Handwritten Cyrillic Postal-Address Interpretation Using Well-Known Recurrent Neural Network Models

Daniyar Nurseitov*, Kairat Bostanbekov†,
Maksat Kanatov‡, Anel Alimova§,
Rassul Tolegenov‖

Abdelrahman Abdallah

National Open Research Laboratory for Information and Space Technologies
Satbayev University
Almaty, Kazakhstan
{*nurseitovdb, †kairat.boss, ‡maksatkanat, §anic2002, ‖tolegenov.rasul}@gmail.com

MSc Machine Learning & Data Science
Satbayev University
Almaty, Kazakhstan
abdoelsayed2016@gmail.com

***Abstract*—This paper demonstrates a work that aims to recognize handwritten postal addresses on envelopes written in Cyrillic (Kazakh and Russian languages) by means of machine learning algorithms. Fulfilling this aim helps to enhance mailing process performance significantly, reducing postal service expenses on delivery. To accomplish handwritten postal-address recognition task, first, a training dataset consisting of 21000 images (500 handwritten samples of 42 classes) were collected. Then, dataset augmentation was applied to enrich this dataset, which resulted in 207438 images. Next, a quantitative comparison of well-known recurrent neural networks (RNN), such as SimpleHTR, LineHTR, NomeroffNet, and Flor's HTR models, has been implemented to choose the best performing model on the dataset given. As experiment results proved, the Flor model demonstrated the best performance with BLABLA% character accuracy rate (CAR) and BLABLA% word accuracy rate (WAR).**

Flor HTR model was then taken as a basis to a new improved model, called *Attention-based Fully Gated CNN-BGRU*. The model combines 5 convolutional blocks (as an encoder), Bahdanau attention mechanism (as an attention part), 2 bidirectional Gated Reccurent Units (as a decoder), and a Connectionist Temporal Classification (CTC) part. This model's performance was reported to be *BLABLA%* (CAR) and *BLABLA%* (WAR) on the aforementioned dataset.

***Keywords*—** CNN, RNN, LSTM, CTC, CAR, WAR

I. INTRODUCTION

Handwriting recognition (HWR) is the process of converting handwritten characters or words into a format that the computer understands. It has an active community of academic researchers studying it for the past few decades as advances in this field help to automate variety of routine processes and paperworks within companies. An example of these types of unpleasant work would be a painstaking search of some scientific evidences within tons of handwritten historical manuscripts by a historian, which requires a lot of time consuming reading work. Converting these manuscripts into a digital format using HWR algorithms would allow the historian to find information within a few seconds. Other examples of routine work that need automation would be the tasks related with signature verification, writer recognition and so on.

Generally, there are 2 types of HWR algorithms depending on the time when recognition task is performed: online and offline. Online HWR method defines techniques that are able to convert handwritten texts extracted from an electronic pen interface into machine format texts instantly [1] [2] [3] [4] [5]. Considering maturity of software solutions in [1] [2] and other commercially available products, it is commonly believed that the problem of online HWR can be considered as being close to solved [6]. In contrast, offline HWR methods are meant to recognize texts which were written beforehand and can be found on documents, such as a scanned document image or an image with handwritten texts on it. The rest part of this section provides a review on some of offline HWR algorithms ever developed.

Extensive research in HWR field resulted in a progress ranging from traditional methods to advanced deep learning based algorithms. In other words, depending on the technology utilized, all HWR methods can be split into 2 basic categories: traditional methods with no deep learning applied and deep learning based algorithms.

Offline HWR with no deep networks applied is not a task of easy type since different people have different handwriting styles, and letters within words might be written in a connected manner. Recognition of a handwritten text would be much easier if the text was written in a format where letters are written separately from each other like in machine printed texts. That is to say that all optical character recognition (OCR) and intelligent character recognition (ICR) engines take advantage of isolated (disconnected) symbols in machine or human printed texts to extract those symbols separately and recognize them. Examples of OCR/ICR engines can be FineReader by ABBYY company [7] [8] and AddressParcel by ParaScript [9]. However, the reality is that not all people write text characters separately.

A large number of early work with no deep learning applied focused on utilising Hidden Markov Models (HMM) to recognize handwritten characters and texts [10] [11] [12] [13] [14] [15] [16]. The main idea behind this HMM algorithm is to extract sequential hidden states (handwritten texts) using observed data available (images of characters/letters). An example of other techniques can be found in papers [17] [18], where the authors leverages a line-oriented approach to HWR task to segment Spanish handwritten text lines into words using dynamic programming based pattern matching algorithm. In other words, handwritten texts are recognized on word-wise manner unlike other character-based HWR methods. In [19], the authors utilised a two-tier beam search approach to Arabic HWR problem. They took advantage of Arabic letter's conditional joining rules to ease this task. Another HWR and typewriting recognition algorithm for Arabic/English language can be found in [20], where region growing based implicit segmentation of a handwritten text image into letters was utilised. After segmentation step, an extracted letter region is resized into a 9×9 matrix and processed with low-pass filter to be introduced into neural network. As the authors claim, the classification accuracy was about 90% and 80% for Arabic and English handwritings respectively. Unlike this approach, the paper [21] demonstrates totally segmentation-free approach to recognition of old Greek handwritten manuscripts. Since

Greek letters (even words to some extent) in these manuscripts are barely separable and contain a lot of closed cavities (which is a specific feature), the authors proposed a novel HWR algorithm that relies on this feature.

Although all abovementioned research works with no deep learning applied were pioneers to solve HWR task, they all had the same common drawback; They needed word/character features, which are specific to a language, to be added to a HWR algorithm manually to increase recognition accuracy each time when a new feature was found.

With prosperity of deep convolutional neural networks, new solutions to a handwriting recognition problem have been offered. Deep neural networks can learn specific features of language characters/letters automatically, thus, pushing HWR accuracy to higher levels.

