DeepSpeech++: Towards Improved ASR with Attention and Conformer

Olan Pinto and Dnyaneshwari Rakshe

Course CSCI 5922, University of Colorado Boulder

1 Abstract

In this project, we have explored different model architectures to enhance the automatic speech recognition (ASR) systems using three different architectures - a DeepSpeech-inspired baseline model, an attention-augmented model, and a Conformer model, using the LibriSpeech Dataset. We started with preprocessing the audio dataset using Mel-Spectrogram transformations and implemented text tokenization using a unigram model. In the baseline model, we have integrated convolutional and recurrent components like Bi-GRU to enhance the model learning performance. In the second architecture, i.e. the attention-based model, we have replaced RNN layers with a Transformer encoder to optimize the sequence modeling through multi-head self-attention. The third architecture, the Conformer model, we combined convolution and attention mechanisms to capture both local and global dependencies in speech data. All the 3 model architectures were trained using Connectionist Temporal Classification Loss i.e. CTC loss, AdamW to handle optimization, and OneCycleLR scheduler to dynamically adjust the learning rate over epochs. We implemented a series of hyperparameter combinations over the training, validation and evaluated model performance on the test set using Word Error Rate (WER) and Character Error Rate (CER) metrics along with the average training time. By comparing the performance of these 3 architectures, we understood that different combinations of hyperparameters could significantly impact the model accuracy. The experimental results for the 3 model architectures highlighted the importance of optimized learning and careful hyperparameter tuning to achieve the best ASR performance.

Keywords: Automatic Speech Recognition (ASR), DeepSpeech, Attention model, Conformer architecture, LibriSpeech dataset, Mel Spectrogram transformation, Word Error Rate (WER), Character Error Rate (CER), Multi-head self-attention (MHSA), Connectionist Temporal Classification (CTC) loss, Convolution, Hyperparameter optimization, ResNet, Bidirectional GRU (BiGRU), Transformer encoder, Data augmentation, Sequence modeling, Recurrent neural networks (RNN), AdamW optimizer, OneCycleLR scheduler, Transcription generation, Text tokenization, Unigram, Token mapping, Feature Extraction, Speech-to-text conversion

2 Introduction

In modern applications, Automatic Speech Recognition (ASR) systems have become an integral part of voice assistants, transcription services, and other accessibility technologies. The progress in ASR systems have been considerable, but these systems can be significantly affected by variations in speaker accents, speech styles, and noisy environments. As the ASR systems continue to grow and spread globally, it becomes important for us to focus on improving its robustness and accuracy for a wide range of speakers.

In this project, we primarily worked with the LibriSpeech dataset. It is a widely used dataset of audio books for ASR research. It contains a large and diverse collection of public domain audio books and serves as an ideal benchmark for developing advanced models across a wide range of variations. Our major focus in this project was to explore and compare different deep learning architectures to enhance ASR performance by addressing challenges like speaker variability. While researching for this project, we found that classic ASR architectures like DeepSpeech, built on RNN-CTC pipelines, often struggle with change in accents or speech patterns that are underrepresented in the training data, leading to reduced accuracy and inclusivity in real-world scenarios. Additionally, attention-based models like Listen, Attend and Spell (LAS) capture the overall context but may underperform in extracting fine-grained temporal features. This highlights the need for more robust models that can effectively generalize across a wide range of speech variations.

In an effort to overcome these challenges, we developed and evaluated three end-to-end ASR architectures: (1) A DeepSpeech-inspired model enhanced with convolutional and BiGRU layers, (2) An attention-augmented model using transformer based encoders for improved sequence modeling, and (3) A Conformer model combining convolutional and attention mechanisms to effectively capture both local and global dependencies in speech. We trained each model using CTC loss with AdamW optimization and OneCycleLR scheduling. We implemented various experiments using different hyperparameter combinations and evaluated the model performance using Word Error Rate (WER), Character Error Rate (CER), and training time as the key metrics. In short, this project helps us understand how architectural designs and careful training strategies can contribute in developing more accurate and generalizable ASR systems to handle real-world scenarios.

