Applied Computer Science, vol. 19, no. 4, pp. 88–102
doi: 10.35784/acs-2023-37

Submitted: 2023-06-23 | Revised: 2023-09-21 | Accepted: 2023-10-01

Keywords: convolutional neural network, model efficiency, handwritten text recognition

Edyta ŁUKASIK [0000-0003-3644-9769]*, *Wiktor FLIS* **

EFFICIENCY COMPARISON OF NETWORKS IN HANDWRITTEN LATIN CHARACTERS RECOGNITION WITH DIACRITICS

Abstract

The aim of the article is to analyze and compare the performance and accuracy of architectures with a different number of parameters on the example of a set of handwritten Latin characters from the Polish Handwritten Characters Database (PHCD). It is a database of handwriting scans containing letters of the Latin alphabet as well as diacritics characteristic of the Polish language. Each class in the PHCD dataset contains 6,000 scans for each character. The research was carried out on six proposed architectures and compared with the architecture from the literature. Each of the models was trained for 50 epochs, and then the accuracy of prediction was measured on a separate test set. The experiment thus constructed was repeated 20 times for each model. Accuracy, number of parameters and number of floating-point operations performed by the network were compared. The research was conducted on subsets such as uppercase letters, lowercase letters, lowercase letters with diacritics, and a subset of all available characters. The relationship between the number of parameters and the accuracy of the model was indicated. Among the examined architectures, those that significantly improved the prediction accuracy at the expense of a larger network size were selected, and a network with a similar prediction accuracy as the base one, but with twice as many model parameters was selected.

1. INTRODUCTION

Character recognition using image analysis techniques is a crucial area of study in computer vision. It is also closely related to natural language processing, as it enables the understanding of the recognized text in the post-processing stage (Islam et al., 2017). Nowadays, the digital form of documents enables the automation of business processes and improves work efficiency, allowing employees of enterprises to effectively focus on their key tasks. Many methods have been developed to efficiently carry out numerous tasks automatically, such as searching for phrases according to given criteria, classifying text according to the subject of its content or improving grammar and text style. Automatic text

* Lublin University of Technology, Faculty of Electrical Engineering and Computer Science, Department of Computer Science, Poland, e.lukasik@pollub.pl
** Lublin University of Technology, Faculty of Electrical Engineering and Computer Science, Poland, wiktor.flis@pollub.edu.pl

translation systems and speech synthesis systems deserve a special place here. Recently, the Transformer architecture (Vaswani et al., 2017) has made it possible to automate previously difficult or impossible tasks using natural language processing methods such as text summarization or generating new text on a given topic. Currently, the recognition of printed text written in the Latin alphabet is easy to perform using widely available programs on personal computers, and the accuracy of their prediction exceeds 98% (Łukasik & Zientarski, 2018). Even so, documents written in natural language do not always exist in computer-accessible form. Most exist only in the traditional paper version, especially in the form of handwritten notes such as historical sources, personal notes written on paper or with a stylus on tablets. This can also include tests written by students, such as colloquia or exams. The use of image recognition techniques allows to check tests faster, allowing to do it using a computer, and using them in research on documents that are historical sources allows not only for more efficient storage and sharing of them, but also for a more in-depth analysis or reading and discovering features invisible to the naked eye.

Despite all the methods and possibilities mentioned above, recognizing handwritten text is still a difficult task due to the variability and heterogeneity of handwritten text. This requires specially adapted models and algorithms that are able to learn and recognize different writing styles. Recognition of individual handwritten characters is even more demanding, because compared to printed writing, the quality of the handwriting in individual cases is not high, and at the same time it has a much greater variety and lower contrast. Often, individual lines of handwritten text do not lie in one straight line, as in printed writing. An additional problem in variants of the Latin alphabet is the presence of diacritics in languages such as Dutch, French, German and Czech - and in this case of Polish, the characters Ą,Ę,Ć,Ł,Ó,Ś,Ń,Ż,Ź. The software in scanning devices usually does not support diacritical marks, omitting them (Gu et al., 2018). This causes that the scanned text is often incomprehensible. The use of advanced image recognition techniques based on deep learning (DL) methods, such as convolutional neural networks, is successful in this field (Idziak et al., 2021; Sharma et al., 2020; Gajoui et al., 2015). The same problem is present in other alphabets with diacritical marks such as Arabic, and the usage of DL methods is documented in (Lutf et al., 2014) where authors used network on Handwritten Arabic Characters Database (HACB) consisting of 52 classes of characters. In Arabic, a different variant of diacritical mark is used depending on whether a diacritical mark is used at the beginning of the sentence, at the end or in the middle. Because of this, the problem of handwriting recognition is still an active and challenging area of research. In the publication by E. Lukasik et al. from 2021, authors trained several such models on a set of scans of handwriting from the Polish Handwritten Characters Database (PHCD) (Tokovarov et al., 2020) and then presented the results of the most promising of them on the test part of subsets of lowercase and uppercase letters with characters diacritics as well as a subset of all available characters.

