VRIJE UNIVERSITEIT BRUSSEL

DEEP LEARNING

Designing a CNN for Optical Character Recognition

Authors:
Margo CABUY
0505785
Zeeshan Hussain KHAND
0590017
Daria SHUMKOVA
0590150

January 11, 2022

Table of contents

1 Introduction

1.1 Deep learning: a quick overview

In the last century, a technological revolution introduced us to a variety of new computer-based applications, including artificial intelligence approaches. Therefore, a lot of research has already been done to improve the accuracy and the predictive capability of the artificial intelligence models with respect to different applications. Deep Learning is the subset of machine learning and artificial intelligence that makes use of hierarchical, complex, non-linear models rather than simple linear ones [1]

Deep Learning neural networks consist of layers of neurons, which get activated either by input data or by processed weights from the previous layer. The central concept of each neural network is the cost function, which estimates the difference between predicted label and ground truth label. Throughout forward propagation, the cost function is calculated, whereas during back propagation it is minimized. The advantage of deep learning, compared to usual machine learning algorithms, is that the algorithm builds the feature set and does not require man-made features. This fastens the learning and ensures more accurate predictions as no human interference is required [2].

Convolutional Neural Networks (CNN) have become one of the most popular methods in deep learning for images due to their ability to learn spatial hierarchies of features [3]. CNNs are composed of convolution layers, pooling layers and fully connected layers. Fully connected layers transfer all the inputs from the previous layer onto the next layer. Convolution layers make use of different filters to detect image features while pooling layers assist in dimension reduction to remove irrelevant details [4]. Several pre-build CNN architectures exist, including the famous AlexNet network, VGG, and High-Resolution network [5]. Nowadays CNN has a variety of applications. In radiology CNNs are widely used to detect, segment and classify cancer and metastases [6, 7]. However, CNNs can also be applied towards non-scientific approaches to directly improve everyday life, e.g. vehicle and pedestrian detection, face-recognition pay systems etc. [8, 9].

1.2 Optical Character Recognition

One of the most popular challenges in Computer Vision since the first AI algorithms were designed, is optical character recognition (OCR). It was found to be useful in an enormous number of applications, including smartphone systems and PDF readers. Prior to using deep learning for this task, a feature extractor was needed before training. However, the pattern recognition accuracy was largely dependent on the ability of the designer to identify appropriate features [10]. Once image classification evolved into more elegant techniques, deep learning - and thus a better recognition accuracy - could be implemented to get better OCR results [11].

Now, many publications are available that use different CNN architectures for OCR. A commercialised architecture used for OCR is LeNet 5, coined by Yann LeCun[10].

1.3 Aim of the study

This project aims to build an OCR system by implementing a convolutional neural network architecture to distinguish between handwritten letters from the EMNIST Letters dataset. The

CNN will additionally be evaluated on handwritten letters that are captured by camera. The model will be built in the Pytorch framework as this framework's application in OCR recognition tasks is poorly researched.

2 Materials and methods

2.1 Programming environment

The interactive python programming environment Google Colab was chosen to write and execute the code. It has several benefits, including compatibility with many libraries used during the project and GPUs lent to fasten the code. Additionally, PyTorch open machine learning framework was used to implement the CNN model as it was a project prerequisite.

2.2 EMNIST dataset

The data used to train and test the neural network for OCR is the widely known EMNIST data. This dataset extends the previously established MNIST data with handwritten character digits. These are derived from the NIST Special Database 19 and converted to 28x28 pixel images to match the MNIST dataset [12]. The EMNIST database has several subdivisions, in this project the EMNIST Letters subset was used. It consists of a set of upper- and lowercase letters, categorized into 26 classes. Additionally, this set is balanced, meaning that each of these 26 classes contains an equal number of samples. In total, the EMNIST Letters dataset contains 145.600 images [13].

As the EMNIST dataset contains a vast amount of data, it was decided to load the data directly from its online source, rather than downloading it to a local PC or personal cloud. A function in the code opens the NIST website – where the different datasets are stored – and loads the EMNIST Letters zipped MATLAB file in. The file is opened, read, and temporarily stored. After the data is locally stored in different training and testing sets, the MATLAB file is removed from memory. The code that performs these operations was found on GITHUB [14].

