AUTOMATIC SPEECH RECOGNITION

PALAMANICKAM

VINAYAK RAI

BASICS

Training loss higher than val loss

refer blog

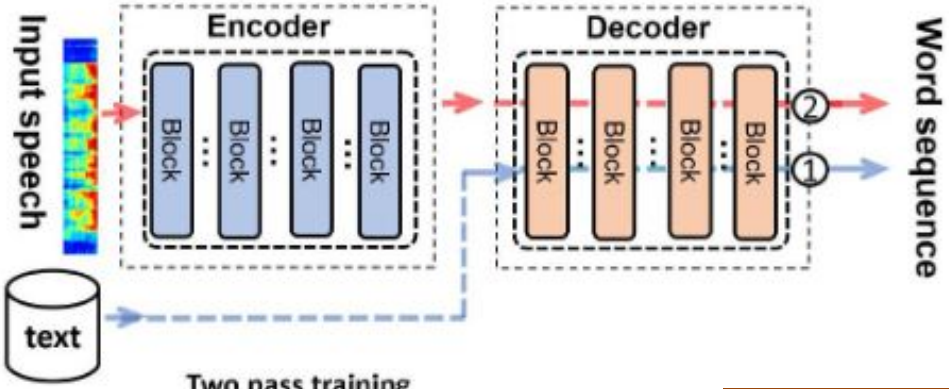
Significant fluctuations in both losses suggest a too-high learning rate, causing overshooting. A very slow decrease suggests a too-low learning rate, making training inefficient.the reported training loss is the average of the loss over all the batches within each epoch. As the model processes different batches, each with unique data distributions and optimal weight configurations, it updates its parameters after each batch (in stochastic optimizers). These frequent updates can introduce inconsistencies, causing fluctuations in the training loss and making it appear less stable.

Both types of regularization increase the training loss by adding these penalties to the base loss, but they are not applied during testing, leading to different calculations.

While a higher training loss might initially seem concerning, recognizing the role of regularization helps contextualize these values and leads to more accurate inferences about model performance.

So finally compute train loss without regularization penalties will give accurate True loss of model which is lower.

Cause: batch norm , data augmentation , dropout , shuffling



Two pass training

① training decoder only with text

② training whole network with speech

Alignment Process

- We follow a 3-step process for alignment -

1. Use existing ASR systems to generate text with timestamps (may be noisy)
2. Align noisy ASR text with reference text – Needleman Wunsch
3. Get sentence level audio-text pairs

ASR Flow

- Spectrogram generator that converts raw audio to spectrograms.
- Acoustic model that takes the spectrograms as input and outputs a matrix of probabilities over characters over time.The neural network is present in this acoustic model
- CTC Decoder (coupled with a language model) that generates possible sentences from the probability matrix.

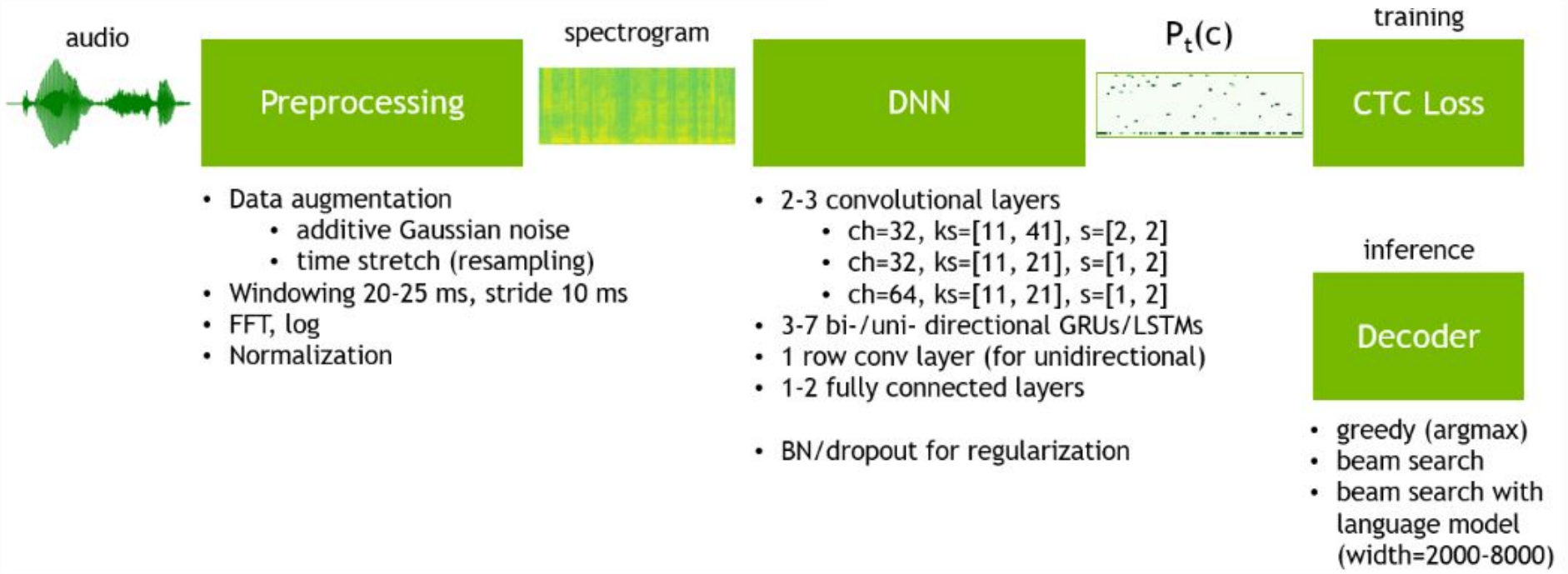
Refers

Sample Model

refers

Parameter	Value
Number of Epochs	30
Early Stopping Patience	7
Training Batch Size	16
Evaluation Batch Size	8
Warmup Size	1/10 of total stepsize
Gradient Accumulation Size	2
Learning Rate	3e-4 (3e-5 for Iban, Quechua, Hupa)

Table 4: Parameters used to train Wav2Vec XLSR-53.



ASR Model

- Feature extraction and padding
- Encoder (2 CNN2d + 1 CNN1d + 5bi-GRU + FC)
- CTC loss and forced alignment of character with the encoder output.
- Inference CTC Decoder
- Statistical Language Model for spelling correction and next word prediction.

Encoder types

RNN encoder:

- Capture the temporal features and sound changes over time.Understand the sequence of sound.
- High computation as the parameter increase and increase the latency too.

CNN encoder:

- Capture the short term pattern and isolated sounds.Learn the rapid change over the time.
- Struggles to learn the long term dependencies and transition of sound(Tsunami)

Transformer encoder:

- Capture the long term relation and global dependencies .
- More than 100 hr data is required and computational expensive(high parameter)

Tradeoff of accuracy and latency:

So we choosed Hybrid RNN = CNN + RNN layers

- It have less latency and low power as compared to hybrid transformer.
- The model parameter can be fixed upto 10 million (conformer 40M)

Feature Extraction:

1. The raw audio data of 24hr , which is splitted into 13100 files of audio with transcript.
2. Each file raw audio is converted into mel spectrogram (128 mel bins , 25 ms each frames) using the librosa library.
3. The mel spectrogram has to be padded in same length by adding zeros.

Encoder :

- So basically , the encoder will receive the 128 mel features for each time stamps as a input.
- The CNN will capture the isolated feature of sound and reduces the dimensions of mel spectrogram(reduce the computation).
- Passing 1d Convolution layer to capture the transition of audio in time axis.
- Now Bi-GRU layer will capture the long term relations of sound over time.
- Finally the fully connected layer projects the encoder output to character probabilities as a output for each time frames.