An open-source Tesseract-OCR engine (originally developed/maintained by Hewlett Packard company in 1984-1994s, and sponsored by Google since 2006) is an example of system that represents both HWR algorithms with no deep learning applied and deep learning based ones. Being able to recognize only English handwritings in its initial version, and having been enabled to recognize handwritten texts in up to 100 languages in version 2 and 3, the Tesseract-OCR engine (version 4) has finally been armed with LSTM (long short-term memory) based RNN models. A Full description of Tesseract-OCR engine architecture (without LSTM added) is given in [22]. An example of work using early versions of Tesseract (without LSTM) can be found in [23]: the authors proposed a solution to recognition of free-flow (the slope of horizontal lines is not fixed) and isolated (disconnected characters) handwritten texts written using lower-case Roman script. According to the authors, the overall recognition accuracy of Tesseract engine was observed as 78.39%. Another research work [24] with overall accuracy about 93-95% tried to solve the problem of recognizing handwritten programming code. It seems that high accuracy rates of these works came from the characteristics of texts (not isolated and not cursive). The accuracy of Tesseract+LSTM model on cursive text recognition task can be found in [25], where "Character error rates" on Cyrillic (Russian) text recognition task falls from 19.06% (base form of Tesseract) to around 4% (Tesseract+LSTM model). Similarly, "Word error rates" have also seen a reduction from 30.83% to around 15%.

Generally, all artificial neural network based approaches to a HWR problem mainly rely on Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), particularly of LSTM (long short-term memory) form [26]. The key difference of using CNN models from using RNN models is that CNN based algorithms are usually utilized to recognize isolated (separate) language characters/digits [27], [28], [29]. A CNN model can also be employed to recognize full texts (with naturally connected handwritten letters in them), but for this task a huge vocabulary dataset (consisting of all possible language words) is required to train that CNN model [30]. Since it is an immense work to collect all language words (with a few dozens of possible handwritten variations for each word), a CNN model itself is generally not a reliable solution to a handwritten text recognition problem. In contrast, RNN models are networks with an architecture which is able to handle the classification (recognition) of sequential data such as handwritten texts [31], [32]. That is to say that these kind of models are able to remember both language characters and sequences of those characters in a training stage, and then use this memory to classify words/texts in a test stage.

In [33], the authors employed Bidirectional LSTM model combined with CTC (Connectionist Temporal Classification) method to recognize unconstrained Bangla (a type of Indian scripts) handwritten texts. According to the authors, the obtained word-level recognition accuracy was around 75%. Initially introduced in [34], the key advantage of using CTC method is elimination of a prior segmentation of input handwritten words/texts into characters. As stated in [35], CTC-incorporated RNN networks showed a significant increase in accuracy resulting in 79.7% and 74.1% for online and offline Latin HWR tasks respectively.

Another example of application of LSTM+CTC based RNN model in HWR problem can be found in a paper [36]. The authors accomplished a comparative evaluation of CTC and Sequence-to-Sequence (Seq2Seq) Learning based LSTM RNNs in a historic handwritten Latin text recognition task. According to the authors, the CTC based model outperformed the Seq2Seq Learning based model with overall 78.10% and 72.79% word-level accuracy respectively.

Generally, it is clear that these days, the most reliable solution to a HWR task is RNN models, particularly with LSTM architecture. Therefore, this proposed work takes an effort to solve the task of offline handwritten Cyrillic postal-address (on envelopes) recognition with a help of RNN models. For this problem to be solved, investigation and comparison of the state-of-art RNN models were implemented (sections II and III). Finally, the best performing model was applied to the postal-address interpretation task.

II. Well-known RNN architectures

After careful consideration of all open-source solutions available, the most popular RNN models were chosen since they seem to be the models that come to attention first when surfing the Internet. Brief descriptions of these models are given below:

A. *SimpleHTR model*

Originally inspired by ANN architectures by [37] and [38], Harald Scheidl proposed a new approach to HWR task [39] in 2018. The model's architecture consists of 5 CNN layers, 2 RNN (LSTM) layers, CTC loss and decoder layers (Figure 1).

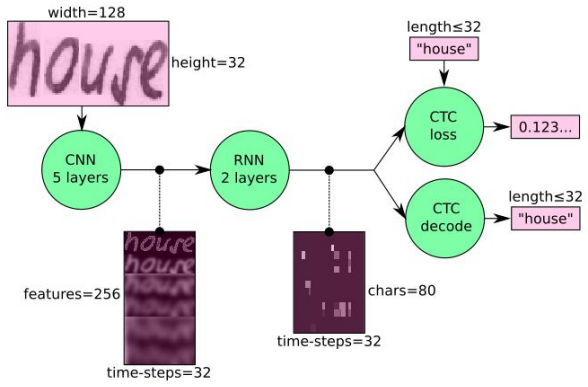

Figure 1. SimpleHTR architecture: green icons are operations; pink icons are data streams [40]

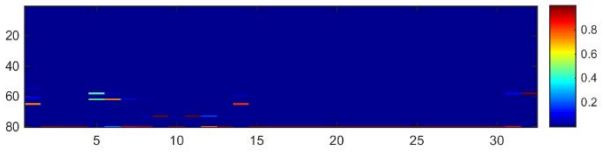

Figure 2. SimpleHTR RNN model's output: tiny brighter bars mean that relevant time-steps (there are 32 time-steps overall on x-axis) gathered high probabilities for one of 80 possible characters (on y-axis) [41]

Below is a SimpleHTR algorithm pipeline in short [40]:

- Input is a grayscale image of fixed size 128 x 32 (W x H)

- CNN layers map this grayscale image to a feature sequence of size 32 x 256
- RNN (LSTM) layers with 256 units map this feature sequence to a matrix of size 32 x 80 (Figure 2): here 32 represents the number of time-steps (horizontal positions) in an image with a word; 80 represents probabilities of different characters at a certain time-step in that image)
- CTC layer may work in 2 modes: LOSS mode - to learn to predict the right character at a time-step when training; DECODER mode - to get the recognized word when testing
- batch size is equal to 50

SimpleHTR model ideally requires handwritten texts to be split into words; otherwise, recognition of a full text line would result in low accuracy since 32 time-steps are insufficient to handle a large number of characters in that text line.

B. *LineHTR model*

LineHTR model [42] is just an extension of the previous SimpleHTR model, which was developed to enable the model to process images with a full text line (not a single word only), thus, to increase the model's accuracy further. Architecture of LineHTR model is quite similar to that of SimpleHTR model, with some differences in the number of CNN and RNN layers and size of those layers' input: it has 7 CNN and 2 Bidirectinal LSTM (BLSTM) RNN layers.