The aim of the article is to analyze and compare the performance and accuracy of architectures with a different number of parameters on the example of a set of handwritten Latin characters from the Polish Handwritten Characters Database (PHCD). Due to the fact that among all the subsets of the examined characters they achieved very high accuracy (over 95% in all but the set of all characters from the PHCD database, where accuracy was above 85%), the focus was on improving the network performance in terms of the size of the model measured by the number of parameters, disk size, and the number of floating-point operations performed by the network during prediction. Both of these parameters are crucial as presented in Hadidi et al. (2019), whether when it comes to the effective use of computing power in the cloud (which is a common solution when implementing neural networks for business applications) or during edge computing implementations. In the latter case the model is uploaded to microcontrollers or small computers based on processors ARM architecture represented by Raspberry Pi and Nvidia Jetson. Computers of this type have low computing power and their components can be optimized specifically for prediction on ready-made models, which allows the creation of various intelligent devices in factories, offices or homes that do not need communication with a central server or internet coverage. This makes predictions much faster, which is extremely important in many applications, such as automatic image recognition or motion detection. It also protects the privacy of their users, due to the fact that there is no need to send data to the outside world, for example from home cameras.

2. RESEARCH EXPERIMENT

2.1. Created architectures

Six convolutional network architectures have been tested. Each of them is based on the basic architecture described in publication by Lukasik et al. (2021). It consists of 4 convolutional layers extracting features from the input images and is completed with a 2-layer multilayer perceptron which finds relationships between them during learning. Its structure is shown in Table 1.

Tab. 1: Layers of baseline architecture

Layer type	Layer output shape
Conv2D	None, 30, 30, 32
Conv2D	None, 28, 28, 32
MaxPooling2D	None, 14, 14, 32
Dropout	None, 14, 14, 32
Conv2D	None, 12, 12, 64
Conv2D	None, 10, 10, 64
MaxPooling2D	None, 5, 5, 64
Dropout	None, 5, 5, 64
Flatten	None, 1600
Dense	None, 256
Dropout	None, 256
Dense	None, 89

Architecture A, presented in Table 2, is an architecture consisting of 2 convolutional layers containing 32 and 64 filters, each of them is followed by the ReLU activation function and the MaxPool2D layer.

Tab. 2: Layers of architecture A

Layer type	Layer output shape
Conv2D	None, 30, 30, 32
MaxPooling2D	None, 15, 15, 32
Conv2D	None, 13, 13, 64
MaxPooling2D	None, 6, 6, 64
Flatten	None, 2304
Dense	None, 5376
Dense	None, 256
Dense	None, 128
Dense	None, 89

The output is then transferred to 4 fully connected layers - having 5376, 256, 128 and 89 neurons in turn. Relative to the base network, it extracts less detailed features, due to having less convolutional layers while being able to model more complicated interactions between them by having more neurons (He et al., 2016).

Architecture B is based on architecture A with the difference that it has 3 fully connected layers with a total of fewer neurons 512, 256 and 89, respectively. Thanks to this, it is much smaller. Its layers are shown in Table 3.

Tab. 3: Layers of architecture B

Layer type	Layer output shape
Conv2D	None, 32, 32, 32
MaxPooling2D	None, 16, 16, 32
Conv2D	None, 14, 14, 64
MaxPooling2D	None, 7, 7, 64
Flatten	None, 3136
Dense	None, 512
Dense	None, 256
Dense	None, 89

In Architecture C the number of convolutional layers is limited to 2 containing 32 and 64 filters followed by ReLU activation and batch normalization, and in the classification layers it has 128 and 64 neurons respectively. The last layer contains 89 neurons as in other architectures. It is presented in Table 4.

