A1003:
음성인식 및 기계학습

초지능창의연구소, 복합지능연구실

박기영

강사 소개

- 초지능창의연구소, 지능정보연구본부, 복합지능연구실
- 1997~2003: 계산및신경망시스템연구실 석,박사
- 2003~2005: 삼성종합기술원 HCILab, Interaction Lab.
- 2005~: ETRI
  - 2007~: 음성처리연구실, 음성지능연구실, 복합지능연구실
  - 2007~2009: 신성장동력산업용 대용량대화형 분산 내장처리 음성인터페이스 기술개발
  - 2010~2014: 모바일 플랫폼 기반 대화모델 적용 자연어 음성인터페이스 기술개발
  - 2015~2018: 언어학습을 위한 자유발화형 음성대화처리 원천기술 개발
  - 2019~2021: 다중 화자간 대화 음성인식 기술 개발
  - 2019~2028: 준지도학습형 언어지능 원천기술 및 이에 기반한 외국인 지원용 한국어 튜터링 서비스 개발
  - 2022~2026: 다화자 동시처리를 위한 인공지능 기반 대화 모델링 기술 개발

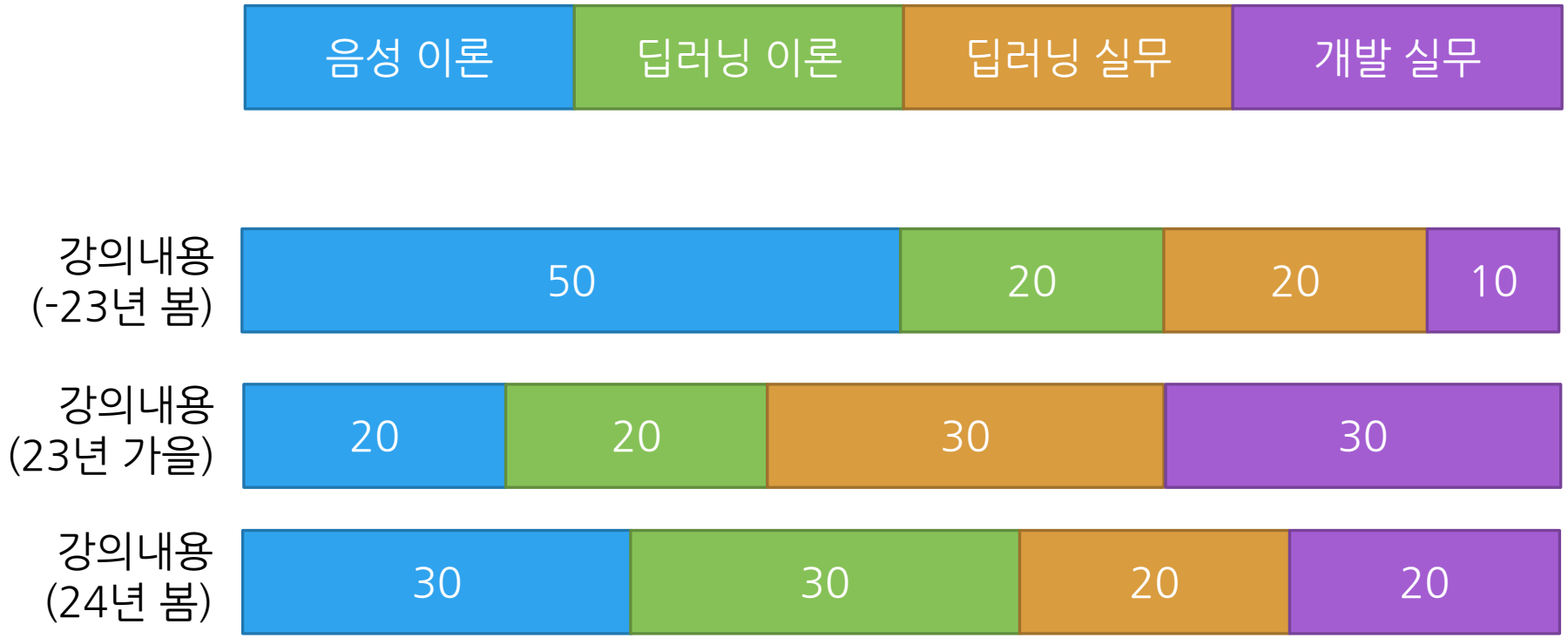
강의내용: Overview

- 음성인식 이론
- 음성인식 실습
- 딥러닝 이론/실습 (Transformer)
- 딥러닝/리눅스 개발환경

강의내용: Overview

- 음성 이론
  - Sampling, FFT, Mel, HMM, GMM, n-gram, AM, LM, wFST, decoder, …
- 딥러닝 이론
  - RNN, LSTM, Transformer, …
- 딥러닝 실무
  - recipe, learning rate, batchsize, …
- 개발 실무
  - terminal, shell,
  - vi, git, tmux, (ana)conda, jupyter, docker, …

강의 목표


음성 이론
딥러닝 이론
딥러닝 실무
개발 실무
강의내용
(-23년 봄)
50
20
20
10
강의내용
(23년 가을)
20
20
30
30
강의내용
(24년 봄)
30
30
20
20

강의일정(-22 가을)

1일차	2일차	3일차
• 음성인식 개요 • (고전적) 음성인식 이론	• 딥러닝기반 (고전적) 음성인식 이론 • 종단형음성인식 개요	• 훈련과정 분석 • Tensorboard
• 실습환경소개 • 인식률 측정하기	• ESPNet 소개 • 종단형 음성인식 recipe 살펴보기	• 음성인식 평가 실습 • 성능 측정
• (고전적) 음성인식 이론 • 특징추출	• 트랜스포머 소개	• 성능개선 방안 • 연구동향
• 훈련DB 소개 • 특징추출 실습	• 음성인식 훈련 실습	• 실습 마무리

Homework

강의일정(23 봄-)

1일차	2일차
• 음성인식 개요 • (고전적) 음성인식 이론	• 종단형음성인식 • Transformer
• 환경설정 • 인식률 측정하기	• ESPnet recipe
• 신호처리 • 특징추출	• 성능개선방안 • 연구동향
• 훈련DB 소개 • ESPnet 설치 및 훈련	• Whisper, 실습 마무리

강의일정(24 봄)

1일차	2일차
• 음성인식 개요 • (고전적) 음성인식 이론	• 종단형음성인식 • Transformer/Conformer/e-Branchformer
• 환경설정 • 인식률 측정하기	• 추론 및 스코어링
• 신호처리 • 특징추출	• 성능개선방안 • 연구동향
• ESPnet 설치 및 코드리뷰	• Whisper, 실습 마무리

