Automatic Speech Recognition with CTC Deep Neural Models

Alessandro Folloni, Gabriele Fossi, Daniele Napolitano and **Marco Solime**
Master's Degree in Artificial Intelligence, University of Bologna
{ alessandro.folloni2, gabriele.fossi, daniele.napolitano4, marco.solime }@studio.unibo.it

Abstract

This project deals with the task of Automatic Speech Recognition (ASR). The approach provided in this paper concerns Deep Learning architectures based on Connectionist Temporal Classification (CTC) loss. The objective is to compare the performance of 4 different models: Deep Speech 2, Jasper, Conformer and Wav2Vec2 which is a pretrained model we finetuned on a small supervised dataset. Some modifications in the model architectures are also implemented and evaluated. After running several experiments, the results show that Wav2Vec2 outperforms all the other models. This finding highlights the efficacy of transfer learning which allows us to achieve top results with very little extra training.

1 Introduction

Humans can communicate with each other in many possible ways, however speech is the primary means of communication (Biing-Hwang and Rabiner, 2005). Automatic Speech Recognition (ASR) is a task whose aim is to allow the machine to understand a person's speech and convert it into a series of words (Saliha Benkerzaz, 2019). With the growth of Big Data and computing power, ASR has become more and more important since it started to find several applications such as voice search, assistance and interactions with mobile devices (for example Siri on iPhone) and machine translation (Jinyu and et al., 2015). Therefore it is of interest to develop systems that can solve this task as good as possible.

In ASR, the most common generative learning approach is based on Gaussian-Mixture-Model based Hidden Markov models, used to represent the sequential structure of speech signals (S. et al., 2016). However, the practical application of Hidden Markov Models require considerable sophistication since simplifying assumptions involved in the application of its principles would lead to a system which has poor accuracy (Gales and Young, 2007). Moreover, with the advent of Deep Learning, the field of ASR has undergone a big shift. The use of multiple layers has allowed the creation of models that can extract intricate features from speech data (Mehrish et al., 2023).

For this reason our approach is based on Deep Learning architectures.

The aim of this project is to compare different existing models and modify their architectures in order to understand whether the modification leads to better results and find the best model. The models that we used for our experiments are Deep Speech 2, Jasper, Conformer and a pretrained model, Wav2Vec2.

The dataset used to train and evaluate the performance of the models is the LibriSpeech audio corpus, the most used dataset for ASR. There are two different versions, one with clean audio and one with non-clean audio, which makes it more challenging.

Several tests were performed. Conformer showed good results thanks to its hybrid architecture that combines convolutional and transformer layers. As expected, the performance of Deep Speech 2 was worse than that of Conformer which uses encoders with relative positional embeddings. However, using inverted ResNet blocks in place of normal ResNet blocks improved the results. On the other hand, the substitution of BiGRUs with Vanilla Prenorm encoders led to a worse performance. Finally, two different values of the strides were tested, namely 2 and 4. The best performance of Deep Speech 2 was achieved setting the stride equal to 2. On the other hand, Jasper did not have an excellent performance. The main issue was related

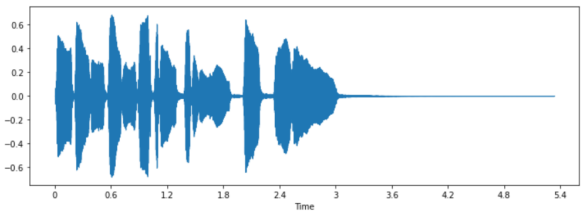

Figure 1: Audio wave: amplitude over time.

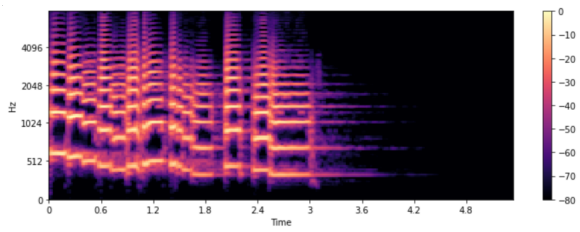

Figure 2: Mel spectrogram: frequency over time.

to the complexity and resource demands of its architecture which did not allow us to fully evaluate its power due to our computational constraints. However, despite the notable performance of Conformer, our findings show that Wav2Vec2, fine-tuned on a relative small dataset outperforms all the other models. Indeed, Wav2Vec2 leveraged unsupervised pretraining on large-scale audio speech data, which provided a significant advantage in terms of feature representation and accuracy. This shows that using a pretrained model integrated with an additional small training achieves superior results, underlining the effectiveness of transfer learning.

2 Background

2.1 Spectrograms

The raw audio files (amplitude over time, Figure 1) were converted into Mel spectrograms: a variation of standard Fourier-transform spectrograms (frequency over time, colored by amplitude, Figure 2). This variation focuses on the lower frequencies, audible for humans, making the spectrogram images more adapt for the speech recognition task. Consequently, in every method outlined in this study, audio inputs were processed as visual representations. For that reason, the initial module of any ASR model described in the next sections require the deployment of Convolutional Neural Networks (CNNs). The reason is to convert the input spectrogram images into high-level feature vectors, known as embeddings, which effectively represent the intricate patterns and characteristics essential for accurate speech recognition.

