Handwritten Digits Recognition Using SVM, KNN, RF and Deep Learning Neural Networks

Yevhen Chychkarov[a], Anastasiia Serhiienko[b], Iryna Syrmamiikh[a], Anatolii Kargin[c]

[a] *Donetsk State University of Management, Mariupol, Ukraine*
[b] *Pryazovskyi State Technical University, Mariupol, Ukraine*
[c] *Ukrainian state university of railway transport, Kharkiv, Ukraine*

Abstract

This article discusses several classification algorithms of recognizing numbers from photographic images or with manual input, namely: support vector machine (SVM), K-nearest neighbors (KNN), random forest (RF) and several variants of neural networks. The success rates of the algorithms in the field of handwriting recognition were compared.

Six variants of recognition technology were analyzed and tested: using classifier from Scikit-learn package and using deep learning neural networks. To construct and train neural networks or train classifiers, a well-known and rather complete base of handwritten digits MNIST was chosen. Two types of neural networks were considered: sequential and convolutional. The training of neural networks was carried out using a variable number of steps (epochs). Recognition images were scaled to a size of 28x28 (784 cells in one-dimensional representation). Preliminary processing of images (filtering, scaling, etc.) was carried out using the OpenCV library. For recognition, each image of a digit was converted to a 28x28 size and fed to the input of a pre-trained neural network.

A technique to select the area of interest in photographs containing hand-written digits for further recognition has been devised. For handwritten digit recognition, the best recognition accuracy is provided by a convolutional neural network, as 97.6% of car ladle digits were recognized correctly with it.

To improve the recognition accuracy for handwritten digits, it is necessary to perform two additional stages of image preprocessing and dataset transformation.

After building recognition models using all the algorithms mentioned above, the recognition accuracy of all handwritten digits on the test program turned out to be within 98-100%. For industrial images regardless of the used neural network version, the recognition accuracy was 96-98%.

Keywords

Scikit-learn; Classifier; Keras; TensorFlow; MNIST; Python; Deep learning; Neural networks; Digit recognition.

1. Introduction

Handwriting recognition is the ability of a computer or device to accept handwritten text as input from sources such as printed documents, photographs, images, or input stream from other devices. Direct input to the touch screen (this is also a handwriting input), which is interpreted as text, is widely used. An example is smartphones or tablets with a touchscreen that can use handwriting (by finger or stylus) input as text or number input.

There are many applications and technologies currently available for handwriting recognition and optical character recognition (OCR).

CMIS-2021: The Fourth International Workshop on Computer Modeling and Intelligent Systems, April 27, 2021, Zaporizhzhia, Ukraine
EMAIL: influence@meta.ua (Ye. Chychkarov); invitery@i.ua (A. Serhiienko); siv.2017@ukr.net (I. Syrmamiikh); kargin@kart.edu.ua (A. Kargin)
ORCID: 0000-0002-4362-5129 (Ye. Chychkarov); 0000-0003-1328-2572 (A. Serhiienko); 0000-0003-0524-6731 (I. Syrmamiikh); 0000-0003-2885-9071 (A. Kargin)

There are a few examples of technical application of optical number or digit recognition from photographs, i.e. recognition of railway carriage numbers, car license plates, product marking, readings of recording devices, etc.

This article discusses several classification algorithms for recognizing numbers in photographic images or with manual input, namely: support vector machine (SVM), K-nearest neighbors (KNN), random forest (RF) and several variants of neural networks. The algorithms success rates in the field of handwriting recognition were compared. In the methods section of this article, brief information is given about handwriting recognition and compared machine learning methods. In the experimental section, the values obtained as a result of the study were compared. Evaluations were made on the compared machine learning algorithms.

2. Analysis of literature data and formulation of the problem

Optical Character Recognition (OCR) is an important area of research in artificial intelligence and character recognition [1]. For optical character recognition, many applications have been developed that solve the problems of text information extraction, automatic recognition of car license plates and railway carriage numbers [2].

By now a wide range of studies have been carried out, including a comprehensive study and implementation of various popular algorithms for recognizing handwritten text, including handwritten numbers. The task of segmentation and recognition of image areas containing handwritten or printed characters is relevant due to the presence of numerous technical applications.

For example, license plate recognition in work [3] or registration and recognition of wagons and tank numbers in work [4-5] are performed using neural network technologies. The functionality of all these systems is approximately the same – they automate the process of reading numbers and store the received information.

Character recognition is a classification task that includes recognizing a set of characters in an image, dividing them into 10 classes in case of numbers or 26 classes in case of the Latin alphabet letters.

Many systems are currently available for identifying printed text. However, according to [2], the identification of handwritten characters is still a problem in the field of pattern recognition. A large number of research and development dedicated to the OCR system, consider extensive handwritten digits recognition capabilities.

There are some problems that hinder the implementation of character recognition, namely:

- low quality of photo or scanned copy, but this problem is partially solved by image preliminary processing.
- presence of distorted characters, especially when working with handwritten documents or numbers, due to the peculiarities of character style.
- similarity between outlines of some characters.

For example, in work [6], the effect of preliminary processing and segmentation of license plate number images on the results of their recognition was noted. For any recognition methods, incorrect character segmentation does not allow to achieve accurate results.

In works [7-9], [12] the study was focused on the comparison of CNN different models with the fundamental algorithms of machine learning to different attributes, such as performance, runtime, complexity, etc., for their explicit evaluation.

In work [7], the author concluded that the Multilayer Perceptron classifier gave the most accurate results with minimum error rate followed by Support Vector Machine, Random Forest Algorithm, Bayes Net, Naive Bayes, j48, and Random Tree respectively. Authors [8] reported a comparison between SVM, CNN, KNN, RFC and were able to achieve the highest accuracy of 98.72% using CNN (which took maximum execution time) and lowest accuracy using RFC. In work [9] the authors made a detailed comparison of SVM, KNN & MLP models to classify the handwritten text and concluded that KNN and SVM predict all the classes of dataset correctly with 99.26% accuracy, but the process was a little complicated with MLP when it failed classifying number 9, for which the authors suggested to use CNN with Keras to improve the classification.

