Baltic J. Modern Computing, Vol. 13 (2025), No. 2, 419-435
https://doi.org/10.22364/bjmc.2025.13.2.07

Optical Character Recognition Tool for Georgian Handwritten Text Recognition Based on YOLOv8

Ana CHIKASHUA[1], Maia ARCHUADZE[2], Magda TSINTSADZE[2], Manana KHACHIDZE[2]

[1] Industrial Analytics IA GmbH, Berlin, Germany
[2] Department of Computer Sciences, Iv. Javakhishvili Tbilisi State University, Georgia

ana.chikashua5118@ens.tsu.edu.ge, maia.archuadze@tsu.ge, magda.tsintsadze@tsu.ge, manana.khachidze@tsu.ge

ORCID 0009-0008-9930-9249, ORCID 0000-0002-9484-1016, ORCID 0000-0001-9316-3295, ORCID 0000-0003-4545-2679

Abstract. Optical character recognition is one of the important tasks in the fields of artificial intelligence, computer vision, natural language processing tasks, and machine learning. It remains a major challenge for less common languages such as Georgian. In addition, we face the complexity and variety of writing forms characteristic of this language, caused by the integration of three different scripts implemented over time.

Discussed in the presented work are three models: convolutional neural networks (CNN), ResNet, and Yolov8, through which the OCR system was created. The methodology presented in this paper is the first attempt to create for the Georgian language an OCR system based on the Yolov8 platform. The paper analyzes the results obtained by all three methods, based on which we conclude that the accuracy of YOLOv8 compared to CNN and ResNet was improved.

After training and evaluating the aforementioned models on the training dataset, it was determined that the YOLOv8 model yielded the best results. The CNN model, incorporating convolutional, pooling, and dropout layers, achieved an accuracy of 0.9230, whereas the ResNet model achieved an accuracy of 0.9302. In contrast, YOLOv8 achieved an accuracy of 0.98848, with a loss value of 0.04479. These results indicate that YOLOv8 performed exceptionally well during live testing. Furthermore, it is noteworthy that during the testing phase, YOLOv8 demonstrated remarkable speed in generating results.

Keywords: OCR, YOLOv8, CNN, ResNet

1. Introduction

In our view, the low accuracy of recognizing Georgian texts by OCR systems is accounted for by several factors. One of them is the uniqueness of the language itself, as its script is a combination of three different scripts: **Asomtavruli**, **Nuskhuri,** and **Mkhedruli**, each with a graphic style of its own. This leads to challenges with variations in handwritten styles. Due to this peculiarity, the same symbol can be tilted, rotated, or presented in

different forms. Some letters may have similar outlines in handwritten text, making it difficult to distinguish them. For instance, "ღ" and "დ," "ღ" and "ლ," "უ" and "ყ," "ვ" and "გ." Additionally, the position of symbols is crucial, as in Georgian, symbols are written above and below the line, each with different meanings. Variations in different fonts further affect the accuracy of the recognition system.

2. Experimental

In addition to notable commercial OCR systems such as Azure OCR, Amazon Textract, and Google OCR, various open-source OCR tools are available today. Whereas this technology is the latest, there are still no OCR products that can recognize all types of text with 100% accuracy.

Pytesseract is a Python package that simplifies the use of Tesseract. It can read all image types supported by the Pillow and Leptonica visualization libraries, including jpeg, png, gif, bmp, tiff, and other libraries (Shubham 2020). EasyOCR can currently recognize text in more than 80 languages, including English, German, Hindi, Russian, and others. (Rosebrock 2020). It is worth noting that most open-source OCR systems do not support the Georgian language at all, (for example, ABBYY FineReader/ABBYY Cloud OCR SDK and OnlineOCR.net.). Based on our experiments, we can conclude that those systems that do support the Georgian language have a low recognition accuracy.

Below are the results of an experiment on a Georgian text for those OCRs that support the Georgian language. Google Cloud Vision's output for a specific example in the Georgian language demonstrates some issues, such as missing characters or incorrect character recognition.

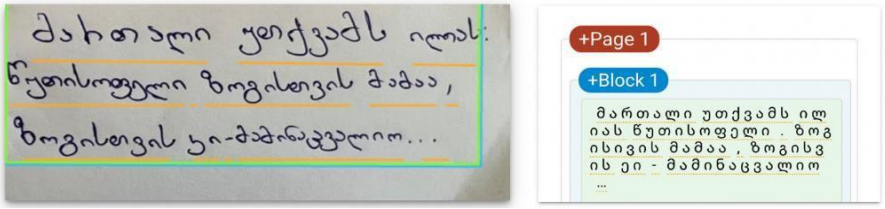


Figure 1. Example of Google Cloud Vision

Tesseract OCR: In this case, the structure of the text and summary are unclear.

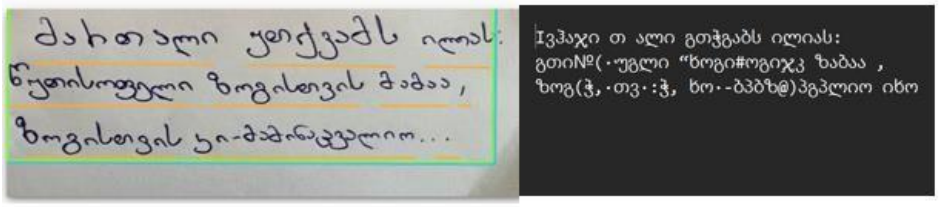


Figure 2. Example of Tesseract OCR

The result of using i2OCR in this case contains several errors.

