applied sciences

MDPI

Article

Synthetized Multilanguage OCR Using CRNN and SVTR Models for Realtime Collaborative Tools

Attila Biró [1,2,3], Antonio Ignacio Cuesta-Vargas [2,3,4], Jaime Martín-Martín [3,5], László Szilágyi [6,7] and Sándor Miklós Szilágyi [1,*]

1 Department of Electrical Engineering and Information Technology, George Emil Palade University of Medicine, Pharmacy, Science, and Technology of Targu Mures, Str. Nicolae Iorga, Nr. 1, 540088 Targu Mures, Romania
2 Department of Physiotherapy, University of Malaga, 29071 Malaga, Spain
3 Biomedical Research Institute of Malaga (IBIMA), 29590 Malaga, Spain
4 Faculty of Health Science, School of Clinical Science, Queensland University Technology, Brisbane 4000, Australia
5 Legal and Forensic Medicine Area, Department of Human Anatomy, Legal Medicine and History of Science, Faculty of Medicine, University of Malaga, 29071 Malaga, Spain
6 Computational Intelligence Research Group, Sapientia Hungarian University of Transylvania, 540485 Targu Mures, Romania
7 Physiological Controls Research Center, Óbuda University, 1034 Budapest, Hungary
* Correspondence: sandor.szilagyi@umfst.ro; Tel.: +40-732-131-974

Abstract: Background: Remote diagnosis using collaborative tools have led to multilingual joint working sessions in various domains, including comprehensive health care, and resulting in more inclusive health care services. One of the main challenges is providing a real-time solution for shared documents and presentations on display to improve the efficacy of noninvasive, safe, and far-reaching collaborative models. Classic optical character recognition (OCR) solutions fail when there is a mixture of languages or dialects or in case of the participation of different technical levels and skills. Due to the risk of misunderstandings caused by mistranslations or lack of domain knowledge of the interpreters involved, the technological pipeline also needs artificial intelligence (AI)-supported improvements on the OCR side. This study examines the feasibility of machine learning-supported OCR in a multilingual environment. The novelty of our method is that it provides a solution not only for different speaking languages but also for a mixture of technological languages, using artificially created vocabulary and a custom training data generation approach. **Methods**: A novel hybrid language vocabulary creation method is utilized in the OCR training process in combination with convolutional recurrent neural networks (CRNNs) and a single visual model for scene text recognition within the patch-wise image tokenization framework (SVTR). **Data**: In the research, we used a dedicated Python-based data generator built on dedicated collaborative tool-based templates to cover and simulated the real-life variances of remote diagnosis and co-working collaborative sessions with high accuracy. The generated training datasets ranged from 66 k to 8.5 M in size. Twenty-one research results were analyzed. **Instruments**: Training was conducted by using tuned PaddleOCR with CRNN and SVTR modeling and a domain-specific, customized vocabulary. The Weight & Biases (WANDB) machine learning (ML) platform is used for experiment tracking, dataset versioning, and model evaluation. Based on the evaluations, the training dataset was adjusted by using a different language corpus or/and modifications applied to templates. **Results**: The machine learning models recognized the multilanguage/hybrid texts with high accuracy. The highest precision scores achieved are 90.25%, 91.35%, and 93.89%. **Conclusions**: machine learning models for special multilanguages, including languages with artificially made vocabulary, perform consistently with high accuracy.

Keywords: CRNN; SVTR; CNN; RNN; multilingual OCR; remote diagnosis; real-time translation; collaborative diagnostics; real-time text detection; assessment

Citation: Biró, A.; Cuesta-Vargas, A.I.; Martin-Martin, J.; Szilágyi, L; Szilágyi, S.M. Multilanguage OCR Using CRNN and SVTR Models for Realtime Collaboration. *Appl. Sci.* **2023**, *13*, 4419. https://doi.org/10.3390/app13074419

Academic Editor: Pengjie Ren

Received: 24 February 2023
Revised: 22 March 2023
Accepted: 27 March 2023
Published: 30 March 2023

1. Introduction

In recent years, thanks to remote diagnosis using collaborative tools [1], multilingual joint working sessions [2] have started to take place in several domains, including health care. One of the challenges in establishing noninvasive, safe, and distant cooperation models, which will be able to provide a solution that enables sharing documents and presentations on the screen in real time [2] to increase the effectiveness of these noninvasive, safe, and distant cooperation models. Traditionally, optical character recognition (OCR) solutions [3–7] have not been successful when a combination of languages, dialects, technical linguistic layers, or different levels or skills of technicians are involved in the process [8,9]. There are also risks of misinterpretations caused by incorrect translations or the interpreters' lack of domain knowledge. Therefore, artificial intelligence (AI) must also improve [10] the OCR part of this technological pipeline to avoid misunderstandings [11,12].

Real-time translation pipelines [2] now include OCR [13–15] as one of their central pillars, which helps solve all translation and interpretation problems. Our research results with the technological pipeline (see Figure 1)—supporting the collaborative videoconferencing tools and platforms—can be applied to healthcare and sports safety.

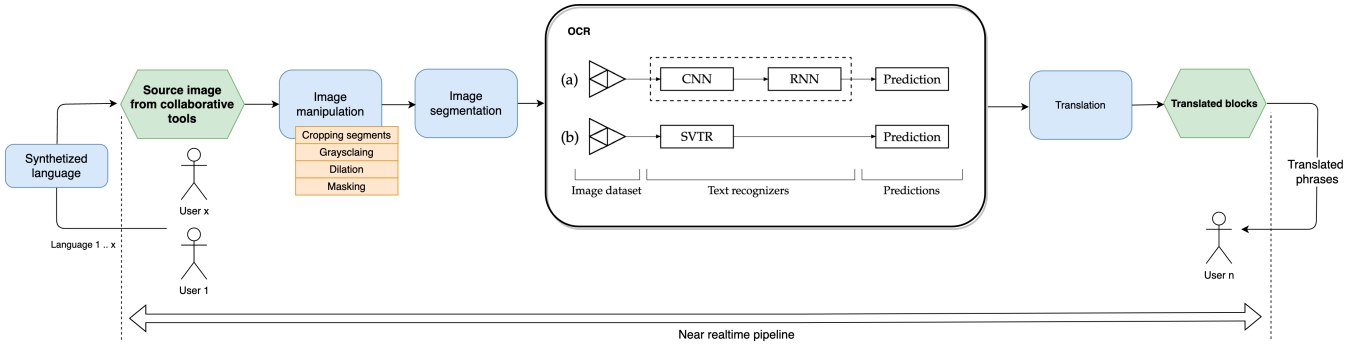

Figure 1. Near real time synthesized language translation pipeline for distant cooperation.

The originality of our approach lies in the fact that it offers a real-time synthesized multilanguage detection solution not only for the various languages spoken by humans [16–18], but also for their sublayers as well as a mixture of technical languages in the wording of texts. This is accomplished through the utilization of artificially created vocabulary as well as a customized method for the generation of training data. The data extraction could be added to neural translation engines [18] to shorten the communication gaps.

Using synthesized language preparation, this study evaluates the feasibility of synthesized language preparation and its adaptability to machine learning (ML)-based OCR engines [19–23] designed to provide efficient solutions for machine learning-based neural translation engines. It validates the effectiveness of analysis for preparing and adapting a synthetic language based on machine learning.

A significant contribution of this paper is how it approaches the problem of developing an artificial intelligence pipeline for interaction with healthcare professionals to facilitate collaborative videoconferencing solutions [2] based on the research presented in this document.

The purpose of the research in both phases was to conduct limited validation-type experiment rounds, summarize their findings, simultaneously measure, compare, and analyze the results, and develop specific requirements concerning the effectiveness of the input data and the effectiveness of the preparation process [2]. These experiments are designed to demonstrate that achieving high accuracy is possible with a tuned model and a suitable algorithm to generate vocabulary and training datasets. In addition, by making use of AI models that have already been established in other industries, we can make predictions with a high degree of accuracy to cover the use cases of "videoconference-based collaborative tools" [1,2,9,24], which can be used for further modeling and translation into

real life. Attendees of video diagnostics could benefit from these models as a component of a more collaborative communication pipeline.

This study takes a unique and cutting-edge approach to remote diagnostics by repurposing pre-existing AI programs and ML algorithms from various other disciplines. It does so through utilizing collaborative use cases, which result in time and resource savings, as well as an increase in the efficiency of multilingual collaboration solutions [2].

2. Novelty of the Approach

In this study, we employ synthetic, multilingual training convolutional recurrent neural networks (CRNN) and patch-wise image tokenization framework (SVTR) model architectures [25] for ML-based OCR engines [14] to (1) test the possibility of custom-generated multilanguage recognition and (2) increase the accuracy and efficiency of OCR.

Mixing Convolutional Neural Networks (CNNs) with Recurrent Neural Networks (RNNs) enables the CRNN kind of OCR engine to recognize text in images. CNN retrieves features from the input image, while RNN models the sequential dependencies between characters in the text. CRNNs effectively recognize text in various contexts, including font sizes, font styles, and orientations. They can also handle many lines of text in an image and recognize scene text accurately.

SVTR is an advanced OCR engine that recognizes text in images by using a combination of deep learning and traditional image processing techniques. The SVTR architectures can recognize scene text in photographs with changing lighting, orientation, and skew. In addition, it has an attention mechanism that allows the network to focus on the most important visual components, hence boosting the noise and distortion resistance of the model [8].

Both CRNN and SVTR offer innovative features in comparison to conventional OCR engines. The advantages include the following: (i) CRNN can recognize texts of various font styles, sizes, and orientations, as well as several lines of text in an image, and is effective at scene text recognition, (ii) SVTR is resistant to noise and distortion due to the attention mechanism it employs to focus on critical portions of an image and can handle scene text in images, (iii) SVTR and CRNN are deep learning-based models that learn from complicated data correlations between the input image attributes and the output text, making them more effective than conventional algorithms, (iv) CRNN and SVTR are more efficient than conventional OCR engines since they can run on parallel systems, such as Graphics Processing Units (GPU), and (v) CRNN and SVTR architectures are adaptive and can be improved with additional training data.

It is crucial to note that OCR systems are complex and that the architecture you choose will rely on the particulars of the problem you are attempting to address. Each architecture has benefits and drawbacks, and the ideal solution will depend on the nature of the problem, the amount and quality of the data, and other factors [26]. In our research, we use and fine-tune CRNN and SVTR models with a PaddleOCR [27] framework.

3. Objectives

Here are the objectives for the research of a Synthetic and/or Multilanguage OCR system using CRNN and SVTR models for real-time collaborative tools:

1. Customize hyperparameters of the PaddleOCR [28] system to accurately recognize text in multiple languages by using CRNN and SVTR models.
2. Compare the performance of CRNN and SVTR models for multilanguage OCR to identify which model is more effective for this task.
3. Evaluate the performance and limitations of the PaddleOCR on a large dataset of multilingual images.
4. Investigate the impact of different multi/synthetic language models and parameters on the PaddleOCR system to identify which languages and vocabulary sets are more challenging for the system to recognize.