2.2 WER and CER metrics

Two critical metrics that serve as benchmarks for evaluating ASR systems, and are commonly used in almost any task-related paper, are the Word Error Rate (WER) and the Character Error Rate (CER). These error-based metrics are essential for quantifying the performance of ASR models, providing a standardized mean to assess and compare the efficacy of various systems.

Word Error Rate (WER) is the de facto standard for measuring the performance of ASR systems. It is calculated based on the number of errors at the *word* level, which includes substitutions, insertions, and deletions when compared to a reference transcription. The formula for WER is given by:

$$WER = \frac{S + I + D}{N}$$

where S is the number of substitutions, I is the number of insertions, D is the number of deletions, and N is the number of words in the reference. A lower WER indicates a higher accuracy of the ASR system, with a WER of zero representing perfect transcription.

Character Error Rate (CER), on the other hand, measures the performance at a more granular *character* level. It is particularly useful for languages where character-level transcription is more relevant or for systems that focus on handwriting recognition. Similar to WER, CER is calculated by the number of character-level errors compared to a reference. The formula for CER is:

$$CER = \frac{S_c + I_c + D_c}{N_c}$$

where S_c, I_c, and D_c represent the number of substitutions, insertions, and deletions at the character level, respectively, and N_c is the total number of characters in the reference.

2.3 Connectionist Temporal Classification (CTC) loss

The Connectionist Temporal Classification is a technique to solve sequence-to-sequence learning and is used in different tasks, including ASR. The idea behind CTC is to interpret a sequence of probability distributions over different labels produced by the network, as a probability distribution over all possible label sequences (Lia and Wanga, 2020). CTC computes the likelihood of a target sequence

y taking into account all possible alignments for the label and a given input length. The likelihood of the correct y given an input x is given by:

$$P(y|x) = \sum_{\pi \in V} P(\pi|x)$$

Where V constains all the alignments of a given length which are compatible to y, including possible insertions of blank tokens (Lee and Watanabe, 2021). For example if the output sequence would be (h, i, s), then some valid paths included in V would be $(h, \_, i, s)$ and $(\_, h, i, s)$.
The CTC loss consists in minimizing the negative log-likelihood:

$$L(y|x) = -\log P(y|x)$$

3 System description

3.1 Models

In order to tackle the challenging task of ASR, we implemented three different models, taking inspiration from SoTA architectures, namely DeepSpeech 2 (Hannun et al., 2014), Conformer (Gulati et al., 2020), and Jasper (Li et al., 2019). While reproducing the main architecture designs and layer patterns, we introduced some modifications at layer level and varied the total number of stages according to our computational constraints. Such models were trained completely from scratch on supervised audio speech data (Panayotov et al., 2015). In addition to that, we imported a pretrained model from Hugging Face, namely Wav2Vec2 (Baevski et al., 2020), and fine-tuned it on a relatively small amount of labeled data. When it came to training and evaluating the model, we followed the official documentation.

3.2 Preprocessing

Since we are dealing with both audio and text, we need to set-up a different pipeline for each modality. Regarding audio data, we converted raw audio waveforms into Mel Spectrograms and applied random masking, both on time and frequency. As proved in SpecAugment (Park et al., 2019), cutting out consecutive blocks has the benefit to enhance generalization capabilities on unseen data and avoid the risk of overfitting. Then, given a batch of spectrograms, we added padding to have them all have the same length. Concerning text normalization, we converted transcriptions to lower lowercase, got rid of special characters, and padded to max sequence length.

3.3 Word piece models

At the beginning of the models we inserted a subsampling module, which consists of a (stack of) convolutional layer(s) with stride greater than 1. The reason was to reduce the spatial dimension of the activations both on time and frequency axis, to keep the number of parameters under control. Due to the choice of the loss (CTC, (Graves et al., 2006)), we were constrained to keep the length of the input (audio speech) greater than the output (text transcription), so in practice we used stride 2 and 4. Regarding the latter, we decided to output a pair of characters (bigrams) for each time-frame instead of single characters (unigrams). Differently from the unigram model, where the output distribution is drawn over a set of 28 distinct characters (alphabet, plus apostrophe and space), the bigram model takes into account pairs of characters, and the output distribution is drawn over all the possible bigram combinations (704). Using the bigram approach, instead of trigrams or whole-words, allowed us to keep the vocabulary dimension under control and increase the striding on the time axis by a factor of 2, as the model generally delivers more information.

As it is common in ASR models, where the input dimension can vary according to the length of the recording and is generally longer than the output dimension, we used the Connectionist Temporal Classification (CTC) loss (Graves et al., 2006). In practice, the models output probability distributions over all the alphabet characters for each time-step in the sequence. CTC models learn to automatically align the transcript during training, so we did not need to manually align the speech audio data to the text transcription beforehand.

3.4 Deep Speech 2

Our implementation of Deep Speech 2 (Hannun et al., 2014) is composed of two major neural blocks: (1) a stack of convolutional layers and (2) a stack of recurrent layers.

