Team
Adonis

Optical Character Recognition on Captcha Dataset

INTRODUCTION

OCR (Optical Character Recognition) is the technology that enables the **conversion of images** containing printed or handwritten text into **machine-readable text.**

Several Machine Learning-based approaches have been proposed for OCR, that include utilizing object detection models like **YoLOv5 and DBNet for text localization**, and employing encoder-decoder architectures, often based on **CNNs and RNNs**, for text recognition. Recent advancements **integrate Transformers** for improved accuracy, with potential exploration of pre-trained CV and NLP models to further enhance performance.

PROBLEM STATEMENT

We were provided with the following problem statement :-

Given an image of captcha, you have to train a model which is capable of predicting the characters in that image correctly.

In order to solve this problem, the following data was made available to us:-

1. **test_images_mlware.zip** : This contained 5000 images on which we had to gather predictions using our OCR model.
2. **train_images_mlware.zip** : This contained the 25000 images with the help of which we trained our OCR model.
3. **train-labels_mlware.csv** : This .csv file contained the actual text present in each image belonging to the training set.
4. **sample_submission_mlware.csv**

PROBLEM STATEMENT

Vizualisation of a few training samples along with their respective labels

Label: 83CQNX

Label: 8Q2YM6

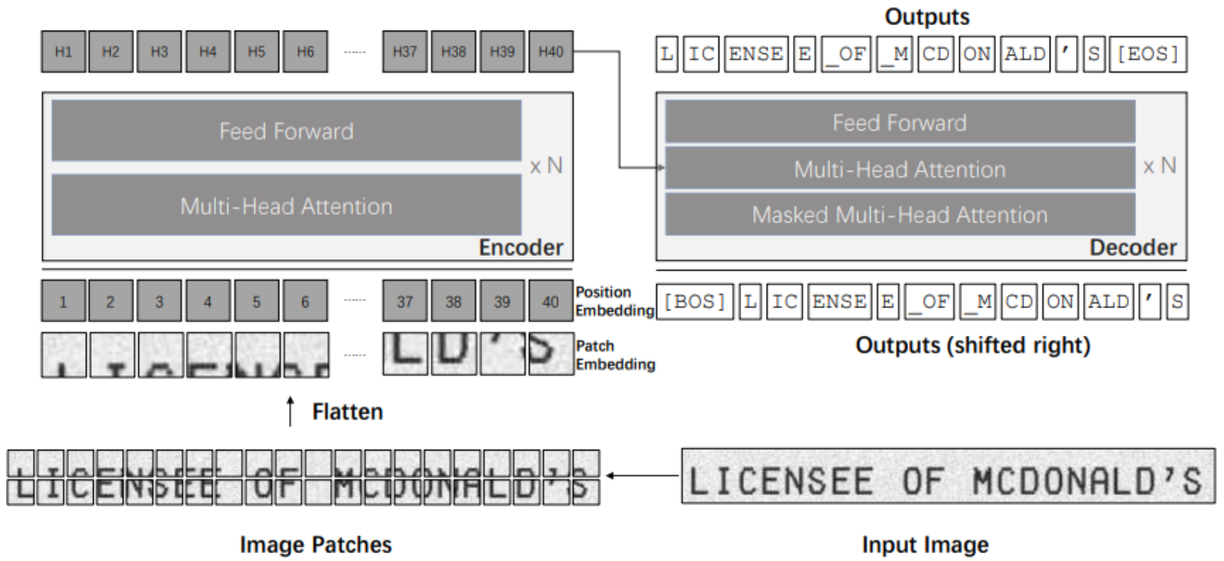
Label: H8SNND

Label: J8EMYU

STRATEGIES : TrOCR

TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models, Minghao Li,I et.al.

After exploring several methods, we found **TrOCR**, an end-to-end text recognition approach that employs **pre-trained image and text Transformer models**, to be a suitable fit for the task at hand. TrOCR excels in printed, handwritten, and scene text recognition by **integrating Transformer architecture for both image understanding and text generation**, emphasizing simplicity for superior performance.