3 Related Work

End-to-End ASR with RNN and CTC Models

- [1] **Deep Speech: Scaling up end-to-end speech recognition**
- [2] **Deep Speech 2: End-to-End Speech Recognition in English and Mandarin**
- [3] **End-to-End Speech Recognition using Deep RNN Models and WFST-based Decoding**

Our strategies differ from these, with the inclusion of attention mechanisms and Conformer architectures in the DeepSpeech framework to more accurately detect global and local dependencies and eventually enhance recognition accuracy using word error and character error rates as an evaluation metric.

Attention Mechanisms in ASR

- [4] **Listen, Attend and Spell**
- [5] **Attention-Based Models for Speech Recognition**

Our paper differs from the above mentioned papers, primarily in the application of strengths in attention mechanisms on conformer models in the DeepSpeech architecture in order to obtain strength in both global attention and local feature extraction for overall ASR while evaluating using WER and CER metrics.

Transformer-based Models for ASR

- [6] **Attention Is All You Need**
- [7] **Transformer-Based End-to-End Speech Recognition with Local Dense Synthesizer Attention**

The previous works like "End-to-End Speech Recognition with Transformer" primarily focused on adapting Transformer-based architectures for ASR. However, our project proposal uniquely incorporates speaker adaptation techniques and integrates Conformer blocks into the DeepSpeech framework. This hybrid approach allows us to enhance both global dependencies using Transformer mechanism and local temporal features using Conformers.

Multi-task Learning for ASR

- [8] **An Overview of Multi-Task Learning in Deep Neural Networks**
- [9] **Multi-Task Learning for End-to-End ASR Word and Utterance Confidence with Deletion Prediction**

These research projects on multitask learning (MTL) in ASR focus on training models for tasks like speaker recognition or noise classification along with transcription. However, our plan is to implement an approach to integrate speaker profiles and accent data into the core ASR pipeline. The primary thought is to pre-train and fine-tune on the LibriSpeech dataset to improve speaker adaptation and ASR performance for different accents and environments, which goes beyond conventional multitask learning strategies.

4 Methodology

4.1 Dataset and Preprocessing

We made use of the LibriSpeech dataset for this project. This dataset consists of audio data compiled from various audiobooks. The LibriSpeech dataset is widely used for many ASR (Automatic Speech Recognition Tasks). For the purpose of this , we used the train-clean-100 version for training, dev-clean for validation and test-clean for final testing. All these subsets of data use 16Khz mono channel recordings of English audiobooks along with their corresponding transcripts.

Text tokenization : We designed a custom unigram tokenizer to help convert transcript texts into formats that can be consumed by our models. The tokenizer maps every lowercase alphabet in the English language and [space] token to unique integer IDs. There are two salient features that are handled by the Unigram tokenizer.

- **Encoding:** Each character in the transcript is converted into a unique integer ID, with a special token `[space]` used for spaces.
- **Decoding:** The integer sequences are mapped back into characters and reassembled into words by replacing `[space]` with a literal space.

This Unigram model gives us a vocabulary size of 29 unique characters (26 alphabets + apostrophe + [space] + CTC blank)

Audio Preprocessing and Augmentation : The raw waveforms that come from the audio data needed to be transformed into some form of representation for our models. Thus, we made use of the Mel Spectrograms configured with 16KHz sampling rate and 128 mel bands. To increase the models robustnedd, we applied data augmentation techniques on our training data.

- **Frequency Masking:** Randomly masks frequency bins (param = 15)
- **Time Masking:** Randomly masks time steps (param = 35)

Validation and test data also go through spectrogram transformation but without any augmentation

Padding and Length Handling : Now since both audio and transcript sequences vary in length, we need to handle both of them appropriately

- **Spectrograms** were zero-padded along the time axis to form a batch tensor of shape [batch_size, 1, n_mels, time_steps], transposed appropriately to match model requirements.
- **Text Sequences** were padded and sequence lengths were computed to assist in CTC loss calculation and decoding

Train-Validation-Test Split

- **Train:** train-clean-100 (100 hours of cleaned audio data)
- **Validation:** dev-clean (3.6 hours of cleaned audio data)
- **Test:** test-clean (3.6 hours of cleaned audio data)

Each waveform-transcript pair was preprocessed using the above pipeline, and stored as tensors to be fed directly into the model for training and evaluation.