Improving the accuracy of handwritten digit recognition is achieved by increasing the complexity of the used deep learning neural networks. For example, in [6-7] a convolution neural network for handwritten digit recognition using MNIST datasets was used. The authors of work [10] used a convolutional neural network with 7 layers including 5 hidden layers along with gradient descent and back prorogation model to find and compare the accuracy on different epochs, thereby getting maximum accuracy of 99.2%. In work [11] the same authors briefly discussed different components of CNN, its advancement from LeNet-5 to SENet and comparisons between different model like AlexNet, DenseNet and ResNet. The research outputs: the LeNet-5 and LeNet-5 (with distortion) achieved test error rate of 0.95% and 0.8% respectively on MNIST data set, the architecture and accuracy rate of AlexNet is the same as LeNet-5, but much bigger with around 4096000 parameters and "Squeeze-and-Excitation network" (SENet) became a winner of ILSVRC-2017 since they had reduced the top-5 error rate to 2.25% and by far the most sophisticated model of CNN that exists.

3. Goal and objectives of the research

This paper provides a reasonable understanding of machine learning and deep learning algorithms like SVM, KNN, RF, CNN, and MLP for handwritten digit recognition. Furthermore, it provides information about the algorithm which is efficient in performing the task of digit recognition. The related work that has been done in this field followed by the methodology and implementation of all the algorithms for their better understanding will be discussed in the following sections of this paper. Next, it presents the conclusion and result. The last section of this paper contains used citations and references.

Goals of this research:

1. Evaluation of recognition accuracy for machine learning and deep learning algorithms such as SVM, KNN, RF, CNN and MLP in relation to real sets of handwritten numbers.
2. Analysis of algorithms influence of an image preliminary processing on recognition accuracy.
3. Evaluation of the possibilities to use the considered algorithms for solving technical problems associated with the processing of handwritten digits noisy images (on the example of recognizing cast iron ladle numbers).

4. Methodology

4.1. Equipment and dataset

The comparison of the algorithms (Support vector machines, KNN, Random Forest, Multi-layered perceptron & Convolutional neural network) is based on the characteristic chart of each algorithm on common grounds like dataset, the number of epochs, complexity of the algorithm, accuracy of each algorithm, specification of the device (Ubuntu 20.10, i5 9th gen processor, 8GB memory) used to execute the program and runtime of the algorithm under ideal condition.

To build and train the model, a well-known and fairly complete base of handwritten digits MNIST was selected [13-15]. This database (MNIST - Modified NIST) contains a total of 70,000 handwritten images of numbers and is part of the larger NIST database [16], which contains handwritten images segmented with images of specially prepared templates populated by respondents from the Census Bureau and students of the US educational institutions. In the MNIST database, images from different authors have been placed in different parts to enhance uniqueness.

4.2. Algorithms and methods used

The Support Vector Machine (SVM) was first proposed by Vapnik and since then has attracted a high degree of interest in the machine learning research community [17]. SVM is a supervised machine learning algorithm. During the training, SVM learns the relationship of each data and tag in the existing training set. Typically, data items are placed in n-dimensional space, where n is the number of features. A particular coordinate represents the value of the feature. Points are classified by finding a hyperplane that distinguishes between two classes. The algorithm chooses a hyperplane that separates the classes correctly. SVM chooses extreme vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed as Support Vector Machine. There are mainly two types of SVMs, linear and non-linear SVM [18]. Kernel function selection is an important step in the process of SVM to solve a problem [19]. SVM is successful in solving classification problems compared to many other techniques.

Decision trees form a classification model as a tree structure for the solution of a problem. The tree structure and rules are easy to understand. This simplifies the implementation of the algorithm. Decision trees method consists of simple sequential decision making operations. One of the most important steps in creating the tree structure is choosing the attribute value for which the branching in the tree will be determined [20,21].

A multilayer perceptron is a class of feedforward artificial neural networks that consists of at least three layers: input, hidden, and output. Except for the input neurons, all neurons use a non-linear activation function.

The number of hidden layers can be increased to any number according to the problem, without limitation on the number of nodes. The specific number of hidden layers or the number of nodes in a hidden layer is difficult to determine due to the unstable nature of the model and is therefore chosen experimentally. Each hidden layer in the model can have different activation functions for processing. For teaching purposes, it uses a supervised learning method called backpropagation. In MLP, a node connection consists of a weight that is tuned to synchronize with each connection as the model is trained. [22]

KNN is one of the classification methods. Nearest neighbor searching is the following problem: We are given a set S of n data points in a metric space, X, and the task is to preprocess these points so that, given any query point q ε X, the data point nearest to q can be reported quickly. This is also called the closest-point problem and the post office problem. Nearest neighbor searching is an important problem in a variety of applications [23 The KNN algorithm is used to assign a new observation to the one of the classes that is most common among the k neighbors of a given element, the classes of which are already known. Classification is made according to the threshold value determined with the average of the k data that appears to be the closest. The performance of the method is influenced by the closest neighbor number, threshold value, similarity measurement and sufficient number of normal behaviors in the learning cluster [24-25].

Random forest algorithm can be used in both classification and regression problems like decision trees. The logic of work is to create more than one decision tree and produce average results with the help of these trees. The reason why this algorithm is called random is that it offers extra randomness during the creation of the tree structure. When splitting a node, instead of looking for the best attribute directly, it looks for the best attribute in a subset of random attributes. This situation creates more diverse trees [21, 26-27].