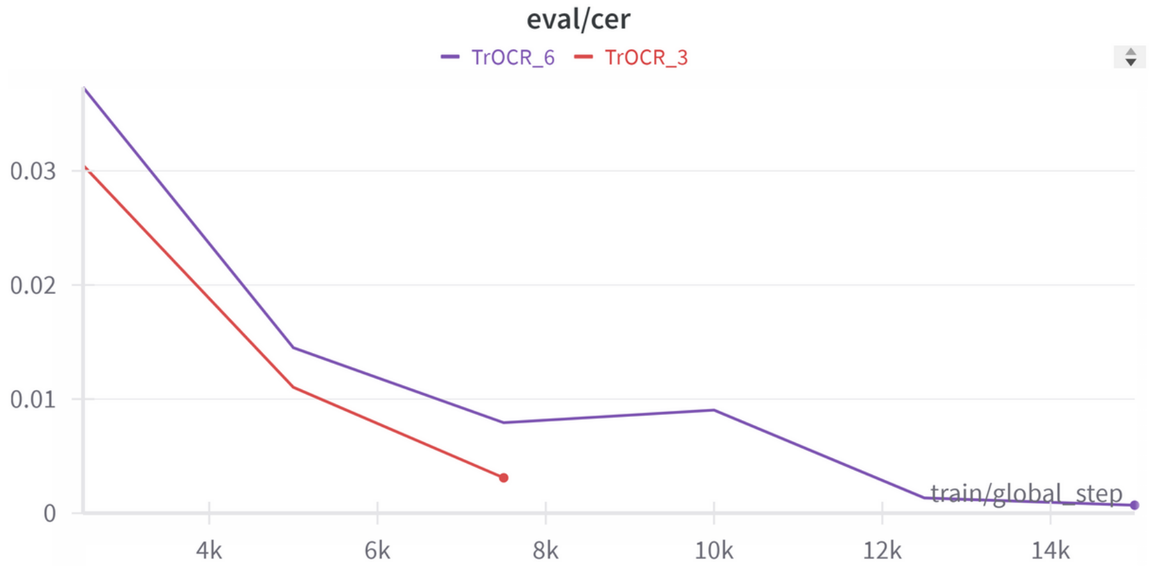
STRATEGIES : TrOCR

From the table below, we can see that TrOCR is the best candidate for our task, considering the fact that it outperforms several existing OCR models.

	Model	Year	Train data	IIIT	SVT	IC03		IC13		IC15		SP	CT	Time	params
				3000	647	860	867	857	1015	1811	2077	645	288	ms/image	$\times 10^6$
Reported results	CRNN [24]	2015	MJ	78.2	80.8	89.4	–	–	86.7	–	–	–	–	160	8.3
	RARE [25]	2016	MJ	81.9	81.9	90.1	–	88.6	–	–	–	71.8	59.2	<2	–
	R2AM [16]	2016	MJ	78.4	80.7	88.7	–	–	90.0	–	–	–	–	2.2	–
	STAR-Net [18]	2016	MJ+PRI	83.3	83.6	89.9	–	–	89.1	–	–	73.5	–	–	–
	GRCNN [28]	2017	MJ	80.8	81.5	91.2	–	–	–	–	–	–	–	–	–
	ATR [30]	2017	PRI+C	–	–	–	–	–	–	–	–	**75.8**	69.3	–	–
	FAN [4]	2017	MJ+ST+C	87.4	85.9	–	94.2	–	93.3	70.6	–	–	–	–	–
	Char-Net [17]	2018	MJ	83.6	84.4	**91.5**	–	90.8	–	–	60.0	73.5	–	–	–
	AON [5]	2018	MJ+ST	87.0	82.8	–	91.5	–	–	–	**68.2**	73.0	**76.8**	–	–
	EP [2]	2018	MJ+ST	88.3	**87.5**	–	94.6	–	**94.4**	**73.9**	–	–	–	–	–
	Rosetta [3]	2018	PRI	–	–	–	–	–	–	–	–	–	–	–	–
	SSFL [19]	2018	MJ	**89.4**	87.1	–	**94.7**	**94.0**	–	–	–	73.9	62.5	–	–
Our experiment	CRNN [24]	2015	MJ+ST	82.9	81.6	93.1	92.6	91.1	89.2	69.4	64.2	70.0	65.5	4.4	8.3
	RARE [25]	2016	MJ+ST	86.2	85.8	93.9	93.7	92.6	91.1	74.5	68.9	76.2	70.4	23.6	10.8
	R2AM [16]	2016	MJ+ST	83.4	82.4	92.2	92.0	90.2	88.1	68.9	63.6	72.1	64.9	24.1	2.9
	STAR-Net [18]	2016	MJ+ST	87.0	86.9	94.4	94.0	92.8	91.5	76.1	70.3	77.5	71.7	10.9	48.7
	GRCNN [28]	2017	MJ+ST	84.2	83.7	93.5	93.0	90.9	88.8	71.4	65.8	73.6	68.1	10.7	4.6
	Rosetta [3]	2018	MJ+ST	84.3	84.7	93.4	92.9	90.9	89.0	71.2	66.0	73.8	69.2	4.7	44.3
	Our best model		MJ+ST	**87.9**	**87.5**	**94.9**	**94.4**	**93.6**	**92.3**	**77.6**	**71.8**	**79.2**	**74.0**	27.6	49.6