5. Investigate the impact of different levels of text blurriness, distortion, and noise on the performance of the OCR system, to identify which image conditions are more challenging for the system to recognize.
6. Investigate the scalability of the OCR system to handle large volumes of images, to ensure that it can handle real-time collaborative tasks efficiently.
7. Investigate the robustness of the OCR system to handle different font types and sizes, to ensure that it can handle real-world scenarios efficiently.
8. Investigate the customized vocabulary on synthesized, multilanguage models.

4. Materials and Methods

4.1. ML-Based Text Recognizers

In our study, we used the following types of modern text recognizers [29] as shown in Figure 2: (a) CNN-RNN models and (b) the SVTR model—a single visual (scene) text recognition model explicitly designed for multilingual applications [30].

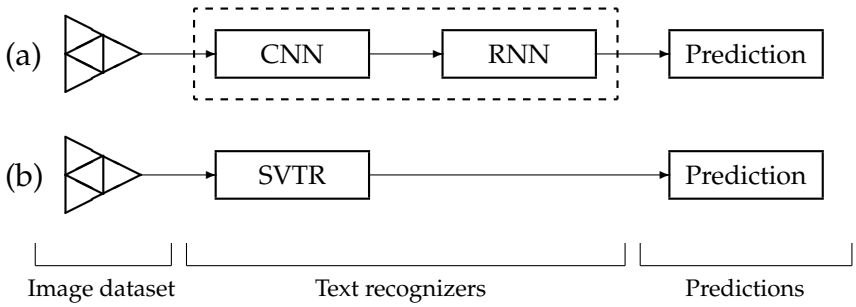

Figure 2. Modern text recognizers using (a) CRNN and (b) SVTR models, adapted from [30].

Our experiments were carried out by using models (a) and (b). Through experiments, we have established that the accuracy and speed of inference play a critical role in decision-making. The dataset and the model have been sufficiently trained to allow us to understand how to optimize the dataset and the model. Optimal input parameters and configurations for model training have been identified through some experiments. In addition to offering several configurable model training options, PaddleOCR has been configured to work with multiple model architectures. The configuration of the most effective hyperparameters has experimented with various model mutations, optimizing the parameters until the optimal ones have been achieved. At the end of the investigations, we trained a model with 94% precision.

4.2. CRNN Model Type

For image-based text recognition, CRNN [31–35] is an increasingly popular modeling technique [26]. The convolutional layers extract characteristics from the input sequence, whereas the recurrent layers capture temporal dependencies [26,29]. The output of the last convolutional layer is fed into a recurrent layer, either Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) layer. The input to the network is composed of the following sequence of 'input vectors' $x_1, x_2, \ldots, x_T$, where T is considered the 'length of the sequence'. The output of this network is composed of the following 'output vector' $y_1, y_2, \ldots, y_T$. The output of the i^{th} the 'convolutional layer' can be defined as:

$$h_i = f(W_i * x + b_i),$$

where W_i is considered as the 'weight matrix' for the ith 'convolutional layer', where b_i is considered as the 'bias vector' for ith convolutional layer, $*$ shows the 'convolution op', and f is an 'activation function'. The output of the recurrent layer is shown as $h_t = g(h_{t-1}, x_t)$, where h_t is considered as the 'hidden state' of the recurrent layer at time t, g is the 'recurrent function,' and h_{t-1} is the 'hidden state' of the recurrent layer defined at time $t-1$.

The LSTM is a recurrent layer used in the CRNN algorithm to capture temporal dependencies in the input sequence. For an LSTM layer, the recurrent function g can be expressed as follows:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c)$$
$$h_t = o_t \odot \tanh(c_t)$$

where i_t, f_t, o_t, and c_t are the 'InputGate', 'ForgetGate', 'OutputGate', and 'CellState' vectors at time t, and W_i, W_f, W_o, W_c, U_i, U_f, U_o, U_c, b_i, b_f, b_o, and b_c are the 'weight matrices' and 'bias vectors' for LSTM [36–38].

GRU is a recurrent layer used in the CRNN to capture temporal dependencies in the input sequence. For a GRU layer, the recurrent function g can be expressed as:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$
$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$
$$\tilde{h}_t = \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

where r_t, z_t, and $\tilde{h}_t$ are the 'ResetGate', 'UpdateGate', and 'CandidateHiddenState' vectors at time t, while W_r, W_z, W_h, U_r, U_z, U_h, b_r, b_z, and b_h are the 'weight matrices' and 'bias vectors' for the GRU.

As shown in the CRNN architecture in Figure 3, the extraction process of beneficial features is performed sequentially, applying convolutions and max-pooling operations. When these are received, they are put into the recurrent unit, which stores information based on how it happened in time.

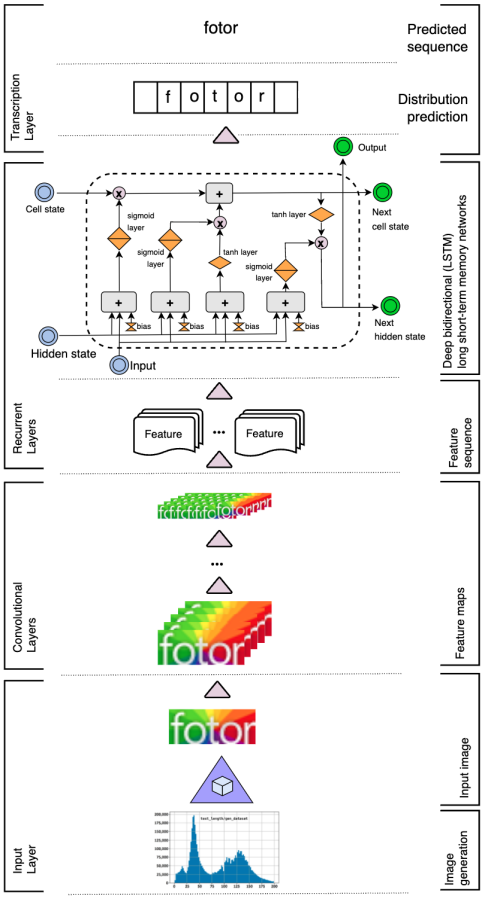

Figure 3. CRNN architecture [33] with CRNN feature extraction, adapted from [35].

4.3. SVTR Model Type

The SVTR [30,39] model, as presented in Figure 4, uses a patch-wise image tokenization framework, eliminating the need for sequential modeling [39].

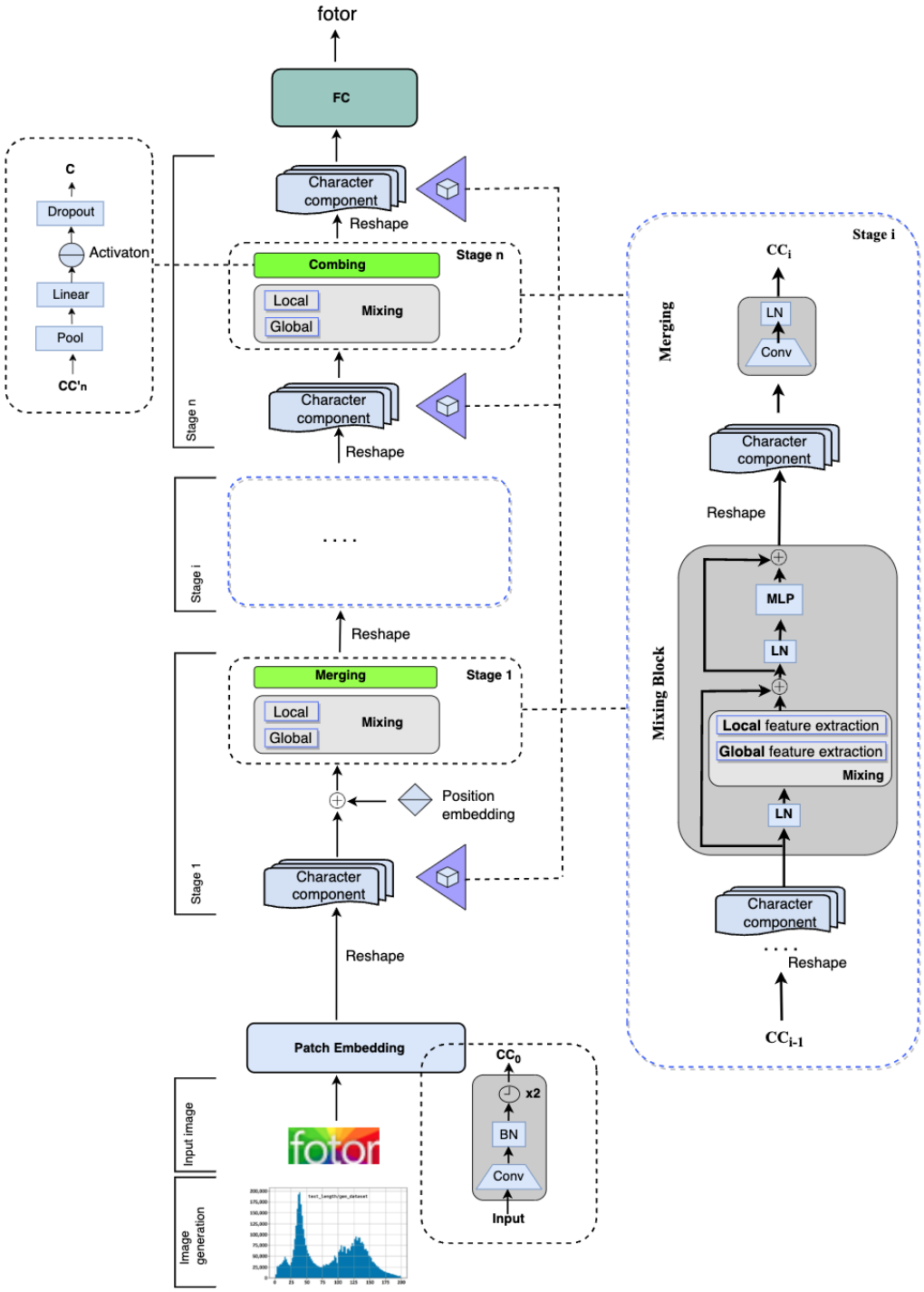

Figure 4. SVTR architecture, adapted from [30].

