

A Novel Model For Transliterating MODI To Devanagari Script

Anshika Jain¹, Maya Ingle^{2*}

¹Research Scholar, School of Computer Science & IT, Devi Ahilya Vishwavidyalaya, Indore, Specialization in Deep Learning, Pattern Recognition, Orcid Id: 0009-0008-1012-1224

^{2*}Professor, Senior System Analyst, School of Computer Science & IT, Devi Ahilya Vishwavidyalaya, Indore, Specialization in Natural Language Processing, Image Processing, Machine Learning,

*corresponding Author; Maya Ingle

*Professor, Senior System Analyst, Department of Computer Science, Specialization in Natural Language Processing, Image Processing, Machine Learning, Devi Ahilya Vishwavidyalaya, Indore,

Abstract

Preservation and digitization of historical documents inscribed in low-resource scripts continue to pose significant threats in computational linguistics and digital heritage conservation. The MODI script, which has been historically utilized for administrative and literary purposes, is now largely out of the way due to the absence of robust transliteration systems. In the current piece of research work, we present an *Encoder-Decoder Deep Sequence (EDDS)* model based on stacked Long Short-Term Memory (LSTM) networks for sentence-level transliteration into Devanagari script of MODI script. The proposed model handles variable-length input and output sequences and effectively captures long-range contextual dependencies. A parallel corpus of 37,796 aligned MODI-Devanagari sentence pairs has employed to train and assess the proposed model. Experimental results determine a BLEU score of 46, character-level accuracy of 85.16%, word-level accuracy of 86.12%, and a Character Error Rate (CER) of 0.09, signifying strong transliteration fidelity. The model further achieves an overall transliteration accuracy of 97.16%, demonstrating strong generalization capabilities to unseen and irregular input sequences. These findings validate the effectiveness of deep sequence learning for low-resource historical scripts and highlight the proposed EDDS model as a scalable and reliable solution for MODI-to-Devanagari transliteration.

Keywords: MODI Script Transliteration, Encoder-Decoder Model, LSTM, Deep Sequence Learning, Sequence-to-Sequence Model.

1. Introduction

The accessibility of ancient information and maintenance of cultural continuity have become essential by means of historical preservation and digitization. The widely used MODI script for literacy and administrative purposes in ancient times has now declined in popularity and is considered a low-resource script in the present. For the processing of low-resource scripts, Sequence-to-Sequence (Seq2Seq) models effectively demonstrate under the recent advancement in Neural Machine Translation (NMT) [1]. In particular, Long Short-Term Memory (LSTM) architecture possesses the capturing capability of sequential dependencies in character and word-level transliteration. Using open-source Calamari OCR with a Convolutional Recurrent Neural Network (CRNN) model, MODI script transliterates into Devanagari on a dataset of 4,334 scanned text lines with ground truth. The model thus achieves character-level accuracy as 88.14% and word-level as 61% [2]. A neural network based on an LSTM model using OCRopus transliterates MODI images into Devanagari text. Using a synthetic dataset of 99,320 MODI images and respective Devanagari transcriptions, the model attains a character accuracy as 94.67% on the test dataset [3]. The primary difficulty in sequence learning for transliteration is handling variable-length input and output sequences while maintaining both semantic and contextual relationships between characters and words. The cursive style of the MODI script along with overlapping characters and irregular spacing complicates the alignment between input and output sequences. In the transliteration of MODI script into another script, it either loses structural coherence or distorts characters that are identical in both scripts [4]. However, a novel sequence model may be developed to attain the enhanced performance for the transliteration of MODI script to Devanagari script.

The preservation and utilization of several scripts is encouraged by transliteration which serves as a vital connection among multilingual representation and digital accessibility of scripts. In Section II, the literature of various transliteration systems for various scripts is presented. The proposed

model *EDDS* is emphasised along with its architectural components and design in Section III. Section IV demonstrates the experimental evaluation incorporating the dataset, training strategy and evaluation metrics. Further, we discuss the outcomes and concluding observations in subsequent sections.