Deep convolutional neural networks (CNNs) are a specialized kind of ANNs that use convolution in place of general matrix multiplication in at least one of their layers [28]. CNN is a deep learning algorithm that is widely used for image recognition and classification. Unlike neural networks with a simpler architecture, which have one or more hidden layers, CNNs are composed of many layers. Such a feature allows them to compactly represent highly nonlinear and varying functions [29]. CNNs involve many connections, and the architecture is typically comprised of different types of layers, including convolution, pooling and fully connected layers, and realize form of regularization [30]. In order to learn complicated features and functions that can represent high-level abstractions (e.g., in vision, language, and other AI-level tasks), CNNs would need deep architectures. Deep architectures, and CNNs, consist of a large number of neurons and multiple levels of latent calculations of non-linearity. According to [31], each level of CNN architecture represents features at a different level of abstraction defined as a composition of lower-level features.

CNN uses a filter (kernel) which is an array of weights to extract features from the input image. CNN employs different activation functions at each layer to add some non-linearity [32].

Many works have noted the importance of image preprocessing for recognizing handwritten numbers or letters.

In particular, [33] applied various preprocessing methods to improve the performance of CNN models. Translations, rotations and elastic deformations have been studied for a variety of in the area of machine learning goals [34-36].

Acording to the authors [33], preprocessing with the combination of elastic and rotation improves the accuracy of the three analyzed networks up to 0.71%.

According to [37], many of today's OCR systems are built following traditional approaches to image processing and work great with printed text but if use them for handwritten text recognition in images it can get unexpected results with poor recognition quality.

The quality of a learned system is primarily dependent of the size and quality of the training set. In work [36] was proposed a simple technique for vastly expanding the training set on base of elastic distortions. In [38] the MINST dataset size was increased by four times by using affine transformations and the prior knowledge of transform invariant properties. In the raised method, the elastic distortions are applied to each sample of the training set to extend nine new samples.

These distortions improve the results on MNIST substantially.

5. Implementation and Computer experiment results

To compare the algorithms based on working accuracy, execution time, complexity, and the number of epochs (in deep learning algorithms) this paper used three different classifiers: Support Vector Machine Classifier, KNN Classifier, Random Forest Classifier, Multilayer Perceptron Classifier.

This set of algorithms was implemented using the Scikit-Learn package in the Python programming language [38].

More complicated Multilayer Perceptron Classifier and Convolutional Neural Network Classifier were implemented using the TensorFlow with Keras frontend package in the Python programming language.

The Keras library contains numerous implementations of widely used building units of neural networks, such as layers, target and transfer functions, optimizers, and many tools to simplify the work with images and text.

5.1. Pre-Processing of Images

To select areas of images containing recognizable digits, we used OpenCV library tools. The findContours function or the Maximally stable extremal region extractor (mser) algorithm were used to highlight the contours of the digits.

To recognize numbers in photographs of real cast iron ladle cars, an algorithm based on boundary detection was used [39]. The algorithm for preprocessing the image and highlighting the area containing the digits of the number included the following stages:

1. image filtering to reduce the noise level (a Gaussian filter was used - cv2.GaussianBlur function);
2. binarization of the image to cut off noise (the cv2.threshold function was used, its parameters were adapted to reliably select the outlines of the digits);
3. highlighting the contrasting borders on the image (the Canny edge detector was used - cv2.Canny function, for which the values of the maximum and minimum values of the gradient were preliminarily selected);
4. filtering to reduce the effect of image heterogeneity (the median filter was used - cv2.medianBlur function);
5. image binarization (cv2.threshold function was also used);
6. morphological transformation (dilatation - function cv2.dilate);
7. selection of contours and their sorting (selection of contours was carried out using the cv2.findContours function);
8. image segmentation, i.e. selection of recognition areas in the form of rectangle set containing previously selected contours of digits (cv2.boundingRect functions were used).

Recognition images to be scaled to a size of 28x28 (784 cells in a one-dimensional representation). Each pixel value of the images lies between 0 to 255 followed by Normalizing these pixel values by converting the dataset into 'float32' and then dividing by 255.0 so that the input features will range between 0.0 to 1.0. Next, one-hot is performed encoding to convert the y values into zeros and ones, making each number categorical.

5.2. Implementation of the considered algorithms

Classical classification algorithms were implemented on the basis of the package Scikit-learn [40]. It is an open source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data pre-processing, model selection and evaluation, and many other utilities.

To implement the Support Vector Machine algorithms, the module sklearn.svm was used. The support vector machines in scikit-learn support both dense (numpy.ndarray and convertible to that by numpy.asarray) and sparse (any scipy.sparse) sample vectors as input. However, to use an SVM to make predictions for sparse data, it must fit for such data. For optimal performance, use C-ordered numpy.ndarray (dense) or scipy.sparse.csr_matrix (sparse) with dtype=float64.

For binary and multiclass dataset classification, scikit-learn implements the SVC, NuSVC, and LinearSVC classes. SVC and NuSVC are similar methods, except for slightly different sets of parameters and they have different mathematical formulations (see section Mathematical formulation). On the other hand, LinearSVC is another (faster) implementation of Support Vector Classification for the case of a linear kernel.

The implementation of C-Support Vector Classification is based on libsvm. The fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples.

The tuning of the classifier was carried out by choosing the type of kernel, the regularization parameter and Kernel coefficient for ‘rbf’, ‘poly’ and ‘sigmoid’ kernel types.

A random forest classifier is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled by the max_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree. The adjustment of this classifier was carried out by selecting the optimal value of max_samples parameter (it is a number of trees in the forest).

Scikit-learn package implements two different nearest neighbors classifiers: KNeighborsClassifier implements learning based on the k-nearest neighbors of each query point, where k is an integer value specified by the user. RadiusNeighborsClassifier implements learning based on the number of neighbors within a fixed radius r of each training point, where r is a floating-point value specified by the user.

The k-neighbors classification in KNeighborsClassifier is the most commonly used technique. The optimal choice of the value k is highly data-dependent: in general, a larger one suppresses the effects of noise, but makes the classification boundaries less distinct.

In this work, KNeighborsClassifier was used with the choice number of neighbors required for each sample.