After the preprocessing phase, we apply an initial convolutional layer that performs striding both on the time and frequency axis; we used 32 kernels, hence obtaining a dimension of 32 on the channel axis. We experimented with two approaches: one based on unigrams (single characters) and one based on bigrams (pair of characters).

Regarding the first stage, we use a stack of residual convolutional layers to learn audio features for

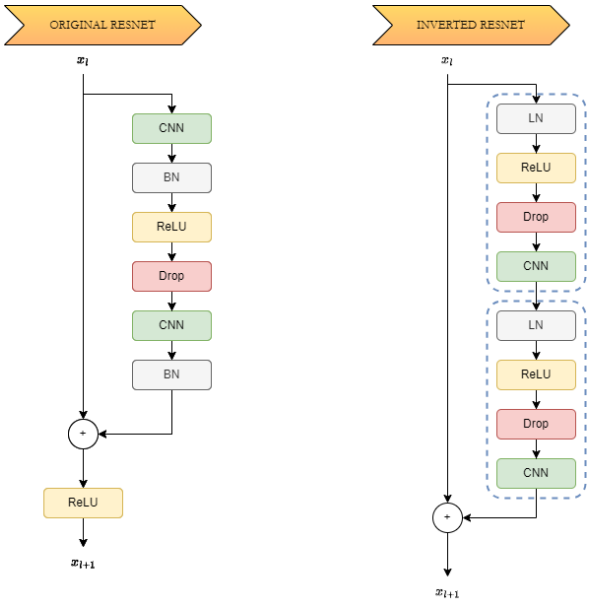

Figure 3: ResNet vs ResNetInv schemas implemented in our version of Deep Speech 2.

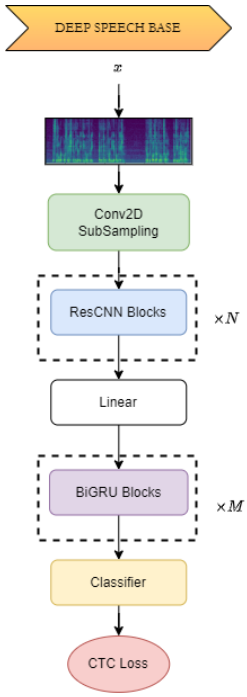

Figure 4: DeepSpeech architecture, using n ResNet blocks and m BiGRU layers.

each time frame, as suggested in Deep Residual Learning for Image Recognition (He et al., 2015). We experimented with two ResNet variants, as illustrated in Figure 3: the first pattern is the regular ResNet block; the second one uses a pre-norm strategy. As stated in Identity Mappings in Deep Residual Networks (He et al., 2016), the second approach is preferable as it leads to faster convergence. Additionally, we used Layer Norm instead of Batch Norm.

Regarding the second stage, we implemented a stack of recurrent neural networks (RNNs), as they naturally adapt to process input sequences. We used bi-directional RNNs because we want the context from both past and future frames. On the other hand, this choice limits us to the case of offline ASR, where the whole recording is needed to produce the output transcription. Differently from DP2 (Hannun et al., 2014) that uses Bi-RNNs, we deployed Gated Recurrent Units (GRU) after applying Layer Norm (Ba et al., 2016) (Figure 4).

Levering the power of Transformer architectures (Vaswani et al., 2023), we also experimented with Attention Layers. Here, we implemented the vanilla Prenorm encoder, without injecting positional embeddings. However, as shown in Conformer (Gulati et al., 2020), using positional embeddings leads to better performance especially on longer audio sequences.

As the final layer we use a linear block (classifier), which consists of 2 linear layers, which adapts extracted features to probability distributions.

3.5 Conformer

Our custom implementation of Conformer is inspired by the original paper architecture (Gulati et al., 2020). In particular, our model is midway between the Small and Medium models. The Conformer Encoder is composed by (1) a convolutional subsampling block that halves the spatial resolution and (2) by a stack of Conformer blocks. Such blocks comprises two feed forward modules that encapsulate a Multi-Head Self Attention block and a convolutional block. In our implementation, we enhanced attention blocks with sinusoidal positional embeddings, as they generally lead to better performances over longer sequences. Regarding the Decoder, we used a simple LSTM layer with 320 hidden units, as in the Conformer Small model. The final block is a linear layer that outputs a probability distribution over the single-character alphabet for each sequence frame (Figure 5).

3.6 Jasper

Differently from the previous architectures that deploy either recurrent or attention layers, Jasper (Li et al., 2019) is a fully convolutional network that leverages the strength of normalization layers (Batch Norm (Ioffe and Szegedy, 2015)) and skip connections. We tried both variants from the paper,

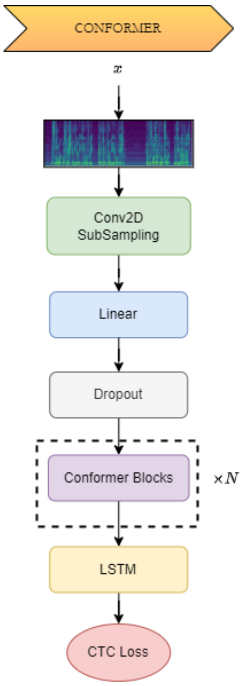

Figure 5: Architecture schema of the Conformer model.

namely Jasper with skip connections embedded in each Conv-BN-ReLU block and Jasper with skip connections towards all consecutive blocks (Figure 6). While such architecture is recurrent and attention free, as it uses only 1D convolutions and normalization layers, the number of parameters is substantially greater from the other 2 models, exceeding 100 million parameters. Indeed, the strategy of Jasper to learn good features is to stack a high number of convolution layers with skip connections.