II. Related Work

Digitizing and preserving historical texts signify a multifaceted challenge in cultural heritage preservation. The transition from handwritten manuscripts to digital text extensively examined for both: linguistics and technological domains. Extensive reviews of machine transliteration techniques demonstrate that hybrid and statistical methods are predominantly employ as it combines rule-based linguistic knowledge and probabilistic modelling to improve accuracy [5]. Deep learning-based transliteration methods such as Recurrent Neural Network (RNN) and convolutional Seq2Seq model inspect 13 language pairs resulting that English-Hebrew pair secures the highest accuracy with a character-level Seq2Seq model and obtains F-score of 0.788 [6]. An attention-based Bi-LSTM encoder-decoder model exploits for reverse transliteration of Romanised Malayalam to its native script. The training dataset comprises of 4.3 million pairs from Dakshina and Aksharantar datasets. Evaluation on IndoNLP-2025 test sets procures a Character Error Rate (CER) as 22.7% for adhoc typing patterns with missing vowels and 7.4% for general typing patterns [7]. An NMT approach is enforced for Kannada-to-English translation using a Sequence-to-Sequence model integrated with an LSTM-based encoder-decoder network. The model effectively captures linguistic patterns between the two languages and acquires a translation accuracy of 86.32% [8]. Similarly, Convolutional Neural Network (CNN) based model is established for character recognition, especially focusing on the gap between MODI and Devanagari script. Thus, it attains 95.44% accuracy with 704 misclassifications, while a deeper three-convolution and three-dense-layer model improves its accuracy to 95.97% with 627 misclassifications on 15,461 test samples [9].

III. Proposed Encoder-Decoder Deep Sequence Model (EDDS)

We present the architectural aspect of our proposed *EDDS* model which aims to transliterate the MODI script into a Devanagari script. It comprises of two sections namely; Encoder and Decoder. This model relies on the principle of representing the input sequence as a context vector and then reconstructing it into the target script over sequential decoding. Here, an input MODI sequence is converted into a variable-length output Devanagari sequence as depicted in Fig. 1. The encoder is built using stacked LSTM units that signifies the deep sequence nature of the model. Each layer captures patterns at a different level of abstraction, where lower layers focus on short-term dependencies i.e. character-level transitions, while higher layers capture broader contextual relationships across the entire sequence. The final hidden state of encoder forms the context vector that a summarized deep representation of the input sequence, which is later passed to the decoder.

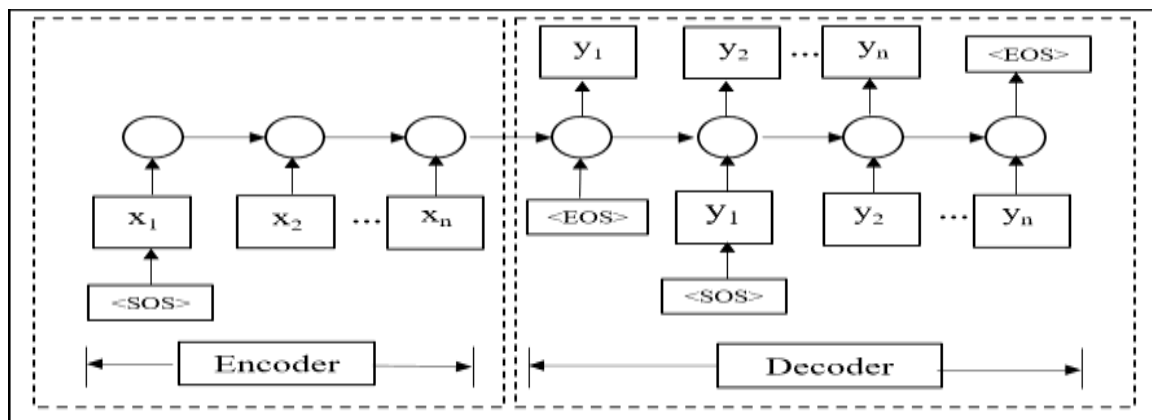


Fig.1 Encoder-Decoder Deep Sequence Model

The decoder is another deep LSTM-based network that unfolds the encoded illustration into the target sequence. By exploiting the deep sequence representation provided by the encoder, the decoder predicts Devanagari symbols one step at a time. Each prediction is based not just on what