CNN Feature Extractor : Input/Output Shape:

- **Conv2d (1 → 32)**
  Input: [1, 1, 128, 477]
  Output: [1, 32, 128, 477]

- **Conv2d (32 → 64)**
  Input: [1, 32, 128, 477]
  Output: [1, 64, 128, 477]
- more channels help extract better high-level acoustic features.

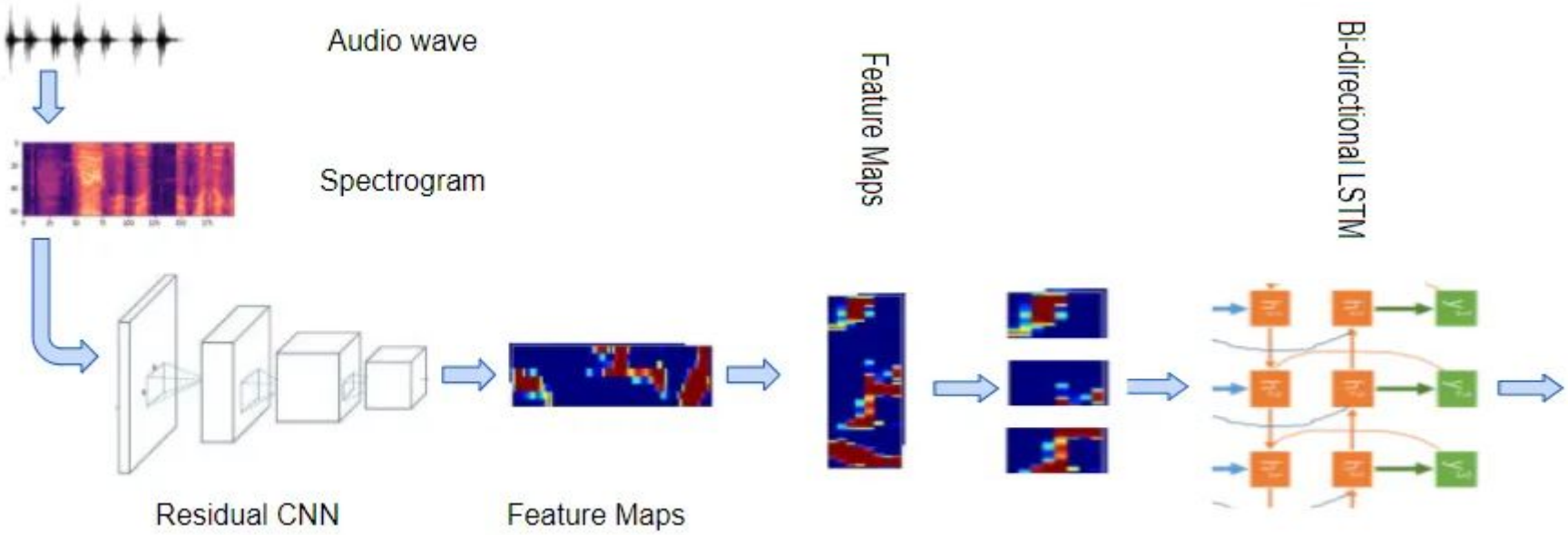
Conv1D Bottleneck Layer

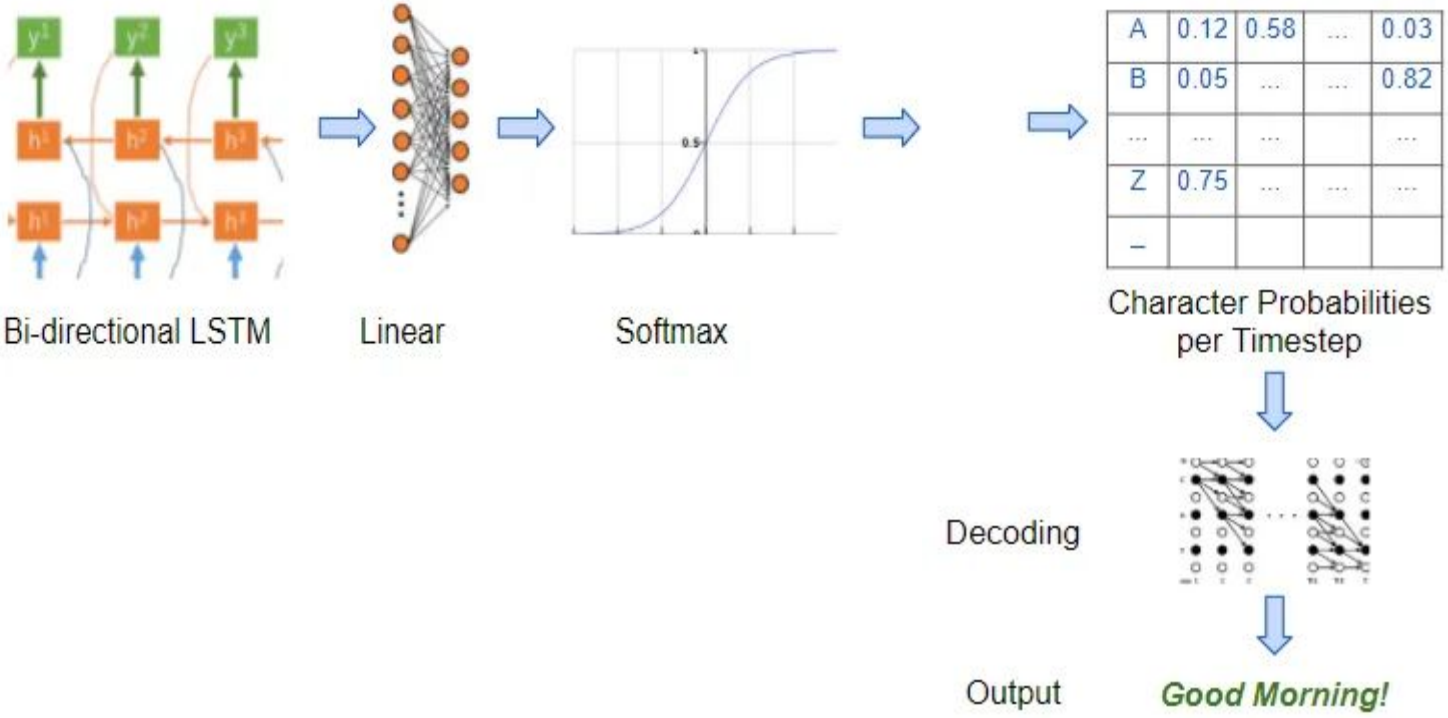
After flattening the CNN output**:**

- **Input:** [1, 8192, 477]

- **Output:** [1, 256, 477]