4.2 Baseline Model Architecture: DeepSpeech Inspired Network

The first model developed was inspired from the DeepSpeech Model Architecture. The primary enhancements were made by adding a residual convolution block and layer normalisation to help improve learning stability and representation power

Architecture Overview : The model consists of 4 main stages, 1) Convolutional Front End 2) Residual Convolutional Blocks 3) Recurrent Block 4) Classifier Head

Convolutional Front End: The initial Conv2D block accepts the input spectrogram of shape [batch,1,128,time]. This then downsamples the input along the frequency and time dimensions. This reduces the computational burden for deeper layers.

Residual Convolutional Blocks: The model includes 3 ResNetInv blocks and each block applies 2 Conv2D layers with ReLU and dropout, all wrapped in a residual skip connection. Layer normalisation is applied along the mel-frequency dimension before each convolution (n_features =128). These blocks help improve feature representation while preserving gradient flow.

Recurrent Block: The CNN output is now reshaped and passed through a linear adapter layer, mapping it to a dimension of size 512. We now have a stack of 5 Bidirectional GRU (BiGRU) layers. Each BiGRU layer consists of layer normalisation, ReLU activation and dropout. The recurrent block captures long range temporal dependencies in the audio signal

4.3 Attention Augmented Model

Now, moving onto the second architecture, the initial stack consists of convolutional and residual layers. This is pretty much retained from the baseline model. However, the sequence modeling component is completely replaced by a Transformer inspired encoder block that operates on spectrogram embeddings.

Front End: The initial Conv2D block accepts the input spectrogram of shape [batch,1,128,time]. This then downsamples the input along the frequency and time dimensions. This reduces the computational burden for deeper layers. This is followed by 3 ResNetInv blocks, and they consist of LayerNorm followed by Relu followed by Dropout and 2 Conv2D blocks. There is a residual skip connection in place around the entire block as well.

Linear Adapter: The output that we get from CNN is reshaped and sent through a linear layer mapping it to a fixed embedding dimension needed for the attention model.

Attention Based Sequence Modeling: The core architectural configuration of this model was handles by the stacked PreNormEncoder blocks (Transformer like encoders). Each block contained a 4 headed Multi-Head Self attention(MHSA) network and a Feed Forward Network(FFN) with 2 linear layers and a dropout. These MHSA and FFN layers are encapsulated with pre layer normalisation adn residual connections to improve the models stability and generalisation.

Classifier Head: The final linear classification head maps the output of the encoder described above to the vocabulary logits. Just like the baseline, this uses Linear, ReLU and Dropout functions

4.4 Conformer Based Model

The third model architecture that we developed was a convolutional and transfoer hybrid architecture and hence the name Conformer. The purpose of this was to to capture both local and global dependencies in speech.

There are three main components to this architecture :

- **Pre-processing module** : ConvSubsampling + Linear
- **Stacked Conformer blocks** : FFN + MHSA + Conv + FFN
- **Post processing module** : LSTM + Classifier

Pre-processing Module: The raw input spectrogram is sent through a Conv2D subsampling block, this reduces the time resolution. The output is flattened and passed through a linear layer (dropout is added here). The output is reshaped to be handled by the encoder.

Conformer Encoder Stack: Every encoder layer consists of the sub modules mentioned above and are executed sequentially with residual connections. The conformer FFN consists of two fully connected layers with a non linear SiLUU activation and dropout. A half residual weight is added in order to preserve training stability. The MHSA component implements multi head self attention. The positional encoding is relative sinusoidal and is created for each time step and then added to the input embeddings. Layer normalisation is applied before attention and residual connection after. The last submodule within the set of stacked conformer blocks is the convolutional block. This captures local structure of the speech using depthwise separable convolutions. There is a pointwise convolution operation, a GRU gate and then a depthwise convolution operation within this module. Batchnorm and SiLU are applied as required. Doing this helps preserve the temporal alignment using symmetric padding. The last bit of the convolutional encoding is the residual connections. The entire stack is repeated several times, thus creating a deep encoder that can model global attention and local phonetic patterns

Post processing module: Finally, an additional LSTM layer cleans/refines the output sequence to capture any additional sequential dependencies. A final linear layer maps the LSTM outputs to logits over the predefined vocabublary

4.5 Training Procedure

The training procedure remained relatively the same across all the 3 models developed. It was trained using a CTC(Connectionist Temporal Classification) loss. This loss function's salient feature is that it enables alignment free training on unsegmented audio transcript pairs.