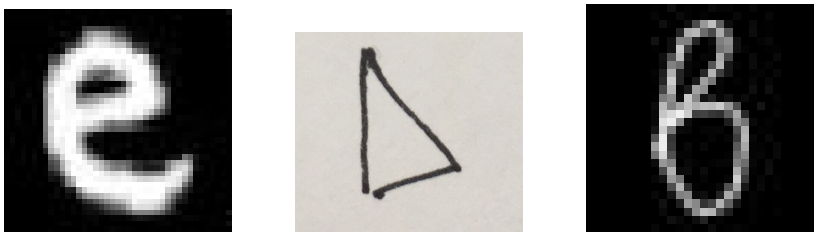
2.3 CNN architecture

The established CNN architecture is based on three existing models that each reported high performance accuracy on EMNIST data [15] [16] [17]. The first model used TensorFlow and therefore needed to be adapted to run inside PyTorch. As the first and second model showed the highest validation accuracy when tested, they were combined. Further, different combinations of convolution, pooling and dropout layers were tested and the best performing CNN was finally implemented in the project and evaluated on Test sets.

This implemented CNN architecture thus started from a convolutional layer with 16 convolution filters of size 5x5. It was followed by a maximal pooling unit with kernel size 2. Another convolution layer - with 32 filters with size 5x5 - was followed by dropout and another maximal pooling unit. Three fully connected layers were put in the end of model and the output was equal to number of classes. Further, the ReLU activation function was used for each layer.

2.4 Data sets

The CNN model was trained on 25.000 images from the EMNIST Letters dataset. Its performance was evaluated on a validation dataset of 6.000 images from EMNIST Letters - to adjust hyperparameters and to choose the best architecture. To further evaluate the performance, the CNN model was tested on 3 Test sets coming from different sources: Test 1 has 6.000 images from EMNIST Letters, Test 2 has 52 Handwritten letters, captured by camera, and Test 3 has 52 letters, drawn using the Tkinter user interface. An example of each of the different test sets is given in Figure 1 as an illustration to the reader. Test 2 consists of 26 lowercase letters and 26 uppercase letters of the English alphabet. It consists of 26 lowercase and 26 uppercase letters of the English alphabet. The handwriting for dataset 2 and 3 was done by the same person. Lastly, EMNIST subsets were randomly shuffled.



(a) Test 1 **(b)** Test 2 **(c)** Test 3

Figure 1: Data used in the different testsets before pre-processing.

2.5 Data preprocessing

To use the EMNIST data in the input layer of the CNN, it needed to be preprocessed further. The EMNIST data was already split up into train and test datasets, so - after loading the data - the images and labels just needed to be separated into different arrays [18]. Next, the arrays containing the images were normalized with 255, to have pixel values between 0 and 1 [19]. Subsequently, the 1D image arrays needed to be reshaped into 2D ones, where the images have a 28x28 pixel size. In order to get a correct image orientation, the array needed to be reshaped using Fortran ordering. Lastly, the numpy arrays needed to be converted into PyTorch tensors.

As the Test 1 dataset originates from EMNIST Letters data, it did not require preprocessing. For the Test 1 and Test 3 sets, however, different preprocessing techniques needed to be applied. The labels for this data were acquired by consistently naming the images in a certain way, thus extracting the ground truth from the image name. These ground truths were appended to an array - for all images in a folder path - and converted into Pytorch tensors. As for the images themselves, they needed to be converted to grayscale and reshaped into 28x28 pixel images. Each pixel value was normalized to fit the range between 0 and 1, as the network was trained on normalized data. An additional step, needed for the Test 2 dataset, was to set a threshold value under which the pixel value became zero. This was needed as the EMNIST letters have a black background. Further, both datasets needed to undergo some transformations - for which the images themselves stayed inert - to have the same dimensions as the EMNIST data. Lastly, the numpy arrays were converted into PyTorch tensors in order to feed them in the input layer of the constructed CNN.

2.6 Evaluation

The model performing multiclass classification, was evaluated using different methods. The accuracy of the trained network on validation data was computed and viewed throughout 20 epochs. Subsequently, the accuracy was calculated for Test 1, Test 2 and Test 3 sets. The accuracy metric is a good first assessment measure as it returns the ratio of correctly classified images to the total number of images in the testset. However, it would behove to use other performance measures during the evaluation such as precision, recall and F1 scores. Precision of multi-class classifiers show the weighted averaged precision of each class. The precision of each class is the ratio of the amount of correctly predicted images in that class to the number of images predicted to be in that specific class. Thus, it displays the amount of images that are predicted to be a certain letter and actually contain an image of that letter. Recall of a multi-class classifier is a weighted averaged recall of each class. The recall of each class is the ratio of the correctly predicted images in a certain class to the actual amount of images in that class. Thus, it displays the number of correctly predicted letters out of the number of actual images containing the letters. The F1 score is a function of both aforementioned precision and recall and shows the relative performance of the classifier. The multi-class weighted-average F1 score is the F1 score of each class weighted by the number of samples in that class. The per-class F1 score is computed using the harmonic mean between precision and recall; and will thus always be a value between precision and recall [20]. Lastly, the confusion matrix was constructed to estimate the performance of the model on different classes of letters. This shows whether there was a bias towards certain letters during misclassification.