Tab. 4: Layers of architecture C

Layer type	Layer output shape
Conv2D	None, 30, 30, 32
BatchNormalization	None, 30, 30, 32
Conv2D	None, 28, 28, 64
BatchNormalization	None, 28, 28, 64
Flatten	None, 50176
Dense	None, 128
Dense	None, 64
Dense	None, 89

Architecture D, which layers are presented in Table 5, includes 4 convolutional layers (two containing 16 and two containing 32 filters) and two fully connected layers consisting of 512 and 89 neurons. In addition, convolutional layers are arranged in pairs where the first layer has batch normalization and the second has MaxPool2D and Dropout operations.

Tab. 5: Layers of architecture D

Layer type	Layer output shape
Conv2D	None, 30, 30, 16
BatchNormalization	None, 30, 30, 16
Conv2D	None, 28, 28, 16
MaxPooling2D	None, 14, 14, 16
Dropout	None, 14, 14, 16
Conv2D	None, 12, 12, 32
BatchNormalization	None, 12, 12, 32
Conv2D	None, 10, 10, 32
MaxPooling2D	None, 5, 5, 32
Dropout	None, 5, 5, 32
Flatten	None, 800
Dense	None, 512
BatchNormalization	None, 512
Dropout	None, 512
Dense	None, 89

Architecture E contains 2 convolutional layers with 16 and 32 filters - each is followed by Batch Normalization and MaxPool2D operation. The flattening layer is followed by batch normalization again before the output layer. Its layers are shown in Table 6.

Tab. 6: Layers of architecture E

Layer type	Layer output shape
Conv2D	None, 28, 28, 16
BatchNormalization	None, 28, 28, 16
MaxPooling2D	None, 14, 14, 16
Conv2D	None, 10, 10, 32
BatchNormalization	None, 10, 10, 32
MaxPooling2D	None, 5, 5, 32
Flatten	None, 800
BatchNormalization	None, 800
Dense	None, 89

Architecture F includes 3 convolutional layers (16, 32 and 64 filters) and two fully connected layers consisting of 256 and 89 neurons. In addition, after each convolutional layer, it contains a SpatialDropout layer, deactivating one of the feature maps randomly in a given layer for one iteration, which has a regularizing effect. Its structure is presented in Table 7. It is the smallest tested network, more than two times smaller than the base one.

Tab. 7: Layers of architecture F

Layer type	Layer output shape
Conv2D	None, 30, 30, 16
MaxPooling2D	None, 15, 15, 16
SpatialDropout2D	None, 15, 15, 16
Conv2D	None, 13, 13, 32
MaxPooling2D	None, 6, 6, 32
SpatialDropout2D	None, 6, 6, 32
Conv2D	None, 4, 4, 64
MaxPooling2D	None, 2, 2, 64
SpatialDropout2D	None, 2, 2, 64
Flatten	None, 256
Dense	None, 256
Dense	None, 89

2.2. Research scenario

The research was carried out on a set of characters from the PHCD database (Tokovarov et al., 2020), a Polish database of handwriting scans containing letters of the Latin alphabet and diacritics characteristic of the Polish language. Each class in the PHCD dataset contains 6 000 scans for each character. The research is carried out on the entire set of characters as well as subsets with the specification of lowercase letters, uppercase letters, lowercase letters without diacritics. In each subset, the data set was divided into training, validation and test sets in the proportions of 70%:15%:15%. For each subset, the network was trained from the beginning 20 times, for a more reliable comparison due to the influence of initialization of the weights (Bouthillier et al., 2021). The Weights & Biases service was used to track the experiment and create graphs, and the training was carried out on a Deep Learning virtual machine available on the Google Cloud platform with an Nvidia Tesla V100 graphics card. The networks were built and trained using Tensorflow library (Abadi et al., 2016) version 2.11, using its Sequential API.

Steps carried out before conducting experiments:

1. Design neural network architectures that will be trained.
2. Creation of a model repository that will contain both data and trained models alongside their results.

Steps carried out during conducting experiments for each model and character subset:

1. Download of train set of data subset.
2. Training of the model.
3. Saving of training results.
4. Download of test set of data subset.
5. Model evaluation and saving the test set results.
6. Model serialization and upload into repository for future reference.

3. RESULTS

With such a research scenario, it was possible to obtain detailed results for the base architecture and 6 selected architectures, obtained separately for each of the data subsets.

Each of the models was trained for 50 epochs, and then the accuracy of prediction was measured on a separate test set. The experiment thus constructed was repeated 20 times for each model to obtain more reliable results. The Table 7 presents the characteristics of researched networks, which do not change between the tested subsets of characters.

