Handwritten Character Recognition and more advanced implementations in order to achieve Handwritten Text Recognition

Arailym Munaitpas
dept. of Computer Science
Nazarbayev University
Astana, Kazakhstan
arailym.munaitpas@nu.edu.kz

Madi Turgunov
dept. of Computer Science
Nazarbayev University
Astana, Kazakhstan
madi.turgunov@nu.edu.kz

Farida Kenzhebayeva
dept. of Mathematics
Nazarbayev University
Astana, Kazakhstan
farida.kenzhebayeva@nu.edu.kz

Yerzhan Yerbatyr
dept. of Computer Science
Nazarbayev University
Astana, Kazakhstan
yerzhan.yerbatyr@nu.edu.kz

Umit Azirakhmet
dept. of Computer Science
Nazarbayev University
Astana, Kazakhstan
umit.azirakhmet@nu.edu.kz

***Abstract*—This paper presents our work on developing systems capable of recognizing handwritten characters and words. Multiple approaches were investigated, including Convolutional Neural Networks (CNN), Dictionary Learning, segmentation pipelines, and sequence modeling with CRNN architectures. The study demonstrates the challenges and potential solutions in moving from isolated character recognition towards full handwritten text recognition.**

***Index Terms*—Handwritten Character Recognition, Handwritten Text Recognition, CNN, Dictionary Learning, OCR**

I. Introduction

Handwriting recognition is an important application of machine learning with a wide range of uses, such as document digitizing and form processing. It is the process of interpreting handwritten texts for further uses with different purposes. In the duration of this course, we have undertaken the challenge of developing an AI model that has the capability of reading and transcribing handwritten characters and words. In order to achieve this task, our team was divided into 2 sub-teams, each developing their own version of this model. The whole process started from obtaining the datasets from different sources, preprocessing them, splitting them into train, test, validation sets and went to and applying different approaches and comparing them using performance evaluation techniques. There were two main approaches that we decided to pursue throughout this endeavor: 1) Image Segmentation + Character Recognition and 2) Text Recognition using BiLSTMs. The next parts of this section contain the descriptions of the approaches. The Methods section describes the datasets and methodologies in detail. The Experiments and Results section presents the performance evaluation of the models. The report concludes with a summary of the project outcomes and key insights.

A. Approach 1: Image Segmentation + Character Recognition

We first employed a Nearest Centroid Classifier (NCC) for character recognition, where we segmented the input image into separate characters and classified them by determining the nearest centroid for each class. However, the NCC algorithm was plagued with overlapping centroids and noise sensitivity issues, restricting its performance as a multiclass classifier. Image preprocessing methods, including thresholding and dilation, aided segmentation by better isolating characters. The merging of nearby characters still constrained segmentation despite these measures. Results indicated that though NCC had its constraints, exploring other sophisticated models could enhance segmentation precision and recognition.

In parallel to NCC, we implemented a convolutional neural network (CNN) model focused on isolated character recognition. The CNN was trained using a dataset of English handwritten characters, encompassing both digits and letters. By leveraging convolutional layers for feature extraction and fully connected layers for classification, the model achieved a significantly higher validation accuracy compared to simpler classifiers. This demonstrated the effectiveness of deep learning-based feature learning in improving the robustness of handwritten character recognition.

B. Approach 2: Text Recognition using BiLSTMs

The second approach we have considered is using a CRNN model. This decision came after researching numerous existing solutions regarding whole-word and text recognition, with a significant portion of them utilizing some sort of combination of a convolutional neural network in tandem with LSTM layers, as well as using CTC loss [1] [2] [3]. The solution would consist of a convolution neural network that would extract features while reducing resolution and converting the

width to timesteps. A recurrent layer would then analyze these sequences to capture contextual information, and, subsequently, decoded using an F-C Layer utilizing CTC loss.

II. Methods

A. Dataset

In this study the datasets IAM Handwriting Forms, HKR, KOHTD and Cyrillic Handwriting Dataset, and an additional English Handwritten Characters Dataset were employed. Each of them plays distinct roles, since they belong to different languages and types of scripts and so has important contributions to the model development.