is right there in the moment, but also on many different features that are gathered during the encoding process, which helps make sure the results in decoding are both clear and fit well with the overall context.

The model initiates with embedding special markers at the beginning and end of the input. i.e. <SOS> (Start of Sequence) and <EOS> (End of Sequence). These markers help the model know where each sequence starts and finishes. The deep encoder uses several layers of recurrent units, such as LSTM or Gated Recurrent Units (GRUs), stacked on top of each other to help extract features in a layered way. The earlier layers focus on recognizing patterns at the level of individual words or tokens, while the later layers look at more general meanings and relationships across the whole sequence. In the step-by-step processing of embedded tokens, each layer improves the representation by creating a full context vector that includes both detailed and general features of the input. Further, the context vector starts the deep decoder, which copies the encoder's structure to rebuild the target sequence. Starting with its own <SOS> token, the decoder uses both the multi-layered context and previously generated tokens to predict each subsequent output symbol in order to continue until an <EOS> token signals. By employing depth in both the encoder and decoder, our model has ability to capture sequential dependencies between input and output representations of varying lengths, making it particularly suitable for script conversion problems.

IV. Experimental Setup and Evaluation Process

The experimentation is conducted in an open-source web-based interactive environment that supports conscious coding, data visualization, graphical outputs, and is fully integrated with the Python programming ecosystem. The execution is performed in Jupyter Notebook on a 64-bit Microsoft Windows platform equipped with an 11th Gen Intel Core i5 processor, 16 GB of RAM, and Intel Iris Xe graphics. We address the experimental working of our model, including some performance parameters, preparing a suitable dataset, and systematically conducting training and testing phases to achieve stable convergence. The parameters namely; Bilingual Evaluation Understudy (BLEU) Score, Word Level accuracy, Character Level accuracy, CER, validation accuracy, and validation loss help us for analysing the performance.

a. Performance Parameters

The applicability of the transliteration proposed model has evaluated using well-established parameters, each of which focuses on some aspect of the model performance. A widely known prominent parameter referred to as the BLEU score, which compares the amount of n-gram overlap between the output produced by the model and the actual output [10]. BLEU measures the degree of correspondence between a model-generated prediction sequence and a reference sequence by computing the overlap of contiguous n-grams, while incorporating a penalty to discourage excessively short outputs. Due to its language-independent formulation and effectiveness in capturing sequence similarity, BLEU has been extensively adopted in neural transliteration and encoder-decoder-based models. For N-gram precision, let C indicate the prediction Devnagari transliteration generated by the EDDS model, and R denotes the corresponding reference transliteration. For an n-gram n , BLEU computes the modified n-gram precision p_n defined as: (eq.1)

$$p_n = \frac{\sum_{g \in C_n} \min(\text{Count}_C(g), \text{Count}_R(g))}{\sum_{g \in C_n} (\text{Count}_C(g))}$$

where C_n is the multiset of all n-grams of length n extracted from the prediction sequence C . $\text{Count}_C(g)$ is the number of times an n-gram g appears in the prediction sequence. $\text{Count}_R(g)$ is the number of times the same n-gram appears in the reference sequence. The *minimum* operation ensures that repeated n-grams in the prediction output are not over-rewarded beyond their maximum occurrence in the reference sequence. To account for both local and contextual sequence consistency, BLEU aggregates modified precisions over multiple n-gram orders (typically $n = 1, 2, 3, 4$) using a weighted geometric mean: (eq.2)

