LDK 2023

Large Vocabulary Continuous Speech Recognition for Nepali Language using CNN and Transformer

Shishir Paudel and **Bal Krishna Bal** and **Dhiraj Shrestha**
Information and Language Processing Research Lab
Kathmandu University, Dhulikhel, Nepal
shishirpaudelofficial@gmail.com
bal@ku.edu.np
dhiraj@ku.edu.np

Abstract

Despite the availability of various algorithms for speech recognition, their performance for low resource languages like Nepali is suboptimal. The Transformer architecture is a state-of-the-art NLP deep learning algorithm that uses self-attention to model temporal context information. Although it has shown promising results for English ASR systems, its performance for Nepali has not been extensively explored. This work implements an end to end CNN-Transformer based ASR system to explore the potential of Transformer for building an ASR for the Nepali language. The study used around 159K datasets extracted from openSLR which was further complemented with original recordings that incorporated sentences representing different tenses, grammatical persons, inflections, direct-indirect speech, level of honorifics, etc to address the grammatical structures of the Nepali language. The end to end CNN-Transformer architecture was trained with varying size of datasets, epochs and parameter tuning. The best resulting model achieved a CER of 11.14%.

1 Introduction

Automatic speech recognition (ASR) systems have gained significant importance in recent years due to its wide range of applications, such as virtual assistants, voice command interfaces, automated customer service systems, transcription services etc. Traditionally, ASR systems were built using separate acoustic, language, and pronunciation modules (Jelinek, 1976) and relied on statistical methods such as Hidden Markov model (HMM) and Gaussian Mixture model (GMM). However, such systems required forcefully aligned data, and had limited ability to model complex phenomena such as coarticulation, speaker variability, context etc., (Rabiner and Juang, 1993). In recent years, ASR systems have shifted towards end to end deep neural network (DNN) models that can directly map speech signals to text without entailing separate modeling of different linguistic features.

Some of the prominent deep neural architecture that can be used to build ASR systems include Convolution Neural Network (CNN), Recurrent Neural Network (RNN) and Transformer. CNNs are efficient in learning local patterns such as spectral or temporal patterns and are mostly employed to extract non-linear features from audio signals. On the other hand RNNs are used to address the temporal relation using the feedback connections and internal status. A problem with RNNs is that they suffer from vanishing gradient problem. Variants of RNN like Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), Gated Recurrent Unit (GRU) and bidirectional LSTM (BiLSTM) try to alleviate the issue however are slow to train and computationally demanding due to their sequential nature. In addition, they are not able to capture the long term dependencies efficiently as the vanishing gradient problem still persists (Zeyer et al., 2019). These shortcomings are solved by the Transformer that employs a multi-headed attention mechanism to compute self-attention. The self attention in Transformer allows each segment in the input to reference every other in the input to capture the long term dependencies (Vaswani et al., 2017). Further the multi head attention allows multiple self attention to be computed simultaneously on different segments of the input that significantly makes the training faster along with capturing of the context for longer sentences (Chernyshov et al., 2021; Dai et al., 2019; Kleinebrahm et al., 2020).

Despite the evolution of ASR systems and deep neural architectures, research is mainly prioritized for prominent languages like English and Mandarin, while for low resource languages like Nepali, ASR systems haven't been explored to that extent (Banjara et al., 2020). Only a few research materials and ASR products based on Nepali language exist today. Further, research carried out in Nepali

ASR include implementation and study of traditional methods such as HMM, RNN etc., while the implementation of recent architecture such as Transformers is largely missing. An efficient ASR based on the Nepali language can be applied to automate various data input systems in different sectors in Nepal, such as banks, hospitals, governmental offices, etc., that could help reduce errors and increase efficiency, ultimately saving time and improving the quality of service.

Nepali is an Indo-Aryan language spoken by 44.64% of the Nepali population and is written using the Devanagari script, which is phonetic (Bal, 2004; Khanal, 2019). Nepali language incorporates a complex system of noun, adjective and verb inflections. Nouns have a system of gender, case and number. Nouns can be inflected to reflect singular or plural, and can be adjusted to seven cases (Bal, 2004) Adjectives in Nepali occur before the noun they modify and they must correspond to gender, case, and number of the noun. Verbs inflect to show contrasts for the grammatical persons, singular/plural, tenses, gender of a subject, grades of honorifics etc.