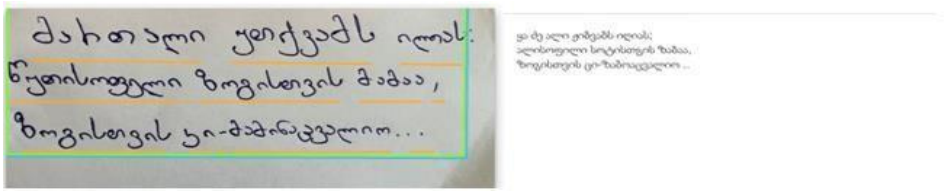


Figure 3. Example of i2OCR

Georgian language is also supported by quite a popular software Nebo (Microsoft 2023) which is a whiteboard-type application available on Android, iOS, and web platforms. It offers remarkable speed and accuracy in calculations. However, Nebo requires that the entries be created directly on the screen of your device. It is important to note that Nebo uses ICR (Intelligent Character Recognition), not OCR (Optical Character Recognition), which typically converts existing text entries into an editable format.

In addition, we ought to mention that scientific studies have implementation regarding ~~to~~ this task for the Georgian language using a convolutional neural network VGG 16 and GoogleNet architecture trained with 200,000 data (Soselia, et al. 2018). In our research, we use CNN, ResNet, and YOLOv8. YOLOv8 represents the latest version, released in 2023, which has provided us with quite good results. The limitation of the previous research is that it only works on individual characters and doesn't consider word recognition.

3. Methodology

Study Limitations and the Process Description:

It is also important to note that the successful operation of the system is contingent upon several key assumptions outlined in this paper.

The input image is a full, straight, (fairly) standard-sized sheet of paper with visible borders;

The text is written horizontally with non-intersecting lines;

Distance between words should be taken into account;

We should also pay attention to the fact that the symbols stand separately and not linked with one another.

The project is divided into two parts. In the first scenario, the model is directly trained, which comprises the following stages:

- Data gathering and labelling;
- Data processing (to format images for the model's input);
- Data augmentation;
- Data partitioning (into training, validation, and testing datasets);
- Model training, validation, and testing;
- Results analysis;
- Model deployment.

This process results in a model capable of working with various inputs, specifically numbers, symbols, and words.

In the second scenario, the focus shifts to segmenting text into individual symbols. Despite the complexity of each step, as elaborated below, the objective is to exclusively provide our trained model with characters it can accurately interpret. Below we consider each of the steps in detail.

3.1. Data Gathering and Labeling

We started the research process by collecting data. We create template that consist of Georgian characters and asked colleges, students, friends to fill this and then create database based on this data. The main idea was to have already labelled characters as many as we can. Working with this model required a database of Georgian characters. This database is a certain table filled by different people. The base consists of 48625 characters.

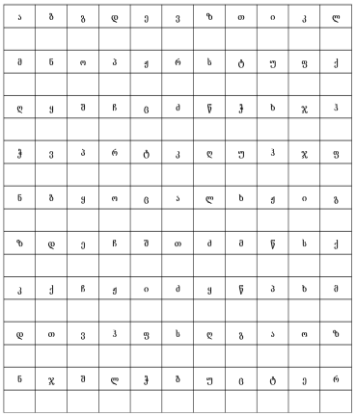


Figure 4. Data collection table

The MNIST (Modified National Institute of Standards and Technology) database, which consists of more than 70,000 scrambled images of numbers, was used to recognize the numbers.

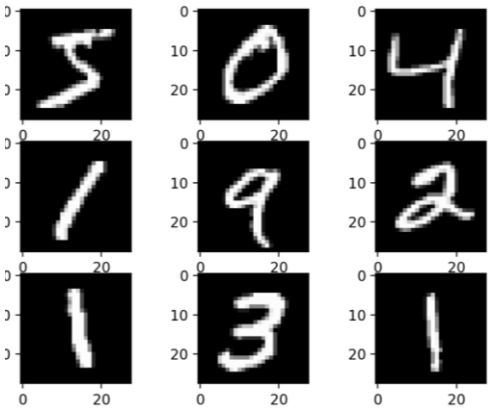


Figure 5. The MNIST database

Data for two models (ResNet and CNN) were taken in the following ratio: 80% for training, 10% for validation, and 10% for testing.

3.2. Data processing

The above-mentioned pre-processing system takes an image of a complete handwritten page and returns images of words. Images of individual words are crucial because they can be fed into Handwritten Text Recognition (HTR) systems.

Preprocessing consists of the following stages:

1. Initial image processing;
2. Edge Removing;
3. Line Removing;
4. Gray-scaling and blurring;
5. Segmentation Edges(Ganny);
6. Filter Connected Components;
7. Segmentation (GMM-based text lines detection);
8. Selection of clusters; 10. Segmentation into words.