STRATEGIES : TrOCR

It was important to first, **fine-tune TrOCR** so that it could be leveraged for our downstream OCR task for Captcha images. For this purpose, we leveraged the pre-trained model **available on HuggingFace**, and implemented a fine-tuning of the pre-trained model by using the Seq2Seq trainer. We made use of the **Weights and Biases** platform to monitor the model progress.

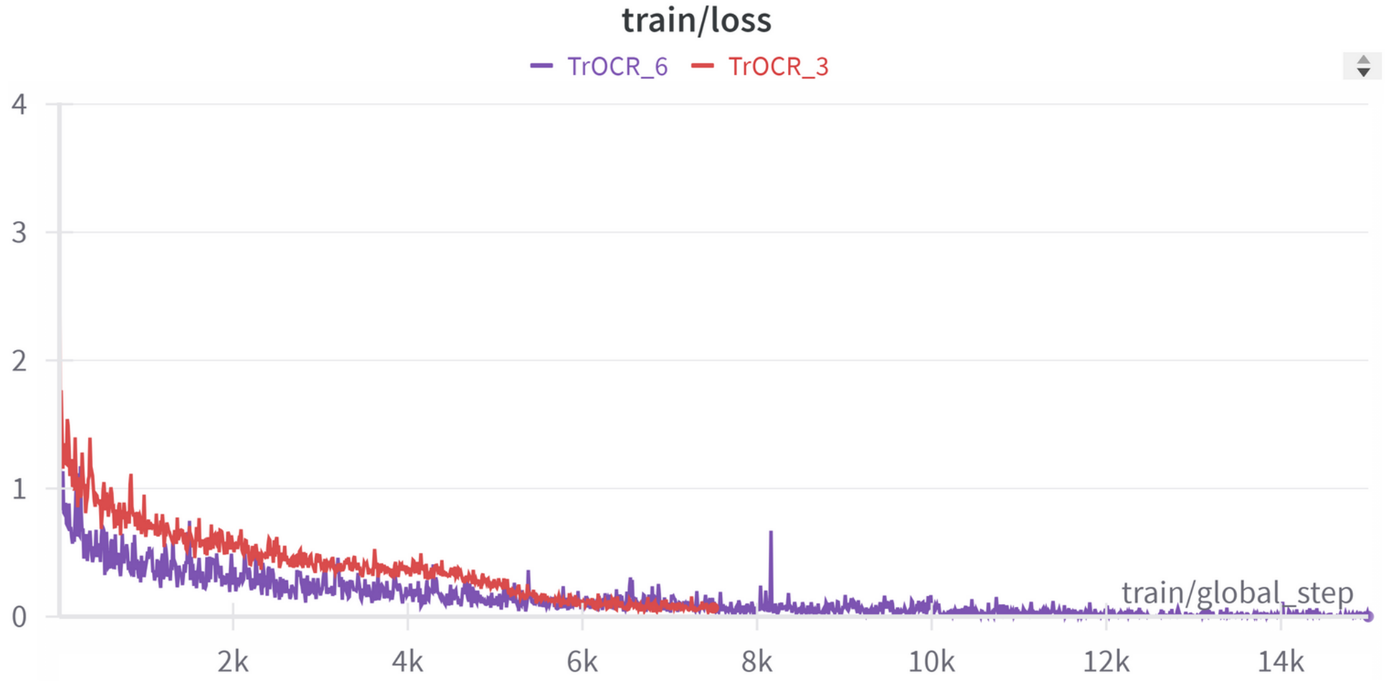


The plot shows the CER loss on the eval dataset. Increasing the number of epochs/time_steps clearly improved model performance, which indicates that model performance can be further improved to achieve even better results.

TrOCR_6 is the model fine-tuned for 6 epochs, while TrOCR_3 is the model fine-tuned for 3 epochs.