The original Multilayer perceptron was also implemented using scikit-learn package. As the experience of setting up MLPClassifier showed, the following parameters have the main influence on the accuracy of character recognition:

- hidden_layer_sizes represents the number of neurons in the i-th hidden layer;
- solver represents the solver for weight optimization;
- learning_rate_initdouble represents the initial used learning rate. It controls the step-size in updating the weights (only used when solver=‘sgd’ or ‘adam’).

It was found that a result acceptable in terms of accuracy is achieved when choosing a sufficiently large number of neurons in the hidden layers, sgd or adam optimizers, and a moderate initial training speed.

All variants of the classifier models after tuning on the MNIST dataset were saved using the joblib module.

The other variant of Multilayer perceptron was implemented using TensorFlow package with Keras interface.

The Keras library contains numerous implementations of widely used building units of neural networks, such as layers, target and transfer functions, optimizers, and many tools to simplify the work with images and text.

The model for recognizing digits included the input and output layers, as well as one or more hidden layers. The model was trained using a variable number of steps (epochs). Recognition images to be scaled to a size of 28x28 (784 cells in one-dimensional representation), so the number of neurons in the input and hidden layers was assumed to be 784.

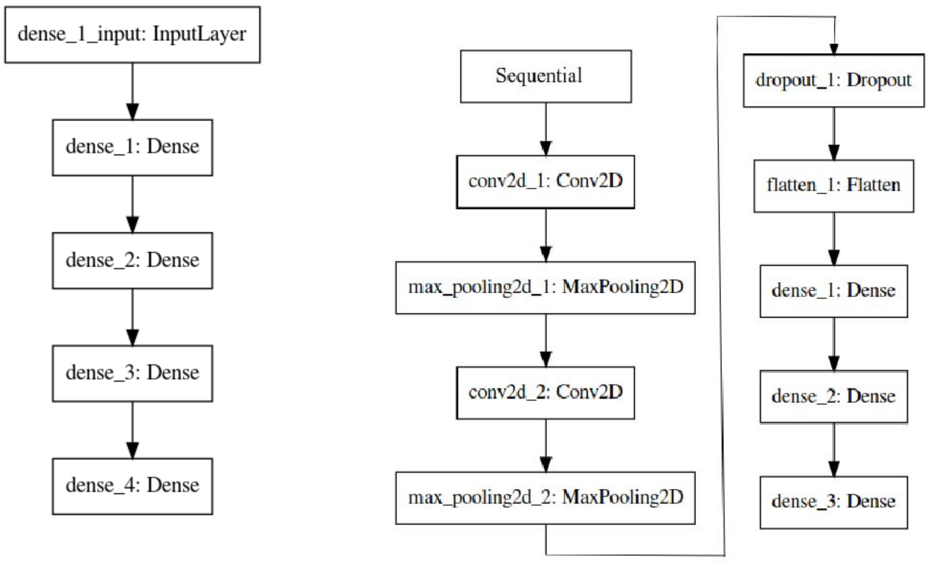
The output level is a layer with 10 nodes tf.nn.softmax, which returns an array of ten probability estimates, the sum of which is 1. Each node contains an estimate that indicates the likelihood that the current image belongs to one of 10 classes.

For recognition, each image of the digit was converted to a size of 28x28, and then fed to the input of a pre-trained neural network.

When setting up the model according to the MNIST test data or a set of images for further recognition, it was found that the recognition accuracy increases slightly with the increase in the number of hidden layers. The model setup is much more sensitive to the choice of the type and parameters of the optimizer and the number of training epochs.

The implementation of handwritten digit recognition by Convolutional Neural Network was done using TensorFlow and Keras.

Variants of structures of the deep learning neural network based on Keras framework, which were used to recognize car number elements, are shown in Fig. 1.



a) sequential neural network (feed-forward network - FNN) with three dense inner layers

b) convolutional neural network (CNN)

Figure 1: Structures variants of the deep learning neural network based on Keras framework, which were used to recognize car number elements

A simpler version of a fully connected neural network (Fig. 1, a) in the test set provided a recognition error level of 1.3-1.5%, but in real examples, the recognition accuracy did not exceed 70% [33].

A more accurate digit recognition of areas of interest was achieved using a convolutional neural network (CNN, see Fig. 1b). The disadvantage of this type of networks is a significantly longer duration of training.

In the sample of 15 ladle car numbers, which included a total of 42 digits (12 three-digit and 3 two-digit numbers), 41 digits were correctly recognized using a convolutional neural network, so that the recognition accuracy was 97.6%. Recognition errors are connected with poorly written numbers.

The stages of constructing and training a model and recognition of practical samples are easily separated due to the possibility of exporting models. To do this, the model.save method (filepath) was used to save the Keras model in one HDF5 file, which contains:

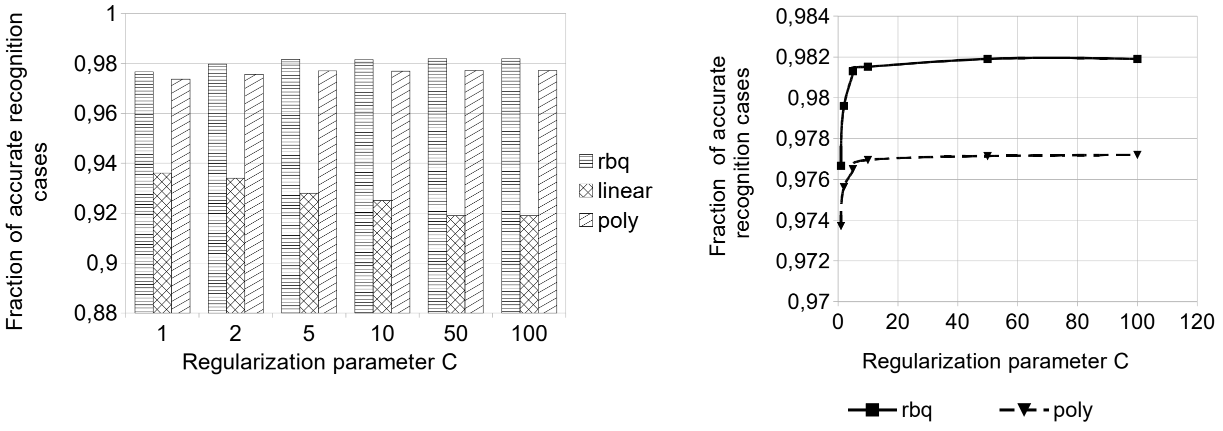
- architecture of the model, allowing to restore the model;
- model weight;
- training configuration (loss calculation function, optimizer)
- state of the optimizer, which allows to resume training exactly where it was stopped.