Let's consider each of them in detail.

3.2.1. Initial image processing

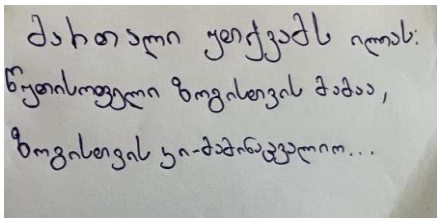


Figure 6 . Original

Ideally, the image should encompass the entire page with all page borders visible. The page should also be as straight as possible and well-lit. 'Well-lit' means that the image is neither too dark nor too bright.

Initially, image processing is required, in order to get the character of a text which would be an input for a model. It is more like an image processing and is the same for each language. This step which includes tasks such as edge detection, grayscale conversion, noise reduction, and the retention of only the components necessary for recognition. These steps are crucial for enhancing the quality of the input data and improving the accuracy of subsequent text recognition. Here's a detailed approach tailored for Georgian handwritten text.

3.2.2. Edge Removing

The first step in preprocessing is to find the page contour, outline the page (leaving only the page). This can be done quite easily by using cv2.findContours (Rosebrock 2020)] and cv2.approxPolyDP to find the largest contour whose approximate shape is a square. cv2.approxPolyDP is a function of the OpenCV library used to smooth the contour by reducing the number of points while preserving the overall shape of the contour. This reduces computational complexity and improves efficiency. The purpose of approxPolyDP is to replace a polygon with another simpler (less squiggly) polygon.

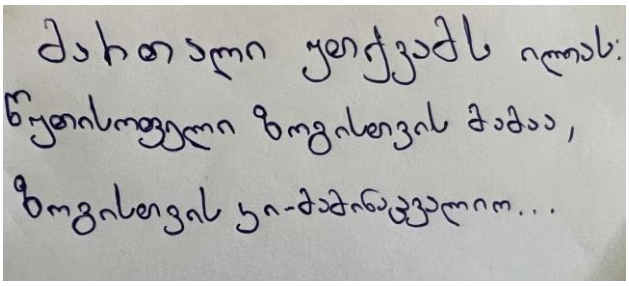


Figure 7. Edge removed

3.2.3. Line Removing

Removing lines on a drawn page is crucial because these lines can meet each other. In related components (these components are discussed below)) they cannot be filtered off. The mentioned methodology uses hough lines and Fourier transform to remove the lines. Doing this several times improves the result of cv2.canny. Therefore, it is necessary to remove them at the beginning. This method does not completely remove the lines, but significantly cuts the line components of the page. (Although in the example we discussed the line sheet is not used, it is still useful, because it gives us quite good results for removing noise).

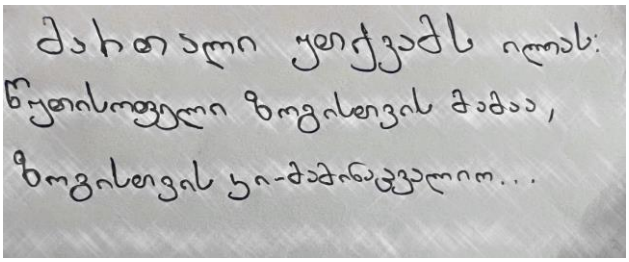


Figure 8. Line removing

3.2.4. Gray-scaling and blurring

The image then becomes gray and blurry for cv2.canny. For blurring we used Gaussian blurring (cv2.GaussianBlur) which in terms of image processing, smoothed out any sharp edges in images and excessive blurring is minimized. In (Microsoft 2023) Gaussian smoothing is used to remove noise that approximately follows a Gaussian distribution.

The end result is a less blurry image, but more "naturally blurry" than any other image that we could get by using other methods.

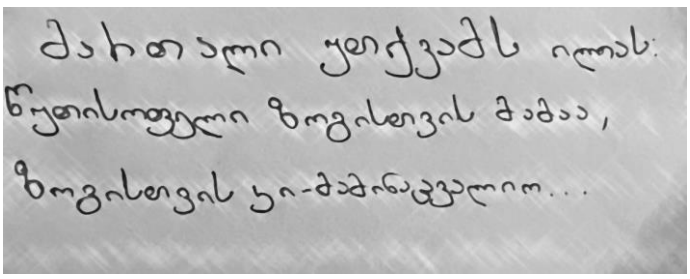


Figure 9. Gray and Blurred

3.2.5. Segmentation Edges (Canny)

This process allows us to localize the text, and we get sharp contours around the text. To get edges of characters we used cv2.canny. This is a good way to detect text in an image.

Invariant parameters are used here to detect all text - any text that is not detected will not make it to the final result, so detecting everything, even noise, is fine. As a result:

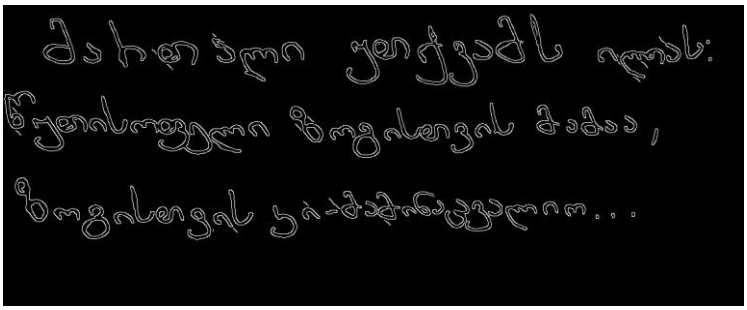


Figure 10. Segmentation Edges

3.2.6. Filter Connected Components

In image processing, connected components refer to distinct regions or sets of pixels in a binary image that are related to each other based on some criterion. These connected components can represent individual objects, regions, or parts of an image.