- **Optimisation and Scheduler:** AdamW optimizer was used to handle sparse updates and control weight decay. A OneCycleLR scheduler was used to dynamically adjust the learning rate over epochs and used a linear annealing strategy. The CTC loss was applied to log-softmax outputs (transposed to [time, batch, class]).
- **Batch and Device Configuration:**Spectrograms and transcripts are all padded and batched and all computations were performed on Kaggle's T100 GPU.
- **Training Loop:** Now, for every batch the spectrograms and target labels were forwarded through the model. The CTC loss was computated based on the predicted log probabilities and target indices. Regular backpropagation for weight updates and optimization steps were applied and the scheduler was updated. Average training loss was computed and tracked for each epoch. Word Error Rate and Character Error rates were used as evaluation metrics for this ASR task.

5 Experiments

We implemented a series of experiments on three ASR architectures - DeepSpeech Baseline Model, an attention-augmented model, and a Conformer-based model. For each model, we executed various hyperparameter combinations, including batch size, learning rate, stride and the number of epochs. We considered Word Error Rate (WER), Character Error Rate (CER), and a reasonable training time as our primary evaluation metrics. WER is basically the fraction of words which were predicted incorrectly, and CER measures small errors such as misspelling and character errors. After identifying the best-performing hyperparameter configurations through validation dataset (dev-clean), we re-evaluated each model on the test dataset (test-clean) using the optimal combination of hyperparameters.

5.1 Overview of Experiments

Total experiments: 27
Total experiments per model:

- DeepSpeech Baseline Model: 9 (8 on validation set + 1 on test set)
- Attention-Augmented Model: 9 (8 on validation set + 1 on test set)
- Conformer Model: 9 (8 on validation set + 1 on test set)

5.2 Results and General Trends

Model	Experiment No.	Batch Size	Epochs	Stride	Learning Rate	Total training time (in mins)	Average training time per epoch (in mins)	Loss	WER	CER
Baseline Model	1	16	10	2	0.0005	254.34	25.43		0.5	0.1998
	2	32	15	2	0.002	308.36	20.55		0.999	0.9848
	3	8	5	1	0.00025	305.02	61.004		0.4888	0.2034
	4	16	25	2	0.0003	621.85	24.87		0.2998	0.1345
	5	8	10	2	0.005	334.12	33.41		1	1
	6	16	20	2	0.007	511.7	25.59		1	1
	7	32	5	2	0.0005	107.29	21.46		0.8245	0.3567
	8	8	10	2	0.0008	336.84	33.68		0.4325	0.1586
	Test Set	8	10	2	0.0008	337.18	33.72	0.5089	0.4318	0.151

Fig. 1. DeepSpeech Baseline Model Performance Summary

Model	Experiment No.	Batch Size	Epochs	Stride	Learning Rate	Total training time (in mins)	Average training time per epoch (in mins)	Loss	WER	CER
Attention Model	1	16	10	2	0.0005	193.22	19.32		0.7747	0.3017
	2	16	15	2	0.002	286.83	19.12		1	1
	3	8	15	1	0.00025	208.03	41.6		0.914	0.4269
	4	16	25	2	0.0003	477.92	19.12		0.5609	0.1989
	5	8	10	2	0.005	211.03	21.1		1	1
	6	16	20	2	0.007	379.89	18.99		1	1
	7	8	10	2	0.0008	196.15	19.61		0.652	0.2384
	8	16	5	2	0.0005	98.97	19.79		0.9076	0.3986
	Test Set	16	25	2	0.0003	475.83	19.03	0.6443	0.5391	0.1833