Tab. 7: Network characteristics obtained for individual architectures for the set of all available characters from the PHCD database.

Architecture	**Param. (mln)**	**FLOPs (mln)**	**Disk size [Mb]**
Baseline	0.484	14.394	1.817
A	13.824	17.216	51.855
B	1.765	5.699	6.633
C	6.452	21.183	23.949
D	0.447	3.976	1.672
E	0.045	1.640	0.170
F	0.098	1.298	0.369

After plotting the relationship between the number of parameters and model accuracy on Fig. 1, the network efficiency can be observed. On the PHCD dataset, after using more than 2 million parameters in network, models see diminished improvement in accuracy results, as the network B is only marginally worse than A and shows very similar results as C.

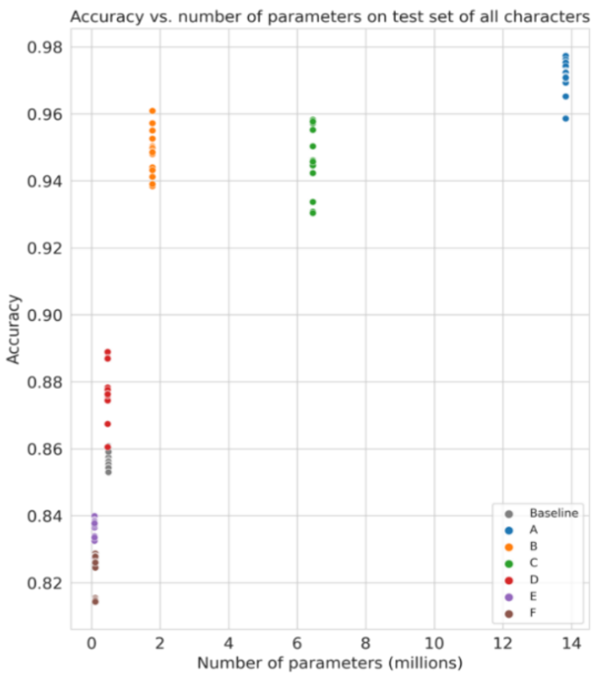

Fig. 1: Relationship between number of parameters and model accuracy.

Tab. 8: Results for the set of lowercase characters with no diacritics.

Architecture	**Mean test acc.**
Baseline	0.9829
A	0.9951
B	0.9930
C	0.9959
D	0.9779
E	0.9280
F	0.9382

Tab. 9: Results for the set of lowercase characters with diacritics.

Architecture	**Mean test acc.**
Baseline	0.9738
A	0.9957
B	0.9959
C	0.9977
D	0.9836
E	0.9432
F	0.9528

Tab. 10: Results for the set of uppercase characters with no diacritics.

Architecture	**Mean test acc.**
Baseline	0.9945
A	0.9971
B	0.9964
C	0.9976
D	0.9901
E	0.9675
F	0.9709

Tab. 11: Results for the set of all characters from PHCD database.

All characters	**Test acc.**	**Baseline % diff.**
Baseline	0.8575	0%
A	0.9719	13%
B	0.9484	11%
C	0.9484	11%
D	0.8780	2%
E	0.8361	-2%
F	0.8228	-4%

The complete accuracy results on the test set for each class (letters) are shown on Figures 2-8.

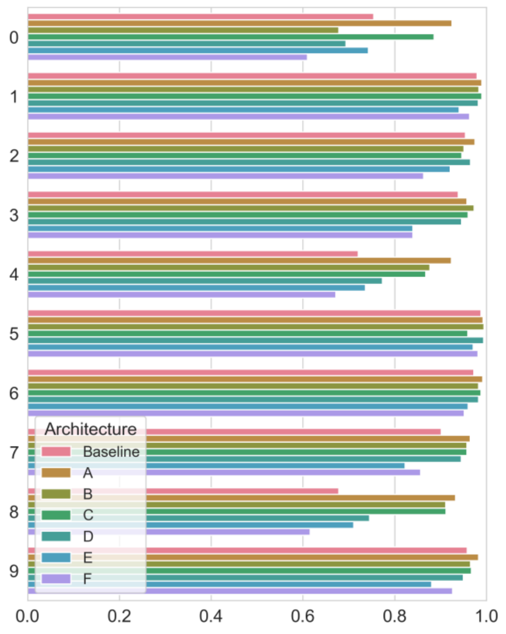

Fig. 2: Digits accuracy comparison.