STRATEGIES : TrOCR

We implemented a **80-20 split between the training and the validation datasets**, and defined a **PyTorch class for the Dataset** for easy processing. First, we tried to fine-tune the model with only 3 epochs(TrOCR_3), which led to a CER loss of 0.02. After this, we fine-tuned our model for a total of **6 epochs(TrOCR_6), which took 585 minutes** (~10 hours) to finish compiling. TrOCR ended up as the **best performer** out of all the approaches that we implemented, achieving a best **CER loss of 0.00088** on the test image dataset.

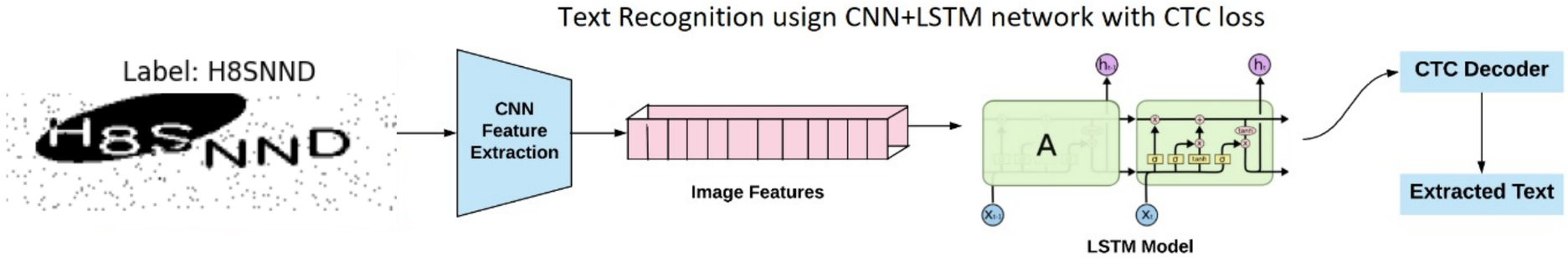


Epoch	Training Loss	Validation Loss	Cer
1	0.260000	0.262227	0.037333
2	0.141500	0.147858	0.014500
3	0.058200	0.116781	0.007933
4	0.060100	0.134317	0.009033
5	0.005500	0.073331	0.001333

Training data for TrOCR_6

STRATEGIES : CRNN

a CRNN model comprises **convolutional layers** for **feature extraction**, capturing local patterns and structures. **Recurrent LSTM cells** follow to **grasp sequential context** and inter-character relationships. These layers collectively enable the model to understand the text's context and make informed predictions.



Central to training is the Connectionist Temporal Classification (CTC) loss function. It facilitates aligning predicted sequences with ground truth labels, accommodating variable-length sequences without explicit alignment during training. This adaptability is crucial for recognizing text of varying lengths.