Below is a LineHTR algorithm pipeline in short [42]:

- Input is a grayscale image of fixed size 800 x 64 (W x H)
- CNN layers map this grayscale image to a feature sequence of size 100 x 512
- BLSTM layers with 512 units map this feature sequence to a matrix of size 100 x 205: here 100 represents the number of time-steps (horizontal positions) in an image with a text line; 205 represents probabilities of different characters at a certain time-step in that image)
- CTC layer may work in 2 modes: LOSS mode - to learn to predict the right character at a time-step when training; DECODER mode - to get the final recognized text line when testing
- batch size is equal to 50

C. *"Nomeroff Net" OCR model*

According to the athors of "Nomeroff Net" automatic numberplate recognition system [43], the OCR architecture solution (Figure 3) was originally taken from https://supervise.ly/.

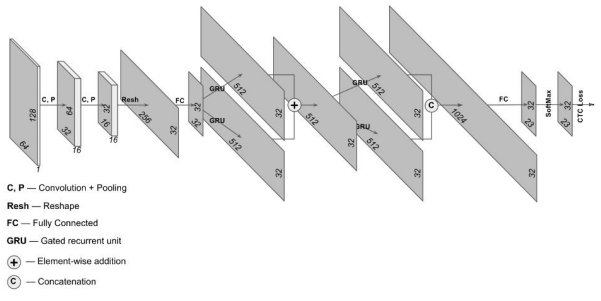

Figure 3. Nomeroff's number plate recognition model architecture [44]

As can be seen from the Figure 3, the Nomeroff Net OCR algorithm pipeline is as follows:

- Input is a grayscale image of fixed size 64 x 128 (W x H)
- This grayscale image is then introduced into 2 subsequent CNN layers, which in turn output feature maps of (16 x 32) x 16 size
- Then these feature maps are reshaped into a single map of 256 x 32 size
- then this map is introduced into a fully connected (FC) layer
- the output from FC layer is directed to 2 parallel RNN layers of GRU (Gated Recurrent Unit) type
- outputs from 2 GRU layers are combined into one by Element-wise addition operation to form a map of 512 x 32 size
- then this map is directed to 2 parallel GRU RNN layers again
- outputs from the last 2 GRU layers are concatenated to form a map of 1024 x 32 size
- then this map is introduced into subsequent FC and Softmax layers, before being directed to CTC decoder to obtain the final recognized text

Although Nomeroff Net OCR architecture was designed to recognize "machine typed" car numbers, it is also worth to check the model's performance on handwritten text recognition tasks. That is why this model is also included into the list of RNN architectures evaluated in this research work.

D. *Flor's HTR model*

The last model evaluated in this research work is HTR model by Arthur Flor [45]. A clear explanation of this work is available via [46].

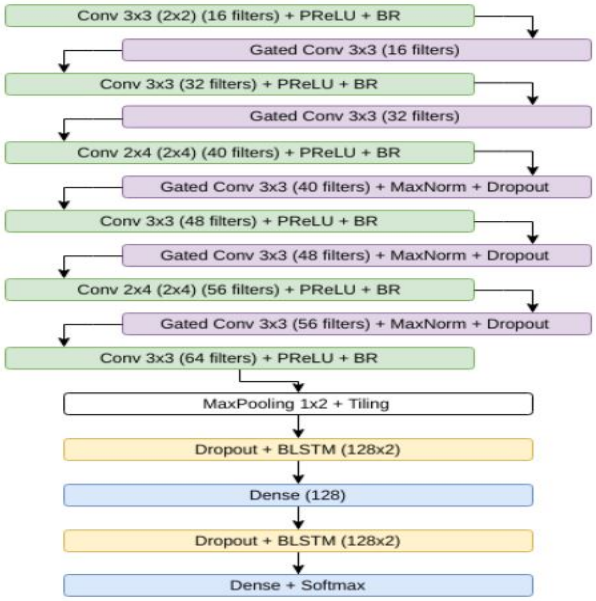

Figure 4. Flor HTR model architecture [46]

As the author stated, the Flor HTR model architecture (Figure 4) is a combination of 2 previous HTR model architectures: by Puigcerver [47], who introduced a model based on CNN-BLSTM architecture with an overwhelmingly large number of ANN parameters (around 9.6 million), thus, having a high level of recognition rate, but being slow; and by Bluche and Messina [48], who proposed a model based on Gated-CNN-BLSTM architecture with a very few parameters (around 730 thousand), thus being compact and faster. Combining these 2 architectures, Arthur Flor obtained Gated-CNN-BLSTM architecture (Figure 4) with low number of parameters (around 820

thousand) and high recognition rate.

While working on the model's architecture, the author also applied Gated approach presented by [49], Batch Renormalization approach by [50], the Parametric Rectified Linear Unit (PReLU) activator approach by [51], and RMSProp optimizer (for training) approach by [52]. The model's CTC layer works based on the Vanilla Beam Search [53] method to decode an output matrix from BLSTM RNN layers, and each character decoded is predicted from a set of 95 characters from ASCII table.

As reported in [46], the Flor HTR model showed 8.58% Character error rate and 27.90% Word error rate, whereas these rates were reported as 9.39% and 29.34% for the Puigcerver model, and 14.30% and 41.17% for the model by Bluche et al.

E. Attention-based Fully Gated CNN-BGRU model

This paper offers a modified version of Flor's HTR model, called Attention-based Fully Gated CNN-BGRU, aiming at improving HTR model accuracy in handwritten Cyrillic text recognition task (Figure 5).

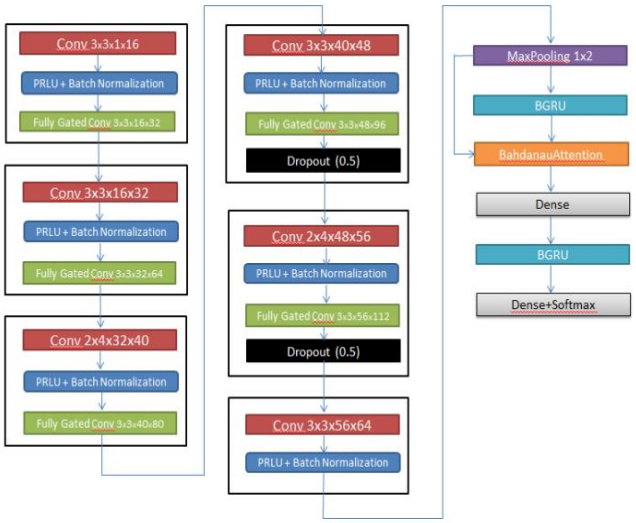

Figure 5. Attention-based Fully Gated CNN-BGRU model

This model's architecture consists of 4 main parts: encoder, attention, decoder, and CTC.

