Noname manuscript No.
(will be inserted by the editor)

Conv-Transformer Architecture for Unconstrained Off-Line Urdu Handwriting Recognition

Nauman Riaz[1*†,2], Haziq Arbab[1*], Arooba Maqsood[1,2], Khuzaeymah Nasir[1], Adnan Ul-Hasan[2†], Faisal Shafait[1,2]

Received: date / Accepted: date

Abstract Unconstrained off-line handwriting text recognition in general and for Arabic-like scripts in particular is a challenging task and is still an active research area. Transformer based models for English handwriting recognition have recently shown promising results. In this paper, we have explored the use of transformer architecture for Urdu handwriting recognition. The use of a Convolution Neural Network before a vanilla full Transformer and using Urdu printed text-lines along with handwritten text lines during the training are the highlights of the proposed work. The Convolution Layers act to reduce the spatial resolutions and compensate for the n^2 complexity of transformer multi-head attention layers. Moreover, the printed text images in the training phase help the model in learning a greater number of ligatures (a prominent feature of Arabic-like scripts) and a better language model. Our model achieved state-of-the-art accuracy (CER of 5.31%) on publicly available NUST-UHWR dataset [1].

This research is partially funded by Higher Commission (HEC), Pakistan's grant for the National Center of Artificial Intelligence (NCAI).

* Authors contributed equally.
1 School of Electrical Engineering and Computer Science, National University of Sciences and Technology (NUST), Islamabad, Pakistan.
† Corresponding authors.

E-mail: {nriaz.mscs20seecs, adnan.ulhassan}@seecs.edu.pk

2 Deep Learning Laboratory, National Center of Artificial Intelligence (NCAI), Islamabad, Pakistan.

Keywords Urdu Handwriting Recognition · Urdu Language · OCR · Transformer · Beam Search

1 Introduction

Communication through written words differentiates humans from other species. It has remained an effective way of communication till date. Despite all technological advancements in speech to text and word processors, handwriting is still the most convenient way of jotting down thoughts, filling forms, and writing addresses.

Automatic text recognition is the process of converting text in images to corresponding editable text. Document digitization (i.e. converting to editable form) has several important applications in the real world. We can preserve our cultural heritage and knowledge of our ancestors for the future generations.

Apart from preservation of history and heritage, digitization plays an equally important role in automating several processes in our daily lives today. Postal automation can significantly reduce mail delivery times. Information extraction from documents like forms or medical records is helpful in developing digital databases and decision support systems. Therefore, text recognition, in general, has been an active area of research for past several decades.

Urdu is national language and among the two official languages of Pakistan. It is the 21^{st} largest first language spoken in the world, with around 61.9 million native speakers. It is usually written in Nastaleeq script and is a derivation from Arabic language. The majority of Pakistanis speak and understand it as their second language [2].

Printed text recognition is considered a solved problem and practical systems exists to reliably convert to editable text/digitize scanned documents in several languages (Google Vision API supports over 100 languages[1]).

Despite being natural to human beings and its wide spread used, it has been a challenging task for computers to recognize handwriting. Humans are highly creative when it comes to handwriting resulting in a vast diversity in writing styles, character formations, etc. Every person has their own writing style and training a model that can recognize an unseen handwriting style is a challenging task. Therefore, we must consider different aspects of writing such as writing styles, type of paper used, width of strokes, human error, and several other factors in addressing handwriting recognition.

Similar to Arabic language, the letters in Urdu scripts (Nastaleeq or Naskh) are joined together to form ligatures[2]. This makes Urdu text recognition highly context sensitive. Moreover, due to joining, some ligatures or characters overlap each others vertically. Some characters are very similar and thus they can be confused with other characters very easily (please refer to Figure 1). Urdu has over $24,000$ unique ligatures [1] and have different joining rules. These challenges are rendering Urdu text recognition as a highly complex task.

There are two major approaches to offline handwriting text recognition. The first is a segmentation-based approach. This approach isolates each letter, ligature or word and recognizes it individually [3]. However, this technique does not work very well especially for Urdu as its text is highly context sensitive [4]. The second approach is non-segmentation based. In this technique, text recognition is modelled as a sequence to sequence modeling task. This has been inspired by [5] for the task of Neural Machine Translation (NMT) [1,6].

There are existing text-recognition systems for symbolic and alphabet-based languages but no such systems exist for languages based on Arabic handwritten script including Urdu [7]. Attempts have been made to digitize information by manual transcription; however,

[1] https://cloud.google.com/vision/docs/languages

[2] https://www.w3.org/TR/alreq/

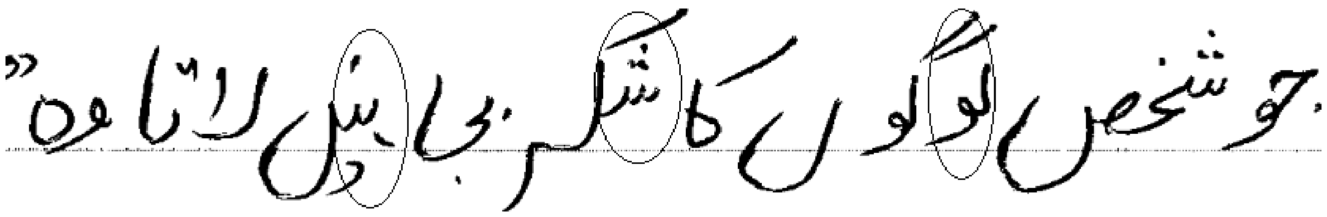


Fig. 1: This figure shows the complexity of a Urdu handwriting. One can see the characters overlapping each other vertically. Moreover, some characters can very easily be confused with other similar looking characters.

manually transcribing such a large volume of data is difficult, time-consuming, and costly [1].

Most research in the field of Urdu text recognition focuses on printed text [8–10], whereas handwriting recognition is wide open for new ideas. The majority of research in Urdu handwriting recognition revolves around stroke-based online handwriting recognition [11] in which touch sensitive devices such as mobile devices take advantage of touch input to recognize text through handwriting. We focus on offline Urdu handwriting recognition which involves the use of handwritten Urdu text images.

The major contribution of this paper is that we propose a CNN+Transformer (Conv-Transformer) architecture for the task of Urdu Handwriting Recognition. A Convolutional Neural Network (CNN) is used to extract the visual information from the image which is then fed to a full-transformer [5] having three encoder and decoder layers stacked on top of each other. The encoded sequence is passed to a transformer decoder that digitizes the handwritten text. The model works in an auto-regressive fashion due to the presence of a transformer. For testing, we proposed a beam search inspired technique to select the most probable outcome amongst the top-k probable output sequences.

The paper is mainly divided in the following sections. Section 2 gives a summary of the related work in the field of offline Urdu handwriting recognition. Section 3 discusses our proposed technique for the task at hand. Section 4 describes the experimental setup including preprocessing steps, data augmentations used, and implementation details. Section 5 provides the findings and their interpretation. Lastly, section 6 concludes the study and provides future research directions.

2 Related Works

Traditionally, Urdu recognition techniques are broadly categorized into holistic and analytical methodologies [6]. Holistic approaches refer to word-level recognition in Roman scripts, whereas in Arabic and Urdu, they refer to partial words or ligatures. Analytical procedures, on contrary, refer to the recognition at the character level. Both printed and handwritten writings are usually categorized in this manner. While recognition of printed Urdu text has improved over the years, the research on handwritten text recognition is still limited.