1) IAM: IAM Handwriting Forms is one of the most widely used datasets used for handwriting recognition. It was developed at the University of Benn, Switzerland, and includes 1,066 forms produced by almost 400 writers [4]. The number of words is 82,227 and the vocabulary consists of 10,841 words. The content is the writings from Lancaster-Oslo/Bergen Corpus (LOB Corpus), which is a collection of standardized British English texts and is mostly used for research purposes in the field of linguistics. The paper describing its development and processes of labelling through segmentation was published in 2002, by Urs-Viktor Marti and Horst Bunke [4]. Despite the fact the dataset was formed over two decades ago, it remains as one of the main datasets that are used for model training with different purposes, including handwriting recognition.

2) KOHTD: To adapt the model for Kazakh language too, Kazakh Offline Handwritten Text Dataset (KOHTD) was used. This dataset contains words from about 3,000 exam papers of Satbayev University and Kazakh National Al-Farabi University [5]. It consists of 922,010 symbols and 140,335 segmented images. The dataset was formed as a part of a research project by researchers from Satbayev University, Kazakh National Al-Farabi University, Assiut University and KazMunayGas Engineering LLP. In the project models such as Flor, Abdallah, Bluche, and Puigcerver were compared for handwritten text recognition [5]. The database is highly valuable since at the time of writing the reports it is the largest (and the only) dataset in Kazakh that can be used for handwriting classification. For our model it was splitted into train, test and validation sets with the ratio 8:1:1, respectively. There were folders for each of them with images in .jpg format. The corresponding .csv files contained names of the images in the first columns, and the transcriptions in Kazakh in the second.

3) HKR: The group of people with the same leading authors also released the Handwritten Kazakh and Russian (HKR) dataset. It was collected as a result of 200 writers filling the forms with most popular words in Kazakh, poems and words such as area, street, village names in Russian. Only a small portion (5%) of the dataset corresponds to Kazakh words. The dataset itself contains approximately 65,000 images, 63,000 sentences, about 716,000 symbols and 106,718 words [6]. Importance of this dataset lies in the fact that the Russian alphabet contains cyrillic letters as does the Kazakh, so it refined the model for cyrillic texts. For the current project it was splitted the same way as the previous database, KOHTD.

4) Cyrillic Handwriting Dataset: There is also another dataset in Russian that was used to train the model. It was presented by Konstantin Verner, SHIFT Lab member, published in Kaggle. The database consists of 73830 segments and was provided by SHIFT Lab [7]. Initially the data was split as 95% to 5%, as train and test data, respectively and was in .tsv format. For the project they were merged and were resplitted as 80%,10%,10% for train, test and validation sets, respectively, and the format was changed to .csv.

5) English Handwritten Characters: For isolated character classification, the English Handwritten Characters dataset was employed [8]. This dataset includes 3,410 images corresponding to 62 classes, covering digits (0-9), uppercase (A-Z), and lowercase (a-z) English letters. Each class contains approximately 55 samples. The images were provided alongside a CSV file containing image paths and labels. This dataset served as the foundation for training the CNN-based character recognition models used in the initial phase of the project.

B. Handwritten Character Recognition using NCC

For handwritten single-character recognition, we first utilized the Nearest Centroid Classifier (NCC) from the course lab assignments, a basic but efficient single-character classification method. The NCC operates by computing the centroid, or average feature vector, for every class within the training set. At recognition time, the model measures the Euclidean distance between a test sample's feature vector and all class centroids, assigning the sample to the class with the nearest centroid. NCC is a simple, computation-light method that serves as a reasonable starting point for character recognition.

C. Handwritten Character Recognition with CNN

To develop a more robust handwritten character recognizer, a convolutional neural network (CNN) model was implemented. The dataset used for training consisted of 3,410 images representing English handwritten characters, including digits and uppercase and lowercase letters, split into 62 classes.

The preprocessing steps included resizing all images to 28×28 pixels, converting them to grayscale, normalizing pixel values to the range $[-1, 1]$, and encoding labels with a LabelEncoder. The dataset was divided into training and validation sets using an 80:20 split, ensuring stratified distribution across classes.

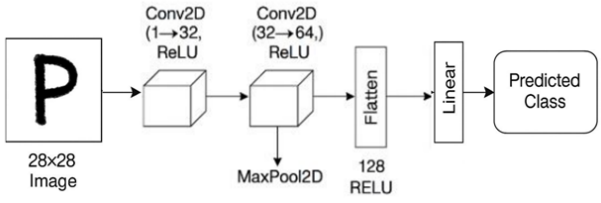

Fig. 1. Architecture of the CNN model.