It uses a single visual model to recognize text in scenes. SVTR can be defined as:

$$y(t) = f[t; z(t)] + e(t),$$

where $y(t)$ is the response variable at time t, $f[t; z(t)]$ is the piecewise linear function with varying thresholds, $z(t)$ are the threshold values, and $e(t)$ is the error term at time t [40]. The piecewise linear function with varying thresholds can be written as:

$$f[t; z(t)] = \beta_0(t) + \beta_1(t)x(t) + \theta_i(t)[\beta_2(t) + \beta_3(t)x(t)],$$

where $\beta_0(t)$ and $\beta_1(t)$ are the 'intercept param' and 'slope param' for the linear segment, $\beta_2(t)$ and $\beta_3(t)$ are the intercept and slope parameters for the threshold segment, $\theta_i(t)$ is the indicator function that equals 1 when $z(t) > \tau_i$ and 0 otherwise, and τ_i is the i^{th} threshold value. The indicator function $\theta_i(t)$:

$$\theta_i(t) = \begin{cases} 1 & \text{if } z(t) > \tau_i \\ 0 & \text{otherwise} \end{cases}.$$

The threshold values $z(t)$ can be modeled by using a threshold function $g(t; \theta)$ that varies with time t and a smoothing parameter λ:

$$z(t) = g(t; \theta) + \epsilon(t),$$

where $\epsilon(t)$ is the 'error term' for the threshold value at time t. A common choice of threshold function is the sigmoid function:

$$g(t; \theta) = \frac{c}{1 + e^{-(t-d)/a}},$$

where c, a, d, and θ are the model parameters, which determine the shape and location of the threshold function.

The SVTR algorithm can be estimated by using the maximum likelihood estimation or Bayesian inference methods. The model parameters can be estimated by using iterative algorithms, such as the EM algorithm, or gradient descent methods, such as the stochastic gradient descent algorithm. The SVTR algorithm provides a flexible and interpretable approach for modeling time series data with non-stationary trends and seasonality. Using the SVTR method, the image text is first decomposed into small patches called character components. Following this, the hierarchy of stages is repeated for the components' mixing, merging, and combining at the component level. To perceive inter- and intra-character patterns, global and local mixing blocks are combined, resulting in a multigrained perception of character components. Due to the use of this model by PaddleOCR3 [27], these model architectures were referred to as version 3 (v3) during this experiment. Figure 4 illustrates the architecture of SVTR.

4.4. Differences between CRNN and SVTR Models

Both CRNN and SVTR are machine learning-based OCR engine architectures proposed to increase the accuracy and efficiency of OCR systems. Here are a few significant distinctions between the two models:

1. Inputs: CRNNs are designed to operate with image inputs, but SVTRs can process pictures, videos, and live video streams. CRNNs use a combination of CNNs and RNNs to detect text in images, whereas SVTRs use a combination of deep learning and conventional image processing techniques.
2. Text recognition: CRNNs are adept at recognizing texts of various font styles, sizes, and orientations, as well as multiple lines of text in an image, and can perform well on scene text recognition, whereas SVTRs are specifically designed to handle scene text in images, which can vary significantly in terms of lighting, orientation, and skew.

3. Attention mechanism: CRNNs lack an attention mechanism, but SVTRs possess one. This attention mechanism enables the network to concentrate on an image's significant parts and increases the model's noise and distortion resistance. Because of the attention mechanism they employ, SVTRs are more resistant to noise and distortion. CRNNs are susceptible to noise and distortion.
4. Training and fine-tuning: Both CRNN and SVTR architectures are adaptive and can be fine-tuned with more data to increase their accuracy. However, CRNN models require substantial training data to produce accurate results. SVTR models, in contrast, can also learn from fewer data. CRNNs are quicker than conventional OCR engines because they can run on parallel architectures such as GPUs. In contrast, SVTRs can run on various platforms, including smartphones and embedded systems.

4.5. Hyperparameter Tuning

Several aspects of the training process and the model were adjusted and/or calibrated to achieve better results. You can find a list of configurable parameters and their meanings in Results secion.

4.5.1. Max Epoch Number

As a result of the training function iterating through the entire training dataset several times, it is required to run through it several times. The number was selected to allow the training to reach its maximum potential because the training can be stopped anytime.

4.5.2. Max Text Length

There is a maximum number of digital characters. Since the project's objective was clearly defined, we decided on a maximum length of 200 characters. However, we have set it to 100 for some experiments to determine whether there are differences. It was observed that the model works better for shorter texts.

4.5.3. Optimizer

The optimizer is responsible for calculating training gradients and fine-tuning the model weights. With beta1 and beta2 values of 0.9 and 0.999, respectively, the Adam optimizer was picked in all instances [2].

4.5.4. Learning Rate

Depending on the multiplier value, the Optimizer can change the weights in each cycle differently [41]. It is essential to highlight the differences between excessively low and high learning rates (LR). The model would learn more slowly if the LR were too low, resulting in a local minimum optimization instead of a global minimum optimization. When the LR is too high, the model cannot find the optimal minimum. Unfortunately, there is no predefined standard rule for selecting the optimal LR, since it varies from case to case. A range of LR values between 0.0001 and 0.005 was tested.

4.5.5. Regularization Logic

An overfitted model can result in inaccurate results. In this case, overfitting occurs when the model learns the exact pixels of each training image instead of understanding the general meaning of the characters. A model trained on a dataset will yield a very high score. However, it will fail when faced with data that have not been trained.

Evaluation metrics can be used to check for overfitting. If the model's evaluation accuracy decreases while its training accuracy continues to increase, it will be deemed overfitted. Using regularizers (L1 and L2) can prevent this by decreasing the learning rate over time, preventing the model from learning the training images exactly. The effect is similar to that of the learning rate and is similar to that of a multiplier. We have experimented with values from 0.000002 to 0.0001 to determine the optimal value.

4.5.6. Model Hidden Layers

Recurrent neural networks (RNNs) have several hidden layers. It is critical to note that this can only be applied to models based on CRNNs. By default, a layer with a width of 128 was used.

4.6. Image Size

For each image in the training set, PaddleOCR can perform some preprocessing steps. This will enable us to make all images the same size. For the models to work, static images are required. Various image heights and widths were tested (no changes were observed in the effects), as well as different image widths. Throughout this study, we used pixels with a width ranging from 100 to 2048. Since we requested 200-character sentences to fulfill real-life use cases, with a width of 2048, we can accommodate at least 10 pixels for each character.

4.7. Batch Size

This parameter specifies the number of images stored in GPU memory simultaneously. As a rule of thumb, we adhere to the principle of "more is better", since the model will be able to learn more broadly, and the training process will be faster. In addition to filling the GPU memory with the value, we also used the value to calculate other hyperparameters. In most cases, the value ranges between 24 and 128 units.

4.8. Data

The training data were created by a dedicated image generation tool, specifically a Python-based tool, which generates "collaborative tool template"-based single and/or multiline images from a fully customizable language corpus. The generator can specify which text sources will be used and the ratios of these sources to be included in the resulting training data. Templates can also be customized per application to meet real-life requirements. The Generator was able to customize the fonts, using the operating system's font database, having an additional external and local source expandability option to cover the robustness criteria of Objectives. The most-used fonts were included in the colaborative applications to cover the real-world scenarios.

For our research trials including video conference and application data, we developed datasets representing real-world use situations, as described in [2]. Consequently, we did not use pre-defined annotated datasets in absence of appropriate options. A dedicated Python picture extractor was developed to extract relevant video pictures. Manual data collection was also conducted by using this method. The validation dataset consisted of hundreds of raw images collected, filtered, and classified through collaborative tools. Several experiments were conducted on the Linux (Ubuntu) platform utilizing Python, either with or without virtualization.

4.8.1. Datasets and Volumes

Table 1 shows the dataset and the evaluation size information. During the experiments, we have investigated large datasets as well, but based on the initial result, these experiments were discontinued due to expected running times. These experiments will be investigated in a future study in a High Performance Computing (HPC) environment.

Table 1. Volume information about Dataset and Evaluation.

Project	Name	Data	Training Size	Evaluation Size
OCR_hun	155k_hu_v2_2	hun_train	77 k	15 k
OCR_hun	155k_hu_v2_1	hun_train	77 k	15 k
OCR_enhu	SVTR	training1	66 k	33 k
OCR_enhu	CRNN	training1	66 k	33 k
OCR_multilang	5M_enhujp_v2_8	training10-200	5 M	101 k

Table 1. *Cont.*

Project	Name	Data	Training Size	Evaluation Size
OCR_multilang	5M_enhujp_v2_6	training9-200	5 M	101 k
OCR_multilang	5M_enhujp_v2_5	training10-200	5 M	101 k
OCR_multilang	5M_enhujp_v2_4	training9-200	5 M	101 k
OCR_multilang	5M_enhujp_v2_3	training8-200	8.5 M	170 k
OCR_multilang	5M_enhujp_v2_2	training8-200	8.5 M	170 k
OCR_multilang	5M_enhujp_v2_1	training7-100	5 M	100 k
OCR_multilang	10M_enhujp_v3_1	training8-200	8.5 M	170 k
OCR_multilang	5M_enhujp_v3_4	training7-100	5 M	100 k
OCR_multilang	5M_enhujp_v3_3	training7-100	5 M	100 k
OCR_multilang	5M_enhujp_v3_2	training7-100	5 M	100 k
OCR_multilang	4M_enhujp_pre_v3_1	training6	4.4 M	88 k
OCR_multilang	4M_enhujp_pre_v3_1	training6	4.4 M	88 k
OCR_multilang	4M_enhujp_v3_5	training6	4.4 M	88 k
OCR_multilang	4M_enhujp_v3_4	training6	4.4 M	88 k
OCR_multilang	2M_enhujp_v2_2	training3	2 M	220 k
OCR_hun	186k_hu_v2_1	training2	186 k	34 k

4.8.2. Vocabularies

Based on the training data, a vocabulary can be created by using a unified dictionary or generating it from the training data. However, the size of the group will influence the training process. Table 2 highlights the different vocabularies (used in the research experiments) and their sizes and types.

Table 2. Vocabulary sizes and types.

Character Dictionary	Dict Size (Characters)	Vocabulary Type
606k_hun_vocab.txt	112	Generated
extended_vocab.txt	201	Unified
training9-200_vocab_min9500.txt	803	Generated
training9-200_vocab_min200.txt	2737	Generated
jpn_latin_dict.txt	4444	Unified
4M_vocab.txt	4721	Generated
4M_min20_vocab.txt	3980	Generated
2M_min2k_vocab.txt	968	Generated
186k_extended_vocab.txt	112	Unified

4.9. Experimental Environment

Our experiments were run in different dedicated environments with different GPU capabilities as follows: Config 1 PC with the following configuration: MBO Gigabyte Z390 Aorus Pro, CPU INTEL Core i7-8700K 3.7 GHz 12 MB LGA1151, DDR4 32 GB 3600 MHz Kingston HyperX Predator Black CL17 KIT2, VGA MSI RTX 2080 Ventus 8 GB, SSD M.2 SAMSUNG 970 Pro 1 TB, Corsair RMx (2018) 750 W Modular 80+ Gold) [2]; Config 2 PC with the following configuration: 2x VGA MSI RTX 2080 Ventus 8 GB; and, in the last phase, Config 3 PC with the following configuration: 6x VGA MSI RTX 2080 Ventus 8 GB.