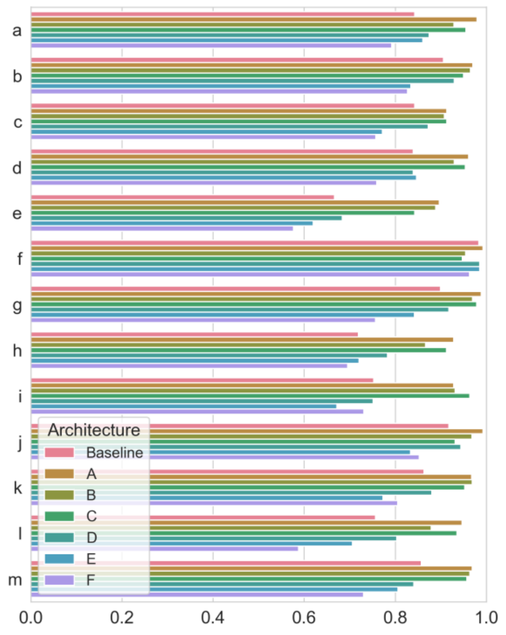

Fig. 3: Lowercase letters accuracy comparison.

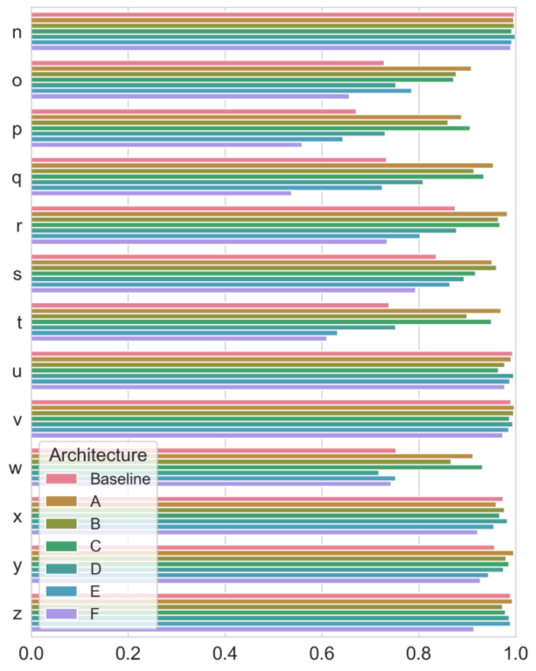

Fig. 4: Lowercase letters accuracy comparison.

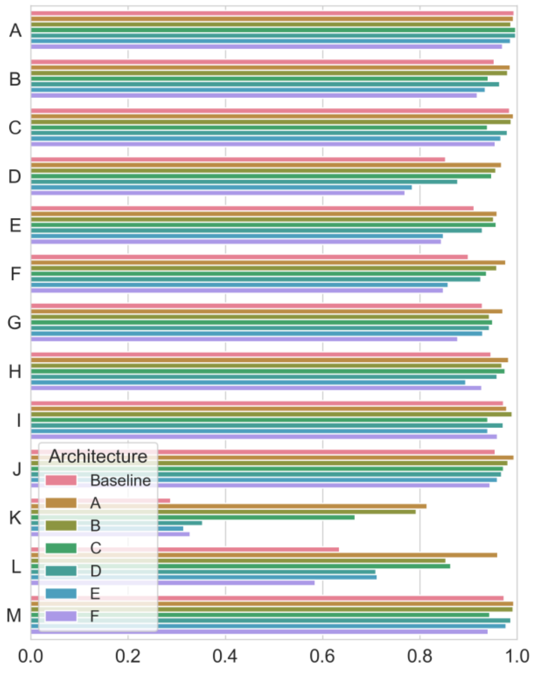

Fig. 5: Uppercase letters accuracy comparison.