To find the components, we used the OpenCV function connected Components With Stats, the ConnectComponentsWithStats function being a part of OpenCV, a popular computer vision library.

Area filtering: Area filtering is a common technique used to filter connected components (regions) in an image based on their area, which refers to the number of pixels in each component. The idea is to keep components that meet a specific size criterion and remove those that are too small or too large.

Filters are based on the height, width, and area of the resulting images. Filtering is performed according to different components;

Filtering the bounds of connected components is based on their dimensions. After this type of filtering, only the connected components that are useful for defining or separating text lines remain;

Filtering of stand-alone 'stray' components: This process once again filters connected components, this time removing 'stray' components along the y-axis in addition to removing lines.

Although simple and unlikely to remove many components, this filtering is crucial for detecting text lines.

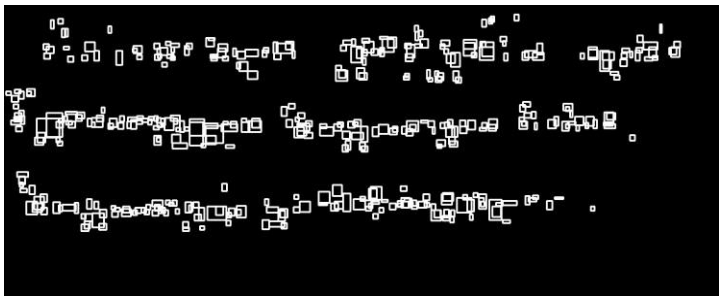


Figure 11. Components with bounding boxes

3.2.7. Segmentation (GMM-based text lines detection)

We used the sklearn.mixture.GaussianMixture function to see how many horizontal lines of the text are there and where there are - this is the assumption of "Horizontal lines". Horizontal lines are not part of the textual content and can be ignored during the recognition process. This helps improve the accuracy of OCR results and prevents horizontal lines from being mistakenly converted to text. Through using it, text lines were grouped according to their y values.

The Gaussian Mixture Model (GMM) is a probabilistic model used to estimate clustering and distribution. It is a generative model which assumes that the data points are generated from several Gaussian distributions. Each Gaussian distribution represents such as mean and covariance, that best explain the data observed.

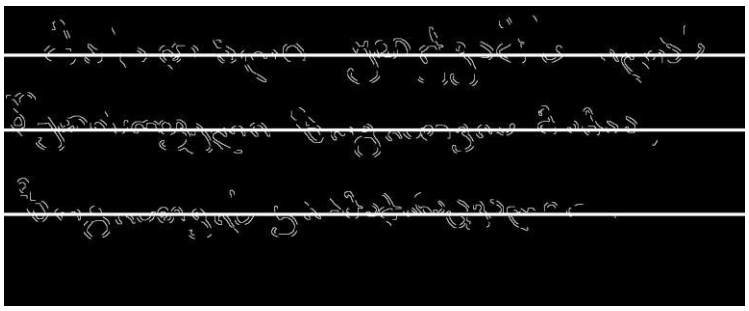


Figure 12. "Horizontal Lines"

3.2.8. Selection of Clusters

To select the number of clusters, we used an approach similar to the elbow method with a lower bound (error) reduction. It is a technique used to determine the optimal number of clusters in a clustering algorithm. The idea of the elbow method is to plot the within-cluster sum of squares (WCSS) for different values of k (number of clusters) and identify the "elbow point" on the graph.

In Gaussian Mixture Model (GMM) clustering, the lower bound is the log-likelihood lower bound (also known as the evidence lower bound (ELBO) in variational inference). This is a value that:

1. Measures how well the model fits the data given a certain number of components.
2. Is unitless, but it's in the log-probability space, typically representing the log-likelihood of the observed data under the GMM model.
   X-axis: Number of GMM components (clusters).
   Y-axis: Lower bound (log-likelihood).

A higher (less negative) lower bound means a better fit. The sharp increase from 1 to - 3 components shows that the model fits the data much better as more clusters are added. After 3 or 4 components, improvements are marginal, so adding more clusters doesn't significantly improve the model.

We determined the optimal number of clusters, 3 in our case.