Sagheer et al. in [2] used the Support Vector Machine (SVM) model for Urdu text recognition. As the preprocessing steps, the images were converted into binarized and greyscale images. To eliminate the salt-and-pepper noise, the median filter was applied to the images. For feature maps, the structural features and gradient features were extracted. For the classification task, the SVM model with the RBF kernel was used. The dataset 'CENPARMI Urdu word dataset' named contained 14,407 samples for training and 3,770 samples for testing. Recognition performance of 97 percent was reported in the paper.

The techniques used in [1], [3], [6], [7], and [12] demonstrate how convolutional–recursive architectures can be used for the effective recognition of recursive text. Hassan et al. in [6] proposed an analytical approach in which the character segmentation is done implicitly using a Convolutional Neural Network as feature extractors, and the classification is done using a Bi-LSTM network. The input image is binarized first. The series of strokes sequences are then mapped according to the transcription. The feature maps are extracted using convolutional layers, which are then transformed into feature sequences and fed into the LSTM layer. The network architecture consists of 7 convolutional layers followed by pooling, batch-normalization and dropout layers in between, and two BiLSTM layers. An average character identification rate of more than 83 percent was obtained in experiments on a sample of 6000 distinct text lines. The dataset used for this study is the UNHD dataset [13]. Furthermore, the authors proposed to extend the work to the recognition of main ligatures separately to decrease the number of character classes. In another similar study, Zia et al. in [1] proposed a handwriting recognition model based on CNN-RNN architecture with an n-gram language modelling. The input layer size is increased from 100 to 128 pixels height to address the issue of limited resolution. Alongside the features are concatenated before being fed into the LSTM layer instead of using the max-pooling layer to eliminate the excessive dimensions. Moreover, the random distortion layer is added just before the input layer in order to distort the images randomly. The paper uses the interpolated n-gram model that combines the strength of lower-order and higher-order grams. To cater for the issue of preventing zero frequency of unknown words, the Kneser-Ney smoothing is applied. The proposed model gave a minimum Character Error Rate (CER) of 5.28 percent on a newly created dataset called 'NUST-UHWR'. This proposed architecture by Zia et al., uses CTC loss at the end which makes it a sequence labelling problem. Treating it as a sequence labelling problem further requires the use of n-gram language modelling to capture the probability of the next character given the previous n-characters. One draw back of this paper is that it is evident from models like LSTM, BERT [14], gpt-3 [15] etc that the task of language modelling is better captured using deep learning models than statistical approaches like n-gram. This motivates and inspires us to model handwriting recognition as a seq2seq task like neural machine translation. While in [3], Naz et al. used the same convolutional–recursive technique, which is state-of-the-art in printed text recognition. They used a 5 layered CNN model for extraction of generic ad abstract level features from MNIST dataset. These features are then fed into Multi-Dimensional Long-Short Term Memory (LSTM) for contextual features and classification. The proposed technique achieved 98.12 percent accuracy on the UPTI dataset. The authors of the paper proposed to extend this work to Persian and Arabic languages.

Similarly, Husnain et al. in [7] also used CNNs to recognize handwritten Urdu characters. For feature extraction, each Urdu handwritten character was analysed in order to extract the structural and geometrical information. These characteristics were then incorporated with the image's pixel-based data in order to produce reliable classification results. These features were then passed through 4 layered-convolutional networks. Finally, towards the end, the fully-connected layer is used for the classification. The paper reported an accuracy of 96.05 percent for character-level recognition of Urdu handwritten text. The dataset contained 800 images of 80 Urdu characters and 10 numerals.

In [12], to further aid the research in Handwriting Text Recognition(HTR), the authors considered three basic aspects of deep HTR systems and proposed effective solutions: 1) retain the aspect ratio of images during the preprocessing step, 2) use of max-pooling for converting the 3D feature map of CNN output into a sequence of features and 3) assist the training procedure via an additional CTC loss which acts as a shortcut on the max-pooled sequential features. The proposed model consists of a CNN backbone, followed by a Column Max-pool layer. After this layer, a recurrent head consisting of Bi-LSTM is used. Apart from the CNN backbone, the recurrent head, the authors also depict the auxiliary CTC shortcut branch which will be the core component of the proposed training modification. The paper also provides a comparison with baseline models on IAM and Rimes dataset. Overall, the proposed model achieved results close to the state-of-the-art models on both datasets.

The seq2seq models often suffer from errors like repeated or skipped words. The authors in [16] addressed this issue using the Connectionist Temporal Classification (CTC)-Prefix Scoring during S2S decoding. In this approach, the invalid paths are penalized during the beam search. The backbone of the proposed architecture is a CNN Block followed by a LSTM based encoder. The feature space is then decoded using a Transformer based decoder. For inference, the character costs of the CTC Prefix-Score, the S2S decoder, and LM are weighted and summed up to obtain the next best characters for decoding. The model was also evaluated on IAM, StAZH and Rimes dataset. The proposed architecture achieved a CER of 2.95% on IAM dataset.

To improve the results of previously mentioned techniques, an attention mechanism was used in [17] and [18]. The attention mechanism helps in achieving global reference for each word/pixel-level prediction. In [17], Michael et al. redesigned an attention-based seq2seq model for the task at hand inspired by the model proposed in [19] for neural machine translation. The model combines with CNN as a feature extractor and an RNN (three Bi-LSTM layers) to encode the temporal context and visual information in the input image, this is the encoder part of the architecture. For the decoder part, a separate LSTM (with 256 hidden units) is used to decode the actual character sequence. The attention is applied between the extracted features and the hidden state of the decoder. Positional embeddings are also injected in the input sequence as they provide relative positions of tokens in any sequence. The proposed architecture gave the minimum CER of 4.87 on the IAM dataset and 4.66 on the BOZEN dataset. In future, the authors aim to improve the encoder part of the architecture by using pre-trained models. While in [18], the authors proposed an end-to-end Transformer-based OCR (TrOCR) model. The input image is divided into patches, concatenated and then flattened to get an embedding matrix that can be fed into the Transformer based encoder. This architecture uses pre-trained image transformer as an encoder and pre-trained text transformer as the decoder. TrOCR treats the handwriting task as a seq2seq problem, where encoder is initialized by weights pre-trained on image net and decoder is initialized by weights pre-trained on wiki-text. The TrOCR model gave the minimum CER of 2.89 on the Synthetic and IAM datasets. Furthermore, the authors proposed to test this model on multi-lingual text recognition problems. The architecture is computationally complex due to the presence of pre-trained transformer encoder like DIET [20] and transformer decoder like Roberta [21]. The model heavily relies on its pre-training and gives good results on IAM dataset after fine tuning. Since decoder is pre-trained on wiki-text which is in the English language. The TrOCR hence becomes infeasible and gives poor results on the task of Urdu handwriting recognition.

In the field of natural language processing, transformer models have produced ground-breaking breakthroughs. But one disadvantage of Transformers is that they require a significant amount of training resources to achieve satisfactory results. The authors in [22] presented a Transformer-based light encoder-decoder architecture for the task of Handwriting Recognition that can be trained effectively on small datasets without the need for additional data. The number of training parameters were reduced to 6.9M as compared to the original 100M. The encoder is based on CNNs and decoder is purely Transformer based and acts as a Language Model. The authors trained the models with a hybrid loss combining both the Connectionist Temporal Classification (CTC) loss [23] and the Cross-Entropy (CE) loss. The model was tested on IAM dataset with and without additional data to compare the efficiency/performance of the proposed model. The proposed light-Transformer achieves results at the level of state-of-the-art Transformer-based models, with a 5.70% CER on the IAM test-set. And with using the synthetic data, proposed architecture is able to get a 4.76% CER. For future, the authors propose to apply this light-Transformer on historical documents.