3 Results

3.1 Architectures comparison

The three selected CNN architectures, originating from different articles, were evaluated on subsets of the EMNIST Letters Train and Validation sets. These models achieved 0.91 and 0.87, 0.93 train accuracy and 0.87, 0.78 and 0.73 validation accuracy, respectively, after 10 epochs (Table 1). The merged first two models - and different combinations of layers - were tested on 10 epochs (Table 1). The architecture consisting of the combination 2 conv, 2 pool, 1 drop was chosen to implement further, as it performed the best on Validation dataset.

Table 1: Train and Validation accuracy, achieved by different models through 10 epochs. conv(1, 16, 5) - torch.nn.conv2d(1, 16, 5), pool - torch.nn.MaxPool2d, drop - torch.nn.Dropout(0.2).

Model/Combination	Train accuracy	Validation accuracy
Model A [15]	0.91	0.87
Model B [16]	0.93	0.87
Model C [17]	0.78	0.73
1 conv(1, 16, 5) + 1 pool	0.92	0.88
1 conv(1, 8, 5) + 1 pool	0.92	0.86
1 conv(1, 8, 7) + 1 pool	0.91	0.87
2 conv + 2 pool	0.92	0.88
2 conv + 2 pool + 2 drop	0.92	0.89
2 conv + 2 pool + 1 drop	0.92	0.90

1 conv + 1 pool + 1 drop	0.92	0.89
1 conv + 1 pool + 2 drop	0.93	0.88

3.2 Evaluation of trained network on validation set

The prediction accuracy of the model on the validation dataset for each subsequent epoch (20 epochs in total) is displayed in Figure 1. It can be seen that the accuracy incrementally increased to 90% for 20 epochs. Adding more epochs would eventually reduce the validation accuracy again, as the trained network started to overfit on the training data. Therefore, it was decided to use 20 epochs for this project.

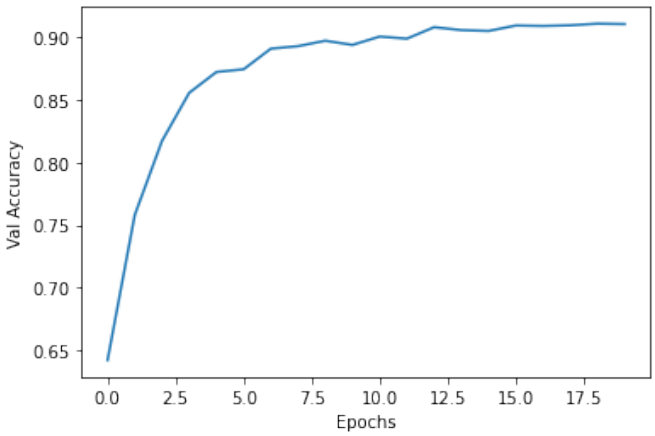


Figure 2: Validation accuracy for a training accuracy with 20 epochs.

3.3 Evaluation of trained network on Test sets

After training and validating the neural network, the model was evaluated on the different Test datasets. Test 1 showed the highest accuracy, F1 and Recall scores of 0.92, 0.91 and 0.92 respectively. The highest precision score (0.92) was demonstrated by Test 1 and Test 3 sets. Accuracy and Recall Scores of Test 2 and Test 3 were comparable between 0.87 and 0.88. The lowest F1 and Precision scores were reported by Test 2 set and were equal to 0.86 and 0.89. These are all list in Table 2.

Test set number & sample size	Accuracy	F1 score	Precision score	Recall score
1 (6000) EMNIST letters	0.92	0.91	0.92	0.92
2 (52) Camera captured	0.87	0.86	0.89	0.87
3 (52) Tk UI	0.88	0.88	0.92	0.88

Table 2: Accuracy, F1, precision, recall scores evaluation of model performance on Test 1, Test 2, Test 3 sets.