3.7 Wav2Vec2

Wav2Vec2 (Baevski et al., 2020) is a model trained on unsupervised audio speech data, namely on the 960 hours split of LibriSpeech corpus (Panayotov et al., 2015). It learns good feature representation using Masked Language Modeling and Contrastive loss (figure 7). We took the architecture and the learned weights from Hugging face and fine tuned the model on a small split of LibriSpeech clean. In particular, we tried 3 different splits of increasing length: 10, 20, and 30 hours. In addition, in another experiment, we fine-tuned the FLEURS dataset (Conneau et al., 2022).

3.8 Decoding

A greedy decoding strategy was used. That means that given the output probability distribution over all characters for each input frame, we simply

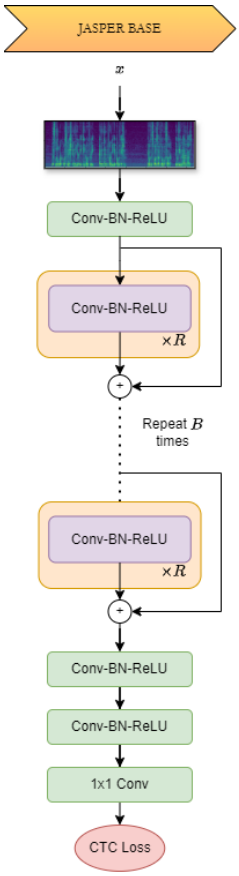

Figure 6: Architecture schema of the Jasper model.

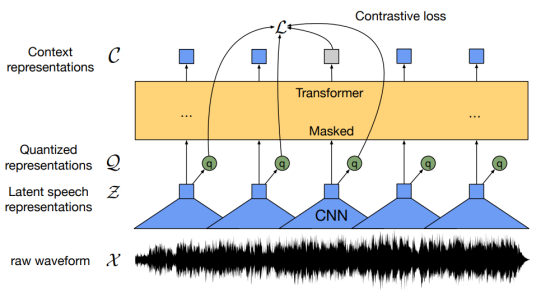

Figure 7: Schema representing the pretraining process for Wav2Vec2. Taken from (Baevski et al., 2020).

picked the most probable character. This turned out to be a simple yet robust approach. Other methods include Beam search strategy together with a Language Model trained on the LibriSpeech corpus, that however were not tried in our experiments.

In general, using a Language Model can provide guidance regarding more plausible transcriptions. Indeed, as shown in the results, our models generally produced grammar mistakes, misspelled words, missing characters; such nuances are almost impossible to catch given the sole audio speech. This is the typical case where having knowledge of the semantic meaning of the sentence – that is naturally provided by a Language Model – can solve utterance ambiguities and produce correct words, despite noisy or unclear recordings. However, as reported in the reference papers, we noticed that the use of LMs had a minor impact on WER/CER, while substantially increasing the complexity in terms of parameters.

We used a simple yet effective post processing pipeline, which particularly had a significant impact on WER. The intuition is that different phonemes tend to be uttered in varying time spans, namely the utterance speed may be faster or slower depending on the narrator. Concretely, given the output model transcriptions, we suppressed repeated characters, so to obtain well-formed and error-free words.

4 Data

4.1 LibriSpeech

All the models were trained and evaluated on the LibriSpeech audio corpus, (Panayotov et al., 2015) since it is the most used dataset for the Automatic Speech Recognition task. It is composed by public domain audio-book recordings, including a total of 1000 hours of English-speaking audio, sampled at 16kHz.
Furthermore, the LibriSpeech dataset is divided in the following subsets:

- **Training Data**. Consists of three partitions, each varying in the duration of speech:
    1. **train_clean100**: Contains 28,539 examples, totaling approximately 100 hours of clean speech.
    2. **train_clean360**: Comprises 104,014 examples, with approximately 360 hours of clean speech.
    3. **train_other500**: Includes 148,688 examples, providing roughly 500 hours of other (non-clean, more challenging) speech.
- **Development (Dev) Data**. It is further divided into two subsets:
    1. **dev_clean**: Comprises 2,703 examples with clean speech.
    2. **dev_other**: Contains 2,864 examples with other (non-clean) speech.
- **Test Data**. Mirrors the dev data division:
    1. **test_clean**: Consists of 2,620 examples with clean speech.
    2. **test_other**: Includes 2,939 examples with other (non-clean) speech.