4.10. Pre-Processing

4.10.1. Exploratory Data Analysis

For ease of processing, some exploratory data analysis (EDA) steps are carried out on the training dataset after it has been generated: (1) counting images, (2) visualizing the

distribution of training examples per category of application templates, (3) loading .tsv files into the DataFrame, and (4) visualizing the distribution of text length. From a scientific point of view, the fourth step is one of the most relevant topics. Figure 5 shows the visually represented text length distribution of the training dataset:

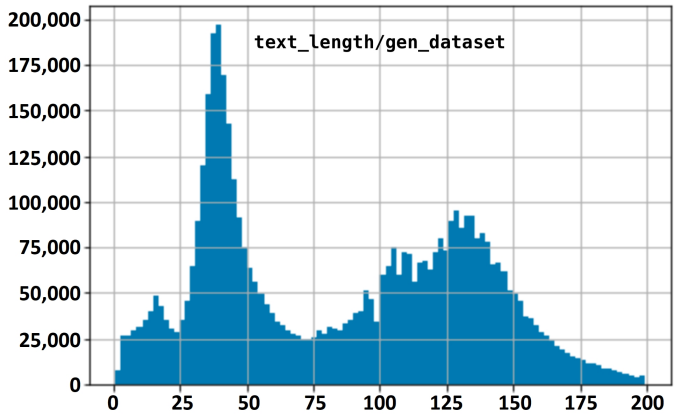

Figure 5. Visual representation of the distribution of the length of texts in the training data.

4.10.2. Vocabulary Generation

Two methods are available to create vocabulary (or dictionaries) in the training process: (1) create a unified vocabulary that is supervised by human experts and (2) generate vocabulary from training data. We begin by applying method number (1), and then customize the vocabulary based on method number (2). With the help of a Python script, we identified the number of character occurrences in the dictionary. It was analyzed whether it contained characters that appeared more than x times. Figure 6 shows a visual representation of character occurrences in the dataset, where value 'x' represents the value of the *x*-axis, which ranges from 0 to 2000.

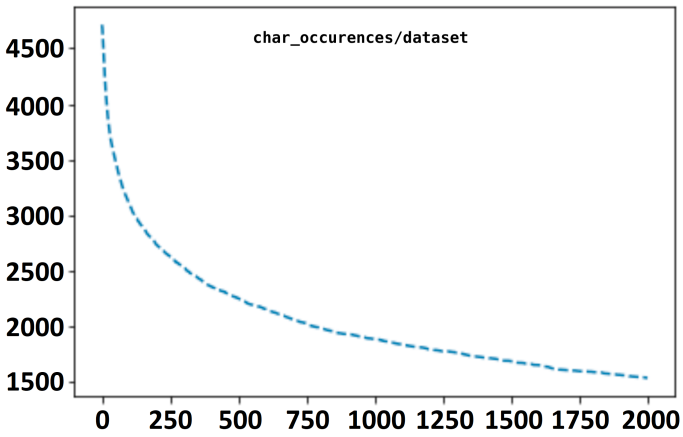

Figure 6. Visual representation of character appearances in the generated training dataset.

On the *x*-axis of the figure, the number 1.732 represents how many characters would be in the dictionary if we removed every character that does not occur more than 1.732 times in the whole dataset. In light of the analysis, let us select 200 as a value. This describes how many times a character appears in the whole dataset to be included in the vocabulary. A filter was applied to these characters, and they were stored for further analysis. All characters that appeared at least 200 times in the vocabulary were stored after this step.

4.10.3. Filter, Split, and Save the Training Dataset

We must use a particular Python script to ensure that the training dataset contains only instances of these vocabulary terms. Moreover, we must check that none of the non-permitted characters appear in the samples. Next, the dataset should be divided into training and validation parts. An important part of model training is the validation of the training procedure. Let us cut off 1% of the training data sample for validation, exclude examples that have non-permitted characters in them, and then save them in Paddle format.

5. Data Analysis

We investigated three different project type during experiments as is shown in Table 3. The volume information of the training datasets was layered into three categories: (1) OCR_enhu: English and Hungarian example only experiments, (2) OCR_hun: Hungarian examples only experiments, and (3) OCR_multilang: when Japanese, English, and Hungarian languages are present in the dataset.

Table 3. Training names, training data, and batch sizes.

Experiment Name	Data	Character Dictionary	Dictionary Size	Train Batch Size	Evaluation Batch Size	Training Size	Evaluation Size
155k_hu_v2_2	hun_train	606 k_hun_vocab.txt	112	128	128	77 k	15 k
155k_hu_v2_1	hun_train	606k_hun_vocab.txt	112	600	256	77 k	15 k
SVTR	training1	extended_vocab.txt	201	128	128	66 k	33 k
CRNN	training1	extended_vocab.txt	201	768	256	66 k	33 k
5M_enhujp_v2_8	training10-200	training9-200_vocab_min9500.txt	803	48	48	5 M	101 k
5M_enhujp_v2_6	training9-200	training9-200_vocab_min200.txt	2737	64	64	5 M	101 k
5M_enhujp_v2_5	training10-200	training9-200_vocab_min9500.txt	803	48	48	5 M	101 k
5M_enhujp_v2_4	training9-200	training9-200_vocab_min200.txt	2737	64	64	5 M	101 k
5M_enhujp_v2_3	training8-200	jpn_latin_dict.txt	4444	50	50	8.5 M	170 k
5M_enhujp_v2_2	training8-200	jpn_latin_dict.txt	4444	30	30	8.5 M	170 k
5M_enhujp_v2_1	training7-100	jpn_latin_dict.txt	4444	50	50	5 M	100 k
10M_enhujp_v3_1	training8-200	jpn_latin_dict.txt	4444	60	60	8.5 M	170 k
5M_enhujp_v3_4	training7-100	jpn_latin_dict.txt	4444	50	50	5 M	100 k
5M_enhujp_v3_3	training7-100	jpn_latin_dict.txt	4444	50	50	5 M	100 k
5M_enhujp_v3_2	training7-100	jpn_latin_dict.txt	4444	50	50	5 M	100 k
4M_enhujp_pre_v3_1	training6	jpn_latin_dict.txt	4444	24	24	4.4 M	88 k
4M_enhujp_pre_v3_1	training6	jpn_latin_dict.txt	4444	28	28	4.4 M	88 k
4M_enhujp_v3_5	training6	4M_vocab.txt	4721	28	28	4.4 M	88 k
4M_enhujp_v3_4	training6	4 M_min20_vocab.txt	3980	28	28	4.4 M	88 k
2M_enhujp_v2_2	training3	2M_min2k_vocab.txt	968	64	64	2 M	220 k
186 k_hu_v2_1	training2	186 k_extended_vocab.txt	112	128	128	186 k	34 k

6. Results

6.1. Settings

Table 4 shows the model settings such as Epoch numbers, Max text length, Training batch size, Evaluation batch size, information on image shape, warm-up, learning rate, and Epoch trains.

Table 4. Setups.

Experiment Name	Data	Character Dictionary	Dictionary size	Train Batch Size	Learning Rate	Max Text Length	Image Shape	Epoch Number	Warmup Epochs
155k_hu_v2_2	hun_train	606k_hun_vocab.txt	112	128	0.0050	100	[3, 32, 512]	2000	2
155k_hu_v2_1	hun_train	606k_hun_vocab.txt	112	600	0.0050	100	[3, 32, 128]	2000	2
SVTR	training1	extended_vocab.txt	201	128	0.0010	150	[3, 48, 320]	100	5
CRNN	training1	extended_vocab.txt	201	768	0.0005	250	[3, 32, 100]	100	0
5M_enhujp_v2_8	training10-200	training9-200_vocab_min9500.txt	803	48	0.0005	200	[3, 32, 1024]	100	1
5M_enhujp_v2_6	training9-200	training9-200_vocab_min200.txt	2737	64	0.0050	200	[3, 32, 1024]	100	1
5M_enhujp_v2_5	training10-200	training9-200_vocab_min9500.txt	803	48	0.0050	200	[3, 32, 1024]	100	2
5M_enhujp_v2_4	training9-200	training9-200_vocab_min200.txt	2737	64	0.0050	200	[3, 32, 1024]	100	2
5M_enhujp_v2_3	training8-200	jpn_latin_dict.txt	4444	50	0.0050	200	[3, 32, 1024]	100	2
5M_enhujp_v2_2	training8-200	jpn_latin_dict.txt	4444	30	0.0050	200	[3, 32, 2048]	100	2
5M_enhujp_v2_1	training7-100	jpn_latin_dict.txt	4444	50	0.0050	100	[3, 32, 1024]	100	2
10M_enhujp_v3_1	training8-200	jpn_latin_dict.txt	4444	60	0.0010	200	[3, 32, 1024]	50	0
5M_enhujp_v3_4	training7-100	jpn_latin_dict.txt	4444	50	0.0001	100	[3, 32, 512]	50	0
5M_enhujp_v3_3	training7-100	jpn_latin_dict.txt	4444	50	0.0001	100	[3, 32, 512]	50	0
5M_enhujp_v3_2	training7-100	jpn_latin_dict.txt	4444	50	0.0001	110	[3, 32, 512]	50	0
4M_enhujp_pre_v3_1	training6	jpn_latin_dict.txt	4444	24	0.0005	200	[3, 32, 768]	50	0
4M_enhujp_pre_v3_1	training6	jpn_latin_dict.txt	4444	28	0.0005	200	[3, 32, 768]	50	2
4M_enhujp_v3_5	training6	4M_vocab.txt	4721	28	0.0010	200	[3, 32, 768]	50	0
4M_enhujp_v3_4	training6	4M_min20_vocab.txt	3980	28	0.0005	200	[3, 32, 768]	50	0
2M_enhujp_v2_2	training3	2M_min2k_vocab.txt	968	64	0.0050	200	[3, 32, 1024]	100	2
186k_hu_v2_1	training2	186k_extended_vocab.txt	112	128	0.0050	100	[3, 32, 512]	2000	2

6.2. Accuracy

Table 5 highlights the outcome (accuracy) information of the models, using a scaled highlighting table of the most promising results, where green shows the best results.

Table 5. Results and accuracy.

