**Report**

**Optical Character Recognition (OCR) for Kazakh Language**

Nazarbay Nursultan, Magzhan Omirtay

Git link: <https://github.com/NazarbayND/ai_final_project>

1. **Image preprocessing**

To solve this problem we have generated data by writing a python script for obtaining Kazakh letters and saved them as a given letter. The script receives a scanned or photographed Kazakh text using PIL and defines words using the pytesseract library. Initially, we tried to get the coordinates of the letters using the pytesseract itself, but it does not show the correct coordinates, although it recognizes the words correctly. For our script to work correctly, we decided to use cv2 library and processed our image with a threshold function. Then we found letter contours using the findContours function, but the contour does not understand the letters and selects all objects in the images.

Graphical user interface, text, application

Description automatically generated

Picture 1 – code for converting image to grayscale

Graphical user interface, text, application

Description automatically generated

Picture 2 – result of converting

Text

Description automatically generated with medium confidence

Picture 3 – code for splitting the words

Graphical user interface, application, table, Excel

Description automatically generated

Picture 4 – result of splitting the words

Graphical user interface, text

Description automatically generated with medium confidence

Picture 5 – code for splitting words into letters

While processing our images, we ran into a problem with the letters "ы"and "й". Because our script split them into two parts and considered them to be different letters as in the Picture I (b). To solve this problem, we merged two adjacent images where these letters met. For this, additional functions were created: horizontal union for "ы"and vertical union for "й". Consequently script separates the words into letters for every 42 Kazakh letters. Further we saved them as a separate image as in the Picture 6. and this process occurred until we collect enough data for data augmentation. After this process, we increase the amount of data to improve model accuracy.

A picture containing histogram

Description automatically generated

1. given word

Text

Description automatically generated

1. initial separation with errors

Diagram

Description automatically generated

1. result after script

Picture 6 - Splitting words into letters

1. **Image data generation**

To get more data, we needed to make minor alterations to our existing dataset. Minor changes such as flips or translations or rotations. For this we have used data augmentation: a technique to increase the diversity of the training set by applying random (but realistic) transformations. Applying these small amounts of variations on the original image does not change its target class but only provides a new perspective of capturing the object in real life. It is quite often used for building deep learning models.

In our work we used several techniques of data augmentation such as Keras ImageDataGenerator, Albumentations Augmentations, ‘imgaug‘ Based Augmentations, TensorFlow-Based Augmentations.

Keras ImageDataGenerator 1 class provides a quick and easy way to augment the images. It provides a host of different augmentation techniques like standardization, rotation, shifts, flips, brightness change, and many more. However, the main benefit of using the Keras ImageDataGenerator class is that it is designed to provide real-time data augmentation. Meaning it is generating augmented images on the fly while the model is still in the training stage.

Text, application

Description automatically generated

Picture 7 – code for ImageDataGenerator augmentation

Shape, polygon

Description automatically generated

Picture 8 - ImageDataGenerator results

Albumentations 2 is a Python library for fast and flexible image augmentations. Albumentations efficiently implements a rich variety of image transform operations that are optimized for performance, and does so while providing a concise, yet powerful image augmentation interface for different computer vision tasks, including object classification, segmentation, and detection.

Graphical user interface, text, application

Description automatically generated

Picture 9 – Albumentations Augmentations

Shape

Description automatically generated

Picture 10 – result of Albumentations Augmentations

imgaug 3 is a library for image augmentation in machine learning experiments. It supports a wide range of augmentation techniques, allows to easily combine these and to execute them in random order or on multiple CPU cores, has a simple yet powerful stochastic interface and can not only augment images, but also keypoints/landmarks, bounding boxes, heatmaps and segmentation maps.

Graphical user interface, text, application

Description automatically generated

Picture 11 – code for imgaug based augmentations

Shape, polygon

Description automatically generated

Picture 12 – result of imgaug based augmentations

In TensorFlow 4 , data augmentation is accomplished using the ImageDataGenerator class. It is exceedingly simple to understand and to use. The entire dataset is looped over in each epoch, and the images in the dataset are transformed as per the options and values selected.