$$\exp \left(\sum_{n=1}^N w_n \log p_n \right)$$

where N is the maximum n-gram length and w_n is the weight assigned to each n-gram order, commonly set to $w_n = \frac{1}{N}$ for uniform contribution. Lower-order n-grams primarily capture character-level correctness, while higher-order n-grams model longer phonetic and orthographic

patterns in Devanagari script. Since n-gram precision alone does not penalize short prediction outputs, BLEU incorporates a *Brevity Penalty (BP)* defined as:

$$BP = \begin{cases} 1 & \text{if } c > r \\ \exp(1 - \frac{r}{c}), & \text{if } c \leq r \end{cases} \quad (\text{eq.3})$$

where c and r denote the lengths of the prediction and reference sequences, respectively. This penalty discourages incomplete transliterations by reducing the score when the generated output is shorter than the reference. The final BLEU score is computed as:

$$BLEU = BP * \exp(\sum_{n=1}^N w_n \log p_n) \quad (\text{eq.4})$$

The BLEU score ranges from 0 to 1, with higher values indicating stronger agreement between the predicted and reference transliterations [11]. A BLEU score below 20 specifies the poor translation quality, above score 50 reflect high quality translation and values exceeding 80 are considered excellent quality. However, BLEU alone is limited in transliteration tasks even a single character mismatch may drastically change the meaning [12]. In order to overcome the problem, the Character-Level and Word-Level accuracies are also measured via *EDDS* model. Character-Level accuracy evaluates transliteration performance at a finer granularity by measuring the proportion of individual characters that are correctly predicted across the evaluation corpus. It provides insight into the model's ability to learn fine-grained character between the source and target scripts. The Character-Level accuracy is computed as:

$$\text{Character - Level Accuracy} = \frac{\text{Number of Coorectly Predicted Characters}}{\text{Total Number of Characters}} \times 100 \quad (\text{eq.5})$$

while Word-Level accuracy evaluates the proportion of complete words that are transliterated correctly by the proposed model. A predicted word is considered correct only if it matches the reference transliteration exactly, with no character substitutions, deletions, or insertions. It is intentionally strict and reflects the model's ability to generate linguistically valid and practically usable transliterations, computed as:

$$\text{Word - Level Accuracy} = \frac{\text{Number of Exactly Correct Transliterated Words}}{\text{Total Number of Words}} \times 100 \quad (\text{eq.6})$$

while Word-Level and Character-Level accuracies quantify correctness at different granular both metrics are limited to exact-match evaluation and do not explicitly capture the severity of transliteration errors [13]. In addition to complementing these measures, CER is employed to quantify the extent of Character-Level discrepancies between the predicted and reference transliterations. It is widely adopted evaluation metric for OCR and transliteration systems, particularly suited for handwritten and noisy textual data. CER measures the degree of dissimilarity between a predicted transliteration and its reference by quantifying the minimum number of character-level edit operations required to convert the predicted output into the reference sequence. CER explicitly captures error magnitude, thereby providing a more informative assessment of transliteration quality, which is measured by:

$$\text{Character - Error Rate(\%)} = \frac{S + D + I}{N} \quad (\text{eq.7})$$

where S denotes the number of character substitutions, D denotes the number of character deletions, I denotes the number of character insertions, and N denotes the total number of characters in the reference transliteration. A more accurate assessment of sequence integrity is provided by the edit-distance-based metric, where lower CER values denote greater dependability [14]. In addition to transliteration-specific evaluation metrics, validation accuracy and validation loss are employed to monitor the learning behavior and convergence characteristics of the proposed model during training. Validation accuracy measures the proportion of correctly predicted samples on a held-out validation set and provides an estimate of the model's generalization capability beyond the training data. Validation loss quantifies the discrepancy between predicted and reference outputs on the validation set and is used to assess training stability and potential overfitting. Together, these metrics support the analysis of model convergence and reliability but are not used as direct measures of transliteration quality.