4.2 FLEURS Dataset

For training purpose, we also conducted some experiments using the FLEURS dataset (Conneau et al., 2022).
It consists of short spoken utterances and their corresponding transcriptions across multiple languages. It includes the following key components::

- **Audio Files**: Short recordings in various languages.
- **Transcriptions**: Text transcriptions.
- **Metadata**: Speaker ID, language, and utterance ID.

The dataset is divided into training, validation, and test sets.
It is useful for speech recognition, language identification, multilingual modeling, and few-shot learning.

Its diversity and quality are essential for multilingual speech research.

5 Experimental setup and results

All computational training tasks were executed on a workstation equipped with a NVIDIA GeForce RTX 3090 Graphics card with 24 GB of VRAM, an Intel Core i7-7700K CPU @ 4.20GHz. For meticulous tracking and management of our experiments, we utilized the Weights and Biases platform. This platform facilitated the systematic logging of key performance metrics, including training loss, Character Error Rate (CER), Word Error Rate (WER), and learning rates throughout the various stages of training.

5.1 Training methodology

The models and its variants were trained taking into account our computational constraints. The training process was conducted over 10 epochs (with exception for Wav2Vec2, utilizing only 5 epochs), with a batch size equal to 16. The AdamW optimizer was employed, which is known for its weight decay regularization to prevent overfitting. This was coupled with a One-cycle learning rate policy that utilizes cosine annealing, 0.3% warmup rate, a maximum value of 1e-4 and a minimum of 1e-5, to ensure a smooth transition in learning rate adjustments throughout the training epochs.
In ASR models, the input dimension can vary according to the length of the recording and usually it is longer than the output dimension. For this reason we decided to use the Connectionist Temporal Classification (CTC) loss. The models output probability distributions over all the alphabet characters for each timestep in the sequence. CTC models learn to automatically align the transcript during training, allowing us to avoid to manually align the speech audio data to the text transcription beforehand.

5.2 Architectures

We evaluated the following models: Deep Speech 2, Jasper, Conformer, and Wav2Vec2. Initially, we adopted the architectures for Deep Speech 2, Jasper, and Conformer as delineated in Section 3. Subsequent iterations involved strategic alterations to their configurations and structures. This iterative process was instrumental in discerning the optimal architectural modifications that would culminate in superior performance outcomes.

Deep Speech 2

For Deep Speech 2, several experiments were performed. The first test consisted in assessing whether the usage of inverted ResNet blocks in place of standard ResNet blocks would improve the performance, as we theorized. To this end, two architectures were compared: the first one consisting of 3 ResNet blocks and 5 BiGRUs, while the second one of 3 inverted ResNet blocks and 5 BiGRUs.
On the other hand, the second experiment was to compare the previous Deep Speech architecture with its variant called Deep Speech Attention. This test consists in establishing the performance of using Vanilla Prenorm encoders in place of BiGRUs. Both configurations had 3 inverted ResNet blocks. The difference was in the use of 5 BiGRUs versus 12 encoders. All the BiGRUs and encoders used an embedding dimension equals to 256. For both the first two tests a stride of 2 on the time axis was used. As a consequence, the ouput would be a probability distribution over 29 characters (Unigram models). Hence, for the third test related to Deep Speech 2, it was of interest to try a stride of 4. In this case, the output would be a probability distribution of 705 pairs of characters (Bigram models). The configurations of the previous test with a stride of 2 were compared with the same configurations but with a stride value equals to 4.

Jasper

For Jasper, we tested its base architecture versus its Dense Residual (DB) version which contains skip connections. Both of these versions are illustrated in the original paper (Li et al., 2019). However, the preliminary results that we obtained were disappointing, therefore we decided not to proceed further with this model, since we could not achieve any meaningful transcriptions.

Conformer

Finally, the Conformer architecture was implemented. It is a mixture between the small and the medium versions which are illustrated in the paper (Gulati et al., 2020). Conformer uses encoders with relative positional embeddings therefore we expected it to outperform Deep Speech Attention which uses Vanilla encoders. For this model we did not perform experiments based on architectural modifications but instead we tried to use different datasets for training. Both the datasets are part of the LibriSpeech collection but they vary in the time length: one contains 100 hours of audio while the other one 360 hours. In both cases a stride of 2 was used.

5.3 FLEURS dataset experiments

In our experimentation with the FLEURS dataset, we encountered several factors that contributed to the observed suboptimal results. We used it as training set of Wav2vec2, while the testing was always performed on LibriSpeech.

The FLEURS dataset, designed to support multilingual speech models, presents inherent challenges due to its diverse linguistic and acoustic variability. This diversity, while beneficial for training models on a wide range of languages and dialects, also

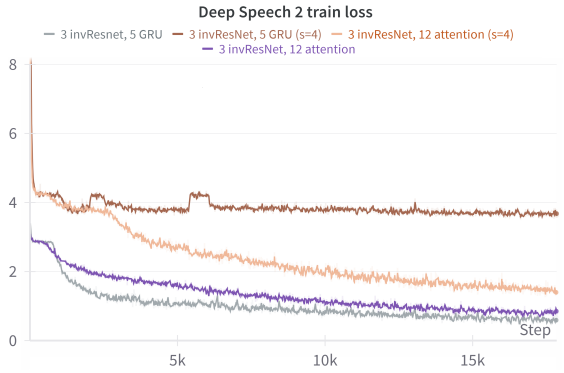

Figure 8: DeepSpeech2 train loss graphs. In the legend, stride=2 when not specified.