강의내용: 1일차

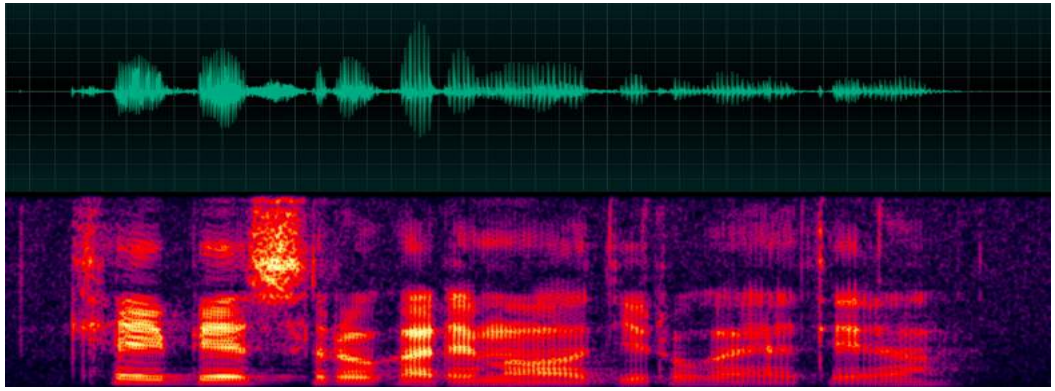
이론	실습
• 음성인식 개요 • (고전적) 음성인식 이론 • 음성인식 성능 평가 방법	• 실습환경구성: Conda, Jupyter • WER 계산
• 음성신호처리 • 특징추출	• Audio 들어보기, • Log Mel Filterbank 추출 • Spectral Augmentation
• 음성인식 툴킷:ESPnet	• ESPnet 코드리뷰 • Data 구조

강의내용: 2일차

이론	실습
• 종단형 음성인식 • Transformer/Conformer/e-Branchformer	• 모델 다운로드 • 추론 및 스코어링
• 성능개선방안	
• 최근연구동향	• Whisper

What is speech recognition

- ASR(Automatic Speech Recognition), STT: Speech-to-text



  - Isolated, Connected, Continuous, Keyword Spotting
  - Speaker Dependent/Independent
- Difference with Image/Video Classification
  - Sequence Generation Problem

History of ASR

1950,60s

- Phonetic Recognizer
- 10 digit recognition
- DTW
- Idea of Continous ASR(CMU)

1970s

- IBM, Bell Lab, …
- DARPA program
  - CMU Harpy:1,011 words vocab., FSN

1980s

- Connected words recognition (Fluently spoken)
- Template based → Statistical Methods
- HMM
- N-gram, Neural Nets.
- DARPA program
  - CMU SPHINX
  - BBN, SRI

1990s

- MCE, MMI
- DARPA programs
  - Natural Lanuage Recognition, ATIS, Broadcast news, Switchboard
- Robust ASR
- Applications

2000s

- Spontaneous speech
- Robust ASR
- Multimodal

50 Years of Progress in Speech and Speaker Recognition Research, ECTI Transactions On Computer And Information Technology, 2005

Applications

1956,
RCA Labs

1975,
1997,
Nuance

DRAGON
NATURALLYSPEAKING

2012,
Google
Voice
Search

2011,
Apple
Siri

2014,
Amazon

2018,
Google
Duplex

Google AI
Google Duplex

Hi, how can I help?
Google Assistant

1997,
삼성
애니콜

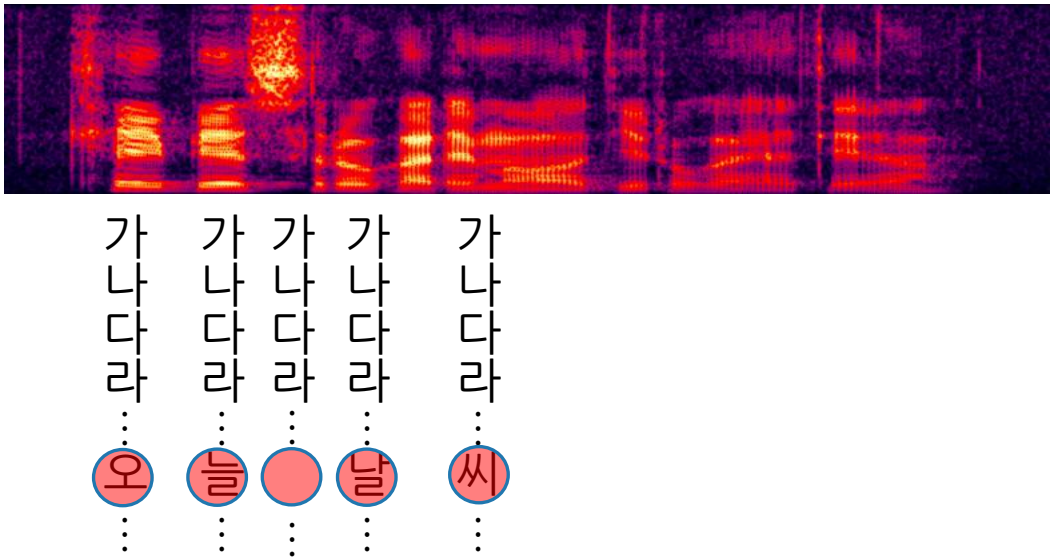
2008,
파인디지털

2012,
다음

2012~
ETRI

How It works

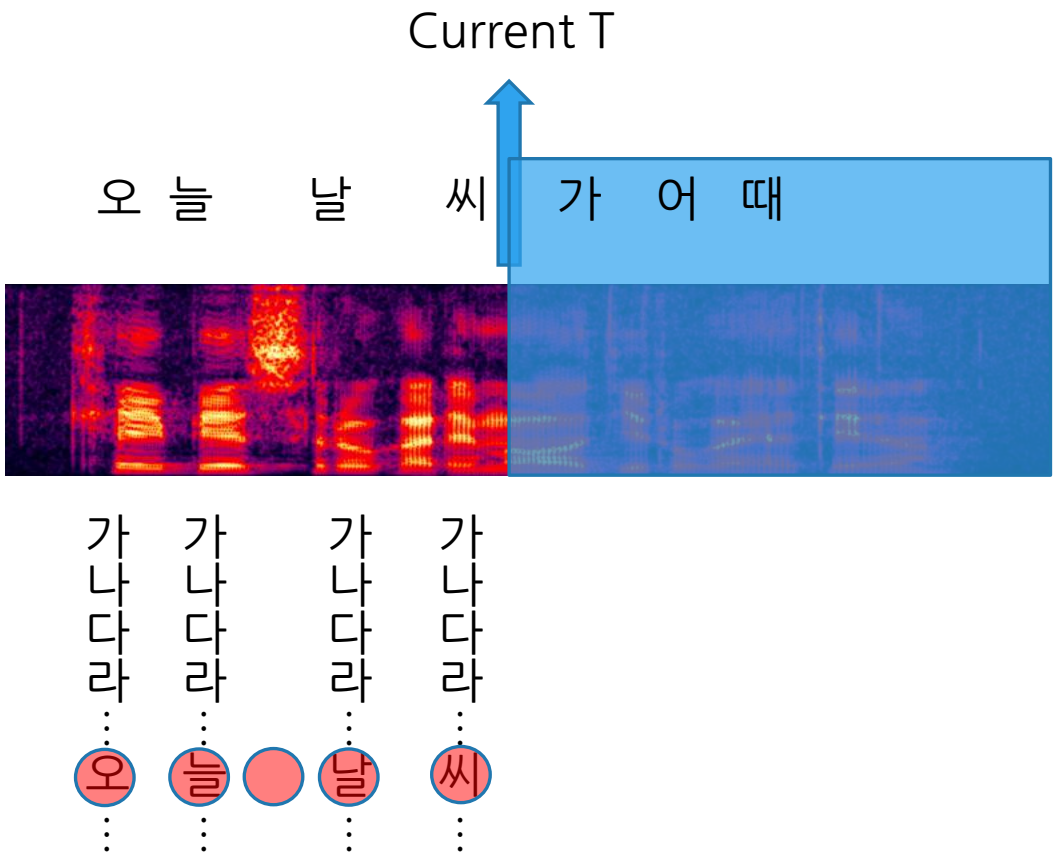
- W* = argmax P(W|X)
  - To Find Most Probable Word Sequence Given Input Signal/Feature