To restore the model, the function keras.models.load_model (filepath) was further used. This function also allows to build a model using the saved training configuration (if the model has never been compiled).

However, when using customized models for recognizing handwritten numbers for other technical objects – carriage numbers, photographs of meter readings, etc., for all model variants (including those based on CNN), the number of errors increased sharply – up to 40-50%, depending on the dataset.

With proper tuning of all the models mentioned above (from KNN to CNN – as the complexity grows), the accuracy of the MNIST test sample estimate was 97.5-98.5%, which is slightly inferior to the best results achieved using preconfigured or convolutional neural networks.

The results of the SVC classifier as part of the scikit-learn package are shown in Fig. 2.



a) comparison of recognition results for different kernels (kernels «rbq», «linear», «poly» were used)

b) comparison of recognition results for different values of the regularization parameter

Figure 2: The results of the recognition accuracy evaluation for SVC classifier with different calibration parameters

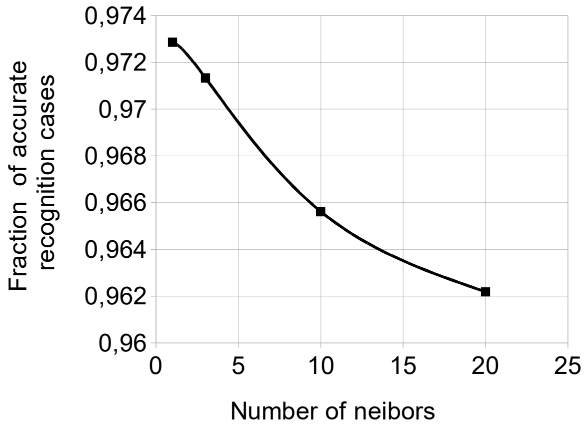
As can be seen from the results shown in Fig. 2, the best accuracy of image recognition from the MNIST sample using the SVC classifier is achieved for the «rbf» kernel and a regularization parameter of at least 50.

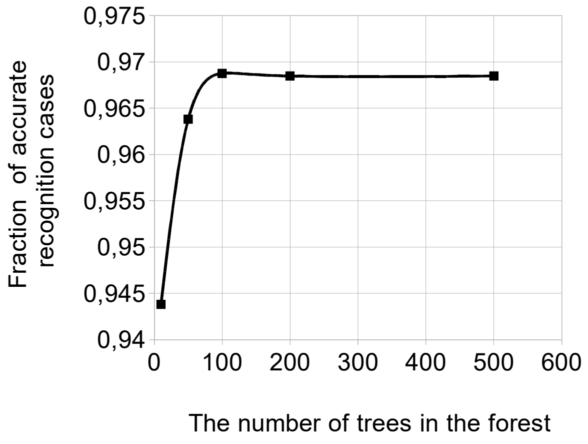
Similar studies were carried out for other classification options - KNN Classifier, Random Forest Classifier, Multilayer Perceptron Classifier. The results of the analysis of the influence of the tuning parameters of the KNN Classifier and Random Forest (RF) Classifier classifiers are shown in Fig. 3.

As can be seen from fig. 3, a decrease in the number of nearest neighbors for the KNN Classifier or an increase in the number of trees in the forest for RF Classifier leads to an increase in recognition accuracy. However, both of the considered classification algorithms do not have any reserves for improving the recognition accuracy by varying the settings.

The settings of the multilayer perceptron had a great influence on the recognition accuracy. The results of the computational experiment are shown in Fig. 4. As can be seen from the graphs presented, the recognition accuracy of digit samples from the test sample increases with an increase in the number of neurons in the hidden layers (it was assumed to be the same for all layers) and with an increase in the number of hidden layers.

The graphs in Fig. 4 are built using the MLP Classifier method from the scikit-learn package. For a similar classifier built using the Keras package, similar results were obtained for the effect on the recognition accuracy of the number of hidden layers and the number of neurons in a layer.





a) KNeighborsClassifier results b) Random Forest Classifier results

Figure 3: The results of the recognition accuracy evaluation for KNeighborsClassifier and Random Forest Classifier with different calibration parameters

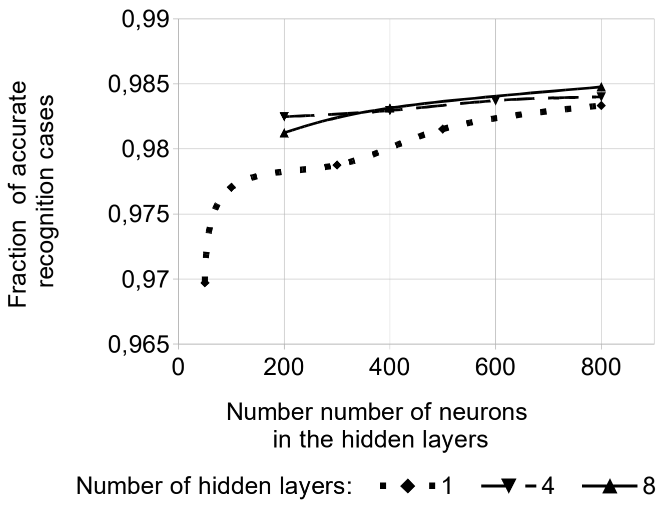


Figure 4: The results of the recognition accuracy evaluation for MLP Classifier (scikit-learning package) with different parameters

When testing various recognition models, some features of the MNIST dataset were established (fig. 5):

- all images of numbers have a clearly defined border area;
- images are grayscale, the brightness of pixels varies across the width of the digit image;
- the images of the numbers are centered in relation to the 28x28 area.