The CNN architecture consisted of two convolutional layers: the first with 32 filters and the second with 64 filters, both

utilizing ReLU activations. A max-pooling operation was applied after the second convolutional layer to reduce spatial dimensions. Following feature extraction, the feature maps were flattened and passed through two fully connected layers, culminating in a softmax classification layer.

The model was trained using the Cross-Entropy loss function and optimized with the Adam optimizer, with a batch size of 64 and a total of 10 epochs. Training and evaluation were performed on a GPU to accelerate computation.

D. Handwritten Character Recognition using Dictionary Learning

To complement our deep learning approaches, we also implemented a Dictionary Learning framework for handwritten character recognition. This method leverages the principle of sparse representation, where each character image is expressed as a sparse linear combination of atoms from a learned dictionary. Our approach is inspired by the sparse coding-based classification paradigm proposed by Zhang and Lu [9], which demonstrated the effectiveness of overcomplete dictionaries combined with linear classifiers for robust pattern recognition.

The workflow involved the following steps:

- Preprocessing handwritten character images into flattened grayscale vectors, normalized between 0 and 1.
- Learning a dictionary using the K-SVD algorithm, which iteratively updates both the dictionary atoms and sparse codes to minimize reconstruction error.
- Representing each input character as a sparse code over the learned dictionary using Orthogonal Matching Pursuit (OMP).
- Feeding the resulting sparse codes into two types of classifiers for final character prediction: (i) a stochastic gradient descent (SGD)-trained logistic regression model, and (ii) a k-nearest neighbors (kNN) classifier based on the extracted sparse features.

Dictionary size and sparsity level were treated as hyperparameters and tuned based on cross-validation performance. In our experiments, we used a dictionary size of 500 atoms and moderate sparsity levels to balance reconstruction fidelity and feature compactness.

The model combining Balanced EMNIST data with Dictionary Learning and SGD-trained logistic regression achieved an accuracy of 62.40%. Meanwhile, the Dictionary Learning model paired with kNN classification attained a slightly higher accuracy of 65.57%. We observed that excessively sparse codes reduced discriminative capacity, while overly dense codes introduced noise, corroborating observations from prior studies [9]. Although Dictionary Learning required longer training times compared to CNNs due to the iterative nature of K-SVD, it provided interpretable feature representations and reasonable accuracy given the reduced training sample size.

E. Segmentation of the image

Segmentation of the image was the next key step of the handwritten character recognition pipeline, as it implies the separation of individual characters from the text. To reduce the complexity of the processing, we first converted the image to grayscale and transformed the grayscale image into a binary image using thresholding algorithms. To segment further, we performed a *dilation* operation on the binary image. Dilation expands the foreground pixels, which serves to bridge close-by characters or characters with tight gaps amongst them. This was performed to ensure that individual characters are recognized as distinct contours. The intermediate result can be seen from Figure 2, where almost all characters were identified, however, some of the characters, namely the point of 'i and the bottom of the second 'e', were recognized as separate characters since they had some space.

Fig. 2. Segmentation after performing Dilation

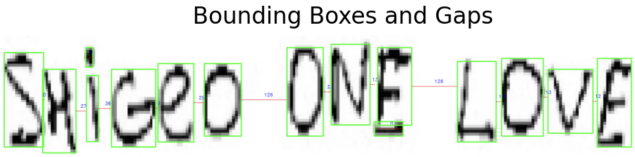

Fig. 3. Bounding boxes and Space

Following dilation, we employed contour detection to detect the boundaries of individual characters. Contours were used to detect bounding boxes around each character, which were subsequently processed for classification. One challenge we faced while performing segmentation was preventing the incorrect detection of spaces between the characters. Horizontal gap measurement was employed to determine when a space existed, with a blank character added to the output. The selection of the gap threshold was key, as it had to provide a balance of detecting separate characters without inserting multiple spaces, as can be viewed from Figure 3, where the gap threshold influenced the accuracy of detecting spaces. Overall, although the segmentation process generated helpful bounds for characterization, merging characters through dilation and the difficulty of accurately identifying spaces resulted in inaccuracies. Refining the segmentation pipeline, such as enhancing the gap detection thresholding and overcoming the shortcomings of dilation, will assist the segmentation quality for a more perfect characterization.