- Considerations
  - Boundary? Segmentation?
  - Output Units? Words, Characters, Phoneme, …
  - Classification Accuarcy? Unit Accuracy vs. Sentence Accuracy

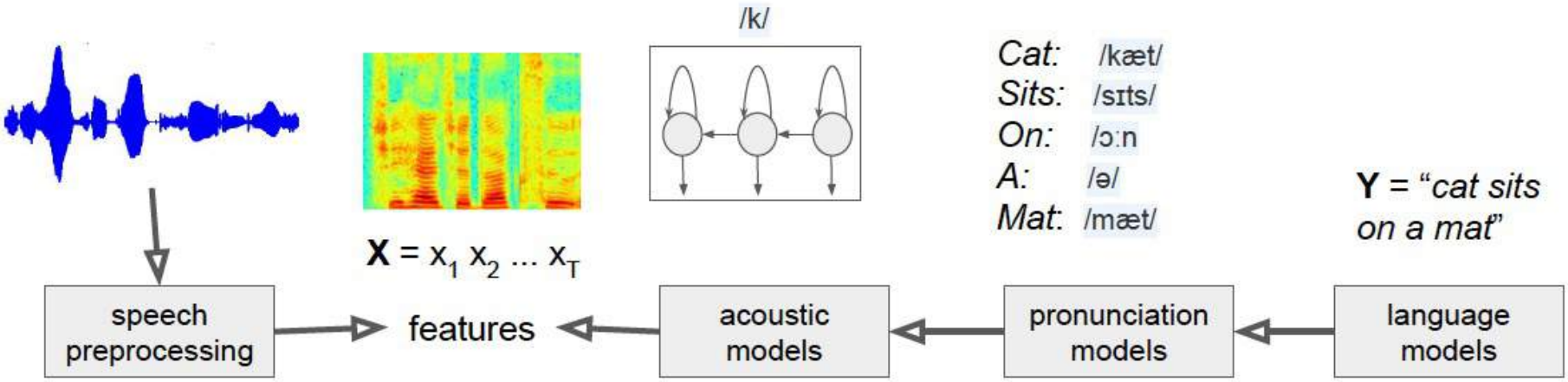
Context/Latency

- Batch or Streaming?



How It really works

- W* = argmax log P(W|X)
- = argmax log P(X|Q)P(Q|W)P(W)
- To Find Most Probable Sequence Among Plausible Words Sequences



https://heartbeat.fritz.ai/the-3-deep-learning-frameworks-for-end-to-end-speech-recognition-that-power-your-devices-37b891ddc380

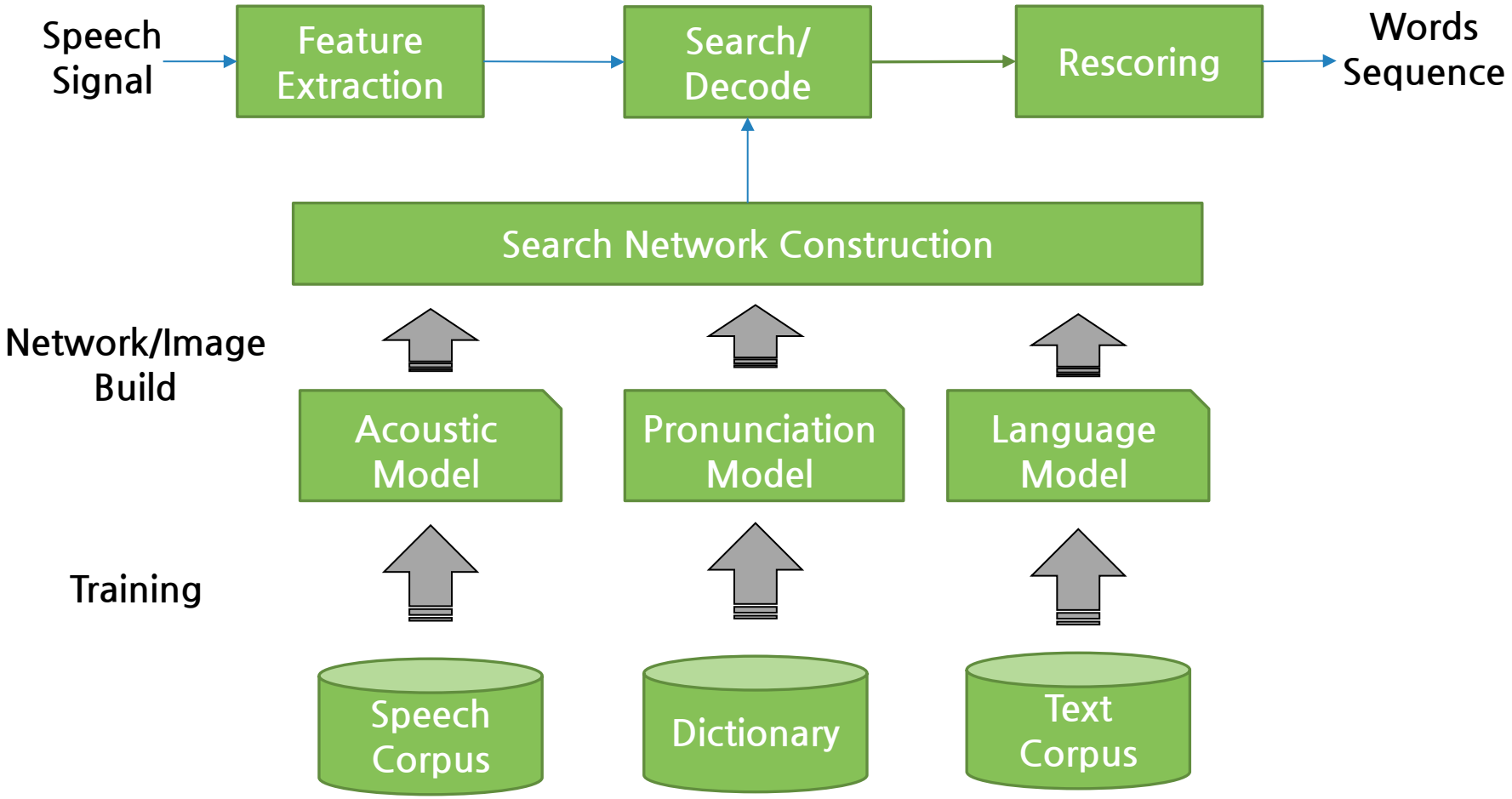
Guess who?