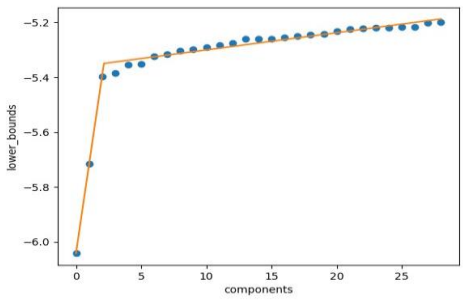


Figure 13. Selection of Clusters

3.2.9. Segmentation into words

One of the main challenges was segmentation into words. After dividing the document into lines, it must be further divided into words. To accomplish this, we find the minimum spacing interval between two words in proportion to the width of the page and the words themselves. This function estimates the minimum spacing threshold that can be used to distinguish individual words in a line of text. It analyzes the horizontal gaps between characters or elements on a line and compares them relative to the page width. By calculating the expected number of words per line based on line width proportions and filtering out lines with insufficient data, it derives an average of the largest spacing values—interpreted as spaces between words. This threshold is crucial for accurately segmenting text into words during document layout analysis. Based on this value, we segment the sentence into individual words. This process entails two main challenges: determining the locations of the separating spaces and obtaining the final word images that can serve as input for Handwriting Text Recognition (HTR).

3.3. Data Augmentation

During the training, we lacked data, which naturally had a significant impact on the accuracy of the model. CNN, ResNet50, required data augmentation using the following techniques:

Morphological changes, its implementation was necessary to correct text lines. Adding noise, which involves removing black pixels or adding white pixels to the image. In the process of research, various methods were used, and in the case of a specific method, we got the following results: RandomRain with a black drop greatly damages the image and makes it more difficult to distinguish. So we tried to add less of this type of augmentation. RandomShadow will blur text with lines of varying intensity. PixelDropout turns random pixels black.

Rotate- ShiftScaleRotate: This parameter is quite problematic. We tried to avoid cutting the text from the initial form. During this process both shape changes and rotation takes place. We paid attention to the fact that the image obtained as a result of rotation corresponds to the predetermined dimensions, which the input image of the model is supposed to have.

Image blurring - to lighten the dark pixels in the image.

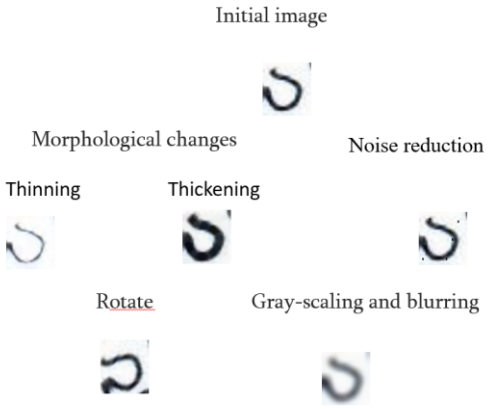


Figure 14. Data Augmentation

As for YOLOv8, we needed no augmentation process because it performs this process by itself and uses Mosaic Data Augmentation for data augmentation. Mosaic Data Augmentation is a simple data augmentation technique in which four images are combined and transferred to the model. This forces the model to learn real objects from different positions.

It is important to note that different models in this study were trained using their respective standard data augmentation strategies. The CNN and ResNet50 models were trained on synthetically augmented datasets, while YOLOv8 utilized its built-in Mosaic Data Augmentation. Although the augmentation methods differ, our aim was not to perform a strictly controlled comparison of architectures under identical input conditions

but rather to evaluate each model in a practical, real-world context using its native training pipeline. YOLOv8 is designed to work seamlessly with Mosaic augmentation, which is considered part of its core strength. Therefore, we argue that the variation in augmentation does not invalidate the comparison but rather reflects realistic usage scenarios where each model is used as intended by its designers. The performance differences observed are still meaningful, as they showcase how each model performs with optimal or standard preprocessing strategies.

3.4. Model Training/ Validate/ Testing

Our research aimed to develop an OCR system for the Georgian language that could provide improved results compared to the previously discussed systems. To achieve this, we employed a new model, distinct from the existing ones, marking the first attempt to create such a system not only for Georgian but also for other languages. Consequently, the research, conducted concurrently with other models, was also interesting and applicable in assessing the model's effectiveness in similar tasks.

We trained three models: CNN, ResNet, and YOLOv8.

The convolutional neural network (CNN) is constructed from several convolutional layers, dropout, and max-pooling (Nielsen 2019). We utilized sparse_categorical_crossentropy for the loss function and Adam for optimization.

Additionally, we experimented with ResNet50, which is a 50-layer neural network (Kaiming, et al. 2015), and applied it to the aforementioned Georgian manuscript data. After 30 iterations, we obtained the following results for both models:

Table 1. Results for CNN and ResNet

Evaluation/Models	CNN	ResNet
Loss:	0.2507	0.2456
Accuracy:	0.923	0.9302
Val_loss:	0.9264	0.6916
Val_accuracy:	0.8018	0.8391

YOLOv8 is the latest version of the YOLO models, officially released on January 10, 2023 (Shubham, et al. 2022), (Cheng 2020), (Krishnakumar 2023), (Jocher 2024) These models demonstrate high performance compared to their predecessors. Notably, YOLOv8 has shown remarkable success in aerial image detection for Unmanned Aerial Vehicles (UAVs) (Yiting, et al. 2023), (Redmon and Farhadi 2017).