Fig. 2. Attention-Augmented Model Performance Summary

Model	Experiment No.	Batch Size	Epochs	Stride	Learning Rate	Total training time (in mins)	Average training time per epoch (in mins)	Loss	WER	CER
Conformer Model	1	8	10	2	0.0005	248.9	24.89		0.3049	0.104
	2	8	15	2	0.002	371.5	24.76		0.9844	0.9513
	3	8	5	2	0.0025	127.2	25.44		0.992	0.9197
	4	8	25	2	0.0003	617.04	24.68		0.749	0.4749
	5	8	10	2	0.005	248.86	24.88		1.0081	0.931
	6	8	20	2	0.007	493.25	24.66		1.0018	0.9486
	7	8	10	2	0.0008	252.25	25.22		1.0278	0.9182
	8	8	5	2	0.0005	128.45	25.69		5.1171	1.7095
	Test Set	8	10	2	0.0005	246.66	24.67	0.3426	0.3025	0.1003

Fig. 3. Conformer Model Performance Summary

Observations: We observed various consistent trends across all 3 ASR models. It was observed that the Conformer model outperformed the other two models by achieving the lowest WER (0.305) and CER (0.1003) on the test set. This result proves that Conformers are designed to effectively capture both local and global dependencies in audio sequences with the help of their hybrid combination of convolution and self-attention mechanisms. On the other hand, the Baseline model showed noticeable inconsistencies in its performance, with the WER values ranging from 0.5 to over 0.9 on validation dataset. This helps us in understanding that the model is highly sensitive to the changes in hyperparameter and is potentially overfitting in some executions. The Attention-based

model exhibited more stable performance across its experiments but did not outperform the Conformer model in final evaluation metrics.

Another pattern that we observed was the trade-off between training time and the model performance. By conducting multiple experiments, we realized that the longer training times, whether it is due to smaller batch sizes or higher number of epochs, did not really produce better WER or CER values. This suggests us that there's a certain point where investing more resources or time won't return better results. Additionally, the experiments where we used small or large learning rates often led to poor results, whereas the moderate learning rates (e.g., 0.0005) provided a balanced mix of stability and accuracy. We have used a fixed stride of 2 for almost all the experiments to maintain temporal consistency, so its effect was not observed independently and therefore we can consider it for future exploration.

In this way, these insights emphasize the importance of model design and well-balanced training configurations in order to achieve optimal performance for ASR applications.

5.3 Error Rate Comparison Across Models:

– **WER Analysis - Baseline Model**

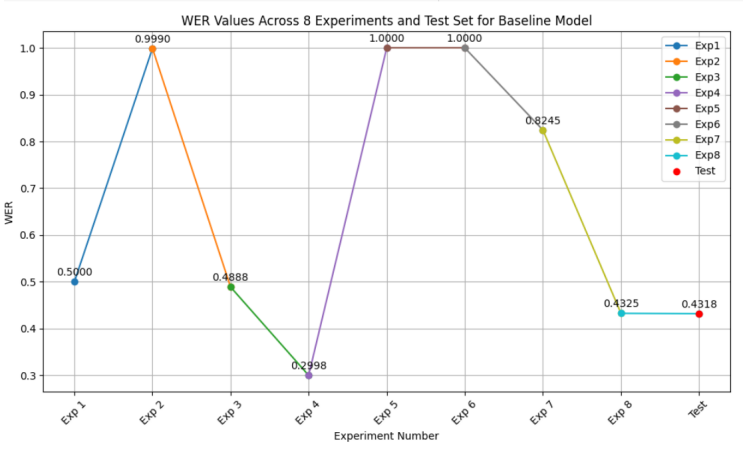

Fig. 4. WER Curve Across Experiments - Baseline Model

This plot highlights Word Error Rate (WER) values for a baseline model across 8 experiments on validation set and 1 experiment on test set. We can observe that the WER values vary significantly, ranging from 0.2998 to 1.000. This indicates that the model with the highest WER i.e. with value 1.000 is a complete failure in generating transcription. The test set WER of 0.4318 suggests that the DeepSpeech Baseline Model performs moderately and the

generated transcription results in nearly half of transcribed words incorrectly. The wide range of WER values highlights that the model's performance is highly sensitive to the different combinations of hyperparameters.