- Find A Criminal Among Suspects Given Evidence
- Criminal = argmax P(Suspect|Evidence)
- Criminal = argmax P(Evidence|Suspect)

  = argmax

  P(Evidence|Behavior)P(Behavior|Suspect)P(Suspect)

Structure of traditional asr



Evaluation Metric

- Types of Error
  - Substitution
  - Deletion
  - Insertion
- Error Rate (cab be > 1)
  - (S + D + I)/N
- Accuracy (can be < 0)
  - 1-(Error Rate)
- WER/CER/SER:
  - Word/Character/Sentence Error Rate

```
REF     : how is the weather today
REC/HYP: how was the better  to day

In Words: WER = 100%, Acc=0%
 - N= 5: how, is, the, weather, today
 - S = 2
 - D = 1
 - I = 2

how is  the weather today
how was the better  to day

In Chars: CER = 25%, Acc=75%
 - N= 20:
h,o,w,i,s,t,h,e,w,e,a,t,h,e,r,t,o,d,a,y
 - S = 3
 - D = 1
 - I = 1

how is  the weather today
how was the better  to day

In Sentence: SER = 100%, Acc=0%
 - N = 1
 - S = 1
```

Quiz

- REF: 오늘 서울의 날씨가 어때
- REC: 음 오눌의 날씨 가 어때
- WER = ?

한국어

- REF: 오늘 서울의 날씨가 어때
- REC: 음 오눌의 날씨 가 어때
- WER=4/4 = 1.0 Acc=0.0
  - N=4, 오늘, 날씨가, 어때요
  - S = 2, D = 1, I = 2, WER=5/4
  - S = 3, I = 1, WER=4/4
- CER= 4/10 = 0.4, Acc=0.6
  - N = 10
  - S = 1
  - D = 2
  - I = 1

```
   오늘   서울의 날씨가  어때
음 오눌의        날씨 가 어때
```

```
오늘 서울의 날씨가    어때
음   오눌의 날씨   가 어때
```

```
    오 늘 서 울 의 날 씨 가 어 때
음  오 눌       의 날 씨 가 어 때
```

측정방법

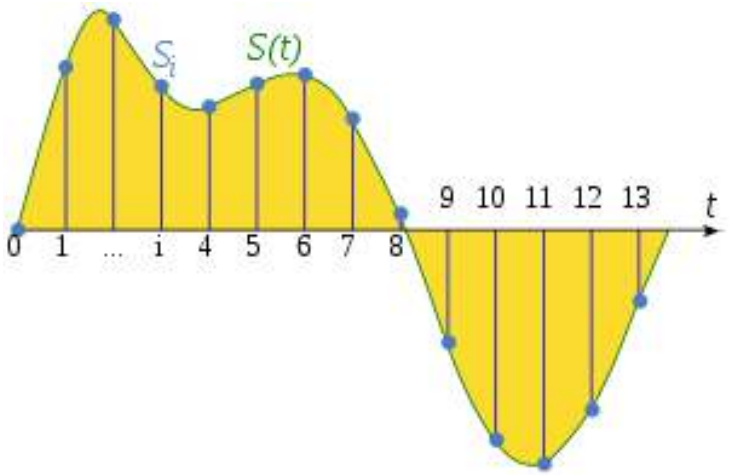
- Edit distance 측정
  - https://en.wikipedia.org/wiki/Edit_distance
- 사용도구
  - HResults (HTK)
  - compute-wer (kaldi)
  - sclite (NIST, ESPnet)
- 예)
  - compute-wer ark:ref.txt ark:rec.txt