Name	Image Shape	Train Batch Size	Learning Rate	Warmup Epochs	Train Epochs	Steps	Train Best Acc	Train Min Loss	Train Best Norm Edit Dist	Eval Best Accuracy
155k_hu_v2_2	[3, 32, 512]	128	0.0050	2	673	407,677	**0.949**	3.623	**0.985**	0.857
155k_hu_v2_1	[3, 32, 128]	600	0.0050	2	1224	157,896	0.301	0.002	0.475	0.272
SVTR	[3, 48, 320]	128	0.0010	5	100	17,225	0.055	0.970	0.177	0.001
CRNN	[3, 32, 100]	768	0.0005	0	100	1521	0.003	0.963	0.119	0.002
5M_enhujp_v2_8	[3, 32, 1024]	48	0.0005	1	21	1,287,568	**0.958**	0.260	**0.999**	**0.938**
5M_enhujp_v2_6	[3, 32, 1024]	64	0.0050	1	23	867,561	**0.938**	0.529	**0.997**	**0.909**
5M_enhujp_v2_5	[3, 32, 1024]	48	0.0050	2	4	350,376	0.885	4.871	**0.976**	0.850
5M_enhujp_v2_4	[3, 32, 1024]	64	0.0050	2	6	209,172	**0.922**	0.628	**0.998**	**0.902**
5M_enhujp_v2_3	[3, 32, 1024]	50	0.0050	2	2	314,836	0.780	13.366	**0.968**	0.736
5M_enhujp_v2_2	[3, 32, 2048]	30	0.0050	2	3	391,741	0.767	10.594	**0.976**	0.716
5M_enhujp_v2_1	[3, 32, 1024]	50	0.0050	2	39	921,738	**0.920**	2.545	**0.989**	0.888
10M_enhujp_v3_1	[3, 32, 1024]	60	0.0010	0	3	179,147	0.483	6.961	0.704	0.448
5M_enhujp_v3_4	[3, 32, 512]	50	0.0001	0	18	408,688	0.750	4.010	**0.904**	0.714
5M_enhujp_v3_3	[3, 32, 512]	50	0.0001	0	14	322,937	0.690	5.179	**0.901**	0.678
5M_enhujp_v3_2	[3, 32, 512]	50	0.0001	0	3	116,924	0.300	2.998	0.574	0.272
4M_enhujp_pre_v3_1	[3, 32, 768]	24	0.0005	0	6	537,619	0.500	1.427	0.681	0.443
4M_enhujp_pre_v3_1	[3, 32, 768]	28	0.0005	2	1	57,629	0.429	2.108	0.668	0.381
4M_enhujp_v3_5	[3, 32, 768]	28	0.0010	0	3	176,068	0.429	2.305	0.737	0.399
4M_enhujp_v3_4	[3, 32, 768]	28	0.0005	0	3	159,235	0.464	1.903	0.704	0.423
2M_enhujp_v2_2	[3, 32, 1024]	64	0.0050	2	100	1,644,300	0.672	41.946	**0.912**	0.648
186k_hu_v2_1	[3, 32, 512]	128	0.0050	2	267	194,505	**0.961**	2.341	**0.990**	**0.927**

6.3. Project Results

6.3.1. English and Hungarian Multilanguage Experiments

Table 6 highlights the most important information and results of hybrid experiments (English–Hungarian) using the CRNN and SVTR models.

Table 6. English and Hungarian experiments.

	Experiment: CRNN Model	Experiment: SVTR Model
Architecture	CRNN	SVTR
Training data	training1	
Training data size	66 k	
Vocabulary	extended_vocab.txt	
Vocab size	201	
Epochs	100	
Max text length	250	150
Image shape (H × W)	32 × 100	48 × 320
Learning rate	0.0005	0.001
Regularization	None	0.00003
Best Train acc	0.26%	5.47%
Best Train loss	0.96	0.97
Best Distance score	11.92%	17.71%
Best Eval acc	0.16%	0.11%

6.3.2. Hungarian Single Language Experiments

Table 7 highlights the most important information and the results of the parameter adjustment experiments ('Training 1', 'Training 2') using Hungarian language vocabulary. Figure 7 shows the best training and evaluation accuracy of the Training 2.

Table 7. Hungarian with parameter checking.

	Experiment: Training 1	Experiment: Training 2
Architecture	CRNN	
Training data	hun_train	
Training data size	77 k	
Vocabulary	606k_hun_vocab.txt	
Vocab size	112	
Epochs	1224	673
Max text length	100	
Image shape (H × W)	32 × 128	32 × 512
Learning rate	0.005	
Regularization	0.000002	
Best Train acc	30.08%	94.92%
Best Train loss	0.002	3.623
Best Distance score	46.55%	98.46%
Best Eval acc	27.18%	85.73%

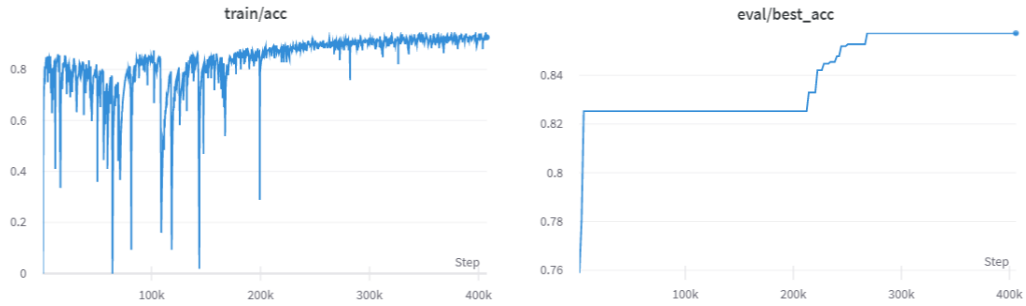

Figure 7. Best training and evaluation accuracy of Training 2.

Table 8 shows the most important information and the results of the parameter validation experiment (Hungarian).

Table 8. Hungarian with parameter validation.

	Experiment: 186k_hu_v2_1
Architecture	CRNN
Training data	training2
Training data size	186 k
Vocabulary	186k_extended_vocab.txt
Vocab size	112
Epochs	267
Max text length	100
Image shape (H × W)	32 × 512
Learning rate	0.005
Regularization	0.000005
Best Train acc	96.09%
Best Train loss	2.3410
Best Distance score	99.02%
Best Eval acc	92.75%

6.3.3. Hybrid Multilanguage Experiments

Table 9 highlights the most essential information and the results of the two multilanguage experiments: 'Experiment 1' (4M_enhujp_v3_4), 'Experiment 2' (4M_enhujp_v3_5).

Table 9. Multilanguage Experiments: 1 and 2.

	Experiment 1: 4M_enhujp_v3_4	**Experiment 2: 4M_enhujp_v3_5**
Architecture	SVTR	
Training data	training6	
Training data size	4.4 M	
Vocabulary	4M_min20_vocab.txt	4M_vocab.txt
Vocab size	3980	4721
Epochs	3	
Max text length	200	
Image shape (H × W)	32 × 768	
Learning rate	0.0005	0.001
Regularization	0.000015	0.00004
Best Train acc	46.43%	42.86%
Best Train loss	1.9033	2.3046
Best Distance score	70.41%	73.69%
Best Eval acc	42.29%	39.87%

Table 10 highlights the most important information and the results of the three multilanguage experiments: 'Experiment 3' (5M_enhujp_v3_2), 'Experiment 4' (5M_enhujp_v3_3), and 'Experiment 5' (5M_enhujp_v3_4). The best training and evaluation accuracy of 'Experiments 4 and 5' could be seen on Figure 8 and Figure 9 respectively.

Table 10. Multilanguage Experiments: 3, 4, and 5.

	Exp 3: 5M_enhujp_v3_2	**Exp 4: 5M_enhujp_v3_3**	**Exp 5: 5M_enhujp_v3_4**
Architecture	SVTR		
Training data	training7-100		
Training data size	5 M		
Vocabulary	jpn_latin_dict.txt		
Vocab size	4444		
Epochs	3	14	18
Max text length	110	100	
Image shape (H × W)	32 × 512		
Learning rate	0.0001		
Regularization	0.00005		
Best Train acc	30%	69%	75%
Best Train loss	2.998	5.1791	4.0096
Best Distance score	57.44%	90.10%	90.44%
Best Eval acc	27.24%	67.76%	71.35%

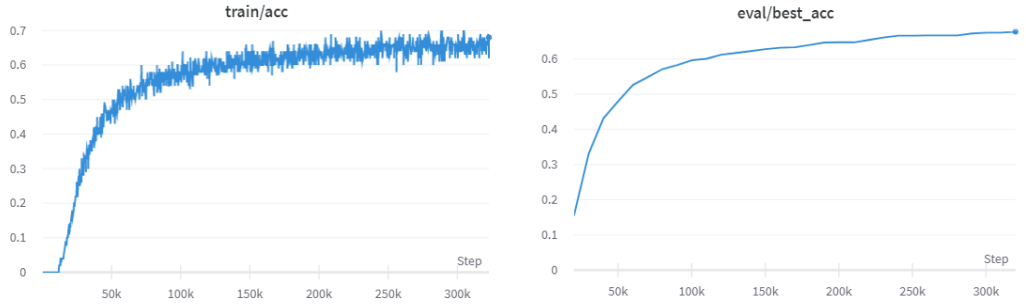

Figure 8. Best training and evaluation accuracy in 'Experiment 4'.

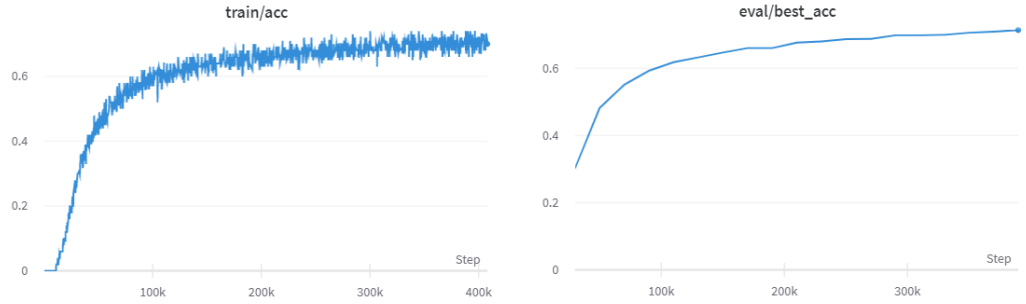

Figure 9. Best training and evaluation accuracy in 'Experiment 5'.

Table 11 highlights the most important information and the results of the two multilanguage experiments: 'Experiment 6' (4M_enhujp_pre_v3_1) and 'Experiment 7' (10M_enhujp_v3_1). The best training and evaluation accuracy of 'Experiments 6 and 7' could be seen on Figure 10 and Figure 11 respectively.

Table 11. Multilanguage experiments: 6 and 7.

	Exp 6: 4M_enhujp_pre_v3_1	**Exp 7: 10M_enhujp_v3_1**
Architecture	SVTR	
Training data	training6	training8-200
Training data size	4.4 M	8.5 M
Vocabulary	jpn_latin_dict.txt	
Vocab size	4444	
Epochs	6	3
Max text length	200	
Image shape (H × W)	32 × 768	32 × 1024
Learning rate	0.0005	0.001
Regularization	0.000015	0.0001
Best Train acc	50%	48.33%
Best Train loss	1.427	6.9611
Best Distance score	68.05%	70.35%
Best Eval acc	44.32%	44.81%

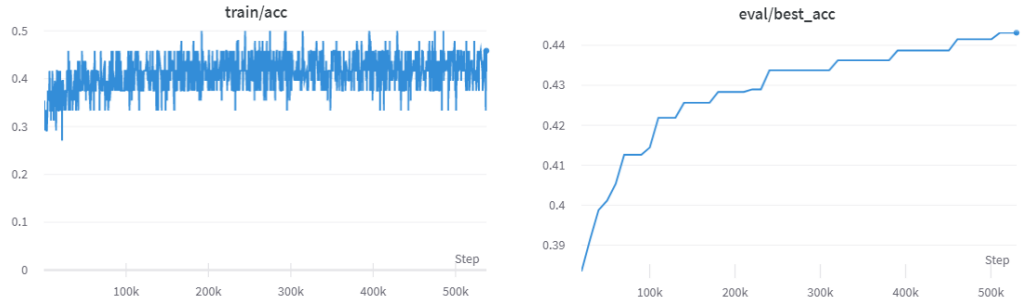

Figure 10. Best training and evaluation accuracy in 'Experiment 6'.