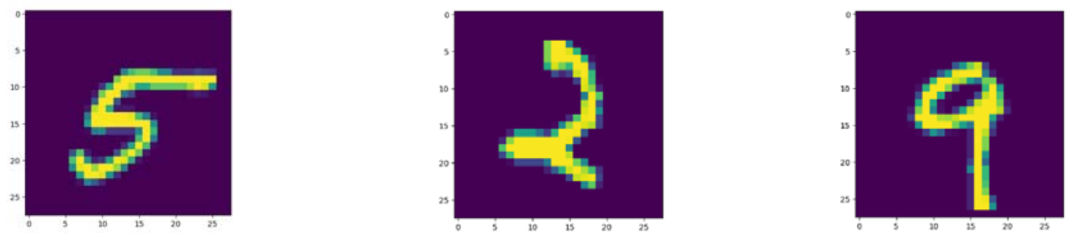


Figure 5: Sample images from the MNIST database

However, the area of interest images for recognition are actually black and white, either due to the binarization of the image at a preprocessing stage or are initially black and white.

To improve the recognition accuracy, it is necessary to perform two additional stages of image preprocessing:

- - after highlighting the area of interest contour exactly along the boundaries of the digit, this part of the image is centered in the square area;
- - the border of the image is added with a width of 15-25% of the size of the square area.

The width of the border area significantly affects the reliability of digit recognition (both printed and handwritten).

Results of the study of influence of the added width of the boundary region in the recognition accuracy set of handwritten characters 0-9 are shown at Fig. 6. A computational experiment for constructing this curve was performed using Scikit-learn implementation of the SVC algorithm.

An analysis of the effect of the width of the added boundary region was carried out for variants of digit recognition using neural networks. The shape of the curve in Fig. 6 for variants with a multilevel perceptron or a CNN network remained exactly the same as for the classification algorithms. An example of recognizing a set of handwritten digits 0-9 is shown in Fig. 7.

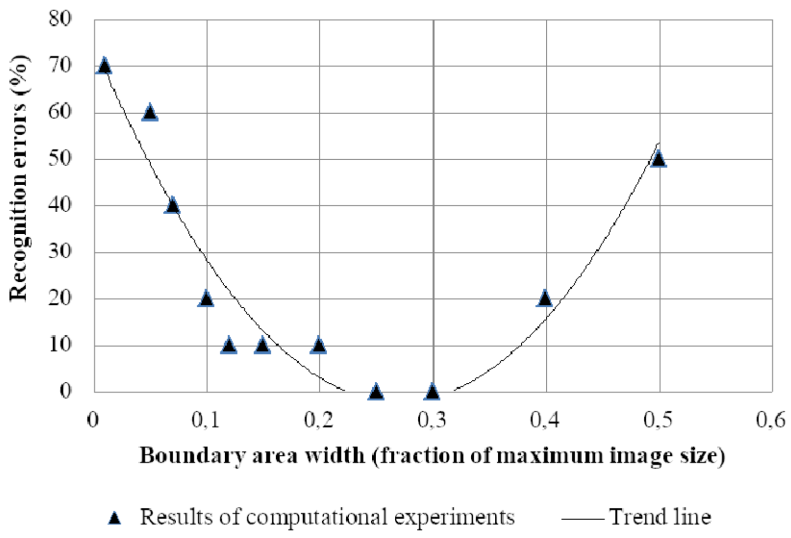
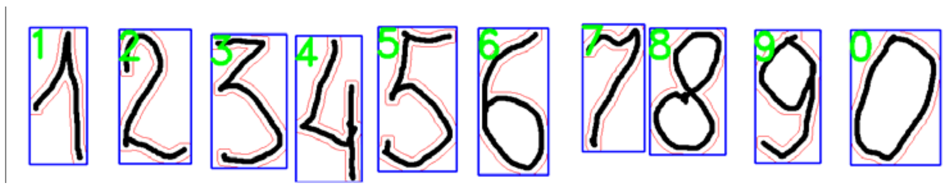
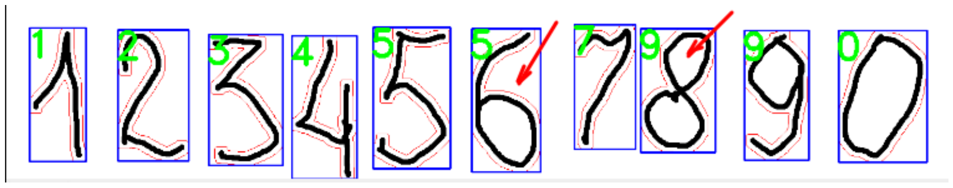


Figure 6: Dependence of the recognition error for a group of handwritten numbers on the width of the added border area



a) Border width 20% of the maximum image size. Recognition accuracy is 100%.



b) Border width 50% of the maximum image size. Recognition accuracy 80% (recognition errors is indicated by arrows).

Figure 7: Examples of recognition results with CNN deep learning network for a group of handwritten digits with different widths of the added border area

To assess the reasons of the recognition failure, a test program in Python was developed that allows to visually draw numbers and monitor the state of the recognition area and the result (Fig. 8).