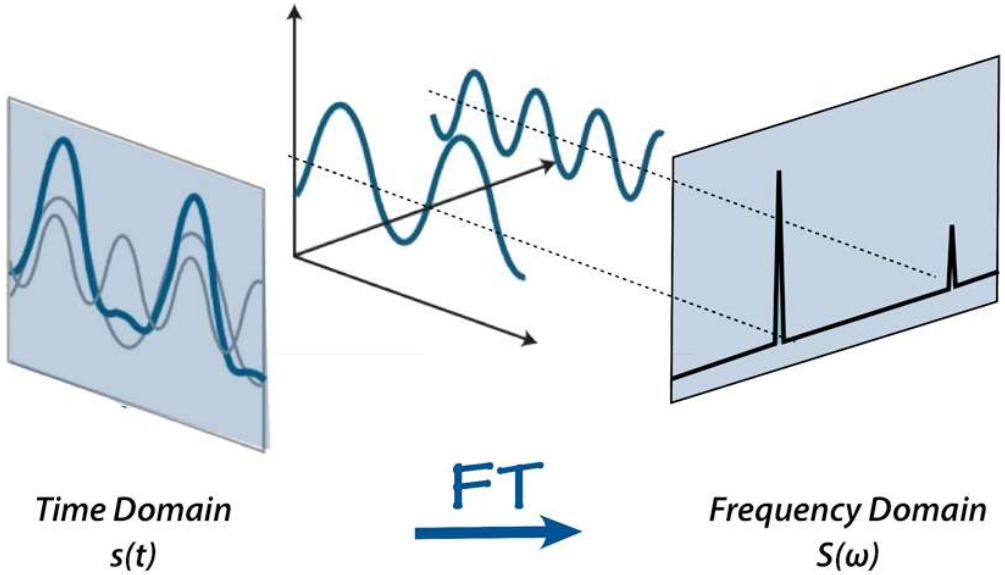
실습

- Conda 설치
- Jupyter notebook

Feature Extraction

Sampling and Spectrum

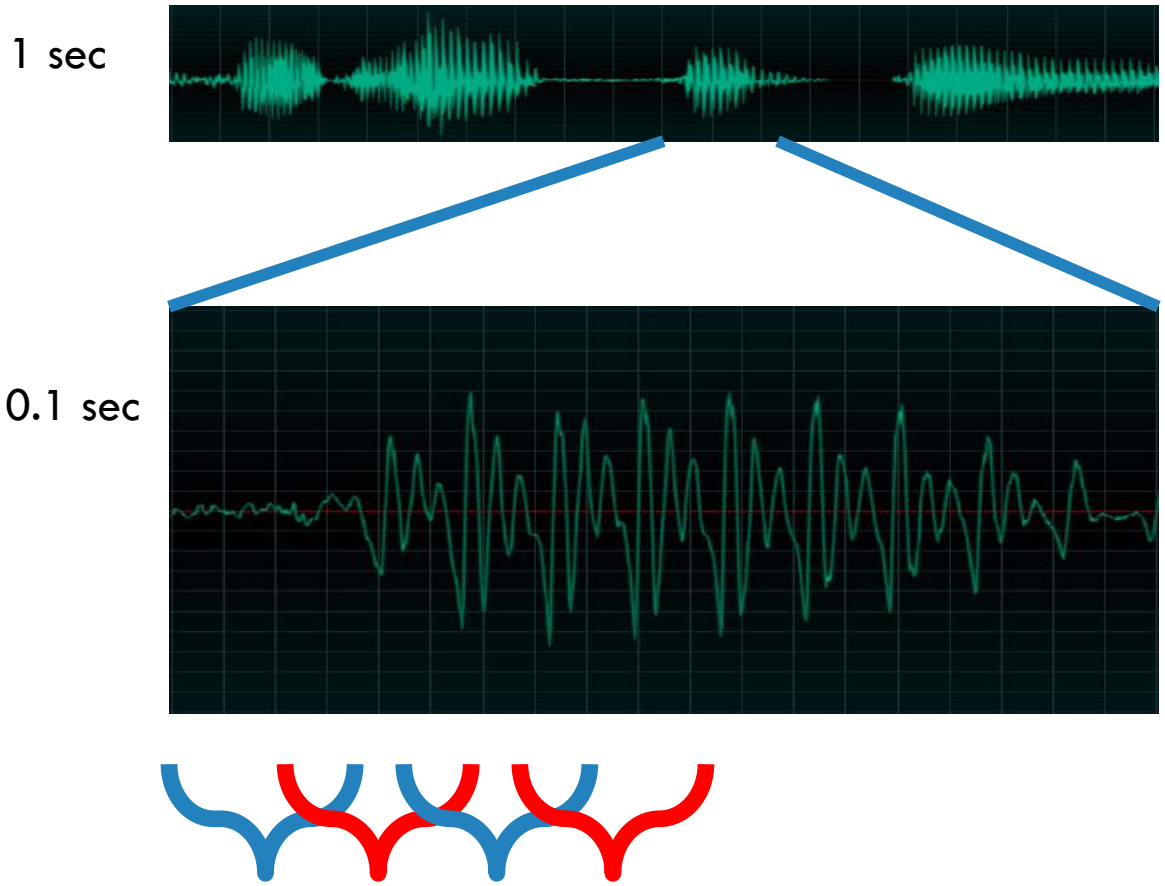




8kHz: Narrowband, 전화망
16kHz: Wideband
44.1/48kHz: High quality audio

https://en.wikipedia.org/wiki/Sampling_(signal_processing)
https://towardsdatascience.com/understanding-audio-data-fourier-transform-fft-spectrogram-and-speech-recognition-a4072d228520

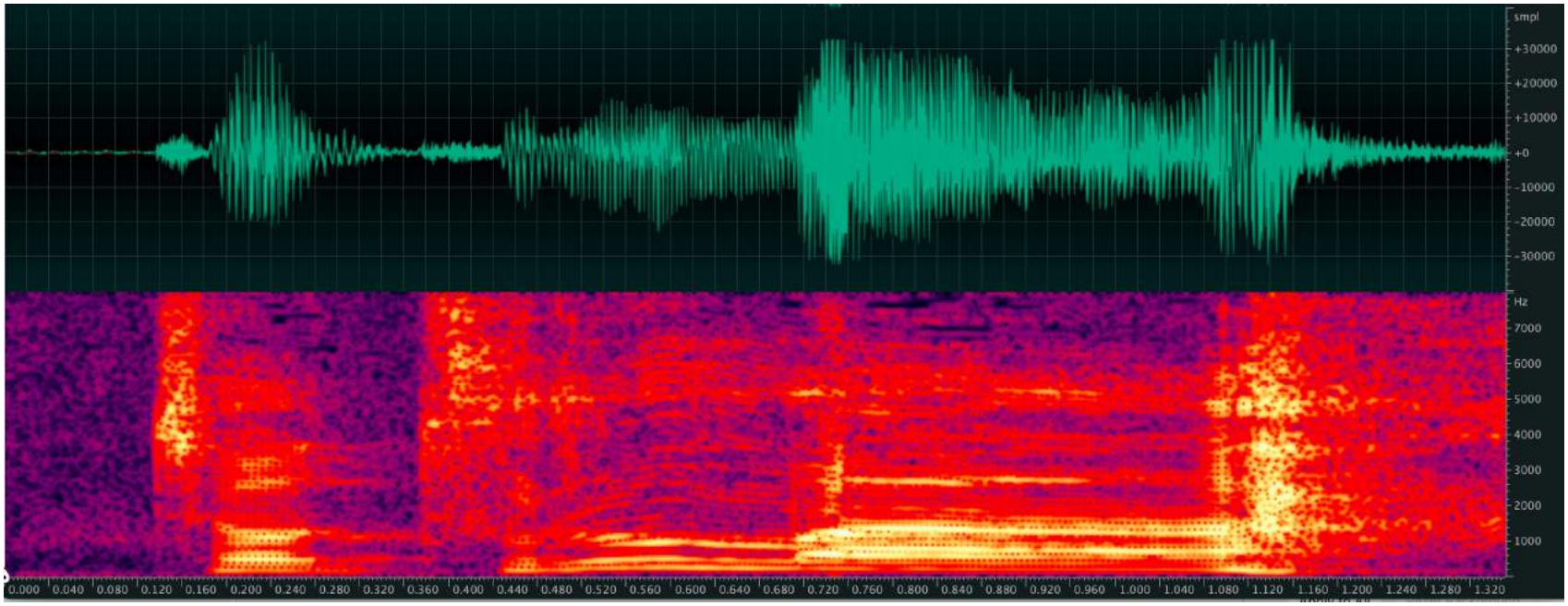
Frame-Wise Processing



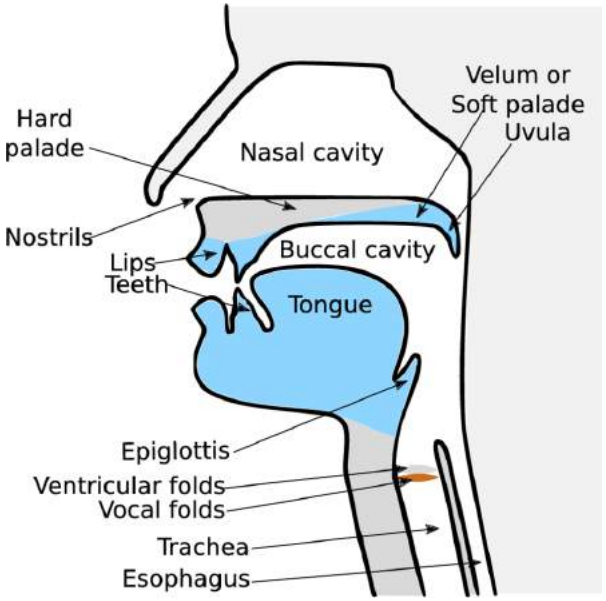
Windowing,
Window length,
FFT size,
Hop size

Spectrogram

- Series Of Spectrum
- Matlab, Python, Adobe Audition, Audacity, …
- Frame Shift, Overlap, Window Length, Windowing, FFT points