– **CER Analysis - Baseline Model**

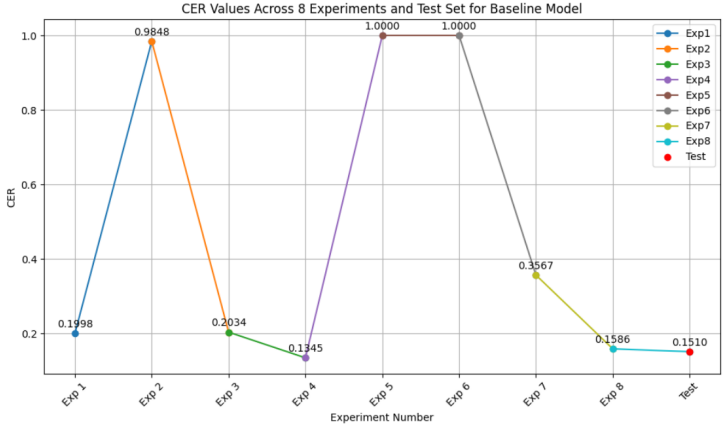

Fig. 5. CER Curve Across Experiments - Baseline Model

The Character Error Rate (CER) values for the baseline model show extreme variability. We can observe that the CER values range from 0.1345 to 1.000. The test set CER is 0.1510, and this indicates that there is a 15.1% character-level error rate, which is significantly better than the worst-performing experiments (e.g., 1.000). Experiment 4 showed a lower CER value of 0.1345, however as it took the highest training time of around 621.85 mins (please refer Fig. 1), we didn't consider this experiment as the optimal combination for test set. Therefore, the test set CER value of 0.1510 is promising but we should consider improving this value for practical use.

– **WER Analysis - Attention-Augmented Model**

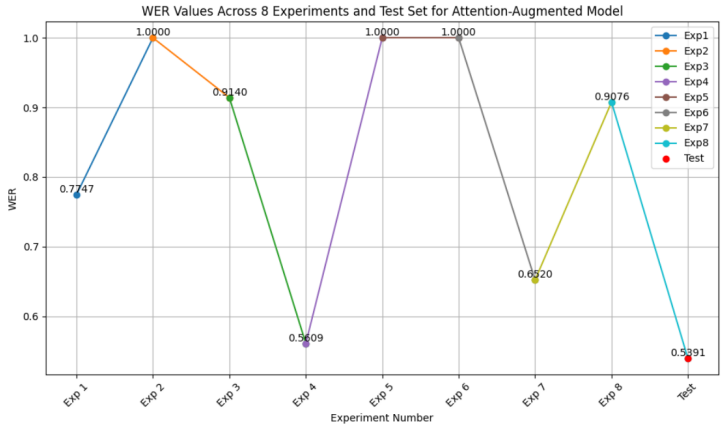

Fig. 6. WER Curve Across Experiments - Attention-Augmented Model

In this graph we can observe Word Error Rate (WER) values across eight experiments and a test set for an attention-augmented model. The hyperparameter combination for the test set produces the lowest error rate of 0.5391. On the other hand, the experiments 2, 5, and 6 exhibit the highest error rates of 1.000, which indicates that there are potential issues with the hyperparameters used. Thus, the results for different combinations of hyperparameters point out the importance of experiment optimization and model adjustments in order to enhance model performance and consistency across different circumstances.