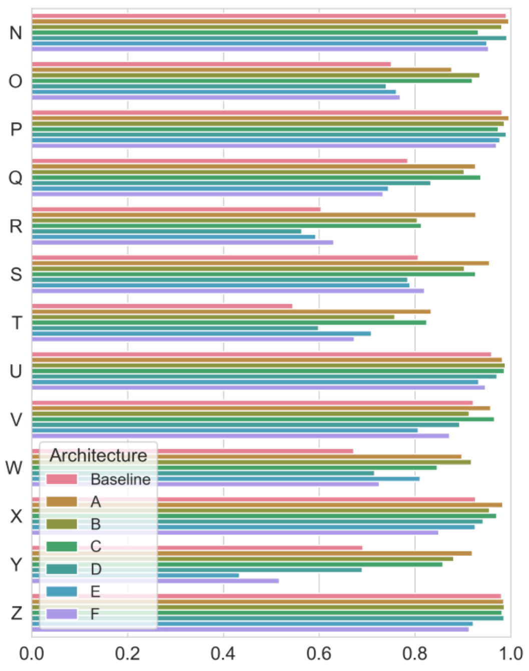

Fig. 6: Uppercase letters accuracy comparison.

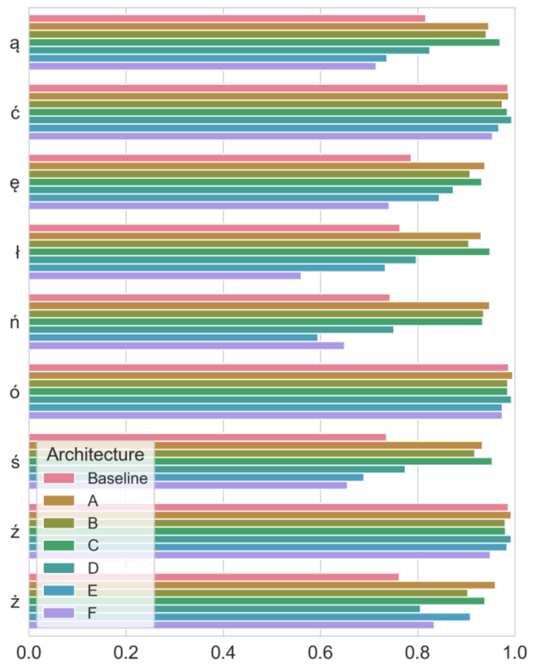

Fig. 7: Lowercase diacritics accuracy comparison.

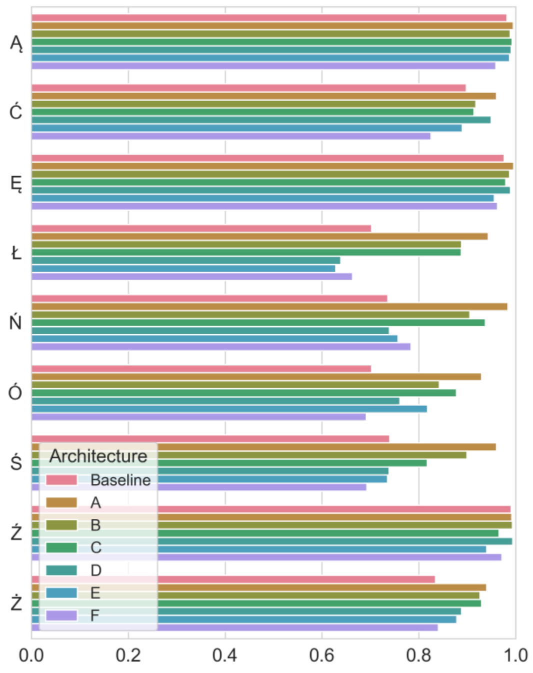

Fig. 8: Uppercase diacritics accuracy comparison.

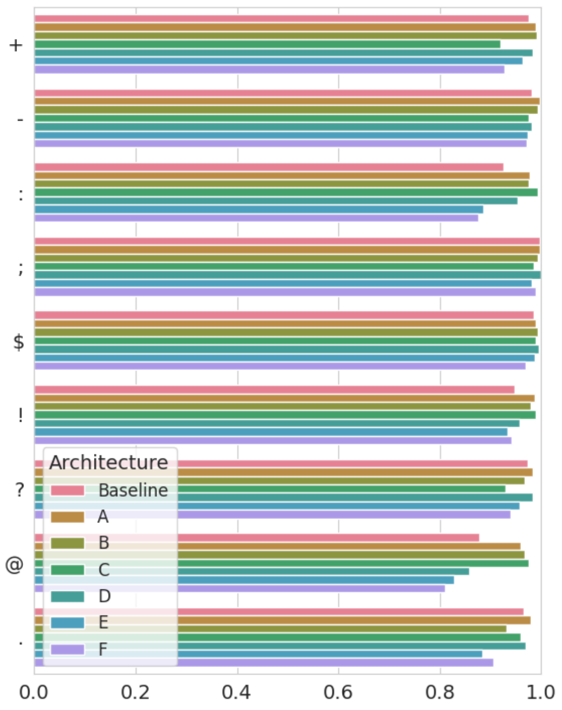

Fig. 9: Special characters accuracy comparison.