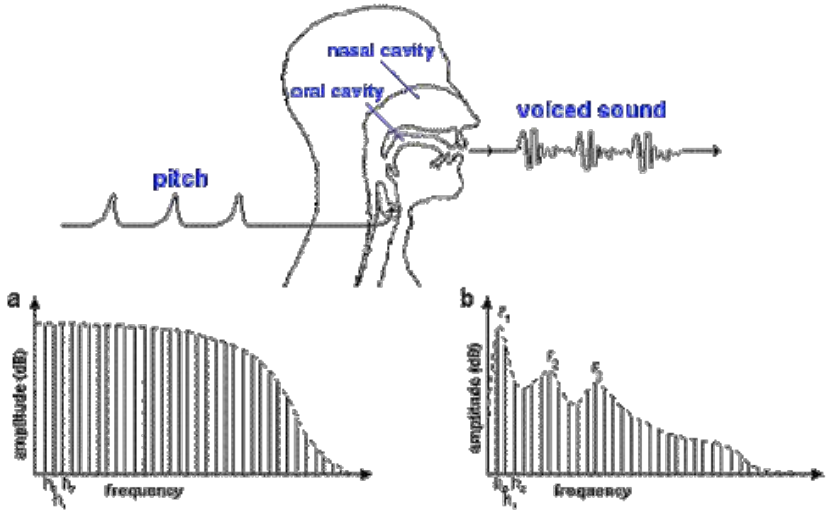


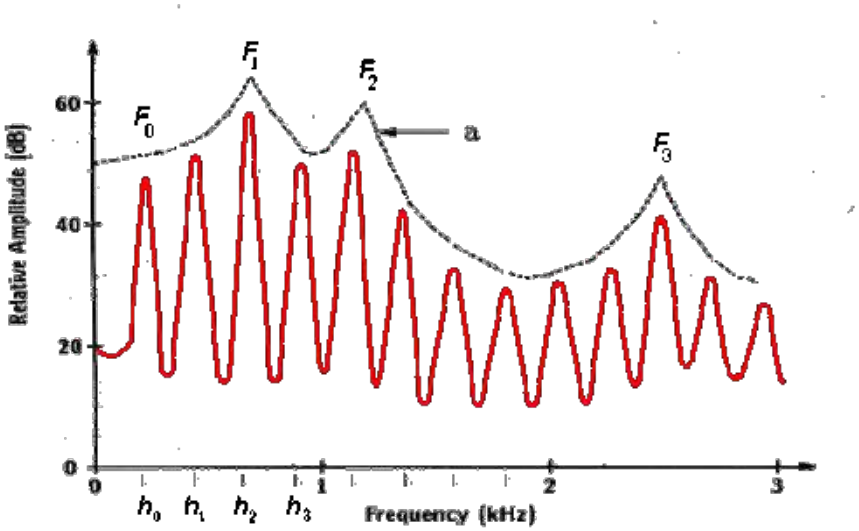
Voice Production



https://www.researchgate.net/publication/318814563_Analyzing_of_the_vocal_fold_dynamics_using_laryngeal_videos
https://www.youtube.com/watch?v=kfkFTw3sBXQ