An encoder part consists of 5 convolutional blocks, each of which is made up of a convolutional layer, Parametric Rectified Linear Unit (PReLU) activator [51] with Batch Normalization, and gated convolutional layer [48]. The Dropout technique is also applied at the input of some convolutional layers (with dropout probability of 0.5) to reduce overfitting issue [54].

As an attention part of this model's architecture, Bahdanau attention mechanism is used [55]. Generally, attention mechanisms encode an input sentence by segmenting it into a fixed number of parts so they can be processed later by a decoder. Bahdanau attention mechanism enabled attention mechanism to focus on (soft-)searching relevant parts of an input sentence, rather than hard segmenting it. The key role of Bahdanau attention mechanism applied between an encoder and decoder is to provide a richer encoding of the input sequence.

Unlike the Flor's architecture which uses bidirectional long-short term memory (BLSTM) layer as a decoder part, this model utilizes a Bidirectional Gated Recurrent Unit (BGRU) [56]. An advantage of using a BGRU design rather than using standard recurrent neural network architectures is that it is robust to a vanishing gradient problem. The GRU is like a LSTM with a forgotten gate, but has fewer parameters than LSTM because it lacks a gate for output. The output sequence of BGRU RNN is mapped to a matrix of size 128x80 (79 Cyrillic characters + 1 for "blank").

Lastly, a CTC part is the same as in the Flor's HTR model. The output from the CTC part is a final result with recognized characters.

III. Experiments and Evaluation

It is obvious that the first step in all supervised machine learning algorithms is a dataset preparation procedure. The basis for this work's dataset was made up of distinct **BLABLA** words and short sentences, written in Russian and Kazakh languages (with about 95% of Russian and 5% of Kazakh words/sentences respectively). Note that both these languages are written in Cyrillic and share the same 33 characters. Beside these characters, the Kazakh alphabet also contains 9 additional specific characters.

This dataset of distinct **BLABLA** words/sentences were boosted by applying various handwriting styles (of approximately 50-100 different persons) to each of these distinct words. This resulted in a final dataset with an overall number of **BLABLA1** handwritten words/sentences in it. Then this final dataset was split into Training (70%), Validation (15%), and Test (15%) datasets. Test dataset itself was equally split into 2 subdatasets (7.5% each): the first dataset was named as TEST1 and consisted of words which do not exist in Training and Validation datasets; the next subdataset was named as TEST2 and made up of words that exist in Training dataset but with totally different handwriting styles. The main purpose of splitting the Test dataset into TEST1 and TEST2 datasets was to check an accuracy difference between recognition of unseen words and the words which were seen in training stage but with unseen handwriting styles.

After training, validation, and test datasets had been prepared, and models been trained, comparative evaluation experiments were conducted.

The first experiment was conducted with SimpleHTR model which showed a performance with average Character Accuracy Rates (CAR) of 38.28% and 90.29% on TEST1 and TEST2 datasets respectively (Figure 6). This big difference in CAR rates shows that SimpleHTR model overfitted to words seen in training stage and demonstrated lower level of generalization. Note that the training stage took 5 epochs when the model ceased to learn from data, and started to show no improvement.

Next experiment was carried out with LineHTR model which was trained on data for 100 epochs. This model demonstrated a performance with average CARs of 29.86% and 86.71% on TEST1 and TEST2 datasets respectively (Figure 7). Similar tendency of overfitting to training data can be observed here as well.

The same experiments were conducted with NomeroffNet HTR model. It took 4 epochs to train the model when it stopped to learn and showed no improvement. Unlike previous models examined, this model showed lower average CAR rate (70.87%) even on TEST2 dataset (Figure 8). The model's CAR value on TEST1 dataset was reported as 23.9%. As can be observed from the figure, NomeroffNet model also suffers from overfitting.

Experiments with Flor's original model reported totally different performance rates. The number of epochs to train the model was about 250. **There is an ISSUE WITH TRAINING LOSS GOING UP AND VALIDATION LOSS GOING DOWN.** Model's CAR rates on TEST1 and TEST2 datasets were 27.78% and 38.46% respectively (Figure 9). According to these rates, although Flor's model demonstrated lower CAR rates on both TEST1 and TEST2 datasets, it seems that the model is more robust compared to previous models, and good at generalization. It also seems that a larger training dataset is needed in order to obtain higher accuracy rates.

As described in subsection E of section II, this research work introduces a model called Attention-based Fully Gated CNN-BGRU

model. It combines some new modifications to Flor's HTR model [45] to improve recognition rate further. Training of this new model took 240 epochs. The CAR rates on TEST1 and TEST2 datasets were reported as 56.23% and 67.06% respectively (Figure 10). As can be seen from the figure, Attention-based Fully Gated CNN-BGRU model resulted in higher CAR and generalization rates overall.

Beside CAR rates, when testing all abovementioned models Word Accuracy Rates (WAR) were also calculated. Overall quantitative evaluation of models is given in the Table I.

The CAR calculcation pipeline should be described.

Why did we develop our own CAR calculcation pipeline. Because Levenstein method does not take into account each symbol comparison

IV. Discussion

The primary goal of this research work was to investigate and quantitatively compare the state-of-the-art RNN models to choose the best performing one in a handwritten Cyrillic postal-address recognition task. This goal also incorporates all efforts put on improving the best performing RNN model. According to experiment results, the Flor's HTR model demonstrated comparatively better results in terms of generalization and overall accuracy (see Table I). This model was then extended to the modified version, called Attention-based Fully Gated CNN-BGRU model.

As figures 6-10 show, generally average CARs of all models tend to be low. It seems the reason for that is large differences between frequencies of Cyrillic characters. In other words, since the dataset includes a small number of Kazakh language handwritings, the language characters have lower frequencies (distribution in dataset) compared to other Cyrillic letters. Consequently, abovementioned models struggle to recognize these characters resulting in very low recognition rates. Hence, this affects the overall average CAR rates. The dataset also includes non-alphabetic characters (such as ". , !" and so on) with small distributions.

SimpleHTR, LineHTR, and NomeroffNet models seemed to prone to overfitting while being trained to Cyrillic handwritings. It seems that enriching the dataset with variety of Kazakh and Russian words, and making it balanced will solve this issue.

Limitations and issues discussed in this section are subject to further work in this research.