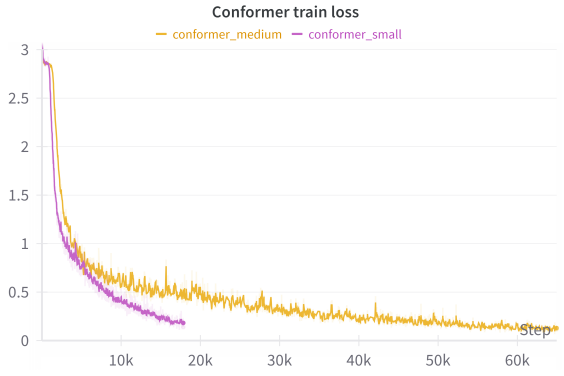

Figure 9: Conformer train loss graphs (10 epochs for both, but different datasets).

introduces complexities. Models trained on such a heterogeneous dataset may struggle to generalize effectively to benchmarks or tasks that prioritize more standardized linguistic or acoustic conditions.

So, despite the reasonable attempt, the different distribution between the two datasets involved, may have contributed in not getting exceptional results. Although we had used many combinations of hyperparameters, our main benchmark (i.e. WER) never improved. Our experimentation underscores the critical impact of dataset distribution on model performance. Although the FLEURS dataset is invaluable for developing and evaluating multilingual speech models, its effectiveness is limited when the evaluation data does not share similar linguistic and acoustic properties.

6 Discussion

Table 6 presents a comparison of our three leading models against the established baselines derived from the foundational literature. Despite the constraints of our modest hardware and resources, which precluded us from achieving equivalent results, it's noteworthy that our fine-tuned Wav2Vec2 model managed to outperform some baseline architectures. Notably, it surpassed the RNN variants of DeepSpeech2 for dev-clean, and Jasper 5x3 for dev-other, possibly suggesting that Wav2Vec2 may possess enhanced robustness against noisy inputs.

6.1 Experiments analysis

Upon further examination of the individual tables reporting the outcomes of our experiments for each approach (Tables 1, 2, 3, 4 and 5), we have drawn the following conclusions.

Deep Speech 2

- The inverted ResNet architecture works better than the standard version for the spectrogram encoding (Table 1), moreover the standard ResNet model was the worst of all the models we tested.

- BiGRUs behaved better than vanilla Prenorm Encoders (Table 2). The choice of the number of encoder layers was guided by the number of parameters of the BiGRU architecture, in order to reach roughly the same size.

- Unigrams (stride 2) performed better than bigrams (stride 4), both for BiGRUs and Vanilla Prenorm Encoders (Table 3).

With reference to Image 8, the training loss graph for each model is consistent with the outcomes presented in the previously analyzed tables.

Conformer

Only one architecture configuration was used, but trained on two versions of LibriSpeech's train splits (train_clean100 and train_clean360), to measure how well it could scale given more train data. As expected, the model trained with more data got better results, but the performance boost is not that significant (Table 4). Interestingly, while the WER is lower for the 360h train, the CER is instead better for the smaller 100h dataset. This could imply that the former model might be overfitting on character-level details when provided with less data.

Taking a look at Image 9, the Conformer trained on train_clean360 managed to reach a lower loss, suggesting that it is effectively utilizing the additional data to refine its parameters and improve

Table 1: ResNet vs inverted ResNet blocks (DeepSpeech)

model type	**# params (M)**	**train-clean**		**test-clean**	
		cer	wer	cer	wer
3 ResNet, 5 BiGRU	23,7	69,89	121,58	70,61	138,69
3 ResNetInv, 5 BiGRU	23,7	**20,02**	**41,95**	**9,61**	**28,57**

Table 2: BiGRUs vs Vanilla Prenorm Encoder (DeepSpeech)

model type	**# params (M)**	**train-clean**		**test-clean**	
		cer	wer	cer	wer
3 ResNetInv, 5 BiGRUs	23,7	**20,02**	**41,95**	**9,61**	**28,57**
3 ResNetInv, 12 Encoders	20,3	25,28	55,99	14,45	44,99

Table 3: Unigram (stride 2) vs bigrams (stride 4) (DeepSpeech)

model type	**stride**	**# params (M)**	**train-clean**		**test-clean**	
			cer	wer	cer	wer
3 ResNetInv, 5 BiGRUs	2	23,7	**20,02**	**41,95**	**9,61**	**28,57**
3 ResNetInv, 5 BiGRUs	4	24	74,58	93,62	74,75	95,55
3 ResNetInv, 12 Encoders	2	20,3	25,28	55,99	14,45	44,99
3 ResNetInv, 12 Encoders	4	20,6	33,74	67,98	33,74	67,98

Table 4: Conformer model trained on both the 100h dataset (small) and 360h (medium). The results below are reported only for WER metrics, tested on the dev splits of LibriSpeech.

model type	**# params (M)**	**train-clean**		**test-clean**		**train time**
		cer	wer	cer	wer	
Conformer 100h	27,3	**4,53**	14,63	**7,39**	22,58	2h 11m
Conformer 360h	27,3	5,88	**12,48**	8,62	**18,18**	6h 44m
model type	**# params (M)**	**dev**		**test**		**train time**
		clean	other	clean	other	
Conformer 100h	27,3	22,37	46,73	22,58	48,76	2h 11m
Conformer 360h	27,3	**17,95**	**36,53**	**18,18**	**38,29**	6h 44m

Table 5: Wav2vec2 fine tuned for 5 epochs on increasingly bigger fractions of the LibriSpeech dataset (in WER).