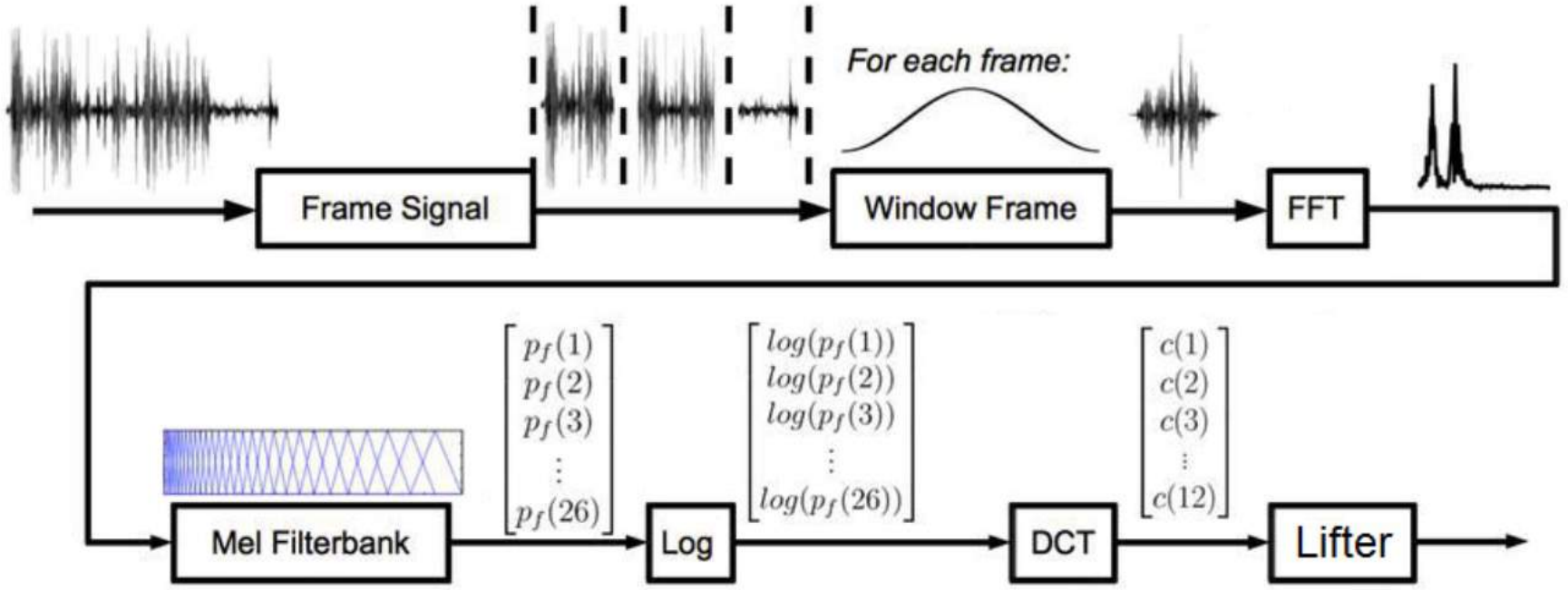
Pitch and Formant





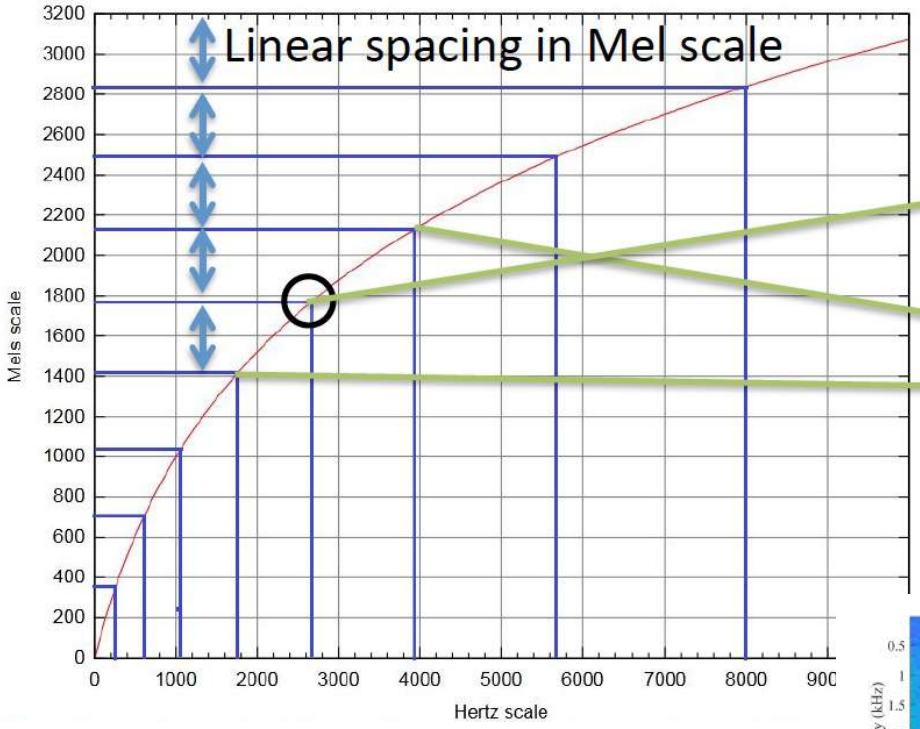
http://147.162.36.50/cochlea/cochleapages/theory/sndproc/sndcomm.htm

Feature extraction: MFCC

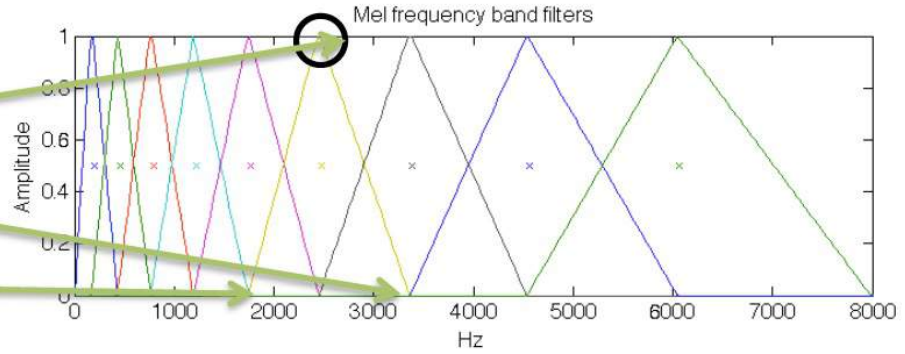


https://hyunlee103.tistory.com/46

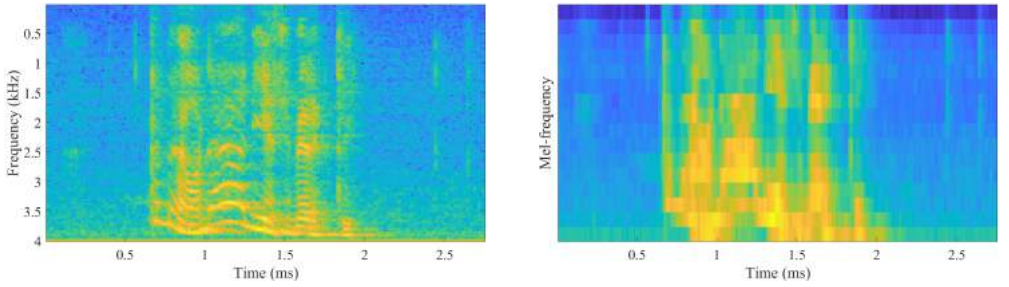
Mel Filterbank



# of filters: 23 → 40 → 80+3



https://hyunlee103.tistory.com/46



https://wiki.aalto.fi/display/ITSP/Cepstrum+and+MFCC

CMVN

- Cepstral Mean Variance Normalization
  - Zero-mean Unit Variance
- CMS: Cepstral Mean Substraction
  - Per Utterance
  - 채널/화자 효과를 제거하고 발성의 특성만 남김
- For Deep Learning
  - Global CMVN
  - For better convergence

Homework

- 음성인식 평가용 테스트데이터 수집
- 본인 (또는 근처 아무나) 목소리를 녹음
  - 16kHz, wav (uncompressed), mono
  - (일단 녹음하고 확인해봅시다)
- 인식률이 되도록 좋게 or 나쁘게 나오도록
  - But don’t be too evil… noise, yell, whisper…
- 10문장 정도, 1문장당 10초 정도.
- wav/text pair (wav.scp, text)

Classical ASR

HMM-based ASR

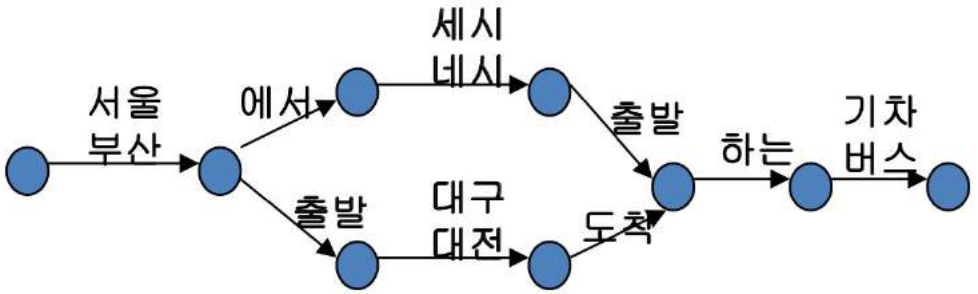
- How we call it?
  - Conventional
  - Traditional/Classical
  - Ancient
- ~2010: HMM의 시대
- Why?
  - 내부 동작을 이해하고 문제점 또는 성능 개선 방법을 찾기 위해서

HMM

- Problem to Solve:
- W* = argmax log P(W|X)
- = argmax log P(X|Q)P(Q|W)P(W)
- P(W): Language Model, P(Wt|Wt-1, Wt-2, …)
- P(Q|Wt): Prononciation Model
- P(X|Q): Acoustic Model

Language Model

- 단어간의 연결 가능성을 이용하여 search space를 제한



- Deterministic Grammar
  - FSN (Finite State Network)
  - JSGF (Java Speech Grammar

```
$time = 세시|네시;
$city = 서울|부산|대구|대전;
$trans = 기차|버스;
sent-start $city (에서 $time 출발 |
출발 $city 도착) 하는 $trans sent-end
```