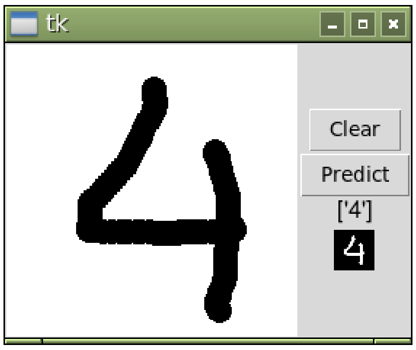


Figure 8: Appearance of the program window for testing handwritten character recognition

After building recognition models using all the algorithms mentioned above, the recognition accuracy of all handwritten digits on the test program turned out to be in the range of 98-100% (one or no errors per 50 drawn digits).

Another reserve for improving the recognition accuracy is setting up the base for building models to work with black and white images. For this, images from the MNIST dataset were converted to black and white.

The results obtained are largely related to the peculiarities of highlighting areas of interest containing digits in the image. The selection of the contour (the findContours function in OpenCV) is carried out exactly along the border of the founded contour. The selected area in the form of a rectangle described around the contour of the digit, without region of interest additional processing, noticeably differs from the base for recognition (MNIST).

A similar result has been achieved in industrial images. Regardless of the used neural network version, the recognition accuracy was 96-98%. For elements of the ladle car number binarization and morphological “closing” operations (MORPH_CLOSE in terms of the OpenCV library) were performed before recognition.

6. Conclusion

This research paper has implemented some models namely Support Vector Machine Classifier, KNN Classifier, Random Forest Classifier, Multilayer Perceptron Classifier, Multi-Layer Perceptron and Convolutional Neural Network for handwritten digit recognition using MNIST datasets. It compared them based on their working accuracy.

1. After a simple setup, all these algorithms demonstrated almost the same accuracy of handwritten digit recognition, differing within +1%.
2. It was found that CNN gave the most accurate results for handwritten digit recognition, but the only drawback is that it took an exponential amount of computing time.
3. To improve the recognition accuracy, it is necessary to perform two additional stages of image preprocessing and dataset transformation:
   - after highlighting the area of interest contour exactly along the boundaries of the digit, this part of the image is centered in the square area;
   - the border of the image is added with a width of 15-25% of the size of the square area;
   - converting images from the MNIST dataset to black and white form.

After building recognition models using all the algorithms mentioned above, the recognition accuracy of all handwritten digits on the test program turned out to be in the range of 98-100% (one or no errors per 50 drawn digits). A similar result was obtained when recognizing the generated sets of digits with different shapes – the recognition accuracy reaches the recognition accuracy on the MNIST test sample. For industrial images regardless of the used neural network version, the recognition accuracy was 96-98%.

7. References

[1] Pramanik, R., Bag, S., (2018). Shape decomposition-based handwritten compound character recognition for Bangla OCR. Journal of Visual Communication and Image Representation, 50, 123-134. doi:10.1016/j.jvcir.2017.11.016

[2] Sarkhel, R., Das, N., Saha, A. K., Nasipuri, M., (2016). A multi-objective approach towards cost effective isolated handwritten Bangla character and digit recognition. Pattern Recognition,58, 172-189. doi:10.1016/j.patcog.2016.04.010

[3] González, A., Bergasa L. M., (2013). A text reading algorithm for natural images. Image and Vision Computing. 31, 3, 255–274. doi: 10.1016/j.imavis.2013.01.003

[4] Sanver, U., (2014). Identification of train wagon numbers. In: Proceedings of the 2014 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference, St. Petersburg, 2014, pp. 63-68. doi: 10.1109/ElConRusNW.2014.6839203.

[5] Lisanti, G.,·Karaman, S.,·Pezzatini, D.,·Del Bimbo, A., (2015). A Multi-Camera Image Processing and Visualization System for Train Safety Assessment. https://arxiv.org/pdf/1507.07815.pdf

[6] Surekha, P., Gurudath, P., Prithvi, R., Ritesh Ananth, V.G., (2018). Automatic license plate recognition using image processing and neural network. ICTACT journal on image and video processing, 08, 04, 1786-1792. doi: 10.21917/ijivp.2018.0251.

[7] Shamim, S. M., Miah, Md Badrul, Sarker, Angona, Rana, Masud, Jobair, Abdullah. (2018). Handwritten Digit Recognition Using Machine Learning Algorithms. Indonesian Journal of Science and Technology. 18. 10.17509/ijost.v3i1.10795.

[8] Dutt, Anuj, Dutt, Aashi, (2017). Handwritten Digit Recognition Using Deep Learning, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET). 6, 7, 990-997.

[9] Norhidayu binti Abdul Hamid, Nilam Nur Binti Amir Sharif, (2017). Handwritten recognition using SVM, KNN, and Neural networks, https://arxiv.org/pdf/1702.00723.pdf

[10] Siddique, Fathma, Sakib, Shadman, Siddique, Abu Bakr. (2019). Recognition of Handwritten Digit using Convolutional Neural Network in Python with Tensorflow and Comparison of Performance for Various Hidden Layers. https://www.researchgate.net/publication/335908757_Recognition_of_Handwritten_Digit_using_Convolutional_Neural_Network_in_Python_with_Tensorflow_and_Comparison_of_Performance_for_Various_Hidden_Layers

[11] Farhana, S., Abu Sufian, Dutta, P., Advancements in Image Classification Using Convolutional Neural Network. https://arxiv.org/pdf/1905.03288.pdf

[12] Wang, H., Zhou, Z., Li, Y., Chen, Z., Lu, P., Wang, W., Liu, W., Yu, L., (2017). Comparison of machine learning methods for classifying mediastinal lymph node metastasis of non-small cell lung cancer from 18F-FDG PET/CT images. EJNMMI Research 7:11 DOI 10.1186/s13550-017-0260-9. https://arxiv.org/pdf/1702.02223.pdf

[13] Arif, R. B., Siddique, M. A. B., Khan, M. M. R., Oishe, M. R., (2018). Study and Observation of the Variations of Accuracies for Handwritten Digits Recognition with Various Hidden Layers and Epochs using Convolutional Neural Network. In: 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT), Dhaka, Bangladesh, 2018. pp. 112-117.