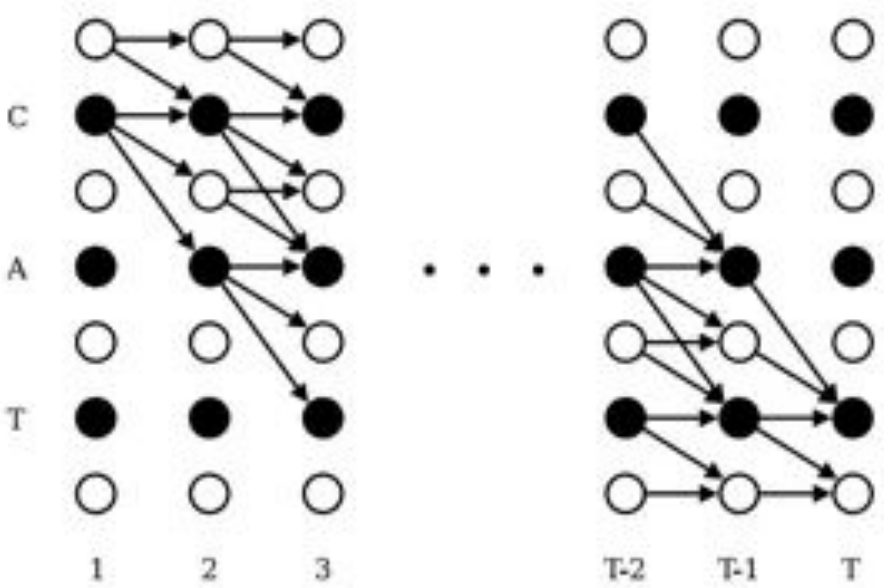
- **GRU Layer Analysis**
- **Final Linear Layers (Decoder)**
  - **LayerNorm + Linear(512 → 256) + GELU + Dropout → Linear(256 → 29)**


Audio wave
Spectrogram
Residual CNN
Feature Maps
Feature Maps
Bi-directional LSTM



https://medium.com/towards-data-science/audio-deep-learning-made-simple-automatic-speech-recognition-asr-how-it-works-716cfce4c706

CTC aligning path



where the entire word is “CAT”. Each column is the time when it’s uttered. At each time step (again, each column), we’re picking which label could go next, which is demonstrated by the arrows.

refers

CTC addresses this by allowing the model to learn to align the input and output sequences during training. the goal is to find the most likely alignment between X

Length: 3,
Output token 'g'

Output on every time step

	T1	T2	T3
_	0.4	0.3	0.2
g	0.6	0.7	0.8

All valid path	Path prob
g g g	= 0.6*0.7*0.8 = 0.336
g g _	= 0.6*0.7*0.2 = 0.084
g _ _	= 0.6*0.3*0.2 = 0.036
_ g _	= 0.4*0.7*0.2 = 0.056
_ g g	= 0.4*0.7*0.8 = 0.224
_ _ g	= 0.4*0.3*0.8 = 0.096

Loss = -ln(sum all path) = -ln(0.336+0.084+0.036+0.056+0.224+0.096) = 0.18392283816

refers

Helps train the model by ensuring that all possible alignments contribute to the loss.The model learns to assign high probabilities to valid alignments.
Conceptually, they both compute path probabilities over all alignments.
CTC Loss (Training): Sums over all paths to get total probability.
CTC Exhaustive Decoding (Inference): Selects the most probable transcription from valid paths.

Key Difference:

- CTC loss computes the probability of the correct transcription (learning objective).
- CTC decoding selects the best possible transcription (inference objective).

CTC Exhaustive Decoding (Inference):

$$y^* = \arg\max_y \sum_{\pi \in A(y)} P(\pi|x)$$

CTC Loss (Training)

$$\mathcal{L}_{CTC} = -\log \sum_{\pi \in A(y)} P(\pi|x)$$

https://www.assemblyai.com/blog/an-overview-of-transducer-models-for-asr/

CTC Advantages

- CTC models are easier to train! A CTC model has a single module, the encoder. This simplicity makes CTC really easy to train.
- There are more resources available for CTC models. Since CTC models have been the most popular architecture for Speech Recognition for so long, there is a large amount of research and open source tools to help you quickly build and train them.

CTC Disadvantages

- CTC models converge slower! Although CTC models are easier to train, we notice that they converge much slower than Transducer models. When training CTC models, more iterations are always needed than Transducers to achieve similar results.
- CTC models are worse with proper nouns. When it comes to proper nouns, CTC models seem to be less accurate. Proper nouns are rare words that may or may not be in the training corpus.
- CTC models require an external Language Model (LM) to perform well.

Number of Filters in Convolutional Layer makes difference

TABLE I. THE ARCHITECTURE OF THE CRNN MODEL

Layers of the CRNN Model	Parameters
Sound Input Preprocessing Layer	To 81 Frequency Bins as Input
Convolutional Embedding Layer (CEL)	Number of Filter = 200, Kernel size = 11, stride = 2, padding = 'valid'
Bidirectional GRU	Dropout = 0.25
Bidirectional GRU	Dropout = 0.25
Bidirectional GRU	Dropout = 0.25
Bidirectional GRU	Dropout = 0.25
Softmax Layer	Target at the 30-character set

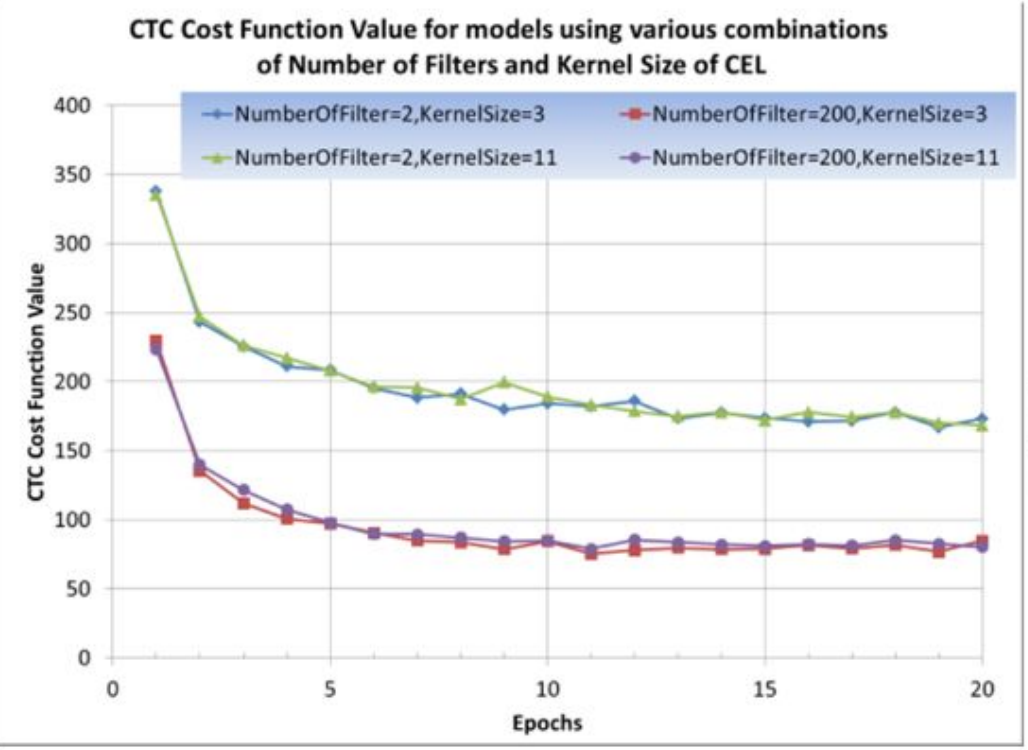
When using the same Number of Filters of CEL, two different Kernel Size curves almost always overlap, which indicates that the Kernel Size of the Convolutional Embedding Layer makes almost no difference in the Training Dynamics of the models.