b. MODI-Devanagari Parallel Text Corpus

Recent advancements in computational linguistics have enabled the construction of corpus-based machine translation systems that map source and target languages in an end-to-end manner. For

this study, a parallel corpus comprising MODI script and its corresponding Devanagari script has been developed to train and evaluate the model. The dataset contains 37,796 aligned sentence pairs with a maximum input length of 11, ensuring one-to-one correspondence between MODI and Devanagari sentences displayed in Fig.2. Firstly, the MODI sentences are written into text files, and then their Devanagari versions are also typed manually using PramukhIME software [15] to ensure their correctness and match the language properly. Building the dataset is very important because it helps overcome the limited resources available for MODI and creates a solid base for evaluation.

લલ્લ લ્લેઈ ગલ્લ	→	क्या कोई गया
ચજ પાંઝ તારિયે છે	→	आज पांच तारीख है
પ્રીથી લ્લે ઉમ્મલ્લ લ્લમ્લ છે	→	किसी को कुछ काम नहीं है
જહઠી જહઠી	→	जल्दी जल्दी
ચંથીલ્લ જૈજ	→	अंशिका जैन
ગુલ્લલ જમ્મુલ દેલ રૂઝિયે મ્લમ્લયે	→	गुलाब जामुन खा कर खुशी मनाओ
ઉમ્લસલ	→	समस्या
ઉમ્લે મ્લમ્લ ઘેઝલ્લ ઝીયે	→	तुम्हें कम बोलना चाहिए
પુજઃ ળધો	→	पुनः पधारे
પ્રેલ પ્રી ચલ્લજ ચ ઝીયે છે	→	कौए की आवाज़ आ रही है

Fig.2 Sample of MODI-Devanagari Text Corpus

c. Training and Testing Process

The training and testing process is set up to effectively learn sentence-level transliteration patterns among MODI and Devanagari scripts. The model is trained in a fully sequence-to-sequence manner since both scripts exhibit variable-length sequences and non-linear structural relationships. In this setup, MODI input tokens are first mapped to continuous latent vectors, after which the decoder generates the corresponding Devanagari tokens. Here, the sequential mapping demands a reliable optimization method so that the Adam optimizer is employed and for strong convergence it is definite by adaptive moments and modifiable learning rate. We set the goal for optimization using categorical cross-entropy loss, which measures how different the expected token probability distribution is from the actual ground-truth distribution at each decoding step. To ensure smoother gradient updates in stages of training, an initial learning rate of 0.001 is applied, which is reduced by a scheduler when the validation loss stops improving. During sequence modelling, the decoder relies heavily on contextual information from preceding outputs. To preserve the dependency stable during the stages of training, the teacher forcing method is used where actual previous token instead of the one it predicted at each step. The key hyperparameters used in training include an embedding dimension of 256, which helps in creating a rich and detailed meaning for each token. A configuration of 512 hidden units is adopted in each LSTM layer for better understand complex patterns in MODI-Devanagari sentences. For ensure stability and training efficiency, the batch size is set to 64 while maintaining learning. The model is trained for 100 epochs to make sure it learns effectively from different sequence patterns and reaches a reliable solution. The system ability also monitored on new or unseen data using validation accuracy and validation loss parameters. Validation accuracy serves as a symbol-level measure of performance, showing how well the decoder can correctly reproduce Devanagari characters for new MODI sequences. Whereas, validation loss is calculated using categorical cross-entropy and shows how closely the model's predicted probabilities match the actual outcomes for data it has not seen before. Keeping track of these metrics over time helps the model settle into a stable state without learning too much from training data.