In this work, we present an end-to-end CNN-Transformer model for Nepali ASR and study the potential of Transformer for low resource languages with the available Nepali datasets. We incorporate variations in the grammar structures, speaking rate, and accent during the training process to enhance the model's ability to generalize over unseen data. Our study sheds light on the performance of contemporary ASR systems for low resource languages and highlights the potential for further research in this area.

2 Past Work

Nepali speech recognition system is one of the least covered topics considering its essence. However, there exist some significant works carried out in Nepali ASR systems. One of the earliest works include a Nepali ASR proposed by Prajapati et al., 2008 that implemented an Ear model based on the human auditory system. Likewise, a HMM based model that was used for processing the Mel Frequency Cepstral Coefficients (MFCC) features from the audio signals was presented by Gajurel et al., 2017. In recent years, deep learning based ASRs were also researched by several authors. Regmi et al., 2019 presented a Nepali ASR based on CNN, RNN and connectionist temporal classification (CTC) combination. The model was trained on a 2 hour Nepali speech data, where, the CNN was used for extracting the MFCC features while RNN was used for processing the sequential data after feature extraction and CTC for decoding. A total of 67 Nepali characters were used to decode the final text and the model provided a Character Error Rate (CER) of 52% on test data.

Similarly, Banjara et al., 2020 also experimented the combination of CNN, with various RNNs on Nepali dataset. However, compared to Regmi et al., 2019, a larger dataset corpora was used which was collected from OpenSLR[1] namely slr43 and slr54 consisting of 158,113 Nepali utterances. From their experiment, the best resulting CNN-GRU-CTC model achieved a CER of 23.72% which is almost half compared to the prior. Their result showed that GRU performs better than normal RNN due to the reduced severity of issues like vanishing gradient in GRU, while also highlighting the significance of a larger dataset in improving RNNs' performance. Likewise, an end to end CNN-BiLSTM-CTC architecture based model was presented by Regmi and Bal, 2021. The authors also used the slr43, slr54 Nepali data corpus which are openly accessible from OpenSLR for training the model. A total of 129 Nepali characters were used for decoding. Their BiLSTM based model provided a CER of 10.3% on test data. The authors also reported that the training for 20 epochs of the CNN-BiLSTM-CTC model required around 8 days.

From the literature study, we found that all the existing researches in Nepali speech recognition have predominantly used traditional statistical methods such as HMM and deep neural networks such as RNN and its variants like LSTM, GRU and BiLSTM. Remarkably, none of these studies have utilized the Transformer model since it is a recent deep neural architecture. Therefore, in this study we aim to explore the possibilities of Transformer model in Nepali language speech recognition by implementing an end to end CNN-Transformer architecture and compare with the existing DNN implementations.

3 Datasets

Nepali speech datasets are not abundantly available. We collected two freely accessible Nepali speech data set corpora namely "SLR43" and "SLR54" provided by the openSLR.org. The first corpus con-

[1] https://openslr.org/

sisted of 2064 utterances collected from 18 female speakers that were mostly of longer length sentences while the second consisted of 157k speech data with mostly shorter length sentences. Further, 6031 original recordings corpus generated as a part of this study named Nep_DS were also added to the collected datasets. Nep_DS consists of various Nepali phrases scraped from Nepali language based websites such as ekantipur[2], setopati[3], hamro patro[4], "Nepali Me" etc., and also several sentences from English websites like BBC translated to Nepali using the google translate api[5]. In addition, we also added several Nepali sentences with varying lengths that address grammatical structures in Nepali language such as different tenses, inflections, grammatical persons, direct and indirect speech, honorifics, etc. The sentences were checked for errors and recorded using Samsung M51 mobile phone, involving 5 speakers. Subsequently, the collected datasets were preprocessed that involved first converting the audio files from .flac to .wav, followed by downsampling the audio from 48 Khz to 16 Khz. The purpose of this preprocessing step was to minimize the computational cost during training. Furthermore, the vocabulary was generated consisting of a total of 119 unique Nepali characters along with 8 additional characters extracted from the text of datasets. The characters were then indexed from 0 to 126, which is shown in Figure 1.