– **CER Analysis - Attention-Augmented Model**

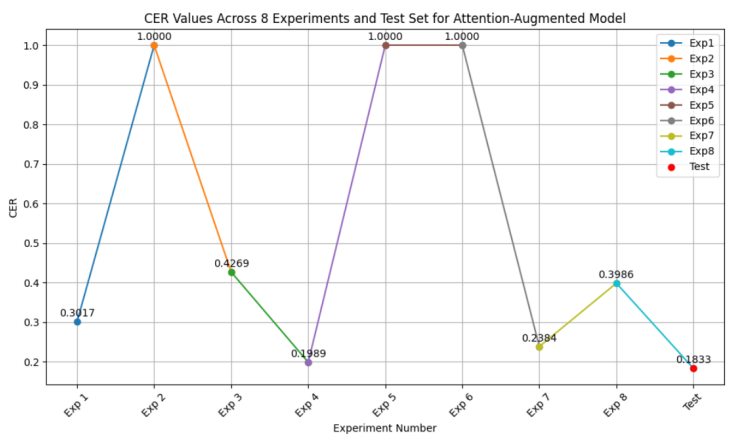

Fig. 7. CER Curve Across Experiments - Attention-Augmented Model

This graph highlights Character Error Rate (CER) trends across eight experiments on validation set and 1 experiment on test set for an attention-augmented model. The test set shows the lowest CER 0.1833, indicating strong accuracy, while Experiments 2, 5, and 6 record the highest error rates 1.000, suggesting significant instability in the model for the associated hyperparameter combinations.

– **WER Analysis - Conformer Model**

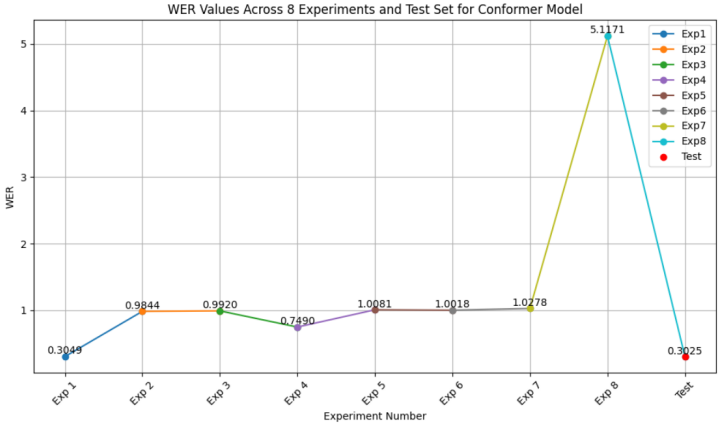

Fig. 8. WER Curve Across Experiments - Conformer Model

Similar to the WER analysis for Baseline and Attention model, this graph illustrates WER trend for a Conformer model. The test set hyperparameter combination demonstrates the lowest error rate of 0.3025, indicating strong model performance. On the other hand, Experiment 8 shows an extreme WER value of 5.1171, and highlights the faulty hyperparameter configuration. Experiments 2 through 7 show variability in WER values where some experiments exceed the WER value of 1.000 and this suggests that there are recurring optimization challenges while considering the hyperparameter combinations.

– **CER Analysis - Conformer Model**

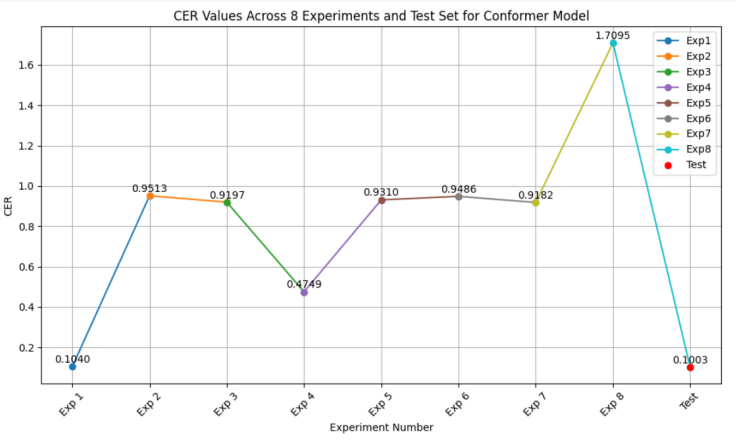

Fig. 9. CER Curve Across Experiments - Conformer Model

This graph analyzes Character Error Rate (CER) values for the Conformer Model. It highlights the significant fluctuations that were observed while implementing experiments using different hyperparameter combinations on the validation and test sets. We received the lowest CER value of 0.1003 for the test set. This helped us in identifying the model with strong accuracy. Whereas Experiment 8 exhibits the highest error rate of 1.7095. Experiments 2, 3, 5, 6, and 7 also show high CER values, which allows us to understand that there could be recurring challenges in hyperparameter combinations. These findings emphasize the importance of refined adjustments to enhance model accuracy and stability across different experiments.