During testing, the trained model works entirely on its own predicting one word at a time. Each word it predicts is then used as input for the next part of the process. In this, the evaluation process preserves the natural sequential dependency structure and provides a realistic assessment of transliteration performance. Additionally, to certify robustness, an additional layer of experimental testing has executed using a chosen manually random.txt file that contains diverse and unseen MODI sentences exactly selected for testing the model. These sentences include variations in uncommon ligature formation and character spacing along with wide range of input lengths. The length spanning from single-word sentences up to sequences of incorporating eleven words. In random set, we also include special symbols namely commas (,), question marks (?), and other punctuation marks that seem in practical text setups. In order to assess stability and sequence-level fluency under unpredictable input patterns, the transliteration outputs resulting from random set have compared with reference Devanagari sequence. Through optimized training and actual monitoring metrics, the *EDDS* model carries fluent sequences with precise token-level transliteration.

V. Result and Discussion

The experimental results demonstrate that our *EDDS* model handles the variable-length sequences and long-term dependencies inherent in MODI script transliteration effectively. MODI manuscripts often exhibit irregular character formations, stylistic variations, and context-dependent symbol usage, which pose significant challenges for conventional rule-based and statistical approaches. By leveraging an LSTM-based encoder-decoder model, our model successfully captures both short-range character dependencies and long-range contextual information across entire sentences, enabling accurate and fluent transliteration into Devanagari script. The *EDDS* model has been evaluated on our MODI-Devanagari parallel corpus using multiple complementary metrics, including BLEU score, Word-Level accuracy, Character-Level accuracy, CER, validation accuracy, and validation loss, ensuring a comprehensive assessment of transliteration quality.

The *EDDS* model achieves a BLEU score of 46, indicating strong sequence-level similarity, resulting into effective preservation of character order, phonetic structure, and contextual coherence despite the low-resource nature of the dataset. At the lexical level, an average Word-Level accuracy of 86.12% reflects the model's effectiveness in generating completely correct transliterated words. It is particularly important for historical document digitization where partial correctness may lead to semantic ambiguity. The Character-Level accuracy of 85.16% demonstrates that the model has learnt robust glyph-to-glyph correspondences between MODI and Devanagari scripts, even in the presence of handwriting-induced distortions. In addition, the CER of 0.09 indicates that only a small number of character-level edit operations are required to transform the predicted outputs into the reference transliterations. It also highlights the robustness of proposed model to substitution, deletion and insertion errors commonly observed in reference input. From a training and convergence perspective, the model attains a validation accuracy of 90.65% and validation loss of 0.65, indicating stable learning behaviour and effective convergence without significant overfitting. Further, the overall accuracy of 97.16% demonstrates the strong generalization capability of the *EDDS* model when applied to previously unseen MODI sentences, as illustrated in Fig. 3. The overall accuracy reflects the model's ability to retain contextual meaning, structural integrity, and semantic consistency, which is essential for the transliteration. As compared to traditional rule-based or purely statistical transliteration methods, the deep sequence-based approach offers superior scalability, adaptability, and robustness to diverse writing styles and sentence complexities.

MODI Input: જાણી યત્ન	Generated translation in DEVANAGARI: नहीं आना
MODI Input: ઉમ્મિ પ્રેરે પછે ?	Generated translation in DEVANAGARI: तुम कैसे आये?
MODI Input: પાઠ પ્રમ્ત સૂતે છે	Generated translation in DEVANAGARI: आप क्या करते हो
MODI Input: મેં ઉમ્મિરે ઉપમત ઠે	Generated translation in DEVANAGARI: मैं तुमसे सहमत हू
MODI Input: મુઝે ગણત જ ઉમ્મરે	Generated translation in DEVANAGARI: मुझे गलत न समझे

Fig. 3 Transliteration of Random Sentences using EDDS Model

VI. Conclusion

With the aim of developing a novel model for transliterating MODI script into Devanagari script, contributing to the preservation of low-resource historical scripts. For attaining it, the *EDDS* model based on LSTM has been trained in a parallel corpus of 37,796 sentence pairs to enable the model to effectively capture long-term dependencies and variable-length sequences. Our model obtained competitive performance, including a BLEU score of 46, character-level accuracy of 85.16%, word-level accuracy of 86.12%, a CER of 0.09, and an overall transliteration accuracy of 97.16%, confirming its reliability and transliteration quality. Collectively, the testing approach determines that the *EDDS* model may handle sentence writing variability and generate precise MODI to Devanagari transliteration output even for words that have rarely encountered over training. These findings authenticate the potential of deep sequence learning for complex low-resource scripts and illuminate its role in digital heritage preservation. With the increment of transformer-based architecture, large dataset pairs may enhance the cross-lingual applicability and transliteration fluency.