0 -	19 ए	38 ढ	57 व	76 ◌ू	95 ज़	114 ८
1 #	20 ऐ	39 ण	58 श	77 ◌ृ	96 ड़	115 ९
2 <	21 ऑ	40 त	59 ष	78 ◌ॄ	97 ढ़	116 °
3 %	22 ओ	41 थ	60 स	79 ◌ॅ	98 फ़	117 ॰
4 ◌	23 औ	42 द	61 ह	80 ◌ॉ	99 य़	118
5 ◌्	24 औ	43 ध	62 ◌	81 ◌ॊ	100 ऋ	अं
6 ◌	25 क	44 न	63 ◌ा	82 ◌ो	101 ऌ	119
7 ◌:	26 ख	45 न	64 ◌	83 ◌ौ	102 ◌ॢ	अं
8 अं	27 ग	46 प	65 ऽ	84 ॐ	103 ◌ॣ	120
9 अ	28 घ	47 फ	66 ◌ा	85	104 ◌ँ	आँ
10 आ	29 ङ	48 ब	67 ◌ि	86	105 ॥	121
11 इ	30 च	49 भ	68 ◌ी	87 ◌ॅ	106 ०	औं
12 ई	31 छ	50 म	69 ◌ु	88 ◌ॆ	107 १	122
13 उ	32 ज	51 य	70 ◌ू	89 ◌े	108 २	अ़
14 ऊ	33 झ	52 र	71 ◌ृ	90 ◌ै	109 ३	123
15 ऋ	34 ञ	53 ऱ	72 ◌ॄ	91 ◌ॉ	110 ४	124 .
16 ऌ	35 ट	54 ल	73 ◌ॅ	92 क़	111 ५	125 ,
17 ए	36 ठ	55 ळ	74 ◌ॆ	93 ख़	112 ६	126 ?
18 ऐ	37 ड	56 ऴ	75 ◌े	94 ग़	113 ७	

Figure 1: Nepali Characters used in proposed ASR

4 Architecture of CNN-Transformer

The proposed Nepali ASR system includes an end-to-end CNN-Transformer architecture as illustrated in Figure 2. At first, audio is transformed into spectrogram using short time fourier transform (STFT). The CNN then processes the spectrogram frames to extract high-level spectral features and these extracted audio feature maps are passed to Transformer. The encoder receives a sequence of feature vectors produced by the CNN and transforms it into a fixed-length vector representation. This is accomplished through a series of self-attention and feed forward layers. Self-attention allows the encoder to attend to different parts of the input sequence, depending on their relevance for the current context (Zeyer et al., 2019). The self-attention mechanism is applied multiple times, with each layer building on the output of the previous layer. The output of the final layer is a fixed length vector representation that summarizes the most important information in the input sequence. This vector representation is fed into the decoder to generate the corresponding text output. The decoder uses self-attention to attend to the previously generated output characters while incorporating information from the input sequence using encoder-decoder attention and generates raw discrete representation. The softmax function in the decoder transforms the raw output discrete vectors into a probability distribution over the 128 Nepali output characters. The character with highest probability is given as text output.

During training, the masking mechanism of the Transformer ensures that only relevant parts of the input sequence are attended. Likewise, masking also prevents the model from attending to future tokens during training, ultimately preventing the model from overfitting (Vaswani et al., 2017). Overall, the combination of the CNN and transformer allows the ASR system to effectively capture both low-level spectral features and high-level temporal dependencies in the input audio signal, which is important for accurate speech recognition.

For the implementation, we have used 3 stacks of 1-D CNNs with each having 64 hidden layers, 11 filter size. The opt for 1 D CNN is to minimize the computation cost, and to handle data acquired from varying sources (Kiranyaz et al., 2021) Likewise, the employed transformer consists of encoder and decoder layer as the one suggested by (Vaswani et al., 2017) while the parameters of the transformer are varied in the experiment to optimize its performance for Nepali dataset. The CNN and Transformer were implemented in python language using the Keras library over TensorFlow platform.