V. Conclusion and Future work

This research work tried to solve a handwritten Cyrillic postal-address interpretation task using well-known RNN models, such as SimpleHTR, LineHTR, NomeroffNet, and Flor's HTR models. These RNN models were first quantitatively evaluated against each other to select the best performing one. According to experiments, the Flor's HTR model demonstrated the highest recognition rate overall.

The authors of this paper took a further step to improve Flor's HTR model, which resulted in a new model, called Attention-based Fully Gated CNN-BGRU model. This model showed **BLABLA**% as CAR, and **BLABAL**% as WAR.

Generally, all models examined in this research proved that there is need for more data, especially containing Kazakh language words. As a future work, a Telegram bot was created to collect a new dataset with predominantly Kazakh language words.

Currently, the handwriting recognition model developed by this research is not ready to use it in a production level, like in a postal company. Development of a web application which enables easy-to-use interface for users are still being developed.

Acknowledgement

This research work was funded by the Ministry of Education and Science of the Republic of Kazakhstan (Grant No AP05135175).

References

[1] Google Inc. Google input tools. https://www.google.com/inputtools/try/. Last accessed 27 March 2020.

[2] Paragon Software Group. Penreader. http://slovoed.com/ru/projects/penreader. Last accessed 27 March 2020.

[3] Niranjan Joshi, G Sita, AG Ramakrishnan, and Sriganesh Madhvanath. Comparison of elastic matching algorithms for online tamil handwritten character recognition. In *Ninth international workshop on frontiers in handwriting recognition*, pages 444–449. IEEE, 2004.

[4] James Arvo and Kevin Novins. Fluid sketches: continuous recognition and morphing of simple hand-drawn shapes. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 73–80, 2000.

[5] Alex Graves, Marcus Liwicki, Horst Bunke, Jürgen Schmidhuber, and Santiago Fernández. Unconstrained on-line handwriting recognition with recurrent neural networks. In *Advances in neural information processing systems*, pages 577–584, 2008.

[6] James A Pittman. Handwriting recognition: Tablet pc text input. *Computer*, 40(9):49–54, 2007.

[7] ABBYY. "finereader". https://finereaderonline.com/ru-ru. Last accessed 27 March 2020.

[8] ABBYY. "intelligent character recognition". https://abbyy.technology/en:%20features:ocr:icr. Last accessed 27 March 2020.

[9] ParaScript. "addressparcel. address recognition on parcels of all shapes, sizes and types". https://www.parascript.com/addressparcel/. Last accessed 27 March 2020.

[10] Horst Bunke, Markus Roth, and Ernst Günter Schukat-Talamazzini. Off-line cursive handwriting recognition using hidden markov models. *Pattern recognition*, 28(9):1399–1413, 1995.

[11] Amlan Kundu and Paramrir Bahl. Recognition of handwritten script: a hidden markov model based approach. In *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing*, pages 928–931. IEEE, 1988.

[12] Amlan Kundu, Yang He, and Paramvir Bahl. Recognition of handwritten word: first and second order hidden markov model based approach. *Pattern recognition*, 22(3):283–297, 1989.

[13] A El-Yacoubi, Michel Gilloux, Robert Sabourin, and Ching Y. Suen. An hmm-based approach for off-line unconstrained handwritten word modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):752–760, 1999.

[14] BS Saritha and S Hemanth. An efficient hidden markov model for offline handwritten numeral recognition. *arXiv preprint arXiv:1001.5334*, 2010.

[15] Azizah Suliman. Hybrid of hmm and fuzzy logic for isolated handwritten character recognition. *Character Recognition*, pages 59–82, 2010.

[16] Alejandro H Toselli and Enrique Vidal. Handwritten text recognition results on the bentham collection with improved classical n-gram-hmm methods. In *Proceedings of the 3rd International Workshop on Historical Document Imaging and Processing*, pages 15–22, 2015.

[17] Aleksander Kolcz, Joshua Alspector, Marijke Augusteijn, Robert Carlson, and Gheorghe Popescu. Visual keyword-based word spotting in handwritten documents. In *Document Recognition V*, volume 3305, pages 185–193. International Society for Optics and Photonics, 1998.

[18] Aleksander Kolcz, Joshua Alspector, M Augusteijn, R Carlson, and G Viorel Popescu. A line-oriented approach to word spotting in handwritten documents. *Pattern Analysis & Applications*, 3(2):153–168, 2000.

[19] Ahmad AbdulKader. A two-tier arabic offline handwriting recognition based on conditional joining rules. In *Summit on Arabic and Chinese Handwriting Recognition*, pages 70–81. Springer, 2006.

[20] Khalid Saeed and Majida Albakoor. Region growing based segmentation algorithm for typewritten and handwritten text recognition. *Applied Soft Computing*, 9(2):608–617, 2009.

[21] Basilios Gatos, Kostas Ntzios, Ioannis Pratikakis, Sergios Petridis, Thomas Konidaris, and Stavros J Perantonis. An efficient segmentation-free approach to assist old greek handwritten manuscript ocr. *Pattern analysis and applications*, 8(4):305–320, 2006.