3.4 Confusion matrix

The Confusion matrix was built for each of the Test sets. For Test 1, it can clearly be seen that the model demonstrated the worst distinguishing ability of letter I from L and vice versa, resulting in 106 mislabeled cases. Moreover, it gave 55 wrong labels when it tried to classify letters G and Q (see Figure 2).

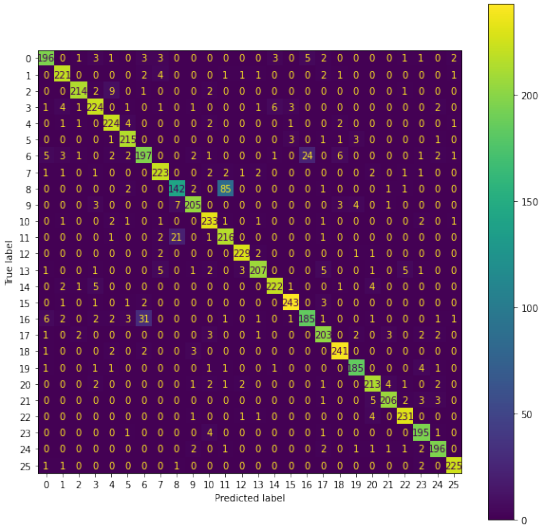
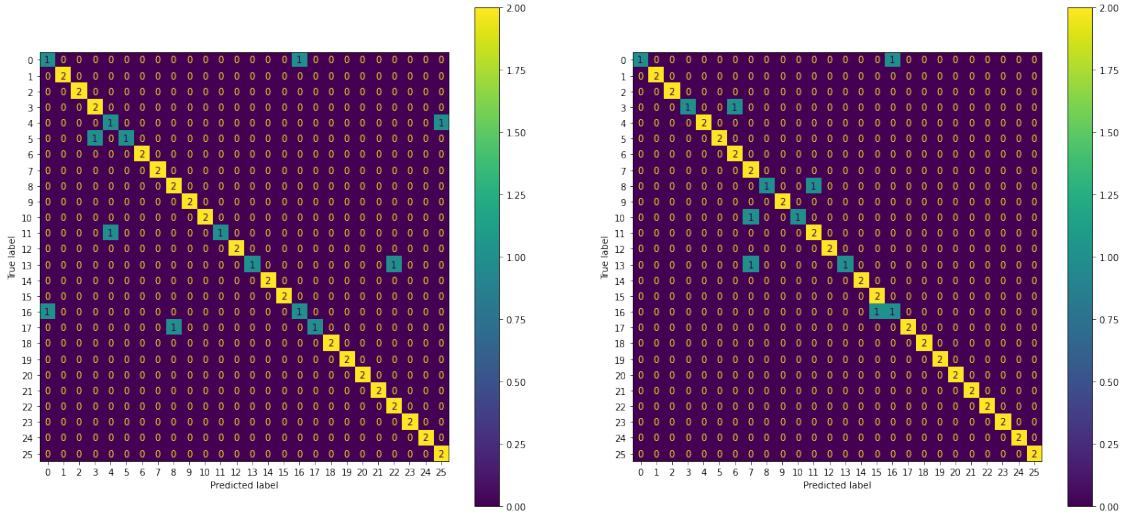


Figure 3: Confusion matrix for Test 1 (from unseen EMNIST Letters set).

Test 2 and Test 3 had only 2 samples for each class, therefore, mislabeling can not be considered informative. Confusion matrices were built to compare the performance of the model between the Test 2 and 3 sets, but not for each class within the same subset. Both Test 2 and Test 3 demonstrated mislabeling of A and Q letters, having 3 wrong predictions out of 8. Overall, Test 2 had 7 mislabeled examples and Test 3 had 6 mislabeled images (see Figure 3). The total amount of images in each Test set was 52.



(a) Test 1 **(b)** Test 2

Figure 4: Confusion matrix for Test 2 (handwritten images captured by camera) and Test 3 (handwritten with Tkinter UI images).

4 Discussion

The constructed CNN model showed 92% accuracy on Test 1 and 90% on Validation dataset. These are subsets of the EMNIST Letters dataset. The accuracy dropped to 87% when the model was evaluated on the Test 2 set, made up from handwritten letters captured by the camera of a smartphone and subsequently preprocessed to be similar with EMNIST images. Test 3 accuracy was comparable to Test 2 and was equal to 88%, in disregard of the fact that its letters were drawn on a computer using the Tkinter user interface. This is possibly due to the robustness of our preprocessing steps.