3 Methodology

In this study, the handwriting recognition is treated as *Seq2Seq* modeling task inspired by the model proposed in [5, 19]. In [19], the authors proposed a full-transformer model to tackle the task of neural machine translation. The encoder and decoder architecture with all attention mechanism not only allows to capture the inter-language dependencies and alignments at embedding level (since attention mechanism does not have any weights) but also learns a language model for the translated language simultaneously. Taking inspiration from this, we also model the task of handwriting recognition as a *Seq2Seq* problem where the goal is to treat the image as a sequence and generate an output sequence

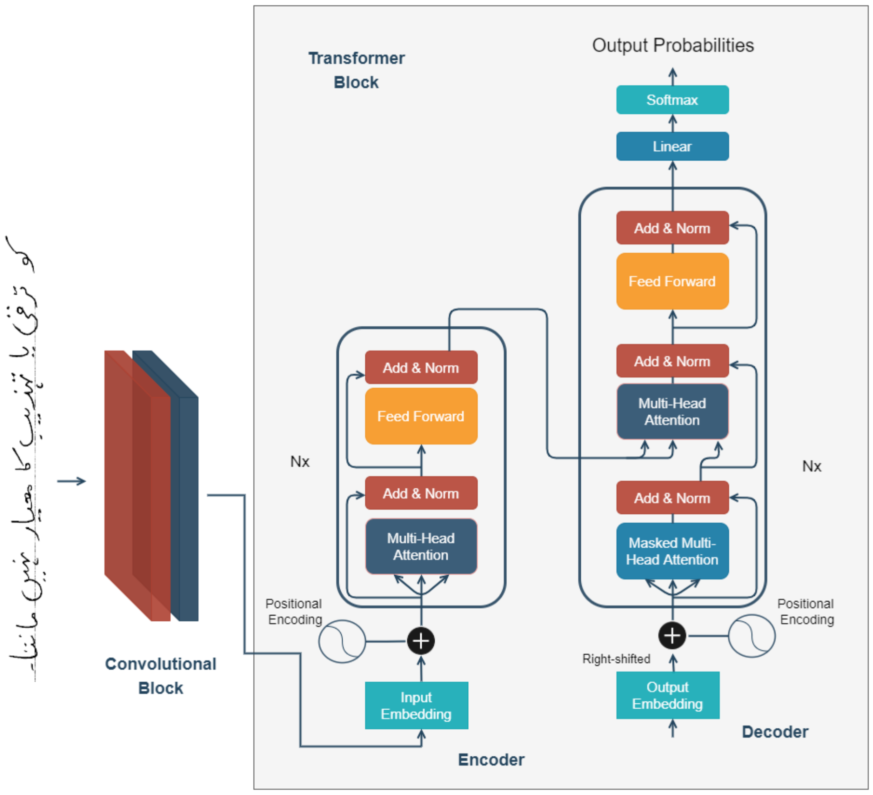


Fig. 2: The model architecture during training phase: It consists of convolution blocks followed by a full-transformer. The transformer contains three encoder-decoder blocks, hence N = 3 in our case.

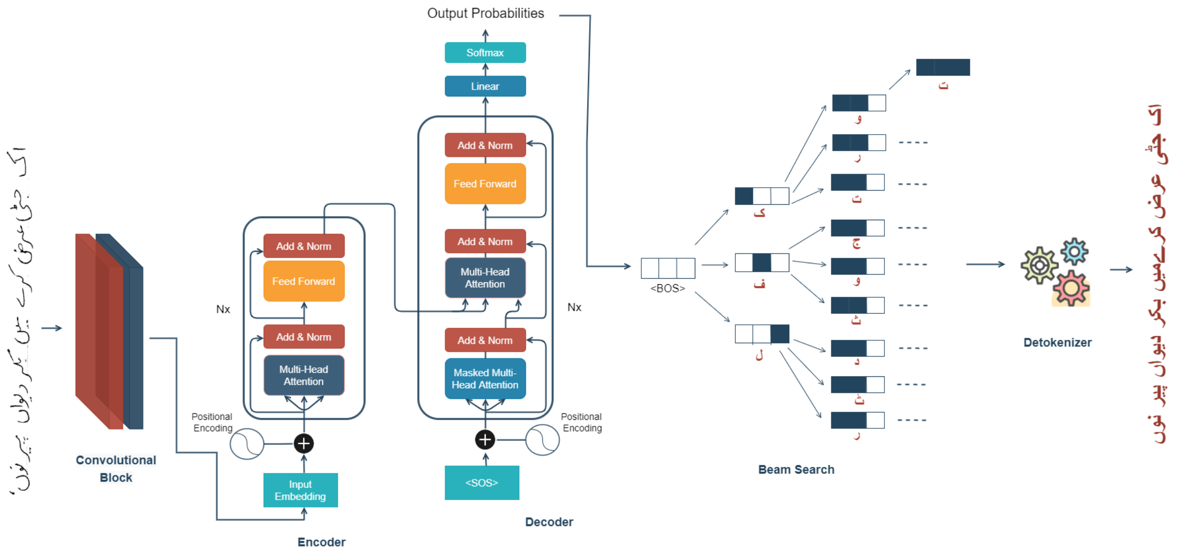


Fig. 3: The model architecture during testing phase: It consists of convolution blocks, a transformer and a beam search mechanism. A BOS (Beginning Of Sentence) token is passed through the decoder. It was mistakenly written as SOS.

of digitized text. It is pertinent to observe that transformer has an n^2 factor in its computational complexity due to the presence of multi-head attention layers, where 'n' is the sequence length [24]. This makes transformers computationally very slow or infeasible for very large sequences as is the case with the handwritten text images. In order to compensate for that, we propose a Conv-Transformer architecture. The convolutional layers at the start act to reduce the spatial resolution of the image and extract important features. After that, the feature maps are fed into a vanilla full transformer that then digitizes the input image. This architecture not only learns to digitize the input image but also learns a language model for the task due to the inherent property of how transformers work. Unlike [1], it does not require a separate n-gram language model.

For training the model, we use teacher forcing on the output sequence to converge the training faster (as shown in Fig. 2). Due to the presence of transformer architecture, the proposed model works in an auto-regressive fashion during the testing or inference phase. We do not use the right shifted labels at a stage of testing or inference. Given an image to be tested, we pass it through convolutional block and then the transformer encoder while a BOS (Beginning Of Sentence) token is passed through the decoder initially. The next character or token is predicted given BOS token and the feature maps extracted from the image. The predicted character is appended with BOS token and the process repeats until an EOS (End Of Sentence) token is encountered. This procedure is the decoding of the output sequence. We use beam search decoding which decodes the output sequence based on the best probability of the sequence (as shown in Fig. 3) rather than the best probability of the next character as in greedy decoding. Beam decoding gives superior results than greedy decoding.

The individual components of the proposed architecture are explained in following sections. Section 3.1 discusses the details of convolutional block. Section 3.2 discusses the significance of the transformer for the task of Urdu handwriting recognition. Section 3.3 discusses the beam search.