F. Handwritten Text Recognition with CRNN+CTC

The second (and more successful) approach consisted of training on whole words using a combination of a convolutional and recurrent layers. Using the model used in [1] as a baseline, we modified it to suit our needs. The final model consists of three main parts. The first amongst them is

the convolutional part, made up of 6 sequential convolutional layers, each of them utilizing the *Conv2D* PyTorch layer, a Normalization layer, a ReLU activator as well as, MaxPool2D (in some of them). The third layer features an extra dropout layer to improve performance and reduce overfitting. Overall, throughout the entire convolutional pipeline, the image is reduced from a 64 by 512 image, to a 4 by 16 image with 512 features.

Afterwards, it is passed along to the recurrent part, which consists of two bidirectional LSTM layers with a dropout layer in the middle. The 16 pixels become 16 timesteps. The first bidirectional LSTM layer takes all the features from the convolutional part, and outputs 512 features (256 from left-to-right and 256 from right-to-left). Then, a dropout of 0.5 is applied and an identical bidirectional lstm further improves contextual data.

The resultant output is sent to a fully connected linear layer, which matches each timestep with a character. In the end, a word or short phrase is served as the output.

CTC loss [3] was incredibly important, and quite likely is the saving grace of this approach. Despite there being 16 timesteps, the characters aren't split perfectly into 16 equal squares on the image. The way CTC loss works is by predicting the distribution of the characters over the entire timeseries. In simple terms, CTC loss enables the model to be trained end-to-end without requiring explicit alignment between the input image features and the target character sequence. It automatically learns to handle variations in writing style, character spacing, and length discrepancies between input and label sequences. It achieves that by adding blank characters and removing repeated characters.

The detailed diagram of the architecture of the model can be seen in Figure 4.

III. Experiments and Results

A. Handwritten Character Recognition with NCC: Results

The binary classification task using NCC was moderately successful. The classifier performed well when distinguishing between two classes (e.g., "Is this character a 'Z' or not?"), achieving reasonable accuracy on simpler tasks. The model correctly classified characters with similar shapes under controlled conditions, as shown in Figure 5. However, when applied to a multiclass classification scenario, the performance declined significantly. NCC struggled with the problem of centroid overlap, as a number of similar characters have similar attributes, making their centroids coincide. This coincidence translated into misclassifications, especially for cases with fine differences, such as 'O and 'Q'.

In spite of these issues, NCC provided a helpful benchmark and emphasized the necessity of a more advanced model for dealing with a larger, more diverse group of characters.

B. Handwritten Character Recognition with CNN: Results

After training for 10 epochs, the CNN model achieved a validation accuracy of 61.29%. Despite the relatively simple architecture and limited number of training epochs, the model demonstrated the effectiveness of convolutional feature extraction for isolated handwritten character recognition.

Sample predictions on the validation set showed successful classification of various characters, such as "S", "U", "P", "8", and "Y", although occasional misclassifications occurred, particularly for characters with similar structures or in cases where handwriting was highly distorted.

The results highlight that even a basic CNN architecture is capable of generalizing over a diverse set of handwritten characters, and suggest that further improvements in accuracy could be achieved with deeper architectures, more extensive data augmentation, and longer training.

C. CRNN Model and multiple alphabets

First, we trained the CRNN Model on purely Latin datasets. This resulted in great, even fantastic performance. With an accuracy of 83.32% (more detailed results in Table **??**), it was well beyond our expectations. However, problems started to arise when we attempted to train with a merged dataset of Latin and Cyrillic characters. This model's accuracy fell down to only 73.8%, which we attributed to the fact that Cyrillic might me more difficult for the model to learn, since more of the dataset contained cursive text, as handwritten Cyrillic is mostly cursive. For this reason, we attempted to train a model that works exclusively on Cyrillic, and obtained an expected accuracy of 64.14%. From manual evaluation, we also found that the model's Latin outputs would lose coherence, possibly due to the fact it had to account for both Latin and Cyrillic. So, we attempted an another solution: instead of teaching all of the data to one model, we would create two separate models - one for Cyrillic, and one for Latin. Then, we would train a basic classifier to differentiate the input between the two, and the results, while not drastically improving the situation, showed promising results.

We trained two classifiers: one using a CNN layer, one using basic PCA/LDA. The PCA/LDA classifier, despite being more simple, proved to have much higher performance at 98% accuracy, whereas the CNN classifier only had 91%.

On Table I, the four approaches are compared. CRNN-Split-Pipeline integrates a PCA/LDA classifier trained to classify alphabets between Cyrillic and Latin. The classifier has an accuracy of 98%. The results, while not significantly better, show improvement over the model that learned both languages at once.