The accuracy is a ratio of correct labeled cases over the total number of observations[21]. Often it is not enough to make informed conclusions about model performance, especially when data is not symmetrically distributed between the classes. Although used Test sets were balanced, it was decided to assess other evaluation metrics of multiclass classification. Therefore, F1 score, which is a balanced mean between precision and recall scores, was calculated. It describes both the precision and robustness of the data. Precision is a ratio of true positives over all positive cases, whereas the recall score is a ratio of true positives over class sample size. Test 3 set demonstrated a higher F1 score than Test 2, but the small sample size does not allow to make a proper comparison. More importantly, Test 1 was reported to have high recall and precision and, therefore, a high F1 score. This gives a basis to conclude that the CNN showed a generalized and balanced performance on the EMNIST dataset, not missing any significant number of instances.

Next, the confusion matrix was evaluated on the different Test sets. For Test 1, the confusion matrix highlighted the difficulties of the model in distinguishing between the letters L, I, G and Q. This can be explained by the spelling similarity. With handwritten datasets, the model experienced difficulties with lowercase A and uppercase Q letters due to individual characteristics of the person's handwriting. Overall, the model is concluded to be able to successfully distinguish between letters from the English alphabet.

Previous studies showed comparable accuracy to the results of this project. The review

of Baldominos et. al. described outputs of five different research groups on EMNIST dataset: Botalb et. al. showed a maximal accuracy of 99.2% [22] [23]. Peng and Yin's study implemented a Markov random field-based CNN and demonstrated a range of accuracy between 87.77% and 99.75% [24]. By changing the number of convolutional and dense layers in the CNN architecture, different authors reported an accuracy of 87.1% over EMNIST By_Class and 99.62% over EMNIST Digits [25] [26].

Further, a study conducted by Meany et. al. compared the performance of different CNN models on the EMNIST dataset - including ResNet [19]. They showed an accuracy for test dataset, consisting of EMNIST Digits, to vary from 84.8% to 88.5%, while for EMNIST Letters it was ranging from 47.5% to 76.9% [19]. Finally, Sakshi et al research compared performance of Knn, ANN and CNN on optical character recognition task and achieved an accuracy of 65.9%, 90% and 92.4% respectively [27].

The project conducted in this report encountered several limitations, including a small validation dataset. Another problem of the research is that the validation dataset comes from another source than the testing dataset. Therefore, further research might change the study design that test and validation data will be of the same size and from the same source. Moreover, as an extension of the project it is possible to develop an app to capture the handwritten letters with a camera and automatically identify them with a deep learning algorithm.

5 Conclusion

The present study concluded that fully automatic recognition of handwritten letters, captured by camera can be implemented using the Pytorch CNN framework and showed an accuracy of 87%, which is in line with the hypothesis. The model trained was evaluated on three type of scenarios. It showed a testing accuracy of 92% on the EMNIST Letters set, 87% on handwritten letters captured by the camera, and 88% on letters drawn on tablet using TinkerUI. This confirms the hypothesis that our model can be implemented in real-world scenarios for the recognition of handwritten letters. Since the model was trained on a subset of the original dataset, we believe that its accuracy and performance will improve if it is trained using the same architecture on the whole EMNIST dataset.

References

[1] Alexander Selvikvåg Lundervold and Arvid Lundervold. *An overview of deep learning in medical imaging focusing on MRI*. May 2019. DOI: 10.1016/j.zemedi.2018.11.002.

[2] Jürgen Schmidhuber. *Deep Learning in neural networks: An overview*. Jan. 2015. DOI: 10.1016/j.neunet.2014.09.003.

[3] Rikiya Yamashita et al. *Convolutional neural networks: an overview and application in radiology*. Aug. 2018. DOI: 10.1007/s13244-018-0639-9.

[4] Víctor Suárez-Paniagua and Isabel Segura-Bedmar. "Evaluation of pooling operations in convolutional architectures for drug-drug interaction extraction". In: *BMC Bioinformatics* 19 (June 2018). ISSN: 14712105. DOI: 10.1186/s12859-018-2195-1.

[5] Laith Alzubaidi et al. "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions". In: *Journal of Big Data* 8.1 (Dec. 2021). ISSN: 21961115. DOI: 10.1186/s40537-021-00444-8.