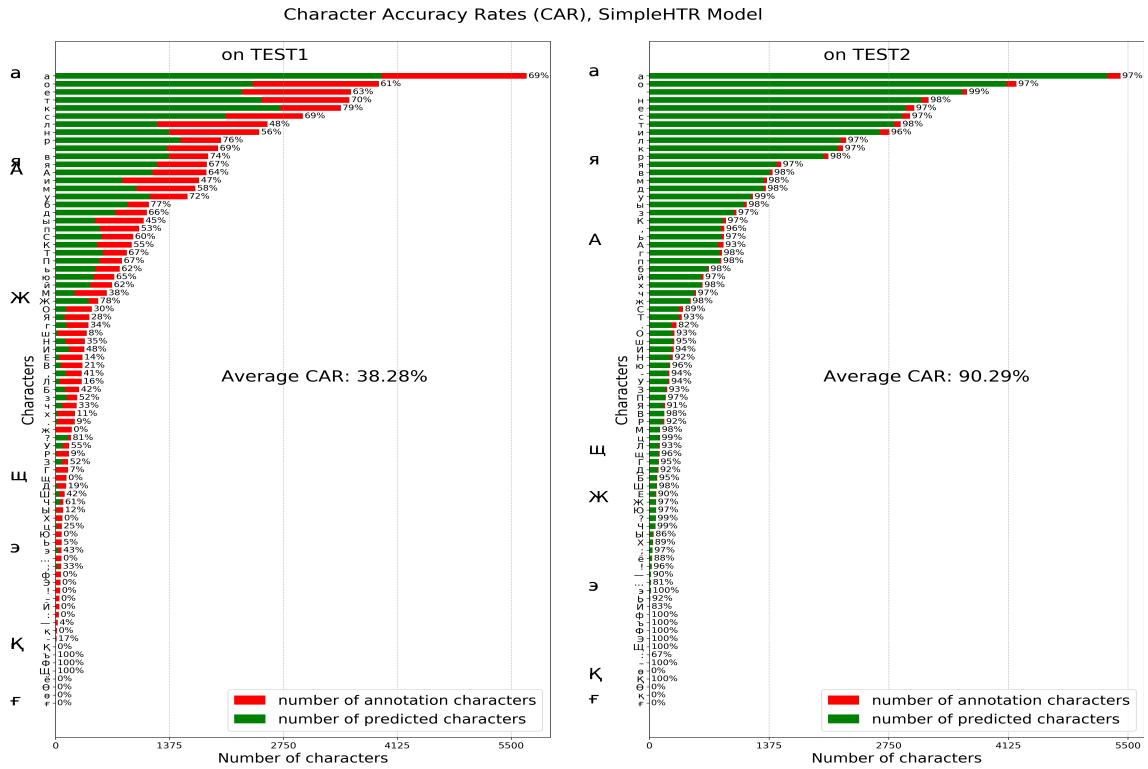

Figure 6. SimpleHTR model performance on TEST1 and TEST2 datasets

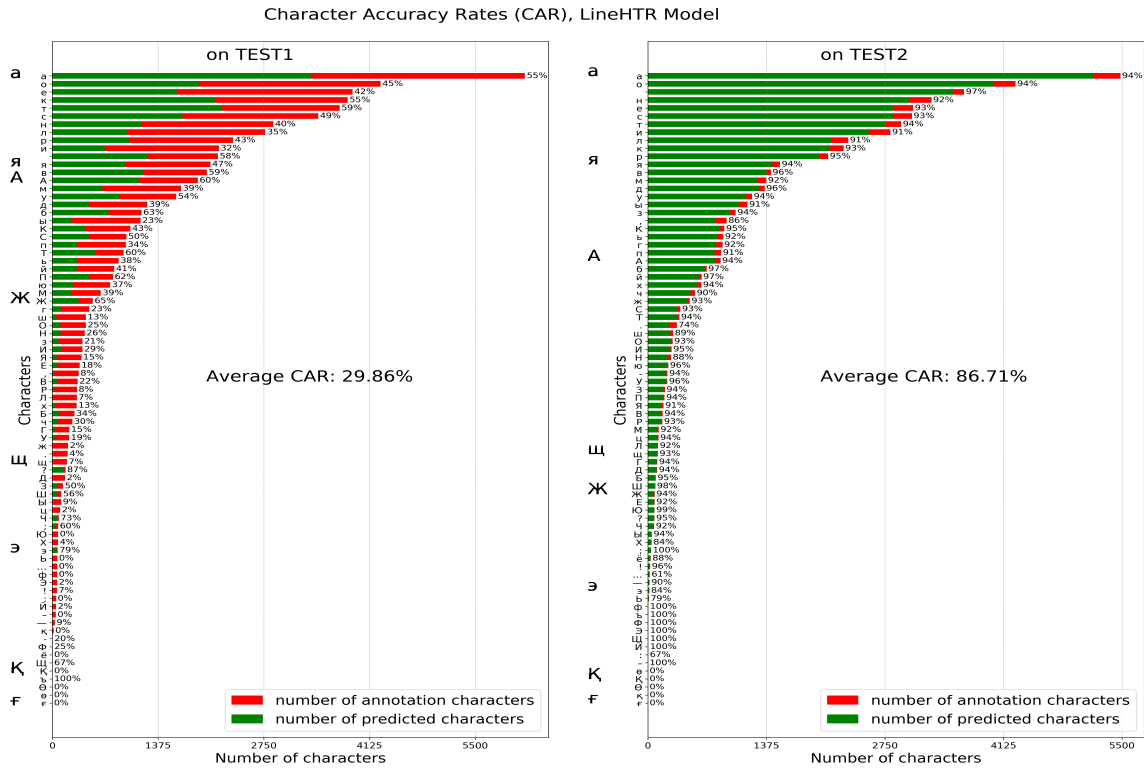

Figure 7. LineHTR model performance on TEST1 and TEST2 datasets

[22] Ray Smith. An overview of the tesseract ocr engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE, 2007.