3.1 Convolutional Neural Network (CNN)

We used stacked CNN layers to extract visual features from the image as CNNs have a strong ability to learn task-specific features [25].Given a greyscale input image of the handwritten text of dimension $W \times H$, the CNN reduces it to $(S \times 1 \times d)$ where S is the width of the feature map having a depth of d after convolution layers. This is then reshaped to $S \times d\_model$ and fed to the transformer encoder where d_model is a hyperparameter and is treated as the input embedding dimensions to the encoder and S is the sequence length. In our case 'S' and 'd_model' are 397 and 256 respectively. The configuration [3] of convolutional blocks of our custom CNN are given in Table 1 and Fig. 4.

3.2 Transformer

Transformers were first introduced in [5]. The architecture is shown in the Fig. 2 which uses attention mechanism to capture long and short term dependencies. This architecture completely replaced RNNs and LSTMs, which struggled to capture long term dependencies due

[3] For details of each Conv. block's dimension, please refer to Fig. 4

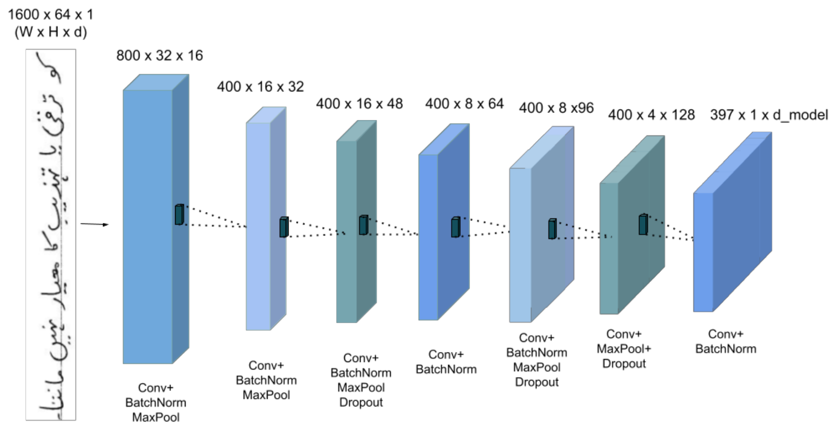


Fig. 4: Custom CNN layers (along with input sequence) used as feature extractor in the proposed architecture.

to vanishing gradient problems [19]. The transformer is an encoder-decoder architecture that uses *self-attention* on the encoder side and *causal-attention* on the decoder side. In the self-attention mechanism, every position in the input embedding attends to every other position whereas in the decoder, the causal-attention restricts the tokens to attend to the previous tokens only. Attention also takes place between the encoder and the decoder known as multi-head encoder-decoder attention. All in all the attention mechanism helps in the handwriting task as it allows the model to know which pixels to attend to in the image while generating a particular character.

The attention mechanism used by the transformer takes 3 input matrices that are Query (Q), Key (K), and Value (V). These are different representations of the input embedding after passing through dense or linear layers. Attention scores are evaluated by the dot product of the hidden states of encoder and decoder (In Encoder Decoder Attention), as shown in Eq. 1.

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

The dot-product is scaled by a factor of the square root of the depth of embedding. These are then converted into probabilities or attention weights using *Softmax*. The multiplication of the attention weights with V vector helps concentrate on positions that are to be focused while generating a particular character. V, K, and Q are split into multi-heads instead of a single attention head since it enables the model to collectively attend to information at various locations from different representational spaces.

In our proposed architecture, the encoder part of the transformer receives the embedding extracted from the CNN module (the feature maps). The embedding is then injected with positional information. We need to add some information about the locations to the input embedding because the transformer's encoder does not have repetition like recurrent neural networks. The positional encoding suggested in [5] has been employed in the proposed work. Then we have three encoder layers, stacked on top of each other, followed by three decoder layers. The number of layers in encoder and decoder

Table 1: Network configuration of custom CNN. (Given below, 'd_model' is the input embedding dimension to the transformer encoder and is being treated as a hyper-parameter. In our case, the value of 'd_model' is 256.)

Layer	Configuration
Conv	$1 \rightarrow 16$, 3×3
BatchNormalization	-
LeakyReLU	-
MaxPooling	2×2
Conv	$16 \rightarrow 32$, 3×3
BatchNormalization	-
LeakyReLU	-
MaxPooling	2×2
Conv	$32 \rightarrow 48$, 3×3
BatchNormalization	-
LeakyReLU	-
Conv	$48 \rightarrow 64$, 3×3
BatchNormalization	-
LeakyReLU	-
MaxPooling	(1,2) (2,1)
Dropout	0.2
Conv	$64 \rightarrow 96$, 3×3
BatchNormalization	-
LeakyReLU	-
Conv	$96 \rightarrow 128$, 3×3
BatchNormalization	-
LeakyReLU	-
MaxPooling	(1,2) (2,1)
Dropout	0.2
Conv	$128 \rightarrow$ **d_model** , 3×3
BatchNormalization	-
LeakyReLU	-

were chosen empirically. The right-shifted output tokens followed by embedding layer are fed to the decoder during the training phase. To predict the final output tokens, a linear layer followed by *Softmax* is used to project the decoder embedding of model dimension to the vocabulary size (v) dimension[4].

3.3 Beam Search

At inference, auto-regressive models must generate context to predict next tokens. This is unlike training when the right-shifted token is supplied as context from the output label. An intuitive initial solution to this problem is making use of greedy decoding. The most probable token is selected at a given index and used as context for further predictions. This approach however has a significant drawback, the output sentence produced may not necessarily be the most probable sequence as a whole given the input. Producing the most likely sentence involves generating all sentences possible and filtering for the most probable one. This is an NP-complete algorithm and hence impractical to implement. Not only does the computational size of this algorithm grow exponentially with each index, but the vocabulary also itself maybe tens of thousands to several million or billion words. For a vocabulary size v, at an index location n, there are v^n possible partially complete sequences each with their own probability[5]. The model must then be run on each of these partial sequences to generate partial sequences of size $n + 1$. This not only increases the computational complexity per iteration, the computational resources and time required to run each subsequent iteration increases exponentially as well. This however is the only algorithm that guarantees the model outputs the highest probable sequence at the end.

Beam search is a heuristic-based approach to make this algorithm more tractable. It captures the essence of the algorithm while remaining computationally feasible. The algorithm works by introducing a hyper parameter 'k', inferred as the number of beams. At the start of inference, the model sorts through 'v' possible

[4] To avoid the contradiction with the value (V) of the transformer equation, the vocab size is being represented as small v.

[5] https://www.width.ai/post/what-is-beam-search

choices for the leading character after the beginning of string token 'BOS' for the highest probable 'k' tokens. At each subsequent iteration after the first iteration, the model is run for each one of these 'k' possibilities and produces a set of k tokens for each one of the previous 'k' sequences for a total of k^2 possible sequences. These sequences are then ranked according to the scoring formula given in Eq. 2.

$$\frac{1}{n^{\alpha}} \sum_{i=1}^{n} (P(y_n)) = \text{Sum of ŷ of length n} \tag{2}$$

where 'n' is the length of the sequence, 'ŷ' is the ground truth label and 'α' is set to 0.7.