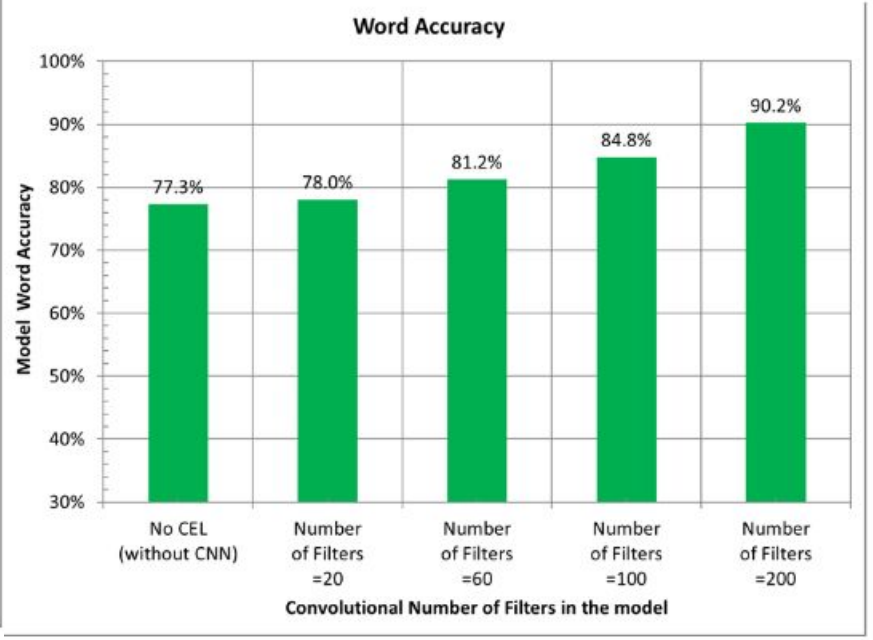
But with the same Kernel Size of CEL, two different Number of Filters curves have a wide gap in CFV, which indicates that **Number of Filters of the Convolutional Embedding Layer makes a large difference** in the speed of the Cost Function Value reduction (i.e. Training Dynamics).

The above observations make sense because the **Number of Filters of the convolutional layer actually is proportional to the size of the Embedding Matrix** of the Sound-2-Vector Embedding

E(SxN) the N should be proportional to the number of Filters of the convolutional layer.

The number of rows N in the matrix is defined as the **Depth of the embedding Matrix** , as this Depth parameter is controlling the size of the **Embedding Matrix E**,which decides how many feature maps can be captured from the Sound Inputs.





Imp asr architecture

https://theaisummer.com/speech-recognition/

This is used to for TTS for diff accent

https://luvvoice.com/

Model–4 testing

original = 'there were more varied and at times especially when beer had circulated freely more uproarious diversions'

predicted = 'there were more varied and at times especially when beer had circulated freely more upproarrious diversiaon'

accent changed female prediction = 'memwa mowre meve and a bing saspesiony wen ma had sopbonag den feeny wmore a poarvios divashouns'

accent changed male prediction = 'bun mord mownd memuvev a bat bh bepeshoa ve wang bewe had dootrematabeed ev maond amoioe bun batamt'

Model — 8 testing

accent changed female prediction = ‘they we mo weadie and at dites yspesiony wen ba hade sopenaid edfet wore abordious ditions’

```
Inference Time: 0.0991 seconds
CPU times: user 2.03 s, sys: 771 ms, total: 2.8 s
Wall time: 176 ms
```

models/asr_model--4.pth – Analysis of Results

- Train Loss (0.1093) & Val Loss (0.5924):
  low overfitting (gap isn't too large).
  Model is generalizing well to validation data.
- CER: 0.1389 (13.89%)
- WER: 0.4819 (48.19%)

Batch = 8 ,Epoch = 17 , 37 sec/epoch , Lr = 0.0001

Increased the Bi-GRU= 4

Epoch 11/150, Train Loss: 0.2035, Val Loss: 0.4866

Epoch 11: CER: 0.1333, WER: 0.4673

And it overfitted at

Epoch 16/150, Train Loss: 0.0819, Val Loss: 0.5477

Epoch 16: CER: 0.1223, WER: 0.4369

Early stopping triggered! At time == **27.18 Min (tripled)**

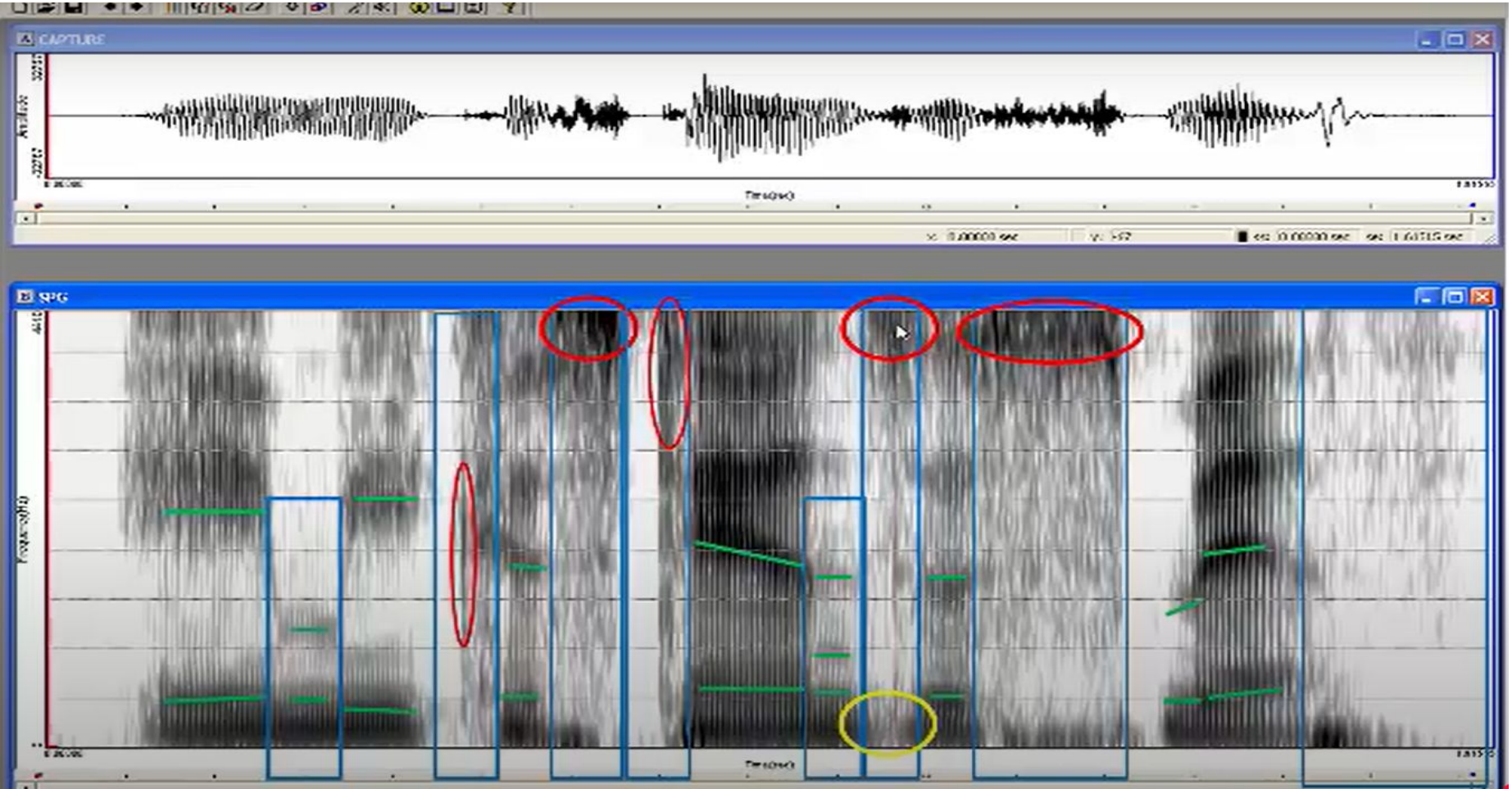
Total params: 16,685,149

How Each Dataset is Used:

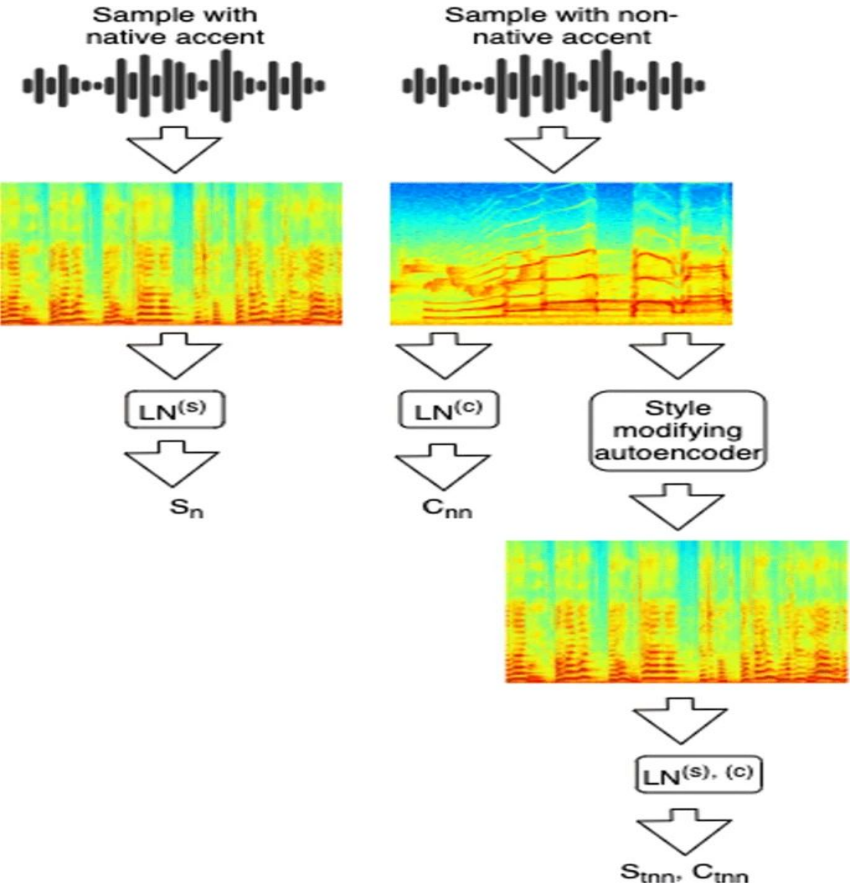
- **Training Data (train_loader):**
  - Training phase.
  - The model learns by computing the loss, backpropagating gradients, and updating weights
  - Training data is for updating model weights.
- **Validation Data (val_loader):**
  - Evaluation phase.
  - The model does not learn from validation data and gradients are not computed, and weights are not updated.
  - It is used to check how well the model generalizes to unseen data.
  - The Early Stopping mechanism monitors the validation loss to stop training if the model starts overfitting

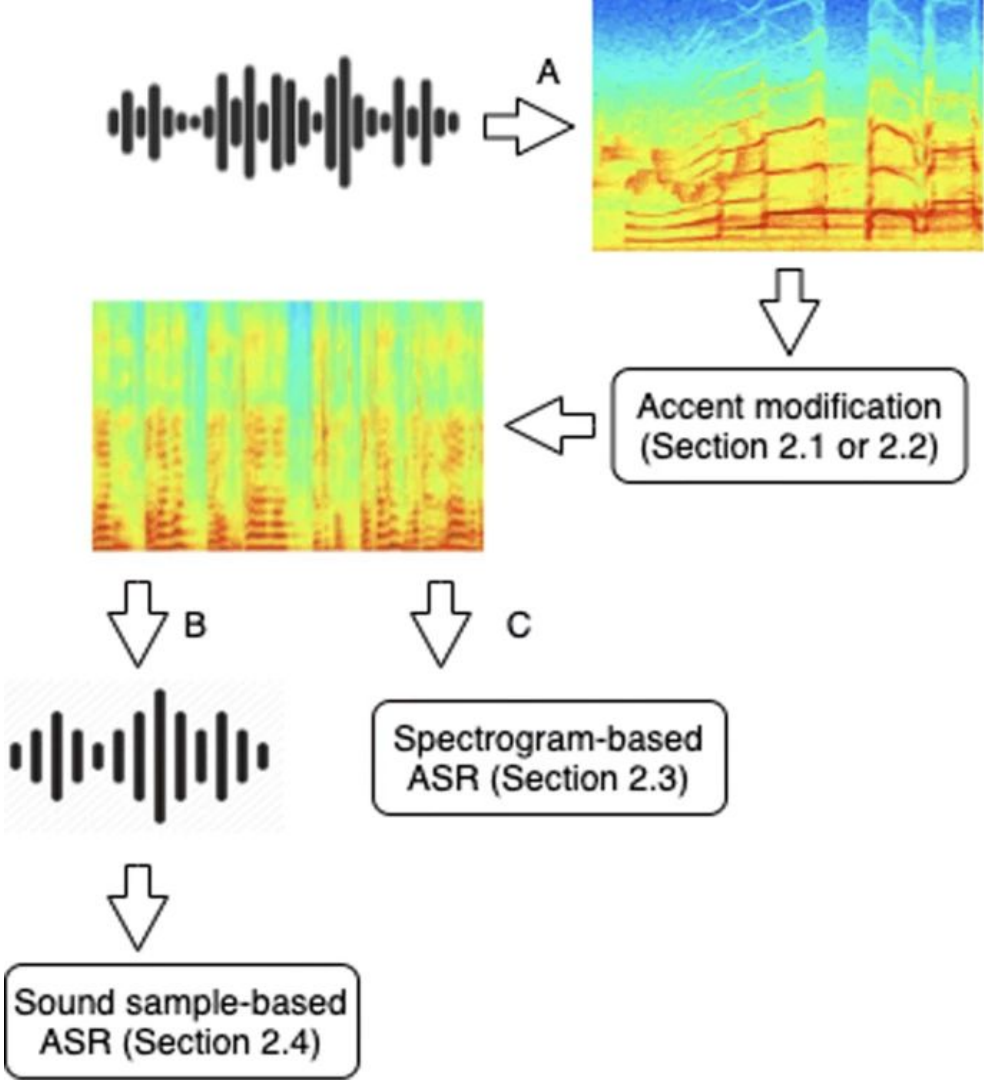
ACCENT HANDLING

Reading Spectrogram : Amy Costanza Smith



https://asmp-eurasipjournals.springeropen.com/articles/10.1186/s13636-021-00199-3





DATA AUGMENTATION

SpecAugment:

Operates on Spectrogram:

SpecAugment applies manipulations like time warping, time masking, and frequency masking directly to the log mel spectrogram.

For large dataset

Raw audio augmentation

Specific Noise Scenarios - If you need to explicitly train your model on a specific type of noise (e.g., car noise) that is likely to be encountered in real-world use cases, adding that noise directly to the audio can be effective.

Limited Data - In situations where you have a very small training dataset, raw audio augmentation can be used to generate additional variations, but be careful not to introduce too much distortion

For small dataset .