[23] Sandip Rakshit and Subhadip Basu. Development of a multi-user

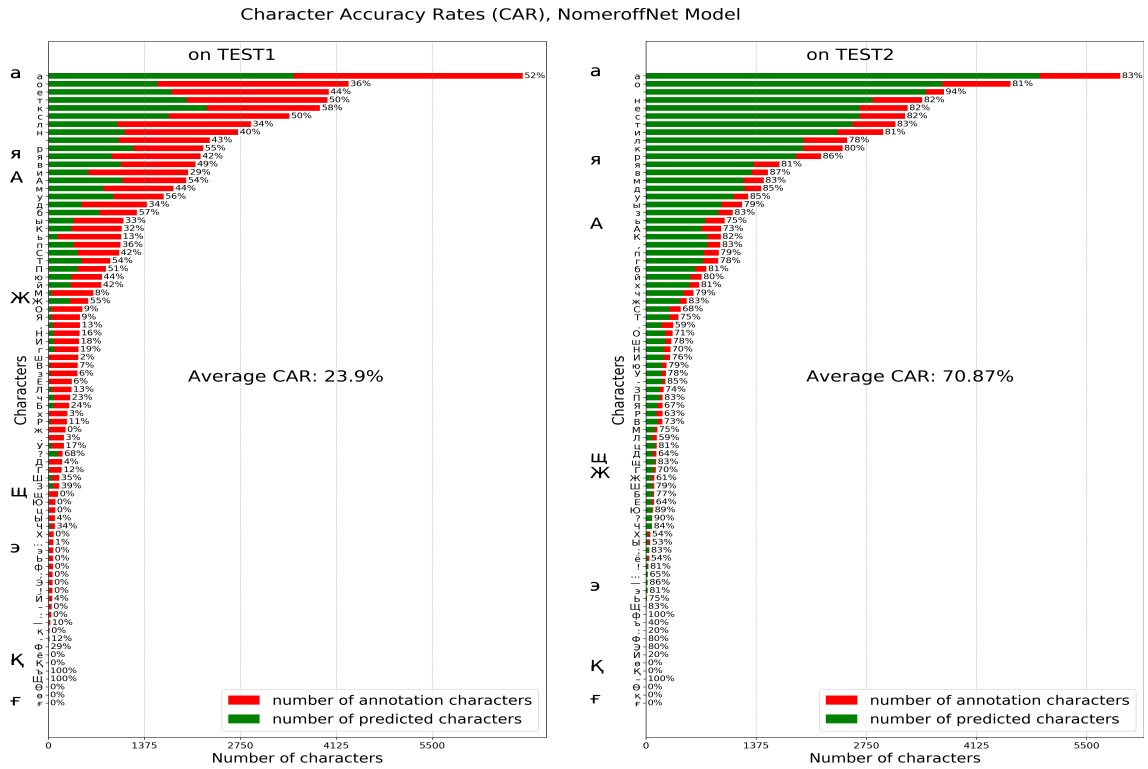

Figure 8. NomeroffNet HTR model performance on TEST1 and TEST2 datasets

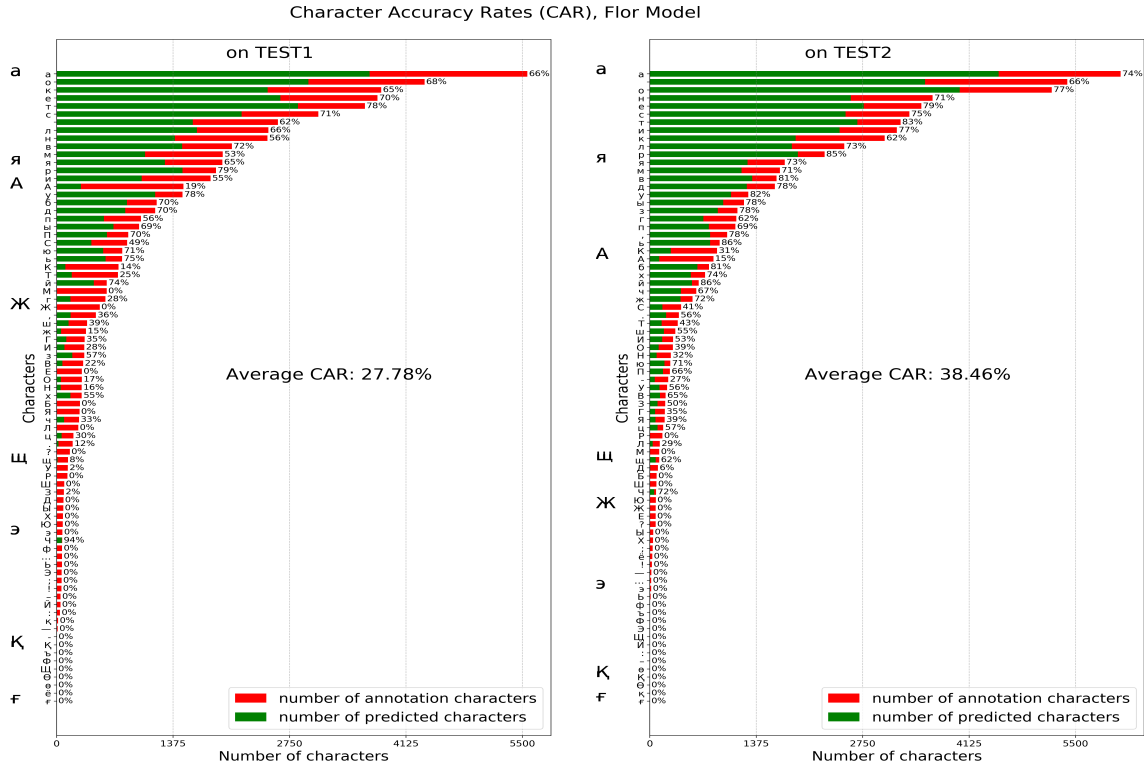

Figure 9. Flor's model (original) performance on TEST1 and TEST2 datasets

handwriting recognition system using tesseract open source ocr engine. *arXiv preprint arXiv:1003.5886*, 2010.

[24] Brian Gonzalez. Iris: a solution for executing handwritten code. Master's thesis, Universitetet i Agder/University of Agder, 2012.