Train set	**dev-clean**	**dev-other**	**test-clean**	**test-other**
LIBRI small (∼10 hours)	22,85	27,82	19,56	28,11
LIBRI mid (∼20 hours)	11,14	21,4	11,42	21,73
LIBRI large (∼30 hours)	**9,02**	**18,61**	**9,45**	**18,8**

Table 6: Comparison between our models (top three rows), and the baseline results from the original papers, for the WER metric.

model type	# params (M)	dev		test	
		clean	other	clean	other
DeepSpeech, 3 ResNetInv, 5 BiGRUs	23,71	37,01	60,77	28,57	63,42
Wav2Vec2 30 hours (ours)	-	9,02	18,61	9,45	18,8
Conformer 360h (ours)	27,3	17,95	36,53	18,18	38,29
Jasper 5x3 BN ReLU (paper)	100	8,82	23,26	-	-
Jasper 10x4 BN ReLU (paper)	201	6,15	17,58	-	-
Jasper DR 10x5 (paper)	333	3,64	11,89	3,86	11,95
Conformer(S) (paper)	10,3	2,7	6,3	2,1	5
Conformer(M) (paper)	30,7	2,3	5	2	4,3
Confomer(L) (paper)	118,8	**2,1**	**4,3**	**1,9**	**3,9**
DeepSpeech2 5 RNN (paper)	38	9,78	-	-	-
DeepSpeech2 7 RNN (paper)	38	9,52	-	-	-
DeepSpeech2 5 GRU (paper)	38	7,79	-	-	-
DeepSpeech2 7 GRU (paper)	38	8,19	-	-	-

its learning. This is consistent with the expectation that more data can lead to better model performance, as it provides the model with more examples to learn from and potentially a better generalization to new data.
However, the marginal performance boost highlights the importance of balancing the quantity of data with the computational resources and time required for training.

Wav2Vec2

The pretrained and fine-tuned model was by far the model with the best performances, even if it used the least amount of data for fine-tuning. The results suggest that the pre-train/fine-tune paradigm is preferable from the trained-from-scratch models. Leaning from unsupervised data has the benefit that (1) ground truths are not needed and (2) the model is almost ready for the downstream task. In Table 5, the same pre-trained model was fine-tuned on increasing fractions of the train_clean100 dataset: 10h, 20h and 30h. While the 10 hours split is sufficient to get decent results, letting the model fine-tune on the 30 hours split greatly improves the performance on the test splits in terms of WER.

6.2 Error analysis

Another way to see how different models behave is to look and compare their outputs. All the results in this section come from the first sample of the test_clean portion of LibriSpeech, which is also a particularly tough example. In the original audio, a male voice reads this sentence with a clear and slow tone:

> "he hoped there would be stew for dinner turnips and carrots and bruised potatoes and fat mutton pieces to be ladled out in thick peppered flour fattened sauce"

Below, the predictions for the same sentence on our trained models. Errors are highlited in red.

DeepSpeech (5 ResNetInv, 3 BiGRUs). WER=0.28:

> "he hoped there would be stoo her dinner turnips and carrates and brozed potatos and fhat muten pieces to be latled out in t thick pepered flower fatend sous"

1) The word "stew" is actually pronounced "stoo" in English, so without sentence comprehension it is impossible to catch the error.

2) The word "carrots" is seemingly pronounced "carrets" in the audio recording. Indeed, the model translated the detected sound without reconciling to a predefined lemma.

3) Regarding the repetition of the “t” before “thick”, we notice that in the audio recording there is indeed a quick "t" pronounced before "thick", that of course is not mirrored in the transcription. So, in this case the audio model correctly identified the sound but was tricked by a human misspelling.

DeepSpeech (5 ResNetInv, 12 encoders). WER=0.45:

> "he hoped ther would be stoo her diner turnipsan carit send brosed potae hos and that muttent peces to be latledoutin e tick pepered flowr thatten sos"

Differently from the 3 BiGRU version, the model not only produces spelling mistakes, but has difficulties to correctly separate words (e.g. "turnips and" becomes "turnipsan", "ladled out in" becomes "latledoutin"). This problem may be due to the absence of positional encodings in the Transformer block, as we really want to be aware of the order of phonemes to avoid misspellings.

Conformer (trained on train_clean360). WER=0.18:

> "he hoped there would be stewver dinner turnips and carrots and bruised potatoes and fat mutton pieces to be latled out in thick peppered flouer fat an sauce le"

The Conformer model has the ability to correctly spell challenging words ("carrots", "potatoes") and correctly separate them. Indeed, it almost guessed the right transcription for the word "stew", wrongly merging it with the subsequent word "for".