For ASR,raw audio file into Mel Spectrogram, we employ SpecAugment and Speed Perturbation , gaussian noise.
not sure whether this can also be applied to MFCCs and whether that produces good results.

Time warping shifts a random point in the spectrogram input with a random distance, while frequency and time masking apply zero masks to some consecutive lines in both the frequency and the time dimensions.

To make the models more robust to temporal variations, the addition of audio data with speed perturbation in the time domain such as has been shown to be effective. They manipulate directly the time series of the frequency vectors which are the features of models, in order to achieve the effect of speed perturbation.

https://sci-hub.se/https://doi.org/10.1109/ICASSP40776.2020.9054130

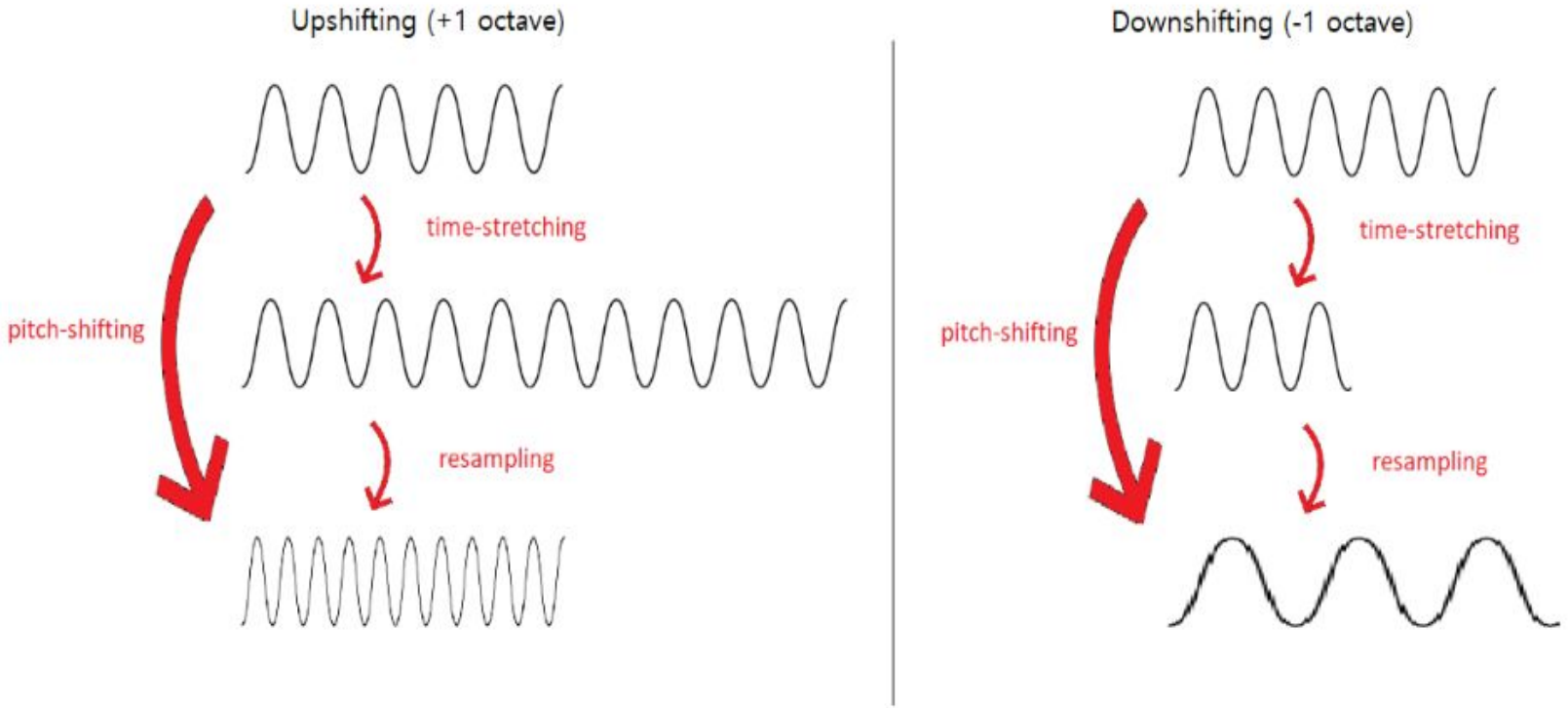
SpecAugment converts ASR from an over-fitting to an under-fitting problem, and performance can be gained by training bigger networks longer.

Label smoothing can make training unstable.

Time warping contributes, but is not a major factor in improving performance.

Pitch Shifting

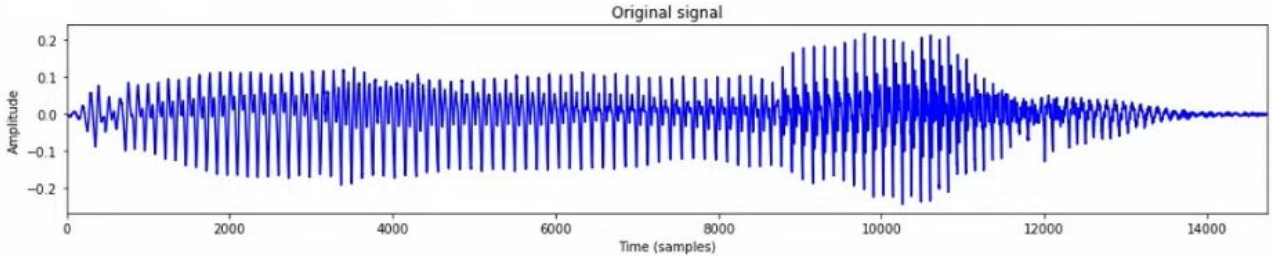
Figure 2.9: Pitch-shifting as a combination of time-stretching and resampling



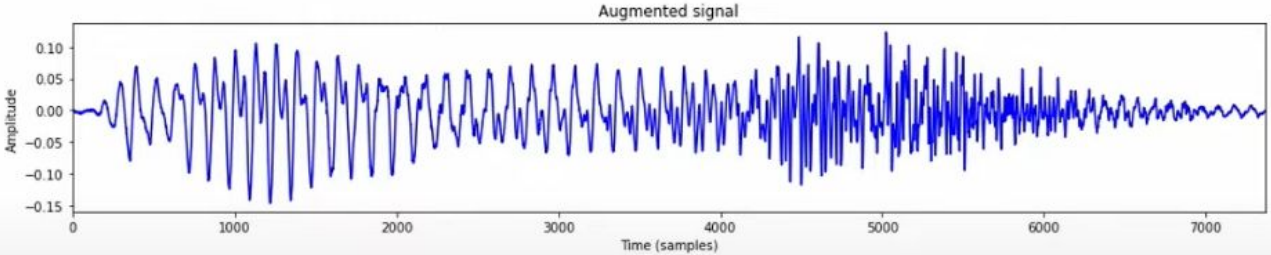
https://www.youtube.com/watch?v=PjKlMXhxtTM