5.4 Prediction Results on Test Set Audio Across 3 Model Architectures:

Original Transcript: He hoped there would be stew for dinner. Turnips, carrots, and roasted potatoes and that mutton pieces would be ladled out into thick, peppered flour sauce.

1. DeepSpeech Baseline Model
   - **Predicted Transcript:** *he hoped there would be sto por dinner turnips and carit and brosed petathose and that muten pieces to be ladled out ton tthick pepered flower in saus*
2. Attention-Augmented Model
   - **Predicted Transcript:** *he hope d ther would be stu her dinner turnetsand carrit send brused petae hos and fhat mutn pecet to be ladl dou in thick pevperdlower thatin sos*
3. Conformer Model
   - **Predicted Transcript:** *He hopered there wuld be stew for dinner turnips, carrit, and brosed petathoes and that mutten pieces to be ladled out into thick pepered flower in saus*

5.5 Limitations and Future Directions

Even though our experiments provided valuable insights into model performance and hyperparameter interactions, several questions remain unanswered. One key limitation is the fixed stride value used across almost all models. Using the same stride value prevented us from evaluating how stride length affects temporal resolution and recognition accuracy. Additionally, although we explored a range of hyperparameters, we believe that the search was not comprehensive.

There might be some impactful combinations that remain untested, such as testing with alternative feature extraction techniques or applying techniques to remove noise from the audio. For future work, we can test the model generalization by exploring how these models perform on more diverse and noisier datasets. We believe that this project experiments hold strong potential for improving our understanding and enhancing the real-world applications of ASR systems by making them more robust to diverse audios, with different speaker characteristics, and for multiple languages.

6 Conclusions

In conclusion, our project highlights the critical impact of model architecture on the performance of automatic speech recognition (ASR) systems. After the implementation and evaluation of three distinct models such as DeepSpeech-baseline, an attention-augmented architecture, and a Conformer model, we found that the Conformer model delivered the best results by achieving the lowest Word Error Rate (0.305) and Character Error Rate (0.1003) on the test dataset. This helps us understand how combining convolutional subsampling, multi-head self-attention, and LSTM-based post-processing effectively capture both local and global dependencies in speech data. We observed that the baseline model was sensitive to hyperparameters, and the attention model showed consistent but moderate improvements. On the other hand, the Conformer model proved the best balance between model complexity and performance. Furthermore, our experimental analysis emphasized that longer training times, extreme learning rates or complex architectures might not always result in better performance. Sometimes, it is important to focus on the model architecture design, thoughtful training strategies and using fine-tuned sets of hyperparameters to achieve optimal model performance. Even though these results from our experimental analysis have been promising, there are certain limitations such as a fixed stride configuration and limited hyperparameter combinations. This creates an opportunity for further exploration to improve the ASR performance across diverse audio environments.

References

1. Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., Ng, A.Y.: Deep speech: Scaling up end-to-end speech recognition (2014)

2. Amodei, D., Anubhai, R., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Chen, J., Chrzanowski, M., Coates, A., Diamos, G., Elsen, E., Engel, J., Fan, L., Fougner, C., Han, T., Hannun, A., Jun, B., LeGresley, P., Lin, L., Narang, S., Ng, A., Ozair, S., Prenger, R., Raiman, J., Satheesh, S., Seetapun, D., Sengupta, S., Wang, Y., Wang, Z., Wang, C., Xiao, B., Yogatama, D., Zhan, J., Zhu, Z.: Deep speech 2: End-to-end speech recognition in english and mandarin. (2015)
3. Miao, Y., Gowayyed, M., Metze, F.: Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding (2015)
4. Chan, W., Jaitly, N., Le, Q.V., Vinyals, O.: Listen, attend and spell (2015)
5. Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., Bengio, Y.: Attention-based models for speech recognition (2015)
6. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need (2023)
7. Xu, M., Li, S., Zhang, X.L.: Transformer-based end-to-end speech recognition with local dense synthesizer attention. In: ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE (June 2021) 5899–5903
8. Ruder, S.: An overview of multi-task learning in deep neural networks (2017)
9. Qiu, D., He, Y., Li, Q., Zhang, Y., Cao, L., McGraw, I.: Multi-task learning for end-to-end asr word and utterance confidence with deletion prediction (2021)