Log probabilities are utilized to prevent numerical underflow and a soft normalization is added with another hyper-parameter 'α' to prevent the model from strictly preferring shorter sequences. The best k-sequences are then selected, and the next iteration is performed. It is to be noted that sequence probabilities are only calculated up till the end of string token 'EOS', while the maximum length of the string is bounded, different inputs may correspond to outputs of different sequence lengths. The model keeps iterating past the EOS token as well and as beam search continues, a particular sequence that already had an EOS token at some previous index may be overwritten by a possible string from the current iteration given the string scores better than the complete sequence. For this reason, after each iteration, all k-sequences are scanned for an EOS token and if present, a separate cache stores the string along with the corresponding score and lets the algorithm continue. While an incomplete string may replace a string with an EOS token during the search, the final score that the incomplete string achieves after predicting an EOS token may be lower than the score of the string that was initially replaced. This is the reason for the presence of the before-mentioned cache. The cache ensures that only the best 'k' finished sequences over the run of the entire search are stored and kept till the end. Once the search reaches the final index location, the highest scoring entry from the cache is returned as the final output. Despite being fallible, this algorithm manages to outperform greedy search by a significant margin and captures the essence of the NP-complete algorithm while remaining computationally realizable and consistent. The model is only run k times at each iteration and there is breadth of the search tree stays limited to 'k' for the entirety of the run.

This paper utilized a custom implementation of beam search, mathematically given in Eq. 2. Traditional implementations generally compute $k \cdot v$ probabilities at each index and search for the best 'k'[6] whereas the approach here only computes $k \cdot k$ probabilities.

Furthermore, given the architecture of the model presented in the paper, the beam search is performed on a character level instead of a word level. The vocabulary size for character-level recognition is generally much smaller since it comprises particularly of unique characters and some ligatures in a language instead of an entire dictionary of the language. This leads to beam search performing even better and closer to the NP-complete algorithm since even moderate values of total beams are much closer to the actual vocabulary size and prove to be significantly more efficient than the num-

[6] https://www.baeldung.com/cs/beam-search

ber of beams being a very small fraction of the total vocabulary size.

4 Experiments

4.1 Dataset

For our experimental setup, we used the NUST-UHWR dataset [1]. The datasets contains image having a single line of Urdu Handwritten text along with their corresponding text labels. The images are unique and contain different text of different styles. The UHWR dataset was divided into training, validation and testing split as described in Table 2. The dataset consists of approximately 10,000 text lines, which are insufficient to train a reasonably good handwriting text recognition engine. The standard method of increasing the training samples is to introduce data augmentation; however, we argue that data augmentation methods are not helpful in training a text recognition system. In [26], the authors showed that ligature coverage has a positive impact in improving the accuracy of a text recognition system. It is specifically true for Arabic like scripts where the number of ligatures are huge. We further argue that printed text corpus can be helpful in training a handwriting recognition system as basic strokes of Urdu writing are similar whether they are handwritten or printed. We have demonstrated the effectiveness of our proposed hypothesis by comparing the results of the same architecture with both methods - 'traditional data augmentation' and 'handwriting plus printed text'.

Two printed text datasets were employed to augment the proposed handwriting recognition algorithm during the training phase. One is the recently proposed Urdu Ticker Text dataset [27] and the other is UPTI 2.0 dataset [1]. The statistics of these two datasets are given in Table 3.

To further test the efficacy of our proposed model on similar languages, we also trained and tested our model on the ADAB dataset [28], which is a handwriting dataset of Arabic language. The data contains text written in Arabic script and 937 Tunisian town/village names were used to create this data[7]. The splits used for this dataset are in the ratio 80:10:10 for training:testing:validation. These splits were generated randomly.

Before feeding the images into the model, few preprocessing steps were performed. To add diversity in the dataset and increase the number of samples, data augmentation techniques were also used. The first step is to formalize the dataset. In a nutshell, we translate data into a format that is simple to utilize. It helps reduce the model training CPU bottleneck. We convert all of the images to greyscale and resize them all to a set height of 64 *px* while maintaining the image aspect ratio. We pad the image width with zeros for batching up till max-length hyper-parameter (set to 1600).

During model training, we employ data augmentation to introduce diversity to the training datasets. Data augmentation can help improve the performance and results of any machine or deep learning model. A total of 10 augmentation functions were used, as suggested by Rehman et al. in [27]. Four of them are color-based augmentations, while the other ones are shape-based augmentations. The image is transformed into multiple dimensions using shape-based augmentations. The augmentation functions include *color inversion, padding*

[7] https://ieee-dataport.org/open-access/adab-database

Table 2: UHWR dataset statistics

Total no. of samples in UHWR dataset	$10,606$
No. of samples used for training	$8,484$
No. of samples used for testing	$1,061$
No. of samples used for validation	$1,061$

Table 3: UPTI-2 and Ticker dataset statistics (used only for Training)

Total no. of samples in Ticker dataset	$19,437$
Total no. of samples in UPTI-2 dataset	$1 million$

the image, adding color correction, adding soft noise to the image, slightly blurring the image, squeezing, degradation effect, rotation of axis, compression of artifacts, and *re-scaling the chunks of images.* Few examples of data augmentations are shown in Fig. 5.

4.2 Implementation Details and Hyper-parameters

The proposed architecture was implemented using PyTorch. The CNN was implemented with leaky-ReLU activation function as it is one of the standard practices[8] and batch normalization for faster convergence as shown in Table 1. The spatial resolution of feature maps was reduced using max pooling layers only whereas the convolutional layers retained their spatial resolution using padding. For transformer, we used 3-encoder and 3-decoder layers as this setting gave us the best results in less computational time. Other settings were tested with different number of encoder and decoder layers. Using higher number other than 3 for encoder and decoder layer of transformer did not yield any improvement in performance. After the transformer, a linear layer was used to transform the output to the shape $(B \times Sq \times v)$, where 'B' is the batch size, 'Sq' is the output sequence length and 'v' is the vocabulary size. In our case, the vocabulary size is the total number of characters encountered in the training data due to the size limitations of our dataset plus the special tokens like *PAD* (padding), *BOS* (Beginning Of Sentence), *EOS* (End Of Sentence) and *UNK* (Unknown character) as we are performing character level handwriting recognition. *Softmax* followed by cross-entropy loss was used for training and validation. We carried out the training of our architecture on a single Nvidia RTX 3080 GPU. Batch size of 16 was used where the right shifted output sequence length was padded with pad token up till the sequence with maximum length in a batch. We used Adam optimizer for the training of our architecture. Learning rate of 0.0003 was used with betas (0.9, 0.98) and epsilon $1e-9$, which are hyper-parameters of the Adam optimizer. Other settings of learning rate diminished the training by either diverging the loss for a higher learning rate or slow convergence for lower learning rates.

4.3 Experiments Performed

The experimentation includes the mixing of different datasets with augmentation techniques in order to analyze the effect on CER (Character Error Rate) of UHWR validation and test split. Printed and handwritten Urdu text datasets were mixed to get more diversity of the Urdu language. This was done in-order for the proposed architecture to learn a better language model. Results in Section 5 verify that printed text aids the model to capture more diversity in the urdu language. Firstly, our model was trained on UHWR train split alone.