4. DISCUSSION AND FUTURE WORK

For the set of lowercase characters with no diacritics (Tab. 8) accuracy of architecture A, B and C is higher than 0.990, D achieves accuracy 0.970 but E and F achieves the accuracy below 0.940. For the baseline architecture it was 0.983. Adding diacritics to the analysed set of lowercase (Tab. 9) keeps these architectures in the same order in terms of the obtained accuracies. For the set of uppercase characters with no diacritics (Tab. 10) the first four architectures give an accuracy above 0.990. The remaining two were below 0.971. Tab. 11 shows the most important results achieved, which directly compares new architectures to the baseline on the same dataset used in publication by Lukasik et al. (2021). It presents a more challenging task for each network than only uppercase or only lowercase letters, and it also contains special characters. Three of the tested architectures (A, B, C) gave a result above 0.94 and three below 0.88. Given the set of all characters the best result was obtained for the network with architecture A. The obtained prediction accuracy is 0,9719. Two more tested architectures B and C gave the same result of 0,9484. The result for the D network was 0.878. For the remaining two, i.e. E and F, it was 0.8361 and 0.8228, respectively.

By looking at accuracy for each letter on Fig. 2-9, we can see that the special characters are not as challenging. Among the worst performing letters are lowercase letter e and uppercase letters R, K, T. In few cases, such as letters ń, ś, Q, the largest networks A, B, C had similar accuracy to the rest of the letters while smaller networks show significant problems with accurately recognizing these letters. The letter which shows the most significant accuracy improvement from Baseline architecture is the letter K. It is usually misclassified as lowercase letter x, h, number 4 or 0, or uppercase letter Q.

According to the presented results, one can distinguish between architecture A, which is the most efficient network in question, and architecture B, which achieves an average accuracy of 2 percentage points lower, while being more than 7 times smaller. Based on the results, it was observed that the more parameters the network contains, the more accurate the results. This was consistent with the findings of Belkin et al. (2019), as models with a large number of parameters perform better than small or medium models. The greater the number of parameters, the faster the network achieved the convergence, at the expense of the larger size of the network on the disk and the number of operations needed for calculations. However, the networks see a diminishing return in accuracy improvement after using more than 2 million parameters, so architectures B and C are the most efficient of those tested.

In addition to the tested total accuracy, the accuracy of prediction for each character was improved, which is particularly visible for the uppercase letter K (from 0.33 in the base network to 0.89 in the A architecture), thanks to which the error rate for this letter is similar as to the rest of them. This is also the letter for which all architectures have the largest number of errors.

The hypothesis that increasing the number of convolutional layers above 2 has no impact on further increasing the accuracy of the model has been proven. The studied architectures were divided into two groups: those having two convolutional layers and the second group having a larger number of these layers. For a statistical comparison of these two groups, a one-sided T-student test was used. A p value of <0.001 was obtained.

Optimization of network architectures A and B can be an interesting topic for future work, for example by using quantization, which consists of representing network weights as

16-bit floating-point variables instead of the default 32-bit, (Choi et al., 2016) or using such method as network pruning, which in addition to regularization, usually results in reduced size of the compressed network, by setting the parameters close to zero to zero (Blalock et al., 2020; Wang et al., 2022). These solution are often found in the implementation of efficient networks for mobile devices and IoT, for example in advanced scanners in libraries.

Presented results are the next step of research the effectiveness of convolutional neural networks in handwritten character recognition. Architectures from this article could be also evaluated on other handwritten character datasets, such as English EMNIST (Cohen et al., 2017) or already mentioned Arabic HACB to rule out any inconsistencies and inaccuracies in researched data set. The measurement of effectiveness of adding more convolutional layers against more fully connected could also be more tested more rigorously using statistical methods. Similarly, effectiveness of different regularisation layers such as MaxPool, Dropout and SpatialDropout can be tested.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *ArXiv, abs/1603.04467*. https://doi.org/10.48550/ARXIV.1603.04467

Belkin, M., Hsu, D., Ma, S., & Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias-variance trade-off. *Proceedings of the National Academy of Sciences, 116*(32), 15849-15854. https://doi.org/10.1073/pnas.1903070116