Time stretching

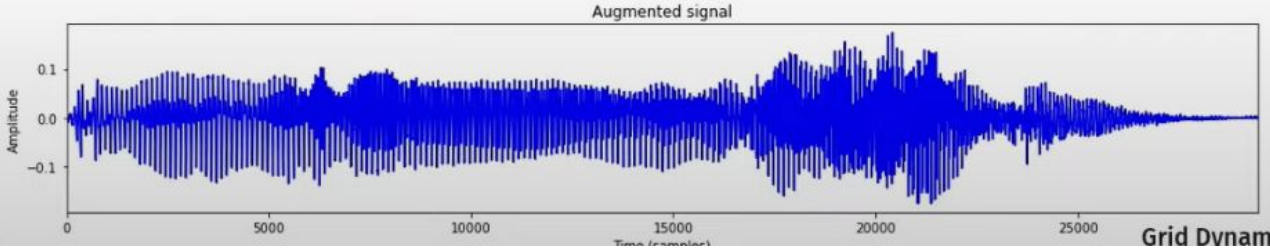
Original



Speed up



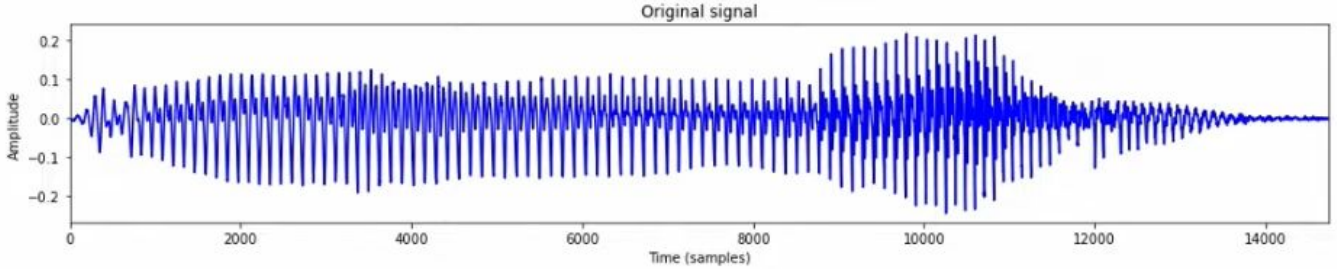
Slow down



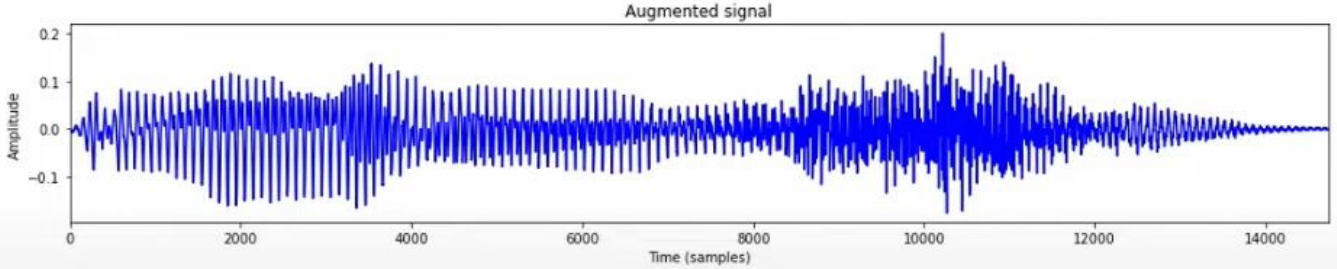
https://youtu.be/BSvy9qB-NrE?feature=shared

Pitch Scaling

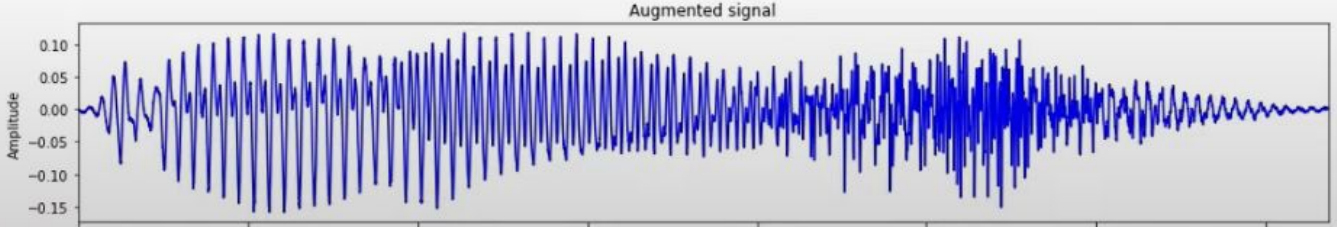
Original



Pitch up



Pitch down



Augmentation	Purpose	Effect on ASR
Speed Perturbation	Simulates different speaking speeds	Improves speaker variability & pronunciation changes
Pitch Shifting	Mimics different speaker vocal ranges (male, female, child)	Enhances accent & speaker generalization
Time Stretching	Slows down or speeds up speech without changing pitch	Helps ASR handle fast & slow speech
Background Noise Addition	Simulates real-world noisy environments	Improves robustness to noise (cafes, traffic, offices)
Reverberation (Echo)	Mimics speech in different room environments	Improves ASR for different acoustic conditions
Volume Perturbation	Adjusts loudness of speech randomly	Helps ASR generalize to different microphone volumes
Band-Pass Filtering	Simulates telephone or low-quality audio	Improves ASR performance on low-bandwidth signals
Microphone Simulation	Emulates different microphone types & distances	Reduces ASR sensitivity to mic variations

Augmentation	Purpose	Effect on ASR
SpecAugment (Best for ASR)	Randomly masks time and frequency regions	Prevents overfitting & forces model to learn robust features
Time Masking	Hides random time segments in spectrogram	Helps ASR generalize to speech gaps & interruptions
Frequency Masking	Hides random frequency ranges	Improves robustness to microphone & channel variations
Time Warping	Distorts time axis to simulate variations in speech speed	Makes ASR robust to different speaking speeds
Gaussian Noise on Spectrogram	Adds small random noise	Prevents overfitting & improves generalization
Mixup & CutMix	Blends two different spectrograms	Helps model learn more diverse speech patterns

Result after augmentation model 3

```
Inference Time: 0.1697 seconds
CPU times: user 3.26 s, sys: 1.36 s, total: 4.62 s
Wall time: 1.27 s
```

Epoch 19/150, Train Loss: 0.2107, Val Loss: 0.2423
Epoch 19: CER: 0.0646, WER: 0.2514
Early stopping triggered!

Aug/Model–3 testing

original = 'there were more varied and at times especially when beer had circulated freely more uproarious diversions'

predicted = 'there were more varied and at times especially when beere had circulated freely more up oarios diversians'

accent changed female prediction = 'merwa mo mevied and atbines saspesciony wen ba had so poen adee feety ore apbrorious dagitions'

accent changed male prediction = 'meren ware more rebeed i bat buns esprihra be ranbeari ard tont premictred thre be morte amordious bunbahancs'

Testing for similar accent different words

Original = ‘We can measure radiation energy, and waves can perform and move a charged particle in their path. It is unnecessary for quantum discussions to examine detailed quantitative electromagnetic relationships, which are defined by Maxwell's equations.’

Model = ‘we can masur ragiatio anergy and waes can performe and move a churged partical an her pot it is unnecessary for cantom desfussions to axamoned detailed contitaid of eluctorming nat occrelatiounships whch aredefined by mactwals equasions’

Augmented model = ‘we canmasure radin ation anergy and waves canperform and move a charged partical an their pagh it is unnecessary for quantom discassions to examine detaled cpontitate of e lectro i nat icgralationships which ar difined by macx wels ecuations’

https://elevenlabs.io/app/speech-synthesis/text-to-speech

Testing for same accent different words

Augmented model = we can measare radeitionenergy and was conterform ad move a charged particale in their path it is unnecessary for quantom discussions to exemine detailed quontitative elect rom magneti relationships o whichae difined by macscwls equasioncs

Model = we han measuratiation anderg and wives conperform and woe the charged pardical in their pa it is anecessary for quantom discussions to exmine detild quontitued if elect from agneati ruelationships o whichare difined by maxwals aquasions

PROBLEMS

If the model predicts too many blanks, words may split incorrectly.