[8] https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

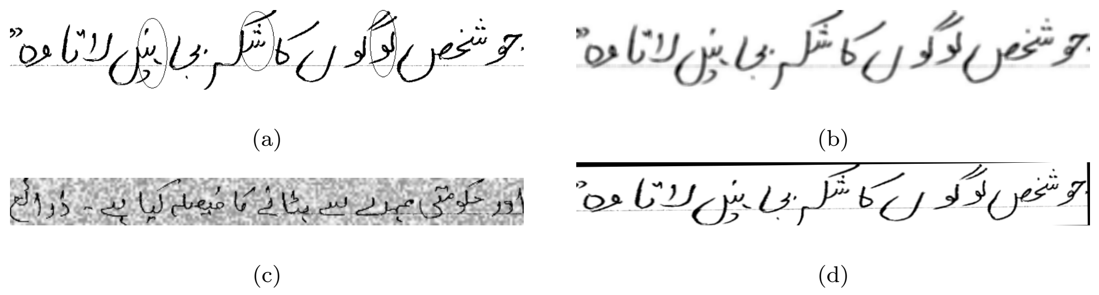


Fig. 5: The figure shows different data augmentation techniques used **(a)** original image **(b)** compression of artifacts and blur effect. **(c)** color inversion, soft-noise and degradation. **(d)** squeezing, rotation and re-scaling effect. The augmentation used for training is a combination of several augmentation techniques yielding significantly different images than original image.

Then we added complete Ticker printed text dataset and to analyze further we added more data including the complete UPTI-2 data for training. The three setups for training include:

1. UHWR train split
2. UHWR train split + Ticker
3. UHWR train split + Ticker + UPTI-2

The addition of more data from different distribution of printed text lead to drop in CER for UHWR validation and test splits, which shows that the transformer indeed learns a language model besides digitizing the input image.

Additionally, to test the effectiveness of our proposed approach in other inflectional languages such as Arabic for offline handwriting recognition, we also performed an experiment using the ADAB dataset.

5 Results and Discussion

5.1 Results

For Urdu Handwriting Recognition, the Character Error Rate (CER) of 6.0% and 6.4% was achieved on UHWR validation and test splits respectively after training our architecture on UHWR train split data. These results were further improved when printed Urdu Handwriting datasets were added with the UHWR train split for training (see Table 4 for details). The combination of printed and handwritten text datasets add more diversity of Urdu Language to the data. This diversity is captured by our architecture and a better language model is learnt as a part of the transformer.

And for the experiment based on Arabic offline handwriting recognition, our proposed model outperformed the Zia et al.'s model in [1]. For our Conv-Transformer model, we achieved the Character Error Rate (CER) of 2.3% and 2.0% on ADAB dataset validation and test splits respectively. (For comparisons, see Table 4). From these results, we can draw a conclusion that our proposed model has the capacity to recognize the complex Nastaleeq script of both Urdu and Arabic language.

5.2 Comparison with Conv-Recursive Architecture

Our proposed architecture is thoroughly compared with the convolutional recursive architecture proposed by Zia et al. in [1], which is the state of the art in Urdu handwriting recognition. The results given in Table 4 show that we beat the state of the art with a margin.

The architecture proposed in Zia et al. [1] uses a separate n-gram word level language model with a character level convolutional recursive deep learning model. Proper reasoning of how these two models generate results is missing so it is a possibility that the n-gram word level language model overrides the predictions of character level deep learning model at the end resulting in reduction in CER from 7.42% without LM to 5.49% with LM on the test set. Overriding the results with an n-gram language model would definitely yield better results over the test dataset since the text uses proper Urdu language words.

Given a handwritten text with random letters, the model proposed by [1] with a separate n-gram would fail. Our proposed architecture uses convolutional transformer that models the problem as the probability of next character given previous character and feature map extracted from image i.e $P(n_c|p_c, c)$ where 'n_c' is the next character, 'p_c' is previous character and 'c' is the feature map extracted from the input image thorough convolutional layers. This performs two learning task simultaneously, i.e., digitizing the input image and learning a character level language model.

The comparison of CER between the two models in Table 4 shows that our proposed architecture outperforms the current state of the art without having a need for a separate language model. Moreover, as more data is added to the UHWR dataset even though belonging to a different distribution of printed text, the model performs significantly better than the state of the art since more data enables the transformer to learn a better language model.

5.3 Comparison with Google's Vision API

Google vision has recently provided an experimental API for Urdu handwriting recognition[9]. We tested this API on UHWR test and validation splits. The results were worse than the state-of-the-art. Google Vision API gave CER of 26.5% and 27.8% on UHWR validation and test splits respectively. The Google vision API model may not have been trained on UHWR dataset and thus we tested the vision API on some random images and compared the results with Zia et al. [1] and our proposed architecture. The details are given in the Section 5.4.

5.4 Smoke Testing on Random Images

Testing on random Urdu handwriting images was performed to check the generalization capabilities of the Zia et al. [1], Google vision API and our proposed architecture. The images were collected randomly by making few individuals write an Urdu script on blank piece of white paper. The scanned images of these handwriting were used for smoke testing. Two test images were selected and fed into each model for predictions. The Fig. 6 shows qualitative results of smoke testing where a comparison between Google vision API, Zia et al.'s model and the proposed model is given. From these results, it is evident that our proposed architecture gave the best CER on these images. As the images for smoke testing was selected at random, we can draw a conclusion that our proposed model generalizes well on hand-

[9] https://cloud.google.com/vision

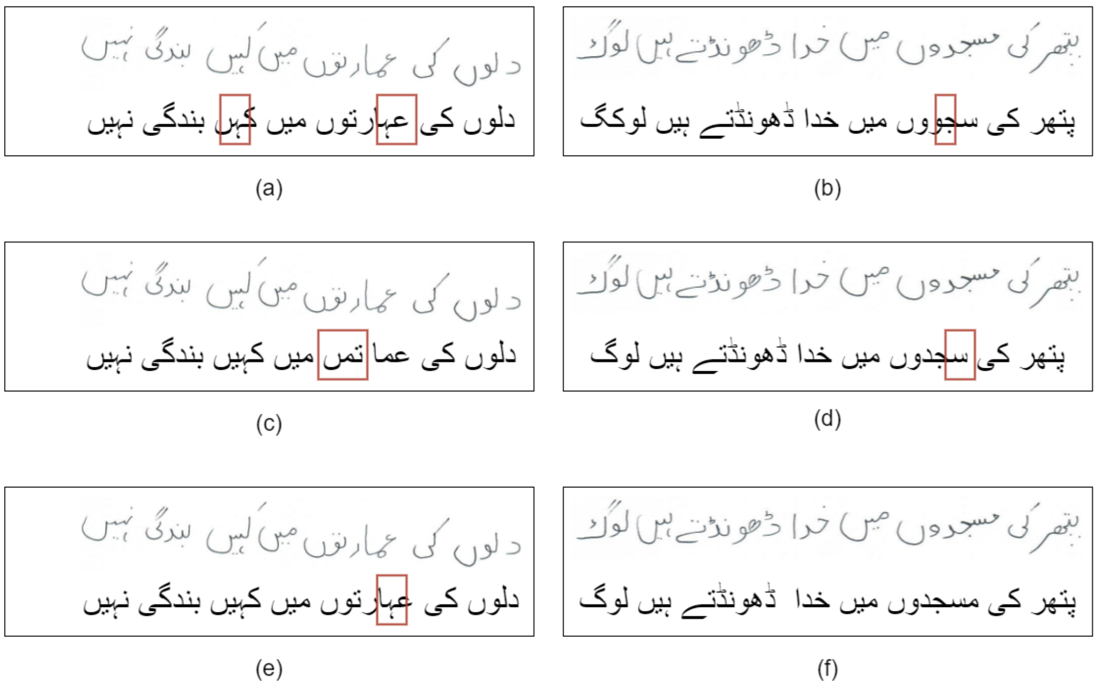

دلوں کی عہارتوں میں کہں بندگی نہیں
(a)
پتھر کی سجوں میں خدا ڈھونڈتے ہیں لوکگ
(b)
دلوں کی عما تمں میں کہیں بندگی نہیں
(c)
پتھر کی سجدوں میں خدا ڈھونڈتے ہیں لوگ
(d)
دلوں کی عہارتوں میں کہیں بندگی نہیں
(e)
پتھر کی مسجدوں میں خدا ڈھونڈتے ہیں لوگ
(f)

Fig. 6: The figure contains the results of Smoke study (as discussed in Section 5.4). Fig (a) and (b) contains results of Google Vision API with CER of 7.8% and 5.7% respectively. Fig (c) and (d) contains results Zia et al. [1]'s model with CER of 5.2% and 5.7% respectively. Fig (e) and (f) contains results of our proposed model with CER of 2.6% and 0% respectively.

written text regardless of complex writing style or overlapping characters in the sample image.