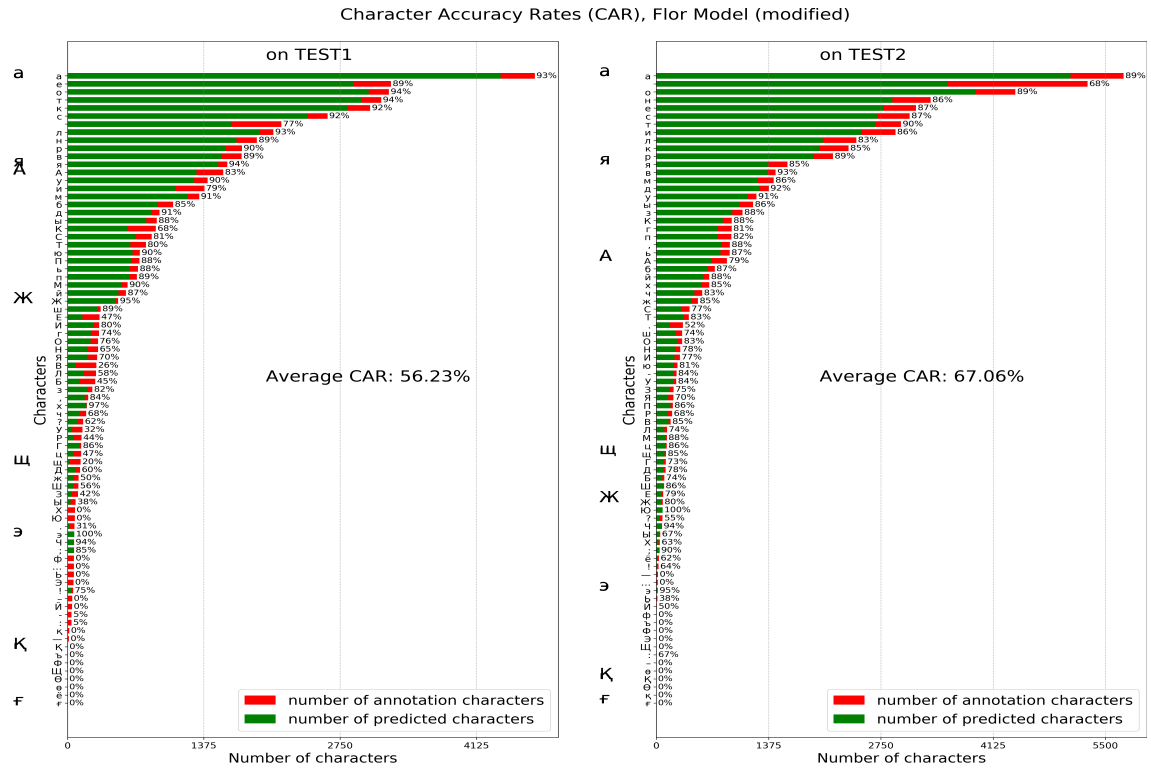

Figure 10. Attention-based Fully Gated CNN-BGRU model performance on TEST1 and TEST2 datasets

Table I
QUANTITATIVE EVALUATION OF MODELS

Model name	TEST1			TEST2		
	WAR,%	CAR (Levenshtein),%	CAR (our),%	WAR,%	CAR (Levenshtein),%	CAR (our),%
SimpleHTR	CAR	CAR	CAR	CAR	CAR	CAR
LineHTR	CAR	CAR	CAR	CAR	CAR	CAR
NomeroffNet	CAR	CAR	CAR	CAR	CAR	CAR
Flor	CAR	CAR	CAR	CAR	CAR	CAR
Attention-based Fully Gated CNN-BGRU	CAR	CAR	CAR	CAR	CAR	CAR

[25] Ray Smith. Training lstm networks on 100 languages and test results. https://github.com/tesseract-ocr/docs/blob/master/das$tutorial$2016/7$Building$%20a%20$Multi-Lingual$%20OCR%20$Engine.pdf$, 2016. $Accessed 11 May$ 2020.

[26] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[27] Andres Calderon, Sergio Roa, and Jorge Victorino. Handwritten digit recognition using convolutional neural networks and gabor filters. *Proc. Int. Congr. Comput. Intell*, 2003.

[28] Haider A Alwzwazy, Hayder M Albehadili, Younes S Alwan, and Naz E Islam. Handwritten digit recognition using convolutional neural networks. *Int. J. Innov. Res. Comput. Commun. Eng*, 4(2):1101–1106, 2016.

[29] Ahmed El-Sawy, Mohamed Loey, and EB Hazem. Arabic handwritten characters recognition using convolutional neural network. *WSEAS Transactions on Computer Research*, 5:11–19, 2017.

[30] Batuhan Balci, Dan Saadati, and Dan Shiferaw. Handwritten text recognition using deep learning. *CS231n: Convolutional Neural Networks for Visual Recognition, Stanford University, Course Project Report, Spring*, 2017.

[31] R Reeve Ingle, Yasuhisa Fujii, Thomas Deselaers, Jonathan Baccash, and Ashok C Popat. A scalable handwritten text recognition system. *arXiv preprint arXiv:1904.09150*, 2019.

[32] Marcin Namysl and Iuliu Konya. Efficient, lexicon-free ocr using deep learning. *arXiv preprint arXiv:1906.01969*, 2019.

[33] Bappaditya Chakraborty, Partha Sarathi Mukherjee, and Ujjwal Bhattacharya. Bangla online handwriting recognition using recurrent neural network architecture. In *Proceedings of the tenth Indian conference on computer vision, graphics and image processing*, pages 1–8, 2016.

[34] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[35] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868, 2008.

[36] Yaroslav Shkarupa, Roberts Mencis, and Matthia Sabatelli. Offline handwriting recognition using lstm recurrent neural networks. In *The 28th Benelux Conference on Artificial Intelligence, Amsterdam (NL)*, pages 10–11, 2016.

[37] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2016.

[38] Paul Voigtlaender, Patrick Doetsch, and Hermann Ney. Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 228–233. IEEE, 2016.

[39] H Scheidl. Handwritten text recognition in historical documents.

Technische Universität Wien, 2018.

[40] Harald Scheidl. Handwritten text recognition with tensorflow. https://github.com/githubharald/SimpleHTR, 2018. Last accessed 11 May 2020.

[41] Harald Scheidl. Build a handwritten text recognition system using tensorflow. https://towardsdatascience.com/build-a-handwritten-text-recognition-system-using-tensorflow-2326a3487cd5, 2018. Last accessed 11 May 2020.

[42] Hoàng Tùng Lâm. Line-level handwritten text recognition with tensorflow. https://github.com/lamhoangtung/LineHTR, 2018. Last accessed 11 May 2020.

[43] Nomeroff Net. Nomeroff net. automatic numberplate recognition system. version 0.3.1. https://nomeroff.net.ua/. Last accessed 11 May 2020.

[44] ria com. Ocr number plate text recognition. https://github.com/ria-com/nomeroff-net/blob/0.2.0/docs/OCR.md. Last accessed 11 May 2020.

[45] Arthur Flor. Handwritten text recognition using tensorflow 2.0. https://github.com/arthurflor23/handwritten-text-recognition/tree/7e19b3f8c33bd67984cfaa1346b31636808748ce, 2019. Last accessed 11 May 2020.

[46] Arthur Flor. Handwritten text recognition using tensorflow 2.0. https://medium.com/@arthurflor23/handwritten-text-recognition-using-tensorflow-2-0-f4352b7afe16, 2019. Last accessed 11 May 2020.

[47] Joan Puigcerver. Are multidimensional recurrent layers really necessary for handwritten text recognition? In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 67–72. IEEE, 2017.

[48] Théodore Bluche and Ronaldo Messina. Gated convolutional recurrent neural networks for multilingual handwriting recognition. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 646–651. IEEE, 2017.

[49] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 933–941. JMLR. org, 2017.

[50] Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *Advances in neural information processing systems*, pages 1945–1953, 2017.

[51] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[52] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[53] Kyuyeon Hwang and Wonyong Sung. Character-level incremental speech recognition with recurrent neural networks. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5335–5339. IEEE, 2016.

[54] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[55] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[56] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.