If the model misses blanks, words may merge incorrectly.

SOLUTION:

- LM ngrams with beam search deocder
- LESS PEAKY CTC FORCED ALIGNMENT BY LABEL PRIORS
- CONV1D
- Wordpiece model
- Use FA tools such as gentle , MFA,VAD

LANGUAGE MODELING

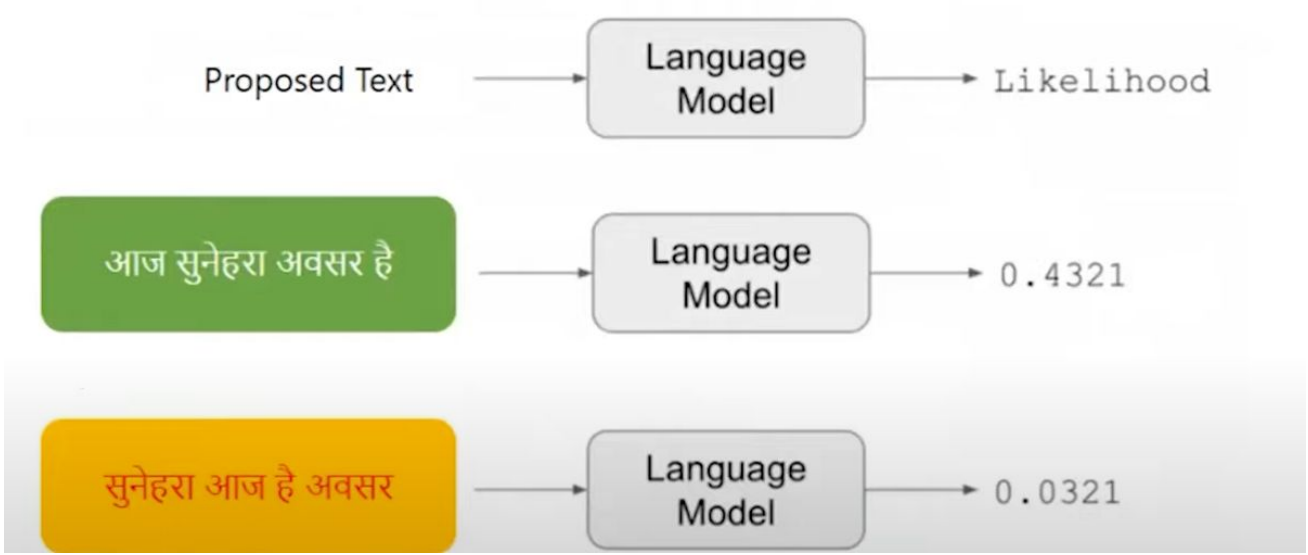
Language Model

It’s a statistical model that is designed to analyze the pattern of human language and predict the likelihood of a sequence of words or tokens

1. **Statistical Language Models:** unigram(ngram or markov model) , bidirectional , exponential, continuous space
2. **Neural Language Models**

https://insights.daffodilsw.com/blog/what-are-language-models-in-nlp

What the LM computes?



https://pytorch.org/audio/main/tutorials/asr_inference_with_ctc_decoder_tutorial.html

Step1: creating ARPA (probs of n-g

Step2: Filtering from lexicon (filter A
using vocab of top k)

Step3: quantize , binarization

- Probability (Likelihood) of a sentence:

$$P(w_0 \ldots w_i) = P(w_i | w_{i-1} \ldots w_0) P(w_{i-1} | w_{i-2} \ldots w_0) \ldots$$

- We generally limit it to N-1 previous words!

$$P(w_i | w_{i-1} \ldots w_0) = P(w_i | w_{i-1} \ldots w_{i-n+1})$$

- *Example: Bigram Language Model*
  - text = "<start> आज सुनेहरा अवसर है <end>"
  - P(text) = P(<start> | आज) * P(आज | सुनेहरा) * P(सुनेहरा | अवसर)
    * P(अवसर | है) * P(है | <end>)

RE VIDEOS

Steps to make ARPA file

https://medium.com/@techhara/n-gram-language-model-arpa-d85c1e5fdcdc

https://youtu.be/nEE7UeMF5_Q

Feature	n-Gram LM	Neural LM (RNN/Transformer-based)
Context Window	Fixed (e.g., 3-gram, 4-gram)	Long-range dependencies (can model entire sentences)
Generalization	Poor for unseen words/sequences	Better generalization using embeddings
Sparsity	Suffers from data sparsity	Learns continuous representations (dense vectors)
OOV Handling	Struggles with unseen words	Handles unseen words better (subword models)
Computation Cost	Fast (efficient WFST-based search)	Slower (requires deep model inference)
Decoding Speed	Faster	Slower, especially for Transformers
Rescoring Quality	Works well for structured text (e.g., domain-specific corpora)	Better fluency and grammatical correctness
Memory Usage	Large storage for n-gram tables	Smaller memory footprint with compact neural representations
Adaptability	Requires explicit retraining for new words	Can adapt better with fine-tuning

TINY PRETRAINED TRANSFORMER

Text based Transformer: Pretrained models — transformers 2.4.0 documentation

Albert-base-v2 **11M parameters,**12 repeating layers, 128 embedding, 768-hidden, 12-heads

T5-small **60M parameters** with 6-layers, 512-hidden-state, 2048 feed-forward hidden-state, 8-heads

Distilbert-base-uncased 6-layer, 768-hidden, 12-heads, **66M parameters**

Distilgpt2 6-layer, 768-hidden, 12-heads, **82M parameters**

Bert-base-cased 12-layer, 768-hidden, 12-heads, **110M parameters.**

Distilroberta-base 6-layer, 768-hidden, 12-heads, **82M parameters**

Tiny-BERT 4 layers, 312 hidden, **14M parameters**

https://github.com/abacaj/awesome-transformers

how-to-calculate-number-of-parameters-in-bert

Set of 24 BERT models

	H=128	H=256	H=512	H=768
L=2	2/128 (BERT-Tiny)	2/256	2/512	2/768
L=4	4/128	4/256 (BERT-Mini)	4/512 (BERT-Small)	4/768
L=6	6/128	6/256	6/512	6/768
L=8	8/128	8/256	8/512 (BERT-Medium)	8/768
L=10	10/128	10/256	10/512	10/768
L=12	12/128	12/256	12/512	12/768 (BERT-Base)

BERT

But we need ASR specific text error data

Pretrained LLM

Auto Encoding

- 10 to 50M params
- Masked lang modeling
- Process all at once in parallel manner
- AlBERT,distillBERT

Auto regressive

- More than 80M
- Causal lang model
- Predict one by one consider full previous context
- GPT2,T%

Train + Validation data tested on ASR model :

- **CER** : 0.027
- **WER** : 0.057

Total unique words in Train+Val: 13614

Total unique words in Test: 5840

Words in Test that are NOT in Train+Val: **894 —---------- 15% is OOV words**

So , we can train the text data sequence for the language model (ngram or neural LM) but we can’t assure that same sequence will repeat in test data or real world data.

NEW DATA

speechiitm@ee.iitm.ac.in

https://sites.google.com/view/englishasrchallenge/home

The above link to iitm data of 250 hr

NPTEL - edu domain data 15000 hr:

https://github.com/AI4Bharat/NPTEL2020-Indian-English-Speech-Dataset?tab=readme-ov-file