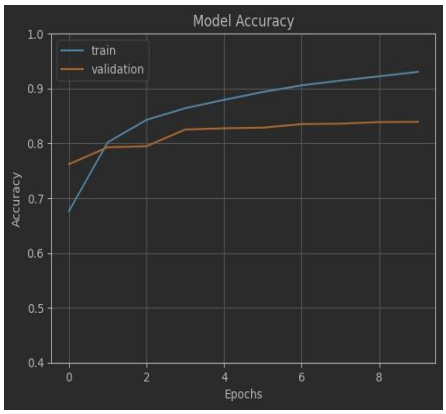
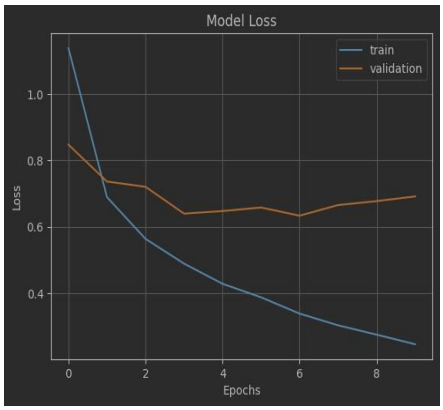


Figure 15. Epochs vs Accuracy (ResNet)



Picture 16. Model Loss (ResNet)

Previous YOLO models have been used for OCR in various languages, such as English, German, French, Latin, and more (Chaudhuri, et al. 2016), (Alghyaline 2022), (Subramanian, et al. 2022), (Farhadi 2018). However, our project marks the first attempt to implement the YOLOv8 model for an OCR system. In our project we employed the 'YOLOv8n-cls' model to analyze the above mentioned data, focusing on solving a classification problem. The model includes an integrated augmentation process, eliminating the need for additional adjustments. After multiple iterations, we obtained model weights, and the best-performing weight can be selected by the user. Additionally, our project generated a CSV file with the following format:

Table 2. Model Weights (csv file)

epoch	train/loss	metrics/accuracy_top1	metrics/accuracy_top5	val/loss	lr/pg0	lr/pg1	lr/pg2
0	0.10073	0.05273	0.22727	0.43404	0.070001	0.0033332	0.0033332
1	0.09822	0.12303	0.34606	0.42814	0.039671	0.0063365	0.0063365
2	0.09025	0.2897	0.68061	0.41418	0.0090113	0.0090099	0.0090099
3	0.07916	0.49455	0.88727	0.39405	0.008515	0.008515	0.008515
4	0.06912	0.58606	0.94667	0.37859	0.008515	0.008515	0.008515
5	0.06345	0.69273	0.96121	0.36818	0.00802	0.00802	0.00802
6	0.05953	0.73697	0.97091	0.3617	0.007525	0.007525	0.007525
7	0.05703	0.75939	0.97879	0.35671	0.00703	0.00703	0.00703
8	0.0549	0.79333	0.98242	0.35381	0.006535	0.006535	0.006535
9	0.05356	0.80364	0.98364	0.3521	0.00604	0.00604	0.00604
10	0.05222	0.81636	0.98606	0.35049	0.005545	0.005545	0.005545
11	0.0512	0.82364	0.98545	0.3487	0.00505	0.00505	0.00505
12	0.05039	0.82848	0.98545	0.34782	0.004555	0.004555	0.004555
13	0.04951	0.83273	0.98485	0.34724	0.00406	0.00406	0.00406
14	0.04835	0.83394	0.98545	0.34676	0.003565	0.003565	0.003565
15	0.04791	0.83576	0.98606	0.34631	0.00307	0.00307	0.00307
16	0.047	0.83939	0.98727	0.3458	0.002575	0.002575	0.002575
17	0.04595	0.84121	0.98788	0.3454	0.00208	0.00208	0.00208
18	0.0452	0.84061	0.98848	0.3451	0.001585	0.001585	0.001585
19	0.04479	0.84061	0.98848	0.34484	0.00109	0.00109	0.00109

As a result of the analysis, we can show the results of model training using the Python library matplotlib:

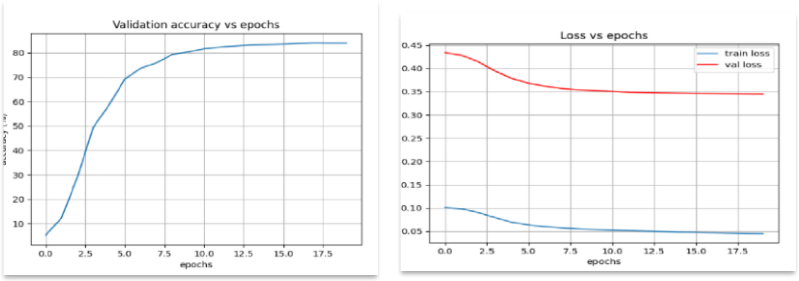


Figure 17. Results of Model Training

While the figure highlights the Top-1 accuracy, it is worth noting that the Top-5 accuracy also yields significant insights. It reflects the model's ability to identify the correct class among its top predictions, offering a more nuanced perspective on overall performance - particularly in cases where multiple classes may appear visually similar. Top-1 accuracy considers only the model's most confident prediction, marking it correct only if it exactly matches the ground truth label. In contrast, Top-5 accuracy is more lenient, deeming the prediction correct if the true label appears within the model's top five highest-probability outputs. This distinction is especially important in complex classification tasks with many categories, where the correct class may not always be the top-ranked prediction but is still among the most plausible candidates.