IV. Conclusion

In conclusion, this project has experimented with several approaches to handwriting recognition. While not all approaches were successful, the insights achieved by them was equally valuable. We have learned that CTC loss can be a good, if not better alternative to text segmentation, as it can also adapt to the situation, whereas more traditional segmentation algorithms cannot.

The NCC classifier provided a baseline for character recognition but faced limitations in multiclass scenarios due to centroid overlap and noise sensitivity. In contrast, the CNN

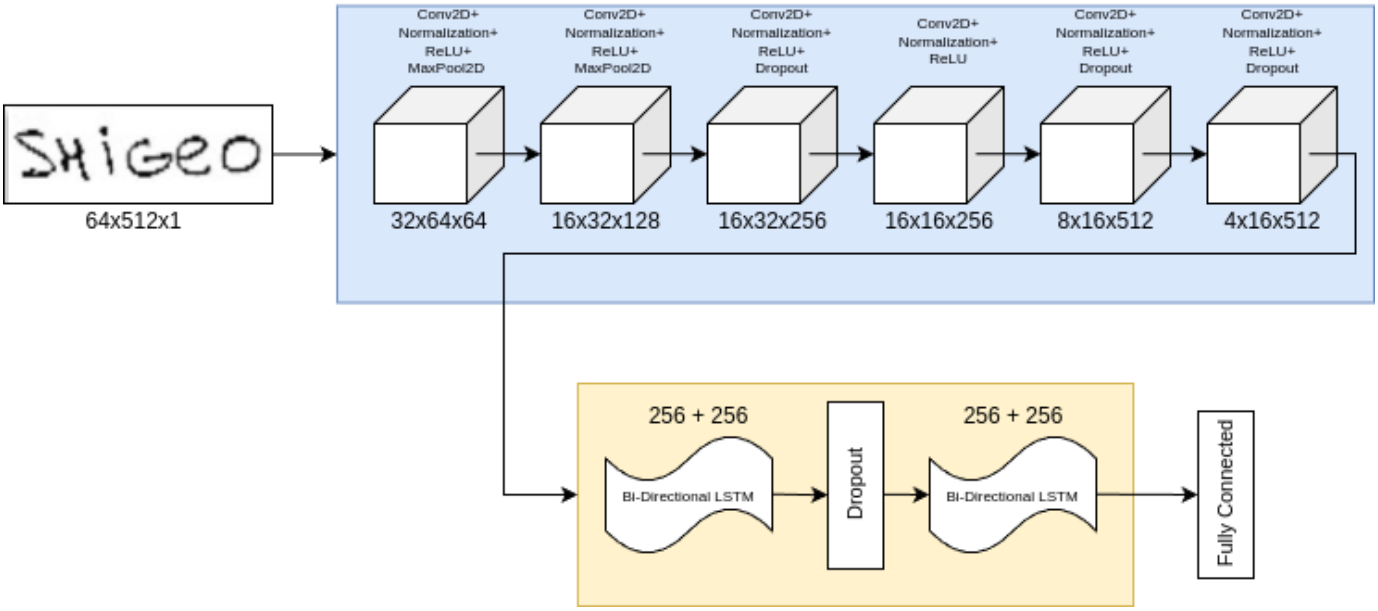

Fig. 4. Architecture of the CRNN Model

Model	Accuracy	Character Error Rate	Length Correlation	Latin Accuracy	Cyrillic Accuracy
CRNN-All-Datasets	73.87%	12.05%	96.81%	83.69%	57.95%
CRNN-Latin	N/A	7%	94.69%	83.53%	N/A
CRNN-Cyrillic	N/A	15.15%	97%	64.14%	
CRNN-Split-Pipeline	75.65%	10.24%	79.03%	84.53%	61.20%