[2] https://ekantipur.com/
[3] https://www.setopati.com/
[4] https://www.hamropatro.com/news
[5] https://pypi.org/project/googletrans/

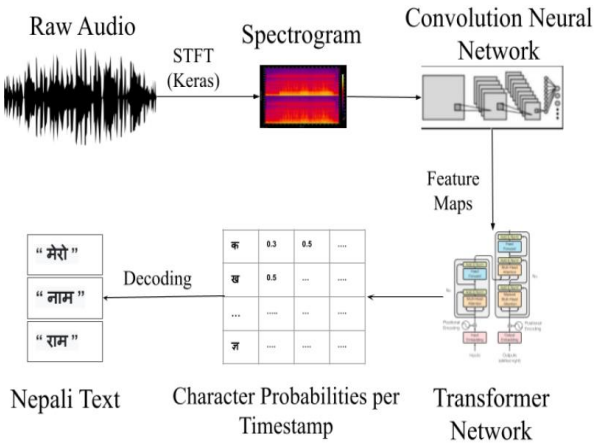

Figure 2: Architecture of the proposed CNN-Transformer ASR

5 Experiments

5.1 Experimental Setup

A total of 14 model training experiments were conducted in two sets to test the potential of the Transformer for recognizing the Nepali speech. The first experiment set involved training the Transformer model on three different Nepali speech datasets: "SLR43", "SLR54", and "Nep_Ds" keeping the training as well as Transformer's parameter values consistent to 200 hidden layers, 2 attention head, 400 FFN, 4 encoders, and single decoder while learning rate was kept 0.001. Different combinations of these datasets were used in the experiments, and alterations were made to the data split ratio and batch size of the training and validation data. The best resulting configuration from the first set was used in the second set, where additional alterations were made to the training parameter i.e., learning rate and Transformer parameters i.e., numbers of attention head, encoder, hidden layer and feed forward network (FFN) . The experiments were carried out on two different machines: Machine 1, which had an Intel i9 processor, RTX 3080Ti GPU, and 32 GB of RAM, and Machine 2, which had an RTX 2060 GPU with other specifications remaining the same. Each trained model was evaluated using CER (Character Error Rate) to analyze the best configuration.

5.2 Experimental Result

In the first experiment set, when the model was trained with a smaller dataset i.e. "SLR43" for 105 epochs, the model overfitted. For training with larger dataset, we introduced early stopping and saving with checkpoints in order to stop the training upon no progress and retain the model with best accuracy. The model performed well when the larger dataset was used i.e. "slr53". For "Nep_DS" as well the model produced a satisfactory result on unseen data. Moreover, the best result was achieved when the all the three corpora ie: "SLR43", "SLR54" and "Nep_DS" were combined with data split ratio kept at 90:10 rather than 80:20 and batch size kept at 64/4 where the obtained CER was 13.97%. This shows that the CNN-Transformer performs better when the dataset has a higher number of examples for training. Furthermore, the model training speed increased when the batch size of the training data was increased although the performance of the model did not improve. The results from the first experiment set are summarized in Table 1.

In the second experiment set, the learning rate (LR) was altered from 0.001 to 0.0095 and then to 0.00001, while the transformer's parameters such as number of attention heads was altered between 2, 4 and 8. Similarly the number of encoders was increased from 4 to 8 and the number of FFN was altered from 400 to 800. After 6 different training sessions with such variations in parameters of training and Transformer we found that the model was able to achieve the least CER value ie: 11.14% when the learning rate was 0.001 and the attention head was increased from 2 to 4, while no progress was seen when changing other parameters. Besides, the training with all three corpora merged together (166K datasets) required around 72 hours for 105 epochs on a RTX 2060 GPU based system while it only took around 12.5 hours on RTX 3080Ti based system. The results from the second experiment set are shown in Table 2.