[6] Shubham Shah et al. "Non-invasive multi-channel deep learning convolutional neural networks for localization and classification of common hepatic lesions". In: *Polish Journal of Radiology* 86.1 (2021), pp. 440–448. ISSN: 18990967. DOI: 10.5114/pjr.2021.108257.

[7] Yan Liu et al. "A deep convolutional neural network-based automatic delineation strategy for multiple brain metastases stereotactic radiosurgery". In: *PLoS ONE* 12.10 (Oct. 2017). ISSN: 19326203. DOI: 10.1371/journal.pone.0185844.

[8] Rujin Ma et al. "Deep learning based vehicle detection and classification methodology using strain sensors under bridge deck". In: *Sensors (Switzerland)* 20.18 (Sept. 2020), pp. 1–26. ISSN: 14248220. DOI: 10.3390/s20185051.

[9] Yu Xin Yang et al. "Face recognition using the SR-CNN model". In: *Sensors (Switzerland)* 18.12 (Dec. 2018). ISSN: 14248220. DOI: 10.3390/s18124237.

[10] Yann Lecun et al. "Gradient-Based Learning Applied to Document Recognition". In: *IEEE* (1998).

[11] J. Mantas and G. Nagy. *Historical Review of OCR research and Development*. Tech. rep. 2. 1987, pp. 207–212.

[12] Cohen G. et al. "EMNIST: extending MNIST to handwritten letters". In: *IEEE* (2017).

[13] NIST. *The EMNIST Dataset details given by NIST*. Mar. 2019. URL: https://www.nist.gov/itl/products-and-services/emnist-dataset.

[14] G. Cohen et al. "EMNIST: an extension of MNIST to handwritten letters." In: (2017). URL: http://arxiv.org/abs/1702.05373.

[15] A. Tripathi. *EMNIST Letter Dataset 97.9%:acc & val_acc: 91.78%*. 2020. URL: https://www.kaggle.com/achintyatripathi/emnist-letter-dataset-97-9-acc-val-acc-91-78#As-the-validation-score-score-went-down-we-won't-be-changing-the-test_split-size.

[16] Vmoon. *PyTorch sets up CNN for handwritten English letter recognition*. 2020. URL: https://www.fatalerrors.org/a/pytorch-0-to-1-for-handwritten-english-letter-recognition.html.

[17] G. Koehler, A. Amantini, and P. Markovics. *MNIST Handwritten Digit Recognition in PyTorch*. 2020. URL: https://nextjournal.com/gkoehler/pytorch-mnist.

[18] Tlindbloom and Dim Dev. *Loading EMNIST Letters dataset into Python*. 2020. URL: https://stackoverflow.com/questions/51125969/loading-emnist-letters-dataset/53547262#53547262.

[19] Meany Connor and Arola Matias. "Optical Character Recognition via Deep Learning". In: *CS230: Deep Learning* (2018).

[20] B. Shmueli. *Multi-Class Metrics Made Simple: Precision, Recall and F1-score*. 2019. URL: https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1.

[21] A. Mishra. *Metrics to Evaluate your Machine Learning Algorithm*. 2018. URL: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234.

[22] Alejandro Baldominos, Yago Saez, and Pedro Isasi. *A survey of handwritten character recognition with MNIST and EMNIST*. Aug. 2019. DOI: 10.3390/app9153169.

[23] A. Botlab et al. *Constrasting Convolutional Neural Network (CNN) with Multi-Layer Perceptron (MLP) for Big Data Analysis*. ICIAS, 2018. ISBN: 9781538672693.

[24] Y. Peng and H. Yin. "Markov Random Field Based Convolutional Neural Networks for Image Classification. In IDEAL 2017". In: *Intelligent Data Engineering and Automated Learning; Lecture Notes in Computer, Science*. Vol. 10585. Guilin, China, 2017, pp. 387–396.

[25] S. Singh, A. Paul, and M. Arun. "Parallelization of digit recognition system using Deep Convolutional Neural Network on CUDA." In: *Third International Conference on Sensing, Signal Processing and Security, Chennai*. Chennai, India, May 2017, pp. 379–383.

[26] S.S. Mor et al. "Handwritten text recognition: With deep learning and Android." In: *Int. J. Eng. Adv. Technol* 8 (2019), pp. 172–178.

[27] S. Sakshi et al. *Optical Character Recognition using Convolutional Neural Network*. IEEE, 2019. ISBN: 9789380544342.