3.5. Results and Analysis

For the evaluation of the OCR system, accuracy, precision, recall, and the F-score were calculated based on the Confusion Matrix for all three models. The results are as follows.

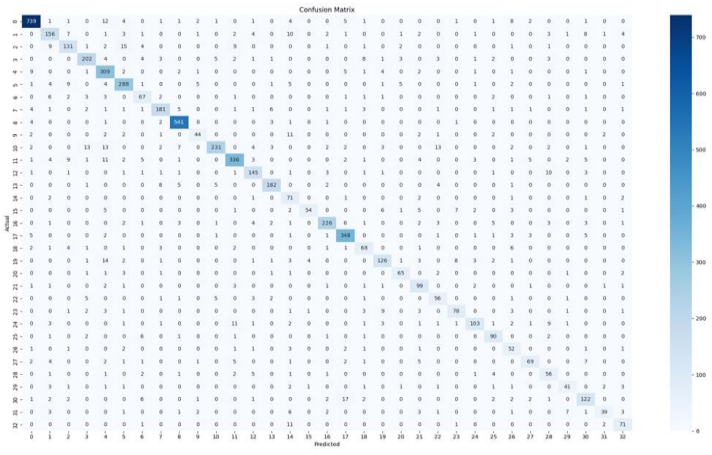


Figure 18. Confusion Matrixes for ResNet

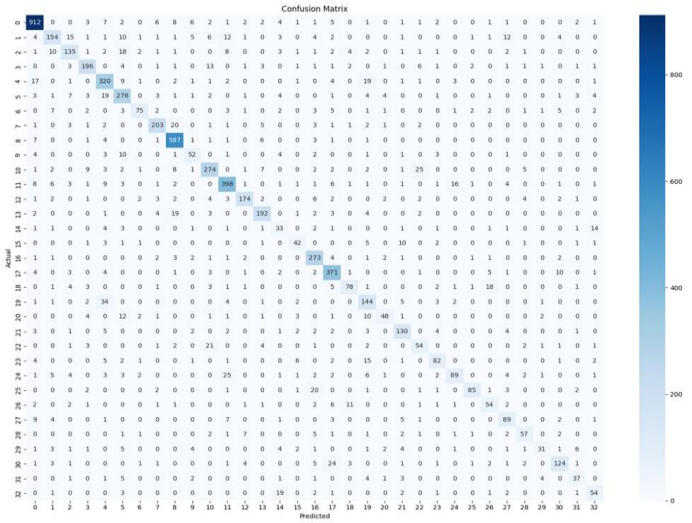


Figure 19. Confusion Matrixes for CNN

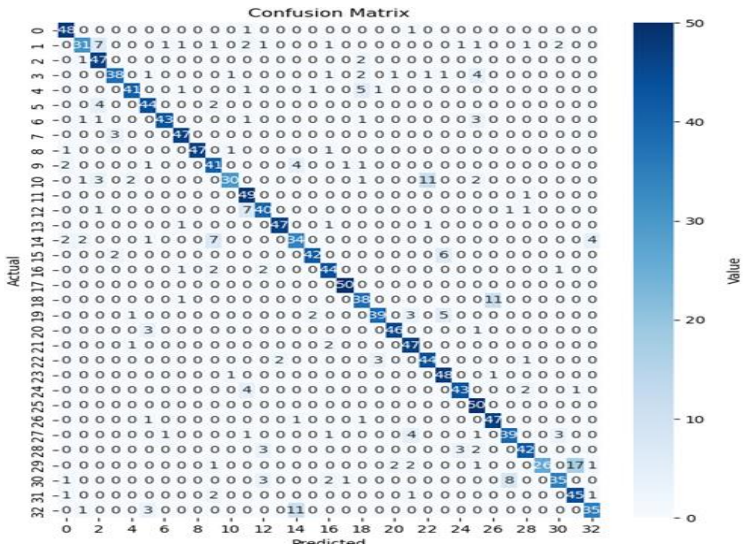


Figure 20. Confusion Matrixes for YOLOv8

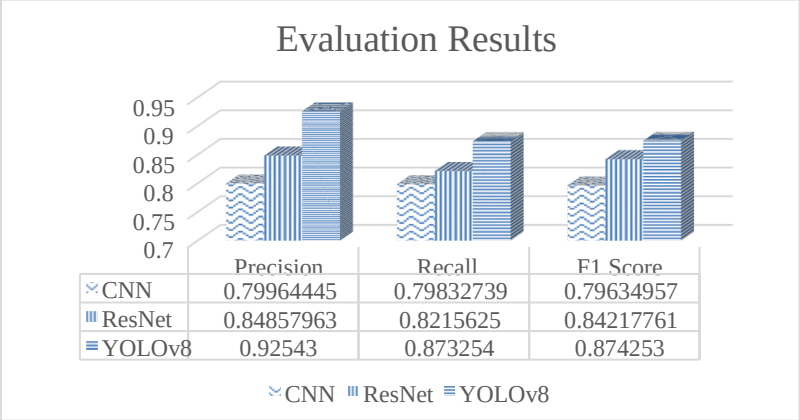


Figure 21. Precision, Recall, F1 Score

As seen from the results, the YOLOv8 model outperformed the other two models on the same data. Advance YOLOv8 model showed some improvements mentioned above, it works better with small letters and major differences that Georgian language characters have. That is why accuracy is much higher and model performance is also faster.