Some of the predictions outputted by the best resulting model on the sample test data is presented in Table 3 which reveals that the model was accurate in most transcriptions. While the majority of the predictions were accurate, a few minor errors were observed, specifically in outputting the corresponding word for numeric utterances. For instance, the numeric sound "२००६" (English translation: "2006") was predicted as "दुई हजार छ" (English translation: "Two thousand and six"). Similarly, the word "आकाशवाणीबाट" was predicted as "आकाशवाणी बाट". Nevertheless, it should be noted that these errors can be neglected as the pronunciation in the predictions precisely matches the reference in both cases.

Expt.	Data	Data Split	Batch Size	Train Data	Test Data	Avg_Epoch_Time(sec) Machine1	Avg_Epoch_Time(sec) Machine2	CER
Ex 1	SLR43	80:20	64/4	1651	413	5.23	39.43	86.98%
Ex 2	SLR54	80:20	64/4	126324	31581	358.75	2337.86	22.77%
Ex 3	Nep_DS	80:20	64/4	4825	1206	12.54	98.7	47.38%
Ex 4	SLR43+54	80:20	64/4	127975	31994	362.36	2440.76	16.57%
Ex 5	SLR43+54+Nep_DS	80:20	64/4	132800	33200	377.51	2658.34	14.74%
Ex 6	SLR43+54+Nep_DS	90:10	64/4	149400	16600	385.02	2586.12	13.97%
Ex 7	SLR43+54+Nep_DS	90:10	128/32	149400	16600	336.44	2498.12	15.34%
Ex 8	SLR43+54+Nep_DS	90:10	256/64	149400	16600	348.37	2434.46	15.89%

Table 1: CNN-Transformer performance results from first experiment set on different datasets, data split ratio and batch size.

Expt.	Attention Head	Encoders	Hidden Layer	FFN	LR	Avg_Epoch_Time(sec) Machine1	Avg_Epoch_Time(sec) Machine2	CER
Ex 9	2	4	200	400	0.00095	417.62	2550.3	15.66%
Ex 10	2	4	200	400	0.00001	364.34	2431.54	16.35%
Ex 11	4	4	200	400	0.001	432.01	2464.23	11.14%
Ex 12	8	4	200	400	0.001	604.68	2888.57	15.71%
Ex 13	4	8	200	400	0.001	533.54	2449.08	16.53%
Ex 14	4	4	200	800	0.001	464.38	2454.29	13.74%

Table 2: CNN-Transformer performance results from second experiment set on various parameter tunings

S.No	Reference	Prediction
1	सुमात्राको टापुमा रहेको तीन इन्डोनेसियाली राष्ट्रिय निकुञ्ज	सुमात्राको टापुमा रहेको तीन इन्डोनेसियाली राष्ट्रिय निकुञ्ज
2	पञ्चमी शब्दले दुई वटा कुरा जनाउँछ	पञ्चमी शब्दले दुईवटा कुरा जनाउँछ
3	२००६ मा उनले	दुई हजार छ मा उनले
4	संसारको पाँचौं अग्लो हिमाल मनासलु यही क्षेत्रमा पर्छ	संसारको पाँचौं अग्लो हिमाल मनासलु यही क्षेत्रमा पर्छ
5	गीतहरूलाई आकाशवाणीबाट प्रसारित	गीतहरूलाई आकाशवाणी बाट प्रसारित

Table 3: Model's predictions on sample test data

6 Discussions

After several experiments and parameter tunings, the proposed CNN-Transformer achieved a CER value of 11.14% for a combined SLR 43, SLR54 and Nep_DS dataset. Table 4 presents the comparison of our model with other deep learning architecture based Nepali speech recognition systems available in the previous literature. In the previous researches, CNN-RNN-CTC implemented by Regmi et al., 2019 achieved a CER of 52% for a small dataset while similar architecture implemented by Banjara et al., 2020 achieved a CER of 23.72% for a larger dataset with around 159K utterances. Similarly, BiLSTM-CTC based model implemented by Regmi and Bal, 2021 provided a CER of 10.3% for the same dataset used by Banjara et al., 2020. From the comparison, it is evident that the CNN-Transformer model proposed in our study outperforms most of the past CNN-RNN-CTC based implementations in terms of CER when trained on a large dataset. Besides, the performance of our model is slightly lower but comparable to the best CER value from the previous researches which was achieved by Regmi and Bal, 2021 with the similar size of dataset using CNN-BiLSTM. Nevertheless, our proposed CNN-Transformer model required only about 14 hour for 20 epochs of training on RTX 2060 GPU which is almost 14 times less than the reported training time for CNN-BiLSTM model presented by Regmi and Bal, 2021 which required 8 days for 20 epochs on RTX 2060 GPU when trained with similar size dataset. As a whole, it can be revealed that Transformer has the ability to recognise Nepali speech as accurately as other state of the art RNN based implementations, while the training time it takes is exceptionally less than RNN and its variants.