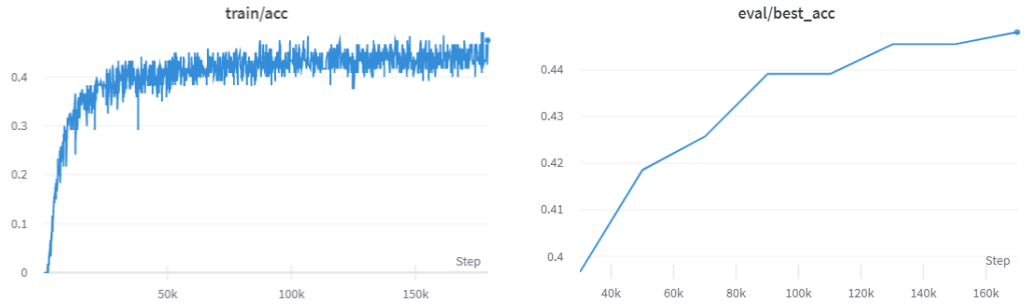

Figure 11. Best training and evaluation accuracy in 'Experiment 7'.

Table 12 highlights the most important information and the results of the three multilanguage experiments: 'Experiment 8' (5M_enhujp_v2_1), 'Experiment 9' (5M_enhujp_v2_2), and 'Experiment 10' (5M_enhujp_v2_3). The best training and evaluation accuracy of 'Experiments 8 and 10' as well as 'Experiment 9' could be seen on Figure 12 and Figure 13 respectively.

Table 12. Multilanguage Experiments: 8, 9, and 10.

	Exp 8: 5M_enhujp_v2_1	Exp 9: 5M_enhujp_v2_2	Exp 10: 5M_enhujp_v2_3
Architecture	CRNN		
Training data	training7-100	training8-200	
Training data size	5 M	8.5 M	
Vocabulary	jpn_latin_dict.txt		
Vocab size	4444		
Epochs	39	3	2
Max text length	100	200	
Image shape (H × W)	32 × 1024	32 × 2048	32 × 1024
Learning rate	0.005		
Regularization	0.000005	0.00005	
Best Train acc	92%	76,67%	78%
Best Train loss	2.5454	10.594	13.366
Best Distance score	98.87%	97.60%	96.81%
Best Eval acc	88.83%	71.56%	73.56%

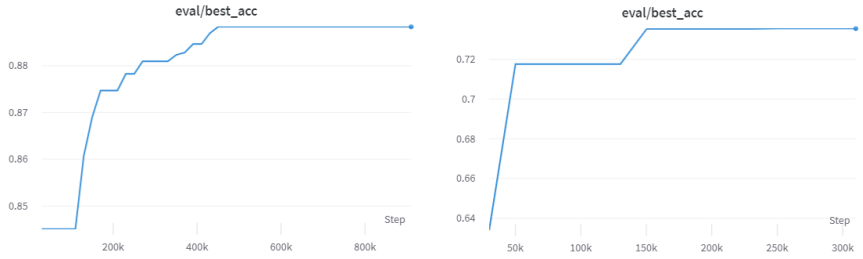

Figure 12. Best training and evaluation accuracy in 'Experiment 8' and 'Experiment 10'.

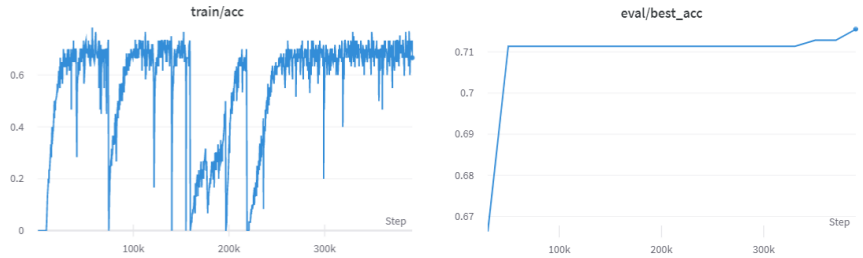

Figure 13. Best training and evaluation accuracy in 'Experiment 9'.

Table 13 highlights the most important information and the results of the four multilanguage experiments: 'Experiment 11' (5M_enhujp_v2_4), 'Experiment 12' (5M_enhujp_v2_6), 'Experiment 13' (5M_enhujp_v2_5), and 'Experiment 14' (5M_enhujp_v2_8). The best training and evaluation accuracy of 'Experiments 11, 12 and 13' could be seen on Figure 14, Figure 15 and Figure 16 respectively.

Table 13. Multilanguage Experiments: 11, 12, 13, and 14.

	Exp 11: 5M_enhujp_v2_4	Exp 12: 5M_enhujp_v2_6	Exp 13: 5M_enhujp_v2_5	Exp 14: 5M_enhujp_v2_8
Architecture	CRNN			
Training data	training9-200		training10-200	
Training data size	5 M			
Vocabulary	training9-200_vocab_min200.txt		training9-200_vocab_min9500.txt	
Vocab size	2737		803	
Epochs	6	31	4	26
Max text length	200			
Image shape (H × W)	32 × 1024		32 × 768	
Learning rate	0.005		0.0005	
Regularization	0.00005	0.0001	0.00005	0.0001
Best Train acc	92.19%	93.75%	88.54%	95.83%
Best Train loss	0.6278	0.5287	4.8713	0.2601
Best Distance score	99.76%	99.75%	97.65%	99.92%
Best Eval acc	90.25%	91.35%	85.02%	93.89%

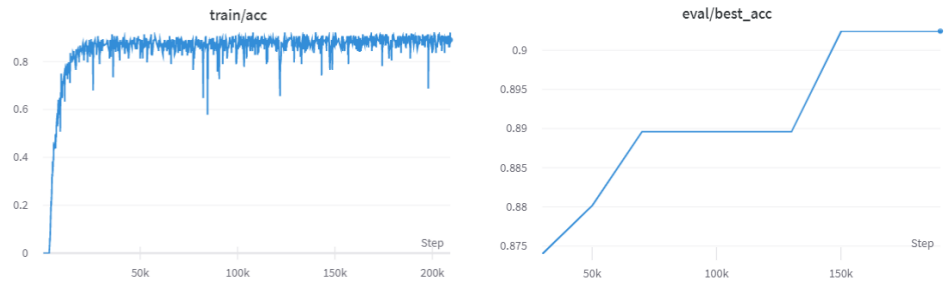

Figure 14. Best training and evaluation accuracy in 'Experiment 11'.

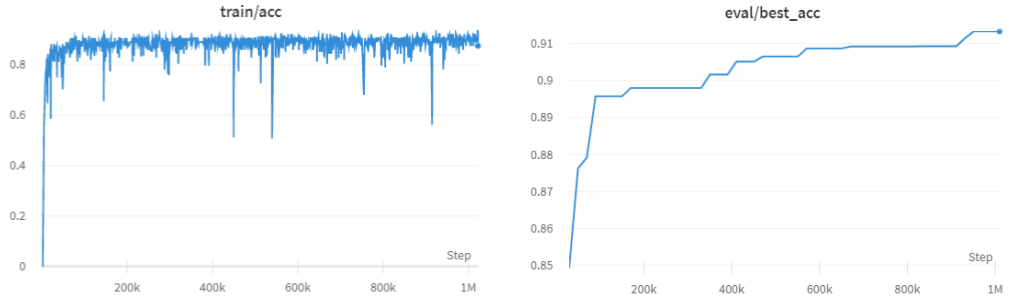

Figure 15. Best training and evaluation accuracy in 'Experiment 12'.

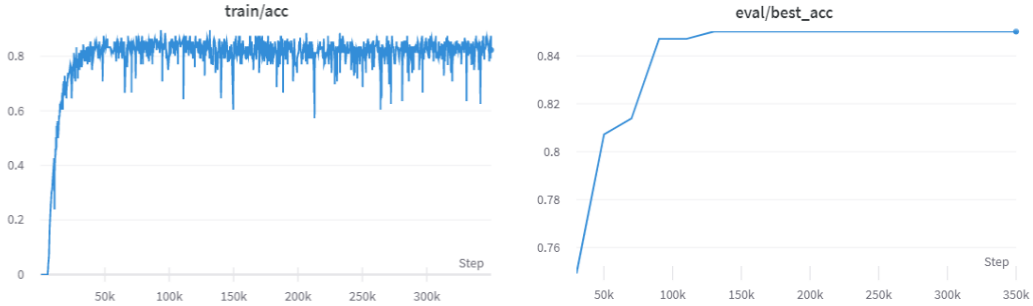

Figure 16. Best training and evaluation accuracy in 'Experiment 13'.

Figure 17 represents the training results of 'Experiment 11' (5M_enhujp_v2_4), 'Experiment 12' (5M_enhujp_v2_6), 'Experiment 13' (5M_enhujp_v2_5), and 'Experiment 14' (5M_enhujp_v2_8).

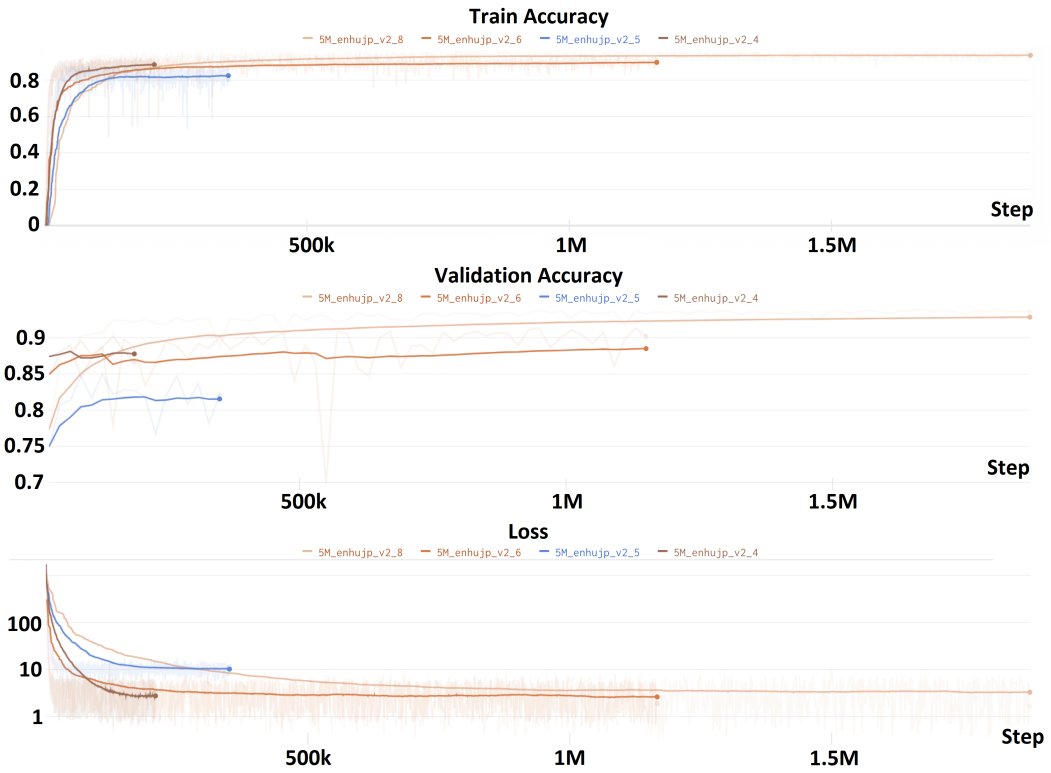

Figure 17. Comparison of experiment 11, 12, 13, and 14.