Text

Description automatically generated

Picture 13 – code for TensorFlow-Based Augmentations

Shape, polygon

Description automatically generated

Picture 14 - TensorFlow-Based Augmentations result

1. **Model Training**

In literature review we found different research approaches in Optical text recognition for many languages but besides the Kazakh language. Including this fact, we made a decision to create our methodology for Kazakh Language in OCR. Our idea is separate every row into words, then these words to characters. After partition steps we recognize every character in trained model on VGG16 architectures. This architecture integrated and trained on every character especially 42 kazakh letters both in uppercase and lowercase. After we trained a model, we estimated performance of our architecture on these metrics such as Accuracy, Mis-Classification, Sensitivity, Specificity, Precision, F1 Score, MCC.

Graphical user interface, text, application, email

Description automatically generated

Picture 15 – Importing necessary libraries

Firstly, we imported all necessary libraries for training our model and all necessary packages.

Graphical user interface, application

Description automatically generated

Picture 16 – Classification of letters

Then, we have classified Kazakh letters to classes. As a result, we came with 84 letter classes for 42 lowercase letters and 42 uppercase letters.

Text

Description automatically generated

Picture 17 – processing image

Then we processed images and saved as a matrix.

Graphical user interface, text, application, email

Description automatically generated

Picture 18 – splitting data for train and test cases

After we processed images we have split the data for train case and test case. And created two arrays, first is for saving image matrix and second one for storing result class of this matrix.

Graphical user interface, text, application

Description automatically generated

Picture 19 – main part

After we separated our data, we started to train our model on that data using transfer learning technique.

Graphical user interface, table

Description automatically generated

Picture 20 – VGG16 model

Table

Description automatically generated

Picture 21 – VGG16 model

The Visual Geometry Group (VGG) was the runner up of ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2014. The main contribution of this model shows that the depth (i.e., number of layers) of a convolutional neural network is the critical component of high recognition or classification accuracy. In this architecture, two convolutional layers are used consecutively with a rectified linear unit (ReLU) activation function followed by single max-pooling layer, several fully connected layers with ReLU and soft-max as the final layer. The 3×3 convolutional filters with stride 2 is applied for performing filtering and sub-sampling operations simultaneously in VGG-E version. There are three types of VGGNet based on the architecture. These three network contains 11, 16 and 19 layers and named as VGG-11, VGG-16 and VGG-19, respectively. There are five convolution layers, three max-pooling layers and three fully connected (FC) layers. The configuration of VGG16 is as follows: number of convolution layers: 16, fully connected layers: 3, weights: 138 Million and Multiplication and Accumulates (MACs): 15.5G. In this architecture, the multiple convolutional layers are incorporated which is followed by a max-pooling layer. In this implementation, we have used VGG16 with less number of feature maps in each convolutional block compared to standard VFF16 model.

In our model we used VGG16 architecture and the result were very high.

1. **Results**

A picture containing calendar

Description automatically generated

Picture 22 – Results of Precision, Recall, F1 Score of 84 letter classes

Graphical user interface

Description automatically generated

Picture 23 – Loss curve on learning

Graphical user interface, text, application

Description automatically generated

Picture 24 – Accuracy curve on learning

This is learning curve of our model. As we can clearly seen from the diagrams that learning curves are very good trained and Training and Validation curves aren’t differ so much

A picture containing application

Description automatically generated

Picture 25 – Final results

In order to evaluate the accuracy of the models performance we used metrics such as accuracy, sensitivity/recall, specificity, precision, F1 score, MCC. A VGG16 architecture gave us high performance results.. The overall results can be seen in the table above.

**Conclusion**

In this project Kazakh recognition system has been developed with its own Kazakh letters dataset created by data augmentation method. We generated dataset of Kazakh letters for training and trained and tested network for OCR. We hope that our work will help many people in retrieving information from text image in Kazakh language. Like students who study in Kazakh and in many cases they have to retype some text from image format to editable text. So, in the future we can make user-friendly interface application based on this work.