original data, the accuracies for both kernels are exactly the same, which can be observed in 6. Similarly to kNN with the HOG features previously, the accuracy increases instead of decreasing like with the original data. With an accuracy of 88.42%, the kernel kNN performs very close to the kNN for data extracted with HOG.

5 Discussion

In the experiment section, we tried to use different kernels for SVM and kNN to find the best model. Even though based on the literature survey, we expected that some models have better performance but we saw different behaviour in our project. For example in one of previous works [3] that has been done in image classification, Log and Power kernels have better performance compared to other kernels. In our project we found that RBF has better performance, this might be because we used 1/3 of data set to train the model.

For kNN we found that the choice of the optimal k depends on the used data. While the original data prefers rather small k, kNN with HOG features has higher accuracy with a higher k. This might be due to our HOG extraction not choosing the optimal number of pixels per cell. This leads to the need of evaluating more neighbors to get the right classification as only assessing the nearest neighbor is not the most helpful. However, it should be kept in mind that the accuracy only changes around 1%. As this is not a very large difference, simplifying kNN to nearest neighbor would be possible.

Another aspect, that is noticeable in kNN is the difference between the Manhatten distance compared to the others. As Minkowski and Euclidean distance assess the straight distance throughout all dimensions, it leads do differing results. However, it is interesting to see that Manhatten distance is more suitable after HOG extraction. Unfortunately, due to our smaller data size in kernel kNN, we cannot direcly compare it to our other two approaches, but we can still see the effects of the HOG extraction. While it negatively affects the linear kernel, it improves the accuracy of the polynomial kernel for kNN. However, discussing the reason for this is out of the scope of our work.

The aforementioned difference of results compared to our literature survey also applies when comparing SVM and basic kNN. kNN gives better accuracy with 95.57% while SVM only has an accuracy of 94.77%. We speculate that our mismatch with literature when it comes to this aspect is due to the data we used, since we have a smaller sample size and our SVM used HOG extraction that might have differed from previous work on this topic.

6 Future work

One of the works that can be done in this subject is trying to improve the performance of our models. Even though the MNIST data set offers such a large set of realistic size, we were unable to use all of it due to the RAM of our computers. Similarly, using too many samples for our kernel kNN took multiple hours. Therefore, we reduce number of samples to run the problem in less than an hour. We are aware that the accuracy can be increased by training the model with the whole number of samples or at least more than half of the samples.

In the first section of of this report we observed that the accuracy of linear SVM increased by increasing the dimension of the feature descriptor. In the future we can run all these experiments on higher dimensional features, assuming that we will have access to a more powerful machine.

Considering other kernels and comparing their performance with kernels that have been used in this study is another work can be done in the future, especially studying performance of conditionally positive definite kernels in kNN.

Another direction to extend this work is to run all the experiment on more complex data sets, for example colour image or speech data.

7 Conclusion

As the previous work suggested, benchmarking the best results from SVM and kNN did not show too much difference in accuracy on our limited MNIST data set although kNN gave a slightly better result compared to SVM.

To come to this conclusion, we applied two types of kernel for SVM, conditionally positive definite kernels and positive definite kernels. In both methods we tried to find best parameters to increase accuracy. We get the best accuracy after applying RBF kernel with default parameters.

For kNN, we tried to find the best k and the best metric to find the highest accuracy. Trying out the classifier on both the original as well as the HOG features, we found that the highest accuracy of 95.57% is achieved using the original data with $k = 3$ and the Minkowski distance. Using kernel kNN only decreased the accuracy and increased computation time. However, this could be due to our implementation or the smaller size of data set as well.

Hence, we conclude that based on our own data, kNN gives us better accuracy. However, we should consider that we didn't use the full data set and applied HOG extraction. Furthermore, SVM needs less computation time with a very small difference in accuracy, so even though in this particular case we get better results from kNN, we cannot deduce that kNN always performs better than SVM.

References

[1] k-nearest neighbor classification – pyimagesearch. `https://customers.pyimagesearch.com/lesson-sample-k-nearest-neighbor-classification/` (last accessed: 02.12.2021).

[2] Mnist data set. `http://yann.lecun.com/exdb/mnist`.

[3] Sabri Boughorbel, J-P Tarel, and Nozha Boujemaa. Conditionally positive definite kernels for svm based image recognition. In *2005 IEEE International Conference on Multimedia and Expo*, pages 113–116. IEEE, 2005.

[4] Tajkia Saima Chy and Mohammad Anisur Rahaman. A comparative analysis by knn, svm & elm classification to detect sickle cell anemia. In *2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST)*. IEEE, 2019.

[5] Y. Chychkarova, A. Serhiienkob, I. Syrmamiikha, and A. Karginc. Handwritten digits recognition using svm, knn, rf and deep learning neural networks.

[6] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[7] Mohammad Mahmudur Rahman Khan, Rezoana Bente Arif, Md. Abu Bakr Siddique, and Mahjabin Rahman Oishe. Study and observation of the variation of accuracies of knn, svm, lmnn, enn algorithms on eleven different

datasets from uci machine learning repository. In *2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT)*. IEEE, 2018.

[8] Laszlo Kozma. k nearest neighbors algorithm (knn). *Helsinki University of Technology*, 2008.

[9] Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database, yann lecun, corinna cortes and chris burges. `http://yann.lecun.com/exdb/mnist/` (last accessed: 02.12.2021).

[10] Cheng-Lin Liu, Kazuki Nakashima, Hiroshi Sako, and Hiromichi Fujisawa. Handwritten digit recognition: benchmarking of state-of-the-art techniques. *Pattern Recognition*, 36(10):2271–2285, 2003.

[11] Leif Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[12] Bernhard Schölkopf. The kernel trick for distances. *Advances in neural information processing systems*, 13, 2000.

[13] scikit learn. sklearn.neighbors.kneighborsclassifier. `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier.fit` (last accessed: 02.12.2021).

[14] Kai Yu, Liang Ji, and Xuegong Zhang. Kernel nearest-neighbor algorithm. *Neural Processing Letters*, 15(2):147–156, 2002.

[15] Cleber Zanchettin, Byron Leite Dantas Bezerra, and Washington W. Azevedo. A knn-svm hybrid model for cursive handwriting recognition. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 10.06.2012 - 15.06.2012.