7. Discussion

The following method was applied in our experiment: (1) we analyzed both CRNN and SVTR models, (2) fine-tuned the hyperparameters, (3) localized the limitations of PaddleOCR, CRNN, and SVTR models, (4) prepared a method for custom vocabulary preparation, and (5) continuously adjusted trained models to achieve the expected goals.

7.1. Single and Hybrid Language Results

7.1.1. English and Hungarian Multilanguage Experiments

There were no promising results from these experiments with CRNN or SVTR. The reason for that might be a relatively small dataset (66k examples) combined with an extended maximum text length. The size of the preprocessed images is another significant consideration. The CRNN model indicates that there will be less than half a pixel for each character in a 100-pixel-width image, which means that 250 characters should appear in a 100-pixel-wide image. Furthermore, the SVTR-based model should have been configured to display 150 characters on 320 pixels, an average of around 2 pixels per character. Since the hyperparameters were only tested in these experiments, they were shut down relatively early.

7.1.2. Hungarian Single Language Experiments

These two experiments ('Training 1' and 'Training 2') used the CRNN architecture for testing purposes (vocabulary, parameters, image sizes), but with some modifications. The size of the dictionary is minimal relative to the amount of the training dataset based on our choice of parameters. Similarly to 'Training 1', we made the same mistake with the image size; we could not display the 1-pixel width for each character. In contrast, 'Training 2', which had a width of 512, was successful. Even though 'Training 1' failed this experiment, 'Training 2' achieved a relatively high score (see Figure 7). The graphs indicate that the accuracy of the assessment stopped increasing after half of the training period. The training data were insufficient for proper model training, but everything else was satisfactory. Experimental parameter validation: The accuracy of this training run improved significantly in relation to its predecessors. Since the final project requirements require a multilingual model, we decided to stop developing it.

7.1.3. Hybrid Multilanguage Experiments

The vocabulary used in both experiments ('Experiment 1' and 'Experiment 2') was relatively large (both were generated from training data). In contrast, the training dataset contains only 4.4 M examples. In addition, the image size might not be sufficient for the 200 long text lines since this would result in an average width of 3.2 pixels per character. These parameters suggest that these models can only reach an accuracy of approximately 40% based on these parameters.

The first experiment was conducted with the v3 _2 ('Experiment 3') model, and the accuracy and evaluation results improved; however, the training accuracy appeared to stop, so the experiment ended. Similarly, the other two models ('Experiments 4' and 'Experiment 5') could benefit from more training time. As a result—see Figures 8 and 9—of having more time to learn from the training data, the following two models were able to achieve significantly higher scores. The precision of the training and evaluation correlates with better deals, as demonstrated by the loss value and precision of the training. Similarly to the two previous ones, they also had to be stopped because their learning slowed down after a while. This could be attributed to their excessive vocabulary.

'Experiment 6' (4M_enhujp_pre_v3_1) was built on a pre-trained Japanese model [28] created by the PaddleOCR team. The majority of the characters in the vocabulary are Japanes, so we determined that the Japanese pre-trained model would be the most appropriate method. As shown in Figure 10, the model did not produce the expected results, and the reliability could not be further increased. PaddleOCR's fine-tuning strategy con-

tributed to this outcome. As a result, the user cannot cut down the top layer of the model and fine-tune it with new data; instead, the model's weights are trained.

'Experiment 7' (10M_enhujp_v3_1) has a big enough training dataset and the correct image size. However, it should be noted that the vocabulary used is relatively large. Moreover, this vocabulary is unified, which may contain unnecessary and missing characters. However, training was only performed briefly (see Figure 11), and training accuracy stopped increasing. Despite this, the accuracy of the evaluations has continued to increase simultaneously. As a result, the model cannot be trained by using the training data.

'Experiment 8' (V2_1) and 'Experiment 10' (v2_3) appear to have reached their maximum potential, based on the evaluation accuracy levels, which have not increased for some time. The evaluation data for these two experiments are shown in Figure 12.

The evaluation of the accuracy (see Figure 13) in 'Experiment 9' (v2_2) appears to be improving even though the training accuracy seems to be stagnant.

To date, 'Experiment 11' (v2_4) has shown the most promising results (see Figure 14), but after a few epochs, it reached its maximum potential. This is explained by a low learning rate.

However, with a slightly lower learning rate, as seen in 'Experiment 12' (v2_6), it seems that the experiment can still have some potential (see Figure 15).

Similarly to the previous two experiments, something similar happens to the first model, 'Experiment 13' (v2_5), which has an excessively high learning rate and stops improving after a while (see Figure 16). With a sufficiently small learning rate, our last experiment (v2_8) reached the best score.

8. Conclusions

As a result of the experiments discussed in this paper, we localized not only the key parameters of multilingual or hybrid (synthetically generated) language (with extended vocabulary) model training but also the limitations of PaddleOCR, time-stealing processes, contradictory documented behaviors, localizing the room of improvements as well. In the following, we enumerate the findings and outcomes:

1. Multiline OCR: As a result of the documentation provided by Paddle OCR and the initial research, it was unclear whether PaddleOCR was capable performing of multiline OCR. We validated that the framework has limited multiline capabilities based on dedicated experiments if the vocabulary is small enough (100–200 characters) and the training examples are carefully selected. To handle multiline capabilities, it is necessary to have a GPU-intensive environment and an extensive training dataset in case we have to use a large or/and extended vocabulary (e.g., when we involve Asian languages with special characters in the vocabulary). This means a long training period as well.
2. Vocabulary generation: There is no vocabulary generator tool in PaddleOCR. It is recommended to perform vocabulary generation manually or by using a dedicated tool.
3. SVTR could not outperform CRNN: PaddleOCR released a version of its software during the experimental period that supports the Visual Transformer-based SVTR model. According to PaddleOCR, this new model architecture may perform better than the previous one (CRNN). Several experiments were conducted with the proposed model architecture, which underperformed the previous model architecture each time. Having conducted several experiments, we decided to return to the CRNN model.
4. Character coding: A high-level view of OCR engines [42] shows that their operation is quite simple. Each character is identified by the model based on its appearance. However, some errors may be caused by this process. Let us assume that there are identically shaped characters that have different character codes. In the training process, the model will look at the given character and determine if it is one of the two options. The model will, however, indicate that it made a mistake if this character belongs to another character code. As a result, during the training process, the model

will make some errors that will manifest themselves during the inference process. The problem can be solved by identifying similar characters and changing them to the selected symbols. They are referred to as homoglyphs [43–45]. The use of homoglyph filtering [44] is recommended.

The main conclusion of machine learning models for special multilanguages, including languages with artificially made vocabulary, is that they perform consistently with high accuracy. The ML models recognized the multilanguage/hybrid texts with high accuracy. The highest precision scores were achieved with 'Experiment 11' (90.25%), 'Experiment 12' (91.35%), and 'Experiment 14' (93.89%).

Author Contributions: Conceptualization, A.B., S.M.S. and A.I.C.-V.; methodology, A.B., A.I.C.-V., J.M.-M. and S.M.S.; software, A.B.; validation, A.B., L.S. and S.M.S.; formal analysis, A.B., J.M.-M. and S.M.S.; investigation, A.B.; resources, A.B. and S.M.S.; data curation, A.B.; writing—original draft preparation, A.B.; writing—review and editing, A.B. and L.S.; visualization, A.B. and L.S.; supervision, A.I.C.-V., J.M.-M. and S.M.S.; project administration, A.B.; funding acquisition, A.B. and L.S. All authors have read and agreed to the published version of the manuscript.

Funding: This study was supported by ITware, Hungary. The work of A.B., J.M.-M. and A.I.C.-V. was supported by University of Malaga. The work of L.S. was supported by the Consolidator Excellence Researcher Program of Óbuda University, Budapest Hungary, and the Sapientia Institute for Research Programs, Romania.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are available upon request. Figures and the results charts of this study can be found at https://doi.org/10.6084/m9.figshare.22151651.v1 (accessed on 22 March 2023).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

OCR	optical character recognition
AI	artificial intelligence
CNN	convolutional neural network
CRNN	Convolutional Recurrent Neural Networks
SVTR	Single Visual model for scene Text Recognition
PaddleOCR	Multilingual OCR toolkits based on PaddlePaddle
WANDB	Weight and Biases
ML	machine learning
NN	neural network
GRU	Gated Recurrent Unit
LR	Learning rate
Adam	alternative optimization algorithm
COVID-19	Coronavirus disease
IT	information technology
LSTM	long short-term memory
RNN	Recurrent Neural Networks
EDA	Exploratory Data Analysis
JSON	JavaScript object notation
GPU	Graphics processing unit
CPU	Central processing unit
GB	gigabyte