Wav2Vec2 (30h). WER=0.25:

> "he hoped there would be stwe for dinner turnips and carrets and brused potatos and fat muttein pieces to be ladled out in thick peppered flower fateined sauce"

The Wav2Vec2 model, due to the good feature representations learnt during unsupervised pretraining, correctly disambiguates between words and produces well form words, while sometimes being tricked by utterance nuances ("stew" becomes "stwe", "carrots" becomes "carrets").

In general, the models produce spelling errors, especially on less common words (almost never on common or short words). In addition, in English language, a lot characters are written but not pronounced, or written differently. A word may also have different pronunciations depending on the context. A viable improvement could be to post-process the transcriptions with a Language Model, which is able to correct the spelling errors given the context. The LM could be trained on the LibriSpeech train split of text transcriptions and used to weight the output text.

7 Conclusion

In this work, we explored and implemented several state-of-the-art architectures to tackle the challenging task of Automatic Speech Recognition (ASR). We experimented with Deep Speech 2, Conformer, Jasper, and a pretrained Wav2Vec2 model, each offering unique insights and capabilities.

Our findings indicated that the pretrained Wav2Vec2 model, fine-tuned on a relatively small amount of labeled data, achieved the best performance. This model leveraged unsupervised pretraining on extensive audio speech data, which provided a significant advantage in terms of feature representation and overall accuracy. The use of the Wav2Vec2 model highlights the effectiveness of transfer learning in ASR tasks, allowing us to achieve superior results with minimal additional training.

The Conformer model also showed promising results, particularly due to its hybrid architecture combining convolutional and transformer layers. This model, inspired by the original Conformer design, performed well on both small and medium datasets from the LibriSpeech collection. Its ability to handle longer audio sequences effectively, thanks to positional embeddings, made it a strong contender in our experiments.

On the other hand, our attempts to implement the Jasper architecture faced challenges. Despite its potential, the complexity and resource demands of the fully convolutional network, which exceeds 100 million parameters, hindered our ability to fully evaluate its performance within our computational constraints. This indicates a need for further optimization and more powerful hardware to leverage Jasper's capabilities fully.

Our initial implementation of Deep Speech 2, while providing valuable insights into the ASR problem, delivered subpar results. The model's reliance on older techniques, such as recurrent neural networks and residual connections, limited its performance compared to more modern architectures. Additionally, our experiments with integrating transformer-based attention layers into Deep Speech 2 did not yield the expected improvements, suggesting that the attention mechanism may require more careful tuning or larger datasets to be effective in this context.

In summary, our experiments demonstrate the critical role of leveraging pretrained models and advanced architectural designs in achieving state-

of-the-art performance in ASR. While Wav2Vec2 emerged as the most effective solution, the Conformer model also showed considerable promise. Future work should focus on optimizing the implementation of Jasper and exploring more sophisticated transformer-based approaches to further enhance ASR performance. Furthermore, the integration of a language model designed to correct spelling inaccuracies in raw transcriptions is expected to significantly enhance the accuracy of the final outputs.

8 Links to external resources

- LibriSpeech: https://www.openslr.org/12

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations.

Juang Biing-Hwang and Lawrence R. Rabiner. 2005. Automatic speech recognition – a brief history of the technology development.

Alexis Conneau, Min Ma, Simran Khanuja, Yu Zhang, Vera Axelrod, Siddharth Dalmia, Jason Riesa, Clara Rivera, and Ankur Bapna. 2022. Fleurs: Few-shot learning evaluation of universal representations of speech.

Mark Gales and Steve Young. 2007. The application of hidden markov models in speech recognition.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks. volume 2006, pages 369–376.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. 2020. Conformer: Convolution-augmented transformer for speech recognition.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. 2014. Deep speech: Scaling up end-to-end speech recognition.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift.

Li Jinyu and et al. 2015. *Robust automatic speech recognition: a bridge to practical applications.* Elsevier.

Jaesong Lee and Shinji Watanabe. 2021. Intermediate loss regularization for ctc-based speech recognition.

Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiev, Jonathan M. Cohen, Huyen Nguyen, and Ravi Teja Gadde. 2019. Jasper: An end-to-end convolutional neural acoustic model.

Hongzhu Lia and Weiqiang Wanga. 2020. Reinterpreting ctc training as iterative fitting.

Ambuj Mehrish, Navonil Majumder, Rishabh Bhardwaj, Rada Mihalcea, and Soujanya Poria. 2023. A review of deep learning techniques for speech processing.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210.

Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. In *Interspeech 2019*, interspeech$_2$019.$ISCA$.

Karpagavalli S., , and Edy Chandra. 2016. A review on automatic speech recognition architecture and approaches. *International Journal of Signal Processing, Image Processing and Pattern Recognition*.

Abdeslam Dennai Saliha Benkerzaz, Youssef Elmir. 2019. A study on automatic speech recognition. *Journal of Information Technology Review*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.