3.6. Model deployment

In conclusion, the deployment of the Georgian language OCR system through Flask, designed to emulate a whiteboard system for uploading whole images and returning typed text, represents a significant step towards bridging the gap between handwritten and digital content in the Georgian language. The deployment details for this system can be found in our GitHub repository (Chikashua 2023). This innovative solution not only leverages advanced optical character recognition techniques tailored to the unique characteristics of Georgian script but also offers a user-friendly experience, making it accessible for a wide range of users. By enabling an easier transformation of handwritten text into digital format, this system plays a crucial role in preserving and digitizing historical documents, improving data accessibility, and promoting the utilization of the Georgian language in the digital age. As technology continues to evolve, this Georgian OCR system serves as a valuable tool in the fields of linguistics, cultural heritage preservation, and document digitization, opening up new possibilities for research, education, and archival work.

4. Conclusion

While this technology represents the latest advancements, there are still no OCR products capable of recognizing all types of text with a 100% accuracy. In conclusion, the goal of this project was to explore one of the intriguing aspects of computer vision related to OCR and the Georgian language.

Tools like whiteboards, with fewer artifacts, distinct colors, and other advantages, tend to yield good results with the mentioned model. However, it also exhibits a reasonable

level of accuracy when processing text written on paper. Progress in the realm of whole document analysis is crucial, although it still requires further development.

Despite the existing advancements in this field, finding a one-size-fits-all solution that consistently delivers accurate results remains challenging. Thus, this paper represents a step towards addressing the problem under study.

References

Alghyaline, Salah. 2022. " A Printed Arabic Optical Character Recognition System using Deep Learning." *Journal of Computer Science*, vol. 18, pp. 1038-1050.

Chaudhuri, Arindam, Krupa Mandaviya, Pratixa Badelia, and Soumya K Ghosh. 2016. "Character Recognition Systems for Different Languages with Soft Computing." In *Studies in Fuzziness and Soft Computing book series*. Springer.

Cheng, Richeng. 2020. "A survey: Comparison between Convolutional Neural Network and YOLO in image identification." *Journal of Physics: Conference Series*, vol. 1453, no. 012139.

Chikashua, Ana. 2023. "Georgian Language OCR with TensorFlow." *GitHub*. https://github.com/AnaChikashua/geo-alphabet.

Farhadi, Joseph Redmon and Ali. 2018. "YOLOv3: An Incremental Improvement." *arXiv preprint arXiv:* 1804.02767.

Jocher, Glenn. 2024. *Ultralytics YOLOv8 Docs.* November 12. https://docs.ultralytics.com/datasets/classify/imagenet.

Kaiming, He, Zhang Xiangyu, Ren Shaoqing, and Sun Jian. 2015. *Deep Residual Learning for Image Recognition.* Microsoft Research.

Krishnakumar, Mukilan. 2023. "Weight & Biases." *Yolov8 locally on a non-GPU machine.* June 26. https://pchenlab.wordpress.com/2023/06/26/yolov8-locally-on-a-non-gpu-machine/.

Microsoft. 2023. *NEBO.* https://apps.microsoft.com.

Nielsen, M. 2019. *Neural Networks and Deep Learning.*

Redmon, Joseph, and Ali Farhadi. 2017. "YOLO9000: Better, Faster, Stronger." *Computer Vision and Pattern Recognition (CVPR),*. USA: Honolulu. pp. 1-23.

Rosebrock, A. 2020. *Getting started with EasyOCR for Optical Character Recognition.* September 14. https://machinelearningknowledge.ai/easyocr-python-tutorial-with-examples/.

Shubham, Prasad. 2020. "https://www.topcoder.com/thrive/articles/python-for-character-recognition-tesseract. ." *https://www.topcoder.com/how-it-works.* December 10.

Shubham, Srivastava, S, Verma Ajay, and Sharma Shekhar. 2022. "Optical Character Recognition Techniques:A Review." *IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS).* Bhopal, India,.

Soselia, D, M Tsintsadze, L Shugliashvili, I Koberidze, S Amashukeli, and S Jijavadze. 2018. "Georgian Handwritten Character Recognition." *Elsevier.*

Subramanian, Srividya, Vineet Kekatpure, Gladina Raymond, Kapil Parab, Shashikant Dugad, and rchana Shirke. 2022. "“TEYSuR - Text Extraction with YOLO and Super Resolution." *in International Conference for Advancement in Technology (ICONAT),*. Goa.

Yiting, Li, Fan Qingsong, Haisong Huang, Zhenggong Han, and Gu Qiang. 2023. "A Modified YOLOv8 Detection Network for UAV AerialImage Recognition." *Drones* vol. 7, no. 304, pp. 2-26.

Received November 19, 2024, revised May 5, 2025, accepted May 6, 2025