References

1. Qaddumi, B.; Ayaad, O.; Al-Ma'aitah, M.A.; Akhu-Zaheya, L.; Alloubani, A. The factors affecting team effectiveness in hospitals: The mediating role of using electronic collaborative tools. *J. Interprofessional Educ. Pract.* **2021**, *24*, 100449. [CrossRef]
2. Biró, A.; Jánosi-Rancz, K.T.; Szilágyi, L.; Cuesta-Vargas, A.I.; Martín-Martín, J.; Szilágyi, S.M. Visual Object Detection with DETR to Support Video-Diagnosis Using Conference Tools. *Appl. Sci.* **2022**, *12*, 5977. [CrossRef]
3. Huang, J.; Pang, G.; Kovvuri, R.; Toh, M.; Liang, K.J.; Krishnan, P.; Yin, X.; Hassner, T. A Multiplexed Network for End-to-End, Multilingual OCR. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 4547–4557. [CrossRef]
4. Li, L.C.; Gao, F.Y.; Bu, J.J.; Wang, Y.P.; Yu, Z.; Zheng, Q. An End-to-End OCR Text Reorganization Sequence Learning for Rich-text Detail Image Comprehension. European Conference on Computer Vision. *LNCS* **2020**, *12370*, 85–100. [CrossRef]
5. Du, Y.N.; Li, C.X.; Guo, R.Y.; Yin, X.T.; Liu, W.W.; Zhou, J.; Bai, Y.F.; Yu, Z.L.; Yang, Y.H.; Dang, Q.Q.; et al. PP-OCR: A Practical Ultra Lightweight OCR System. *arXiv* **2020**, arXiv:2009.09941.
6. Du, Y.N.; Li, C.X.; Guo, R.Y.; Cui, C.; Liu, W.W.; Zhou, J.; Lu, B.; Yang, Y.H.; Liu, Q.; Hu, W. et al. PP-OCRv2: Bag of Tricks for Ultra Lightweight OCR System. *arXiv* **2021**, arXiv:2109.03144.
7. Nguyen, T.T.H.; Jatowt, A.; Coustaty, M.; Doucet, A. Survey of Post-OCR Processing Approaches. *ACM Comput. Surv.* **2021** *54*, 6, 124. [CrossRef]
8. Zhao, Z.P.; Zhao, Y.Q.; Bao, Z.T.; Wang, H.S.; Zhang, Z.X.; Li, C. Deep Spectrum Feature Representations for Speech Emotion Recognition. In Proceedings of the Joint Workshop of the 4th Workshop on Affective Social Multimedia Computing and first Multi-Modal Affective Computing of Large-Scale Multimedia Data, Seoul, Republic of Korea, 26 October 2018; pp. 27–33. [CrossRef]
9. Fischer-Suárez, N.; Lozano-Paniagua, D.; García-González, J.; Castro-Luna, G.; Requena-Mullor, M.; Alarcón-Rodríguez, R.; Parrón-Carreño, T.; Nievas-Soriano, B.J. Use of Digital Technology as a Collaborative Tool among Nursing Students—Survey Study and Validation. *Int. J. Environ. Res. Public Health* **2022**, *19*, 14267. [CrossRef]
10. Li, X.; Zhang, Y.; Yuan, W.; Luo, J. Incorporating External Knowledge Reasoning for Vision-and-Language Navigation with Assistant's Help. *Appl. Sci.* **2022**, *12*, 7053. [CrossRef]
11. Bulut, S.Ö. Integrating machine translation into translator training: Towards 'Human Translator Competence'? *Translogos Transl. Stud. J.* **2019**, *2*, 1–26. [CrossRef]
12. Bizzoni, Y.; Juzek, T.S.; España-Bonet, C.; Chowdhury, K.D.; van Genabith, J.; Teich, E. How human is machine translationese? Comparing human and machine translations of text and speech. In *Proceedings of the 17th International Conference on Spoken Language Translation*; Association for Computational Linguistics: Online , 2020; pp. 280–290. [CrossRef]
13. Zhang, B.; Bapna, A.; Sennrich, R.; Firat, O. Share or Not? Learning to Schedule Language-Specific Capacity for Multilingual Translation. In Proceedings of the International Conference on Learning Representations, Virtual, 3–7 May 2021; pp. 1–19. Available online: https://openreview.net/pdf?id=Wj4ODo0uyCF (accessed on 22 March 2023).
14. Shekar, K.C.; Cross, M.A.; Vasudevan, V. Optical Character Recognition and Neural Machine Translation Using Deep Learning Techniques. In Innovations in Computer Science and Engineering. Lecture Notes in Networks and Systems; Saini, H.S., Sayal, R., Govardhan, A., Buyya, R., Eds.; Springer: Singapore, 2021; Volume 171, pp. 277–283. [CrossRef]
15. Yang, J.; Yin, Y.W.; Ma, S.M.; Zhang, D.D.; Li, Z.J.; Wei, F.R. High-resource Language-specific Training for Multilingual Neural Machine Translation. *Int. Jt. Conf. Artif. Intell.* **2022**, 4436 –4442. [CrossRef]
16. Qi, J.W.; Peng, Y.X. Cross-modal bidirectional translation via reinforcement learning. *Int. Jt. Conf. Artif. Intell.* **2018**, 2630–2636. [CrossRef]
17. Shin, J.H.; Georgiou, P.G.; Narayanan, S. Towards modeling user behavior in interactions mediated through an automated bidirectional speech translation system. *Comput. Speech Lang.* **2010**, *24*, 232–256. [CrossRef]
18. Ding, L.A.; Wu, D.; Tao, D.C. Improving neural machine translation by bidirectional training. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*; Association for Computational Linguistics: Online; Punta Cana, Dominican Republic; 2021; pp. 3278–3284. [CrossRef]
19. Kaur, J.; Goyal, V.; Kumar, M. Improving the accuracy of tesseract OCR engine for machine printed Hindi documents. *AIP Conf. Proc.* **2022**, *2455*, 040007. [CrossRef]
20. Rijhwani, S.; Anastasopoulos, A.; Neubig, G. OCR Post Correction for Endangered Language Texts Pages. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), online, 16–20 November 2020; pp. 5931–5942. Available online: https://aclanthology.org/2020.emnlp-main.478.pdf (accessed on 22 March 2023).
21. Gunna, S.; Saluja, R.; Jawahar, C.V. Improving Scene Text Recognition for Indian Languages with Transfer Learning and Font Diversity. *J. Imaging* **2022**, *8*, 86. [CrossRef]
22. Ignat, O.; Maillard, J.; Chaudhary, V.; Guzmán, F. OCR Improves Machine Translation for Low-Resource Languages Pages. *arXiv* **2022**, arXiv:2202.13274.
23. Park, J.; Lee, E.; Kim, Y.; Kang, I.; Koo, H.I.; Cho, N.I. Multi-Lingual Optical Character Recognition System Using the Reinforcement Learning of Character Segmenter. *IEEE Access* **2020**, *8*, 174437–174448. [CrossRef]
24. Gifu, D. AI-backed OCR in Healthcare. *Procedia Comput. Sci.* **2022**, *207*, 1134–1143. [CrossRef]
25. Bartz, C.; Yang, H.J.; Meinel, C. STN-OCR: A single Neural Network for Text Detection and Text Recognition. *arXiv* **2017** , arXiv:1707.08831.

26. Lowe, A.; Harrison, N.; French, A.P. Hyperspectral image analysis techniques for the detection and classification of the early onset of plant disease and stress. *Plant Methods* **2017**, *13*, 80. [CrossRef] [PubMed]
27. PaddleOCR. Available online: https://github.com/PaddlePaddle/PaddleOCR (accessed on 22 March 2023).
28. Paddle Japanese Model—Japan Ultra-Lightweight OCR Model. Available online: https://github.com/1849349137/PaddleOCR (accessed on 22 March 2023).
29. Wu, H.; Prasad, S. Convolutional Recurrent Neural Networks for Hyperspectral Data Classification. *Remote Sens.* **2017**, *9*, 298. [CrossRef]
30. Du, Y.K.; Chen, Z.N.; Jia, C.Y.; Yin, X.T.; Zheng, T.L.; Li, C.X.; Du, Y.N.; Jiang, Y.G. SVTR: Scene Text Recognition with a Single Visual Model. In Proceedings of the 31st International Joint Conference on Artificial Intelligence Main Track, JICAI-ECAI Vienna 2022. *arXiv* **2022**, arXiv:2205.00159.
31. Kloft, M.; Stiehler, F.; Zheng, Z.L.; Pinkwart, N. Predicting MOOC Dropout over Weeks Using Machine Learning Methods. In *EMNLP Workshop on Analysis of Large Scale Social Interaction in MOOCs*; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 60–65. [CrossRef]
32. Al-Nabhi, H.; Krishna, K.; Abdullah, A.; Shareef, A. Efficient CRNN Recognition Approaches for Defective Characters in Images. *Int. J. Comput. Digit. Syst.* **2022**, *12*, 1417–1427. [CrossRef]
33. Kang, P.; Singh, A.K. CTC—Problem Statement. The AI Learner. 2021. Available online: https://theailearner.com/ (accessed on 22 March 2023).
34. Shi, B.; Bai, X.; Yao, C. An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2298–2304. [CrossRef] [PubMed]
35. Keren, G.; Schuller, B. Convolutional RNN: An Enhanced Model for Extracting Features from Sequential Data. In Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 24–29 July 2016. Available online: https://arxiv.org/pdf/1602.05875.pdf (accessed on 22 March 2023).
36. Wu, H.; Prasad, S. Semi-Supervised Deep Learning Using Pseudo Labels for Hyperspectral Image Classification. *IEEE Trans. Image Process.* **2018**, *27*, 1259–1270. [CrossRef] [PubMed]
37. Gan, Z.; Singh, P.D.; Joshi, A.; He, X.D.; Chen, J.S.; Gao, J.F.; Deng, L. Character-level Deep Conflation for Business Data Analytics. *arXiv* **2017**, arXiv:1702.02640.
38. Lee, T. EMD and LSTM Hybrid Deep Learning Model for Predicting Sunspot Number Time Series with a Cyclic Pattern. *Sol. Phys.* **2020**, *295*, 82. [CrossRef]
39. Zhuang, J.; Ren, Y.; Li, X.; Liang, Z. Text-Level Contrastive Learning for Scene Text Recognition. In Proceedings of the 2022 International Conference on Asian Language Processing (IALP), Singapore, 27–28 October 2022; pp. 231–236. [CrossRef]
40. Jung, M.J.; Lee, H.; Tani, J. Adaptive detrending to accelerate convolutional gated recurrent unit training for contextual video recognition. *Neural Netw. J.* **2018**, *105*, 356–370. [CrossRef]
41. Brownlee, J. Understand the Impact of Learning Rate on Neural Network Performance. *Deep. Learn. Perform.* **2019**. Available online: https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks (accessed on 22 March 2023).
42. Schneider, P.; Maurer, Y. Rerunning OCR: A Machine Learning Approach to Quality Assessment and Enhancement Prediction. *J. Data Min. Digit. Humanit.* **2021**, *2022*, 1–14. [CrossRef]
43. Almuhaideb, A.M.; Aslam, N.; Alabdullatif, A.; Altamimi, S.; Alothman, S.; Alhussain, A.; Aldosari, W.; Alsunaidi, S.J.; Alissa, K.A. Homoglyph Attack Detection Model Using Machine Learning and Hash Function. *J. Sens. Actuator Netw.* **2022**, *11*, 54. [CrossRef]
44. Majumder, M.T.H.; Rahman, M.M.; Iqbal, A.; Rahman, M.S. Convolutional Neural Network Based Ensemble Approach for Homoglyph Recognition . *Math. Comput. Appl.* **2020**, *25*, 71. [CrossRef]
45. Suzuki, H.; Chiba, D.; Yoneya, Y.; Mori, T.; Goto, S. ShamFinder: An Automated Framework for Detecting IDN Homographs. In Proceedings of the IMC'19: ACM Internet Measurement Conference, Amsterdam, The Netherlands, 21–23 October 2019; pp. 449–462. [CrossRef]