TABLE I
DETAILED RESULTS OF DIFFERENT CRNN APPROACHES

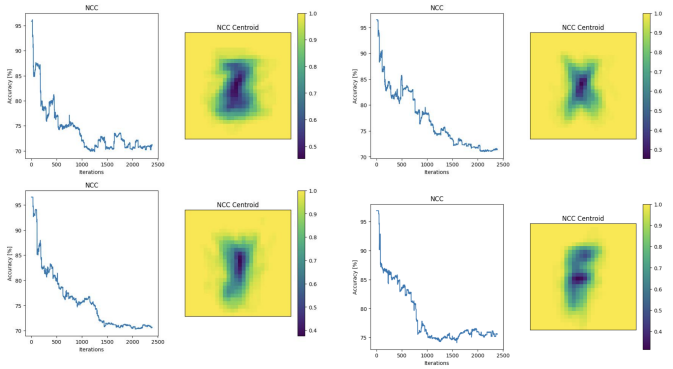

Fig. 5. Handwritten Character Recognition using NCC

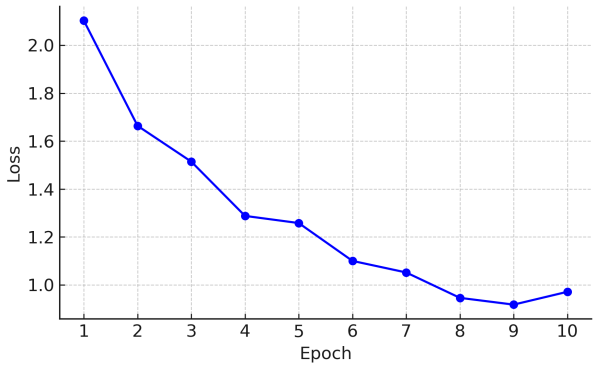

Fig. 6. Training loss of the CNN model.

model demonstrated the effectiveness of deep learning for isolated character recognition, achieving a respectable accuracy despite its simple architecture. This highlighted the potential for further improvements through deeper networks, data augmentation, and extended training.

Key challenges included accurate segmentation of connected characters, sensitivity to handwriting variations, and balancing model complexity with computational efficiency. Future work could explore advanced segmentation techniques, hybrid architectures, and expanded datasets to improve generalization across languages. Additionally, integrating language models and optimizing the CTC decoder could further enhance text recognition accuracy.

In summary, this project validated the superiority of deep learning approaches like CNNs and CRNNs for handwriting recognition while emphasizing the need for tailored solutions in multilingual contexts. The insights gained lay a foundation for developing scalable, adaptive systems capable of handling diverse handwriting styles and scripts.

REFERENCES

[1] K. Markou, L. Tsochatzidis, K. Zagoris, A. Papazoglou, X. Karagiannis, S. Symeonidis, and I. Pratikakis, “A convolutional recurrent neural net-

work for the handwritten text recognition of historical greek manuscripts," in *Pattern Recognition. ICPR International Workshops and Challenges* (A. Del Bimbo, R. Cucchiara, S. Sclaroff, G. M. Farinella, T. Mei, M. Bertini, H. J. Escalante, and R. Vezzani, eds.), (Cham), pp. 249–262, Springer International Publishing, 2021.

[2] S. Nakarmi, S. Sthapit, A. Shakya, R. Chulyadyo, and B. K. Bal, "Nepal script text recognition using CRNN CTC architecture," in *Proceedings of the 3rd Annual Meeting of the Special Interest Group on Under-resourced Languages @ LREC-COLING 2024* (M. Melero, S. Sakti, and C. Soria, eds.), (Torino, Italia), pp. 244–251, ELRA and ICCL, May 2024.

[3] H. Zhan, Q. Wang, and Y. Lu, "Handwritten digit string recognition by combination of residual network and rnn-ctc," in *Neural Information Processing* (D. Liu, S. Xie, Y. Li, D. Zhao, and E.-S. M. El-Alfy, eds.), (Cham), pp. 583–591, Springer International Publishing, 2017.

[4] U.-V. Marti and H. Bunke, "The iam-database: An english sentence database for offline handwriting recognition," *International Journal on Document Analysis and Recognition*, vol. 5, no. 1, pp. 39–46, 2002.

[5] N. Toiganbayeva *et al.*, "KOHTD: Kazakh offline handwritten text dataset," *Signal Processing: Image Communication*, vol. 108, p. 116827, October 2022.

[6] D. Nurseitov, K. Bostanbekov, D. Kurmankhojayev, *et al.*, "Handwritten kazakh and russian (hkr) database for text recognition," *Multimedia Tools and Applications*, vol. 80, pp. 33075–33097, 2021.

[7] C. Werner, "Cyrillic handwriting dataset." https://www.kaggle.com/datasets/constantinwerner/cyrillic-handwriting-dataset, 2021. Accessed: April 23, 2025.

[8] T. E. de Campos, B. R. Babu, and M. Varma, "Character recognition in natural images," in *Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal*, February 2009.

[9] K. Zhang and J. Lu, "Handwritten character recognition via sparse representation and multiple classifiers combination," in *2010 IEEE International Conference on Information Theory and Information Security*, pp. 1139–1142, 2010.