[14] Siddique, F., Sakib, S., Siddique, M.A.B., (2019). Handwritten Digit Recognition using Convolutional Neural Network in Python with Tensorflow and Observe the Variation of Accuracies for Various Hidden Layers. Preprints 2019, 2019030039. doi: 10.20944/preprints201903.0039.v1.

[15] LeCun, Y., The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist.

[16] Grother, P.J., Nist special database 19 – handprinted forms and characters database // National Institute of Standards and Technology (NIST), Tech. Rep. 1995. https://www.nist.gov/srd/nist-special-database-19.

[17] Vapnik, V., The Nature of Statistical Learning Theory. NY: Springer-Verlag (1995).

[18] Srivastava, Durgesh, Bhambhu, Lekha. (2010). Data classification using support vector machine. Journal of Theoretical and Applied Information Technology. 12. 1-7. http://www.jatit.org/volumes/research-papers/Vol12No1/1Vol12No1.pdf

[19] Gunn, S.R., Support vector machines for classification and regression, (1998). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.10.7171&rep=rep1&type=pdf

[20] Safavian, S. R., Landgrebe, D., (1991). A survey of decision tree classifier methodology. In: IEEE Transactions on Systems, Man, and Cybernetics, vol. 21, no. 3, pp. 660-674, May-June 1991, doi: 10.1109/21.97458.

[21] Géron, A., "Hands-On machine learning with Scikit-Learn and TensorFlow concepts," O'Reilly Publishing, 2017.

[22] Luca Parisi. M-arcsinh: An Efficient and Reliable Function for SVM and MLP in scikit-learn. https://arxiv.org/pdf/2009.07530.pdf

[23] Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R. Wu, A.Y., (1998). An optimal algorithm for approximate nearest neighbor searching in fixed dimension. Journal of the ACM, 45, 6, 891-923. https://graphics.stanford.edu/courses/cs468-06-fall/Papers/03%20AMNSW%20-%20JACM.pdf

[24] Babu, U., Chintha, Aneel, Venkateswarlu, Y. (2014). Handwritten Digit Recognition Using Structural, Statistical Features and K-nearest Neighbor Classifier. International Journal of Information Engineering and Electronic Business. 6. 62-68. 10.5815/ijieeb.2014.01.07.

[25] Toghi, Behrad, Grover, Divas. (2018). MNIST Dataset Classification Utilizing k-NN Classifier with Modified Sliding Window Metric. https://www.researchgate.net/publication/327742790_MNIST_Dataset_Classification_Utilizing_k-NN_Classifier_with_Modified_Sliding_Window_Metric

[26] Bernard, S., Heutte, L., Adam, S., (2007). Using Random Forests for Handwritten Digit Recognition. Proceedings of the International Conference on Document Analysis and Recognition, ICDAR. 2. 1043-1047. 10.1109/ICDAR.2007.4377074.

[27] Gilles Louppe. UNDERSTANDING RANDOM FORESTS: from theory to practice https://arxiv.org/pdf/1407.7502.pdf, arXiv:1407.7502v3 [stat.ML] 3 Jun 2015

[28] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning (Adaptive Competition and Machine Learning). The MIT Press, p. 779, 2016.

[29] Dalto, M., (2015). Deep Neural Networks for Time Series Prediction with Application in Ultra-Short-Term Wind Forecasting, IEEE, 1657–1663. https://www.researchgate.net/publication/312928742_Deep_neural_networks_for_time_series_prediction_with_applications_in_ultra-short-term_wind_forecasting.

[30] Ferreira, A., Giraldi, G., (2017). Convolutional Neural Network Approaches to Granite Tiles Classification, Expert Systems with Applications, 84, 1–11, https://doi.org/10.1016/j.eswa.2017.04.053

[31] Bengio, Y., (2009). Learning Deep Architectures for AI. http://www.cs.cmu.edu/~10701/slides/deep_learning_paper.pdf.

[32] Nwankpa, C.E., Ijomah, W., Gachagan, A., (2018). Marshall, S., Activation Functions: Comparison of Trends in Practice and Research for Deep Learning, https://arxiv.org/pdf/1811.03378.pdf

[33] Tabik, Siham & Peralta, Daniel & Herrera-Poyatos, Andrés & Herrera, Francisco. (2017). A snapshot of image Pre-Processing for convolutional neural networks: Case study of MNIST. International Journal of Computational Intelligence Systems. 10. 555. 10.2991/ijcis.2017.10.1.38.

[34] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. Proc. IEEE, 86(11):2278–2324, 1998.

[35] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. IEEE Signal Process. Mag., 29(6):82–97, 2012.

[36] P. Y. Simard, D. Steinkraus and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," Seventh International Conference on Document Analysis

and Recognition, 2003. Proceedings., Edinburgh, UK, 2003, pp. 958-963, doi: 10.1109/ICDAR.2003.1227801.

[37] Stark, F., Hazirbas, C., Triebel, R., Cremers, D. CAPTCHA Recognition with Active Deep Learning, In: GCPR Workshop on New Challenges in Neural Computation, 2015. https://vision.in.tum.de/_media/spezial/bib/stark-gcpr15.pdf

[38] Lauer, F., Suen, C. Y.,Bloch, G. (2007). A trainable feature extractor for handwritten digit recognition. Pattern Recognition,40(6), 1816-1824. doi:10.1016/j.patcog.2006.10.011

[39] Scikit-learn. User guide. https://scikit-learn.org/stable/user_guide.html

[40] Serhiienko, A., Chychkarov, Y., Syrmamiikh, I., (2020). Information Technology of Hot-metal Ladle Car Handwritten Numbers Recognition from Photo Image. In: Computer Modeling and Intelligent Systems (CMIS-2020): Proceedings of the Third International Workshop (Zaporizhzhia, April 27 – May 1, 2020 y.). – Zaporizhzhia, 2020, Vol. I–2608, P. 900–912. http://ceur-ws.org/Vol-2608/paper67.pdf