Blalock, D., Ortiz, J. J. G., Frankle, J., & Guttag, J. (2020). What is the state of neural network pruning?. *ArXiv, abs/2003.03033*. https://doi.org/10.48550/arXiv.2003.03033

Bouthillier, X., Delaunay, P., Bronzi, M., Trofimov, A., Nichyporuk, B., Szeto, J., Sepah, N., Raff, E., Madan, K., Voleti, V., Kahou, S. E., Michalski, V., Serdyuk, D., Arbel, T., Pal, C., Varoquaux, G., & Vincent, P. (2021). Accounting for variance in machine learning benchmarks. *ArXiv, abs/2103.03098*. https://doi.org/10.48550/ARXIV.2103.03098

Choi, Y., El-Khamy, M., & Lee, J. (2016). Towards the limit of network quantization. *ArXiv, abs/1612.01543*. https://doi.org/10.48550/arXiv.1612.01543

Cohen, G., Afshar, S., Tapson, J., & Van Schaik, A. (2017). EMNIST: Extending MNIST to handwritten letters. *2017 International Joint Conference on Neural Networks (IJCNN)* (pp. 2921-2926). IEEE. https://doi.org/10.1109/IJCNN.2017.7966217

Gajoui, K. E., Allah, F. A., & Oumsis, M. (2015). Diacritical language OCR based on neural network: Case of amazigh language. *Procedia Computer Science*, *73*, 298–305. https://doi.org/10.1016/j.procs.2015.12.035

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., & Chen, T. (2018). Recent advances in convolutional neural networks. *ArXiv, abs/1512.07108*. https://doi.org/10.48550/arXiv.1512.07108

Hadidi, R., Cao, J., Xie, Y., Asgari, B., Krishna, T., & Kim, H. (2019). Characterizing the deployment of deep neural networks on commercial edge devices. *2019 IEEE International Symposium on Workload Characterization (IISWC)* (pp. 35-48). IEEE. https://doi.org/10.1109/IISWC47752.2019.9041955

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770-778). IEEE. https://doi.org/10.1109/CVPR.2016.90

Idziak, J., Šeļa, A., Woźniak, M., Leśniak, A., Byszuk, J., & Eder, M. (2021). Scalable handwritten text recognition system for lexicographic sources of under-resourced languages and alphabets. *In International Conference on Computational Science 2021* (pp. 137–150). Springer. https://doi.org/10.1007/978-3-030-77961-0_13

Islam, N., Islam, Z., & Noor, N. (2017). A Survey on optical character recognition system. *ArXiv, abs/1710.05703*. https://doi.org/10.48550/arXiv.1710.05703

Lukasik, E., Charytanowicz, M., Milosz, M., Tokovarov, M., Kaczorowska, M., Czerwinski, D., & Zientarski, T. (2021). Recognition of handwritten Latin characters with diacritics using CNN. *Bulletin of the Polish Academy of Sciences: Technical Sciences*, *69*(1), e136210. https://doi.org/10.24425/bpasts.2020.136210

Lutf, M., You, X., Cheung, Y., & Chen, C. (2014). Arabic font recognition based on diacritics features. *Pattern Recognition, 47*(2), 672–684. https://doi.org/10.1016/j.patcog.2013.07.015

Łukasik, E.,& Zientarski, T. (2018). Comparative analysis of selected programs for optical text recognition. *Journal of Computer Sciences Institute,* 7, 191-194. https://doi.org/10.35784/jcsi.676

Sharma, R., Kaushik, B., & Gondhi, N. (2020). Character recognition using machine learning and deep learning - a survey. *2020 International Conference on Emerging Smart Computing and Informatics (ESCI)* (pp. 341-345). IEEE. http://doi.org/10.1109/ESCI48226.2020.9167649

Tokovarov, M., Kaczorowska, M., & Milosz, M. (2020). Development of extensive polish handwritten characters database for text recognition research. *Advances in Science and Technology Research Journal*, *14*(3), 30-38. https://doi.org/10.12913/22998624/122567

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *ArXiv, abs/1706.03762*. https://doi.org/10.48550/arXiv.1706.03762

Wang, H., Qin, C., Bai, Y., Zhang, Y., & Fu, Y. (2022). Recent advances on neural network pruning at initialization. *ArXiv, abs/2103.06460*. https://doi.org/10.48550/arXiv.2103.06460