References

- [1] Felix Stahlberg, "Neural Machine Translation: A Review", Journal of Artificial Intelligence Research, vol. 69, pp. 343-418, 2020.
- [2] S. Joseph & J. George, "Machine Transliteration of Handwritten MODI Script to Devanagari using Deep Neural Networks", International Journal of Computing, Vol. 23, Issue 2, pp. 219-225, June 2024.
- [3] Kundaikar, T., Fadte, S., Karmali, R., Wagh, R., Pawar, J.D., "Modi document transcription to Devanagari", Ingénierie des Systèmes d'Information, Vol. 29, No. 3, pp. 909-915, 2024.
- [4] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le., "Sequence to sequence learning with neural networks", In Proceedings of the 28th International Conference on Neural Information Processing Systems, Vol. 2, pp. 3104-3112, 2014.
- [5] K. Kaur and P. Singh, "Review of Machine Transliteration Techniques," International Journal of Computing Applications, Vol. 107, No. 20, pp. 13-16, 2014.
- [6] S. Kundu, S. Paul, and S. Pal, "A Deep Learning Based Approach to Transliteration", Proceedings of the Seventh Named Entities Workshop, pp. 79-83, 2018.
- [7] Bajiyoo Baiju, Kavya Manohar, Leena G. Pillai, and Elizabeth Sherly, "Romanized to Native Malayalam Script Transliteration Using an Encoder-Decoder Framework", In Proceedings of

- the First Workshop on Natural Language Processing for Indo-Aryan and Dravidian Languages, pp. 174-178, 2024.
- [8] Nagaraj, P.K., Ravikumar, K.S., Kasyap, M.S., Murthy, M.H.S., and Paul, J., "Kannada to English machine translation using deep neural network", *Ingénierie des Systèmes d'Information*, Vol. 26, No. 1, pp. 123-127, 2021.
 - [9] Sawant, S., Sharma, A., Suvarna, G., Tanna, T., Kulkarni, S., "Word transcription of MODI script to Devanagari using deep neural network", In 2020 3rd International Conference on Communication System, Computing and IT Applications, 2020.
 - [10] Ehud Reiter, "A Structured Review of the Validity of BLEU", *Computational Linguistics*, Vol. 44 (3), pp. 393-401, 2018.
 - [11] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu., "Bleu: a Method for Automatic Evaluation of Machine Translation", In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA, pp. 311-318, 2002.
 - [12] Abhinav Jha and Hemprasad Yashwant Patil, "A Review of Machine Transliteration, Translation, Evaluation Metrics and Datasets in Indian Languages", *Multimedia Tools Application* 82, 23509-23540, 2023.
 - [13] Umer Butt, Stalin Varanasi, and Günter Neumann, "Low-Resource Transliteration for Roman-Urdu and Urdu Using Transformer-Based Models", In *Proceedings of the Eighth Workshop on Technologies for Machine Translation of Low-Resource Languages (LoResMT 2025)*, pp. 144-153, April 2025.
 - [14] Clemens Neudecker, Konstantin Baierer, Mike Gerber, Christian Clausner, Apostolos Antonacopoulos, and Stefan Pletschacher, "A survey of OCR evaluation tools and metrics", In *Proceedings of the 6th International Workshop on Historical Document Imaging and Processing (HIP '21)*. Association for Computing Machinery, New York, pp. 13-18, 2021.
 - [15] PramukhIME software <https://www.pramukhime.com/type/marathimodi>