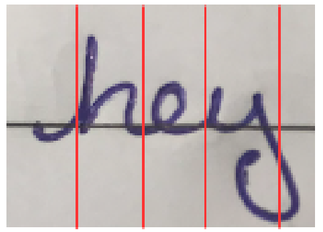


Image for annotation

-hey-

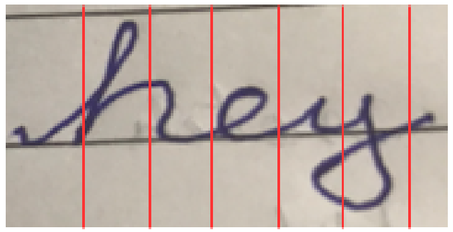


Image for annotation

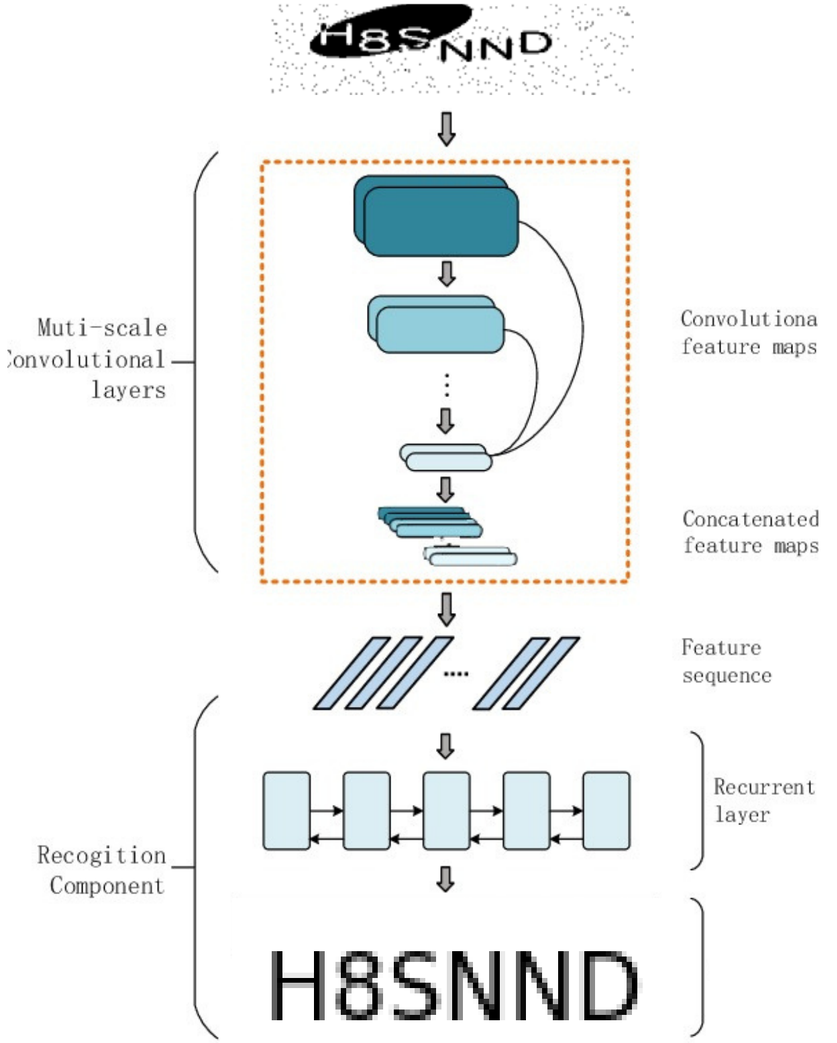
-hheyy-

STRATEGIES : CRNN

We used the Keras framework to create a sequential model of the following architecture; 5 convolution layers, followed by a Bidirectional LSTM and finally a dense layer to obtain the model output.

We trained this model for 100 epochs using early stopping and a learning rate scheduler to optimize the training process.

The CER loss plateaued at a value of 0.04, which was not comparable to existing state-of-the-art benchmarks. This became an incentive to try and implement state-of-the-art approaches.



STRATEGIES : Deep-Text-Recognition

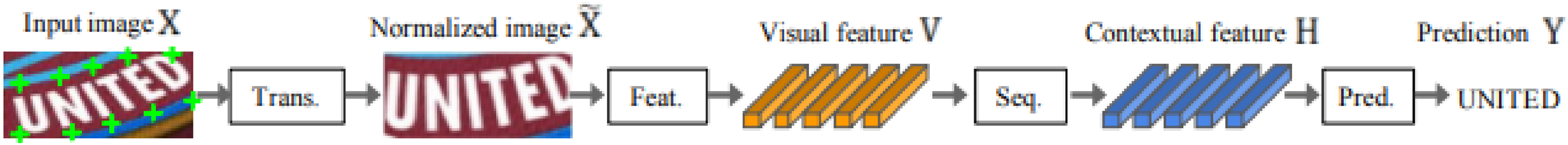
What Is Wrong With Scene Text Recognition Model Comparisons? Dataset and Model Analysis , Jeonghun Baek, et.al

1. Transformation (Trans.) normalizes the input text image using the Spatial Transformer Network to ease downstream stages.

2. Feature extraction maps the input image to representation that focuses on the attributes relevant for character recognition, while suppressing irrelevant
features

3. Sequence modeling captures the contextual information within a sequence of characters for the next stage to predict each character more robustly, rather than doing it independently.

4. Prediction estimates the output character sequence from the identified features of an image.



CONCLUSION

This problem statement allowed us to explore various machine learning-based methods for carrying out OCR. While our best performing model was able to achieve a CER loss of 0.00088, from our observations in the TrOCR section, we believe that the model can be scaled further and we can obtain even better results, had we trained for more epochs.

Owing to time constraints and the significant computational cost of finetuning on a large dataset, we could fine tune the model for six epochs. Other possible pre-trained models could have been selected for fine-tuning, and one could also change the hyperparameters and try for better results.

We thank the problem setters and the team of MLWare for providing us with an interesting problem statement that allowed us to learn immensely.

Thank You!