5.5 Ablation Studies

Ablation study was also performed on the architecture in order to test the contribution of the transformer decoder in learning a language model that reduces CER on UHWR test set. Moreover, we also perform ablation study to test the contribution of Conv layers as well in the performance of our architecture.

5.5.1 Ablation Study - Only encoder CTC

We completely removed the decoder layers in our architecture and used only the Conv plus transformer encoder to train on UHWR dataset. We used CTC loss as in Zia et al. [1]. This model is similar to Zia et al. [1] with difference of transformer encoder in-place of Recurrent neural networks like GRU. With this setting it was evident from validation and testing results that the model was performing similar to Zia et al. [1] without n-gram language modelling. The conv. and transformer encoder gave us the CER of 7.28% on UHWR validation split and 7.4% on test split. We used same hyper parameter settings as in the case of our full conv-transformer architecture with the change in the loss function i.e. CTC loss.

5.5.2 Ablation Study - Encoder decoder

We also tested our architecture by removing the Conv Layers from it in order to test the impact of Convolution Network as whole before a full transformer. Image was directly fed to the transformer Encoder after positional embeddings. This setting gave us better results than Zia et al. [1] but it was harder for the model to converge during the training. We got a CER of 6.97% and 7.1%

Table 4: This table gives a comparison of CER of UHWR valid and test splits between Conv-Recursive [1] (current state-of-the-art) and our proposed model for Urdu handwriting recognition. For conv-recursive model, n-gram Language Model (LM) was trained on UPTI-2 dataset only.

Dataset (used for training)	Conv-Recursive Model [1]		Conv-Transformer Model (Proposed)	
	valid CER	test CER	valid CER	test CER
UHWR train split	7.25% (no LM)	7.35% (no LM)	**6.0%**	**6.4%**
UHWR train split + Ticker	8.15% (no LM)	8.3% (no LM)	**5.35%**	**5.5%**
UHWR train split + UPTI-2	5.28% (with LM)	5.49% (with LM)	**5.12%**	**5.34%**
UHWR train split + Ticker + UPTI-2	5.27% (with LM)	5.5% (with LM)	**5.14%**	**5.31%**
(Arabic) dataset	5.2% (no LM)	5.0% (no LM)	**2.3%**	**2.0%**

on UHWR validation and test splits respectively. Given sufficient data, a full transformer without conv layers can be used for training and testing. With the limited data that we have, convolution layers play a major role in giving the state of the art results. Again, the same hyperparameter settings were used to train this variant of our architecture.

5.6 Analysis of Failure Cases

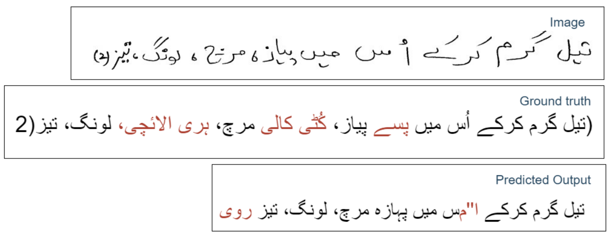
Some failure cases of our model has been shown in Fig. 7. The characters predicted corresponding to the input image show that the failure in prediction was encountered where either the image has some distortion Fig. 7(b) or the writing of a character closely resembled some other character Fig. 7(a). These errors could be reduced by pre-training of our transformer decoder on a Urdu language modeling task so that if a character in the input image is ambiguous, the architecture can still predict it based on the probability of next character given previous character.

6 Conclusions and Future Directions

We modeled the task of Urdu handwriting recognition as a Seq2Seq learning problem inspired from [5] and proposed a Conv-Transformer architecture that eliminated the need for a separate language model. Moreover, the convolution layers at the start of a full transformer acts to reduce the spatial resolution of the Urdu handwritten text images and extract important features. The feature maps with reduced spatial resolution than the input image compensate for the n^2 complexity of the Multi head Attention layers of the transformers leading to reduced training and inference running times.

To the best of our knowledge we are the first one to propose a deep learning architecture that trains simultaneously on Urdu printed and handwriting dataset to yield state of the art result for unconstrained Urdu handwriting recognition.

Future direction includes the pre-training of our architecture on a big dataset before generalizing to a specific



Image

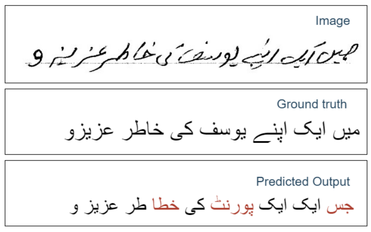
Ground truth

(تیل گرم کرکے اُس میں پسے پیاز، کُٹی کالی مرچ، ہری الائچی، لونگ، تیز(2

Predicted Output

تیل گرم کرکے ا"مس میں پہازہ مرچ، لونگ، تیز روی

(a)



Image

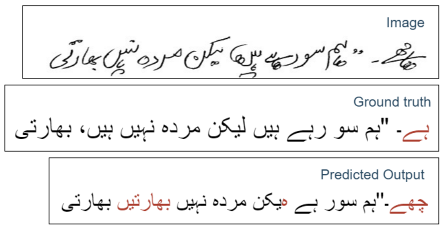
Ground truth

میں ایک اپنے یوسف کی خاطر عزیزو

Predicted Output

جس ایک ایک پورنٹ کی خطا طر عزیز و

(b)



Image

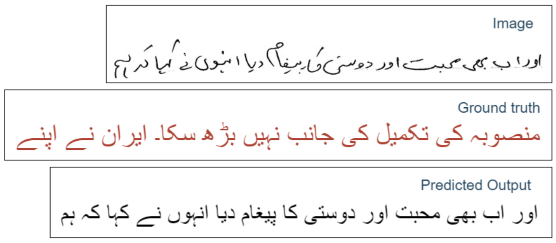
Ground truth

ہے- "ہم سو رہے ہیں لیکن مردہ نہیں ہیں، بھارتی

Predicted Output

چھے-"ہم سور ہے ہیکن مردہ نہیں بھارتیں بھارتی

(c)



Image

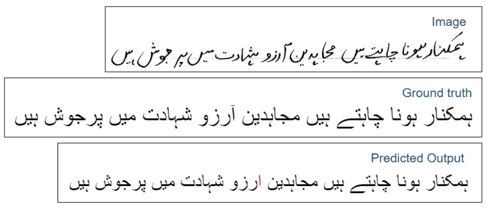
Ground truth

منصوبہ کی تکمیل کی جانب نہیں بڑھ سکا۔ ایران نے اپنے

Predicted Output

اور اب بھی محبت اور دوستی کا پیغام دیا انہوں نے کہا کہ ہم

(d)