Papers	Model	Dataset	Dataset Size	CER(%)
Regmi et al., 2019	CNN-RNN-CTC		2 Hours	52
Banjara et al., 2020	CNN-RNN-CTC	SLR 43+54	159K	23.72
Regmi and Bal, 2021	BiLSTM-CTC	SLR 43+54	159K	10.3
This study	CNN-Transformer	SLR 43+54 +Nep_DS	166K	11.14

Table 4: Comparison of the proposed CNN-Transformer model with other deep neural based Nepali ASR

7 Conclusion

In conclusion, this study explored various algorithms used in Nepali ASR. Further, we implemented the Transformer architecture in combination with CNN to build an ASR for Nepali language. Various experiments were conducted to analyze the performance of the CNN-Transformer model on different Nepali datasets with several parameter tunings. The training and validation datasets were extracted from openSLR and augmented with 6031 original speech recordings developed for this study named "Nep_DS". The best resulting CNN-Transformer model obtained an accuracy of 11.14% CER on test data, outperforming many RNN based Nepali ASR in terms of both accuracy and training speed.

Data Availability

The "Nep_DS" corpus generated in this study will be made publicly available at `https://ilprl.ku.edu.np/` upon the publication of this work.

References

B. K. Bal. 2004. Structure of nepali grammar. *PAN Localization Working Papers*, pages 332–396.

Janardan Banjara, Kaushal Raj Mishra, Jayshree Rathi, Karuna Karki, and Subarna Shakya. 2020. Nepali speech recognition using cnn and sequence models. In *2020 IEEE International Conference on Machine Learning and Applied Network Technologies (ICMLANT)*, pages 180–185. IEEE.

Artem Chernyshov, Valentin Klimov, Anita Balandina, and Boris Shchukin. 2021. The application of transformer model architecture for the dependency parsing task. *Procedia Computer Science*, 190:142–145.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988.

Avaas Gajurel, Manish K. Ssarma, Anup Pokhrel, and Basanta Joshi. 2017. Hmm based isolated word nepali speech recognition. In *2017 International Conference on Machine Learning and Cybernetics*, volume 1, pages 71–76.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Frederick Jelinek. 1976. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556.

Rajendra Khanal. 2019. Linguistic geography of nepalese languages. *The Third Pole: Journal of Geography Education*, 18:45–54.

Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel Inman. 2021. 1d convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, 151.

Max Kleinebrahm, Jacopo Torriti, Russell McKenna, Alessandro Ardone, and Wolf Fichtner. 2020. Using neural networks to model long-term dependencies in occupancy behavior. *Working Paper Series in Production and Energy*.

C. Prajapati, J. Nyoupane, J. Shrestha, and S. Jha. 2008. Nepali speech recognition.

Lawrence Rabiner and Biing-Hwang Juang. 1993. *Fundamentals of speech recognition*. Prentice-Hall, Englewood Cliffs, NJ.

Paribes Regmi, Arjun Dahal, and Basanta Joshi. 2019. Nepali speech recognition using rnn-ctc model. *International Journal of Computer Applications*, 178(31):1–6.

Sunil Regmi and Bal K. Bal. 2021. An end-to-end speech recognition for the nepali language. In *Proceedings of the 18th International Conference on Natural Language Processing*, pages 180–185.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.

Albert Zeyer, Parnia Bahar, Kazuki Irie, Ralf Schlüter, and Hermann Ney. 2019. A comparison of transformer and lstm encoder decoder models for asr. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 8–15. IEEE.