Image

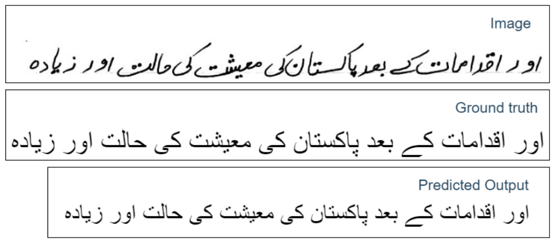
Ground truth

ہمکنار ہونا چاہتے ہیں مجاہدین آرزو شہادت میں پرجوش ہیں

Predicted Output

ہمکنار ہونا چاہتے ہیں مجاہدین ارزو شہادت میں پرجوش ہیں

(e)



Image

Ground truth

اور اقدامات کے بعد پاکستان کی معیشت کی حالت اور زیادہ

Predicted Output

اور اقدامات کے بعد پاکستان کی معیشت کی حالت اور زیادہ

(f)

Fig. 7: The figure shows examples of some the input with with its ground-truth and predicted text given below. **(a)** contains example of label noise in the dataset. Given an image, the provided ground truth contains extra words/characters that are not present in the input image hence increasing the CER of this example. The model correctly predicts the characters that are present the input image. **(b)** contains an example of a distorted input image. The model is unable to predict the true label as the input images contains few characters/literals that are difficult to recognize due to the writing style of the writer. **(c)** contains an input image that contains a complex and calligraphic writing style which makes it difficult for the model to make a correct prediction. **(d)** contains an example of label noise in the dataset. Given an image, the provided ground truth is incorrect, but the predicted output is correct as per the input image. This shows that our model is efficient enough to produce correct results. The high CER is due to the mismatched ground truth. **(e), (f)** contains the cases where the model perfectly recognized the handwritten text hence giving a very low CER.

task. The Conv-Transformer encoder can be pre-trained on a vision task like ImageNet classification and the transformer decoder can be trained on a language specific language modeling task. This pre-training would greatly enhance the accuracy on a task specific datasets after fine-tuning on them as the convolution and transformer architectures have good generalization capabilities.

Conflict of Interest Statement

The author(s) disclosed no possible conflict of interest.

References

1. N. Zia, Muhammad F. Naeem, Syed K. Raza, Muhammad M. Khan, A. Ul-Hasan, and F. Shafait. A convolu-

tional recursive deep architecture for unconstrained urdu handwriting recognition. *Neural Computing and Applications*, pages 1–14, (2021).

2. Malik W. Sagheer, Chun L. He, N. Nobile, and C. Y. Suen. Holistic urdu handwritten word recognition using support vector machine. In *2010 20th International Conference on Pattern Recognition*, pages 1900–1903, (2010).

3. S. Naz, Arif I. Umar, R. Ahmad, I. Siddiqi, Saad B. Ahmed, Muhammad I. Razzak, and F. Shafait. Urdu nastaliq recognition using convolutional–recursive deep learning. *Neurocomputing*, 243:80–87, (2017).

4. K. Asad, M. Z. Asghar, S. Anam, I. A. Hameed, S. H. Asif, and A. Shakeel. A survey on sentiment analysis in urdu: A resource-poor language. *Egyptian Informatics Journal*, 22(1):53–74, (2021).

5. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, (2017).

6. S. Hassan, A. Irfan, A. Mirza, and I. Siddiqi. Cursive handwritten text recognition using bi-directional lstms: A case study on urdu handwriting. In *2019 International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML)*, pages 67–72, (2019).

7. M. Husnain, Malik M. Saad Missen, S. Mumtaz, Muhammad Z. Jhanidr, M. Coustaty, M. Muzzamil Luqman, J. Ogier, and G. Sang Choi. Recognition of urdu handwritten characters using convolutional neural network. *Applied Sciences*, 9(13):2758, (2019).

8. A. Ul-Hasan, S. Ahmed, F. Rashid, F. Shafait, and Thomas M. Breuel. Offline printed urdu nastaleeq script recognition with bidirectional lstm networks. In *2013 12th International Conference on Document Analysis and Recognition*, pages 1061–1065, (2013).

9. S. Naz, A. I. Umar, R. Ahmed, I. Siddiqi, S. Ahmed, M. I. Razzak, and F. Shafait. Urdu nastaliq recognition using convolutional-recursive deep learning. *Neurocomputing*, 243:80–87, 2017.

10. S. Naz, A. I. Umar, R. Ahmed, M. I. Razzak, S. F. Rashid, and F. Shafait. Urdu nasta'liq text recognition using implicit segmentation based on multi-dimensional long short term memory neural networks. *SpringerPlus*, 5, 2016.

11. K. Khan and I. Haider. Online recognition of multi-stroke handwritten urdu characters. In *2010 International Conference on Image Analysis and Signal Processing*, pages 284–290, (2010).

12. George Retsinas, Giorgos Sfikas, Basilis Gatos, and Christophoros Nikou. Best practices for a handwritten text recognition system. In *International Workshop on Document Analysis Systems*, pages 247–259. Springer, 2022.

13. S. Ahmed, S. Naz, Salahuddin, M. Razzak, and A. Umar. Ucom offline dataset – a urdu handwritten dataset generation. *International Arab Journal of Information Technology, 14(2) · March 2016*, 03 (2016).

14. J. Devlin, M. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding.

15. Tom B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, (2020).

16. Christoph Wick, Jochen Zöllner, and Tobias Grüning. Rescoring sequence-to-sequence models for text line recognition with ctc-prefixes. In *International Workshop on Document Analysis Systems*, pages 260–274. Springer, 2022.

17. J. Michael, R. Labahn, T. Gruning, and J. Zollner. Evaluating sequence-to-sequence models for handwritten text recognition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1286–1293. IEEE, (2019).

18. M. Li, T. Lv, L. Cui, Y. Lu, D. Florencio, C. Zhang, Z. Li, and F. Wei. Trocr: Transformer-based optical character recognition with pre-trained models. *arXiv preprint arXiv:2109.10282*, (2021).

19. H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou. Training data-efficient image transformers amp; distillation through attention. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10347–10357. PMLR, 18–24 Jul 2021.

20. H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers distillation through attention, (2020).

21. Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, (2019).

22. Killian Barrere, Yann Soullard, Aurélie Lemaitre, and Bertrand Coüasnon. A light transformer-based architecture for handwritten text recognition. In *International Workshop on Document Analysis Systems*, pages 275–290. Springer, 2022.

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

24. N. Riaz, S. Latif, and R. Latif. From transformers to reformers. In *2021 International Conference on Digital Futures and Transformative Technologies (ICoDT2)*, pages 1–6, (2021).

25. K. O'Shea and R. Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

26. M. F. Naeem, N. Zia, A. Awan, A. Ul-Hasan, and F. Shafait. Impact of ligature coverage on training practical urdu ocr systems. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 131–136, (2017).

27. A. Rehman, A. Ul-Hasan, and F. Shafait. High performance urdu and arabic video text recognition using convolutional recurrent neural networks. In *International Conference on Document Analysis and Recognition*, pages 336–352. Springer, (2021).

28. Houcine Boubaker, Abdelkarim Elbaati, Najiba Tagougui, Haikal El Abed, Monji Kherallah, Volker Märgner, and Adel M. Alimi. Adab database, 2021.