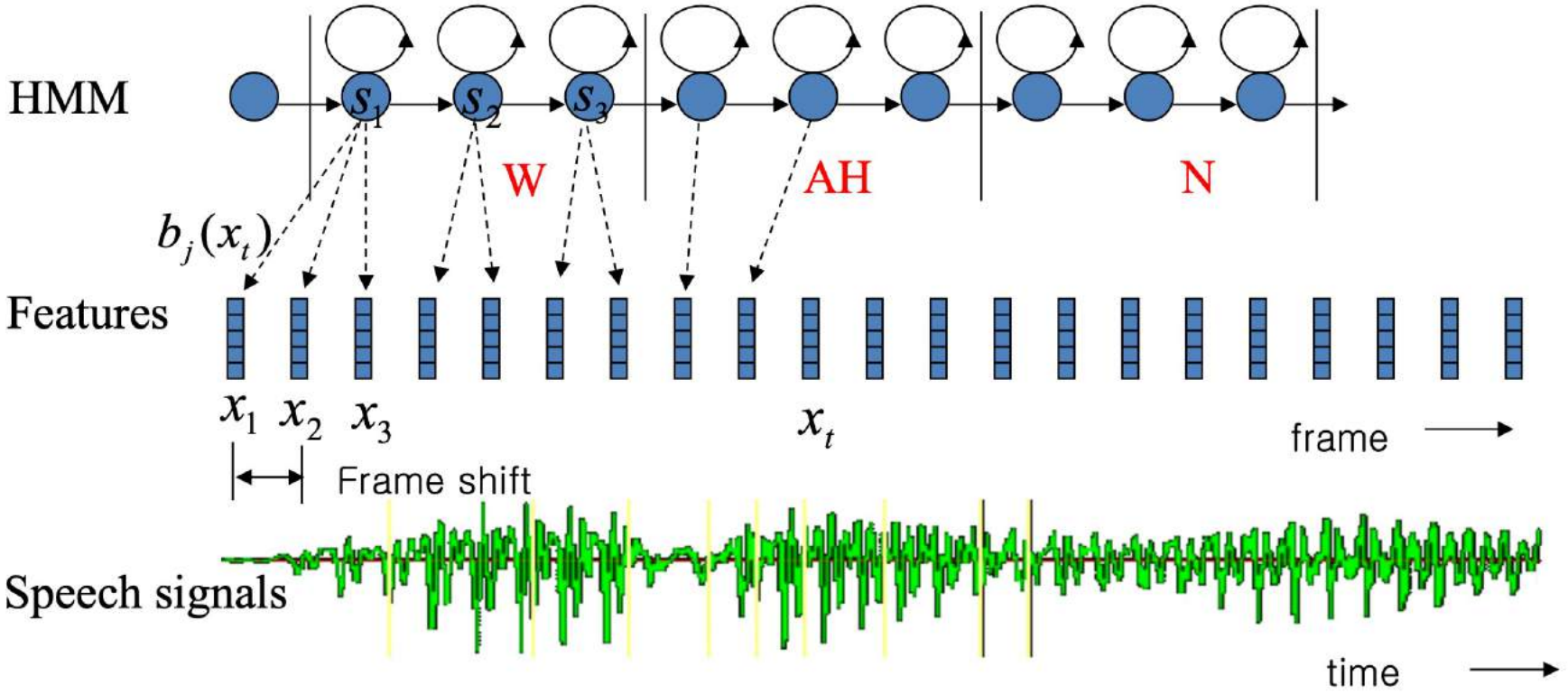
- Stochastic Grammar
  - N-gram

```
P(에서|서울)=0.2 P(세시|에서)=0.5
P(출발|세시)=1.0 P(하는|출발)=0.5
P(출발|서울)=0.5 P(도착|대구)=0.9
…
```

Prononciation Model

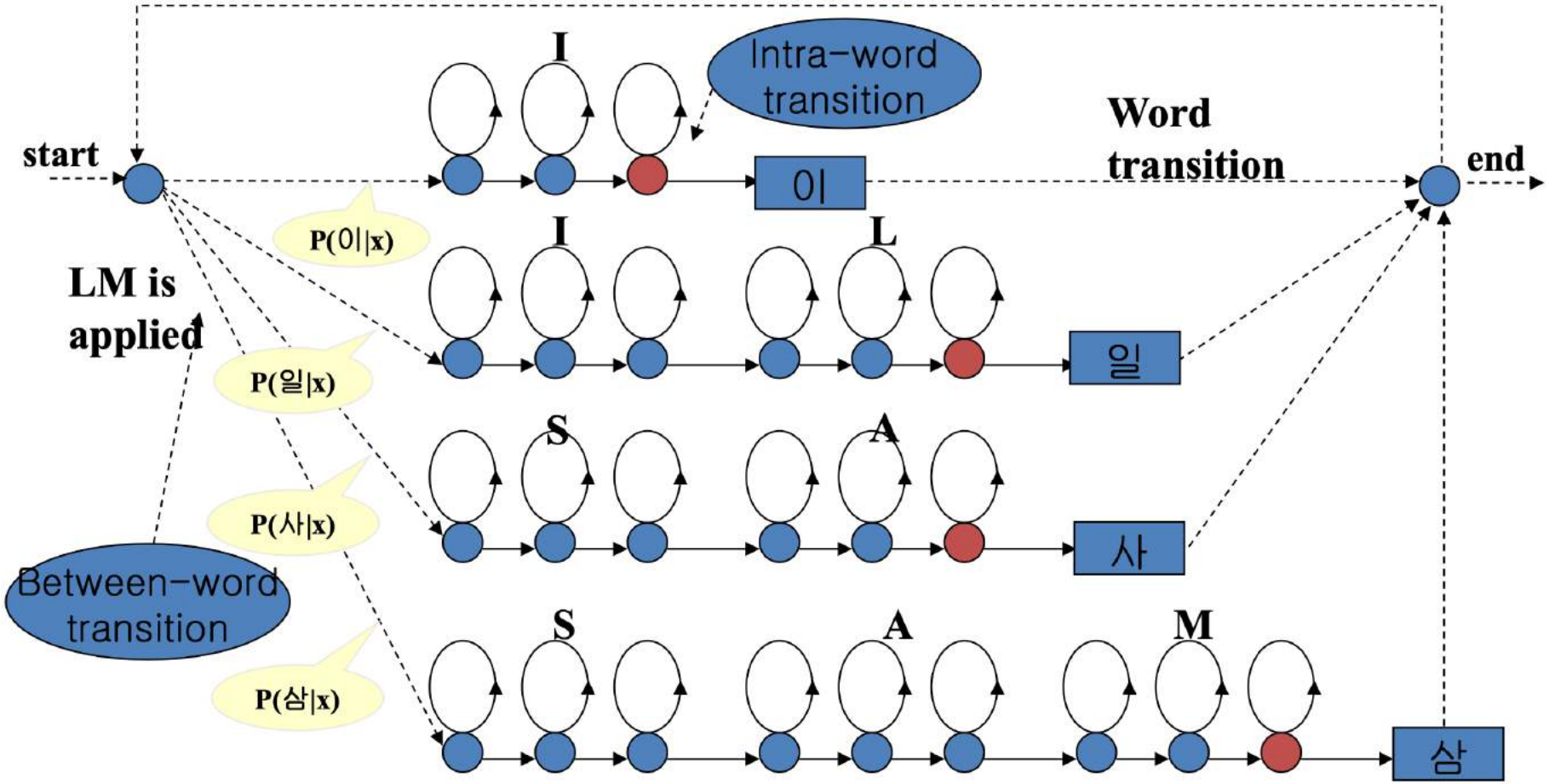
- How a word is pronunciated
- Very Langauge-Dependent and Requires Expert Knowledge
  - 대한민국: /d E h a xn m i xn g u xg/
  - 2NE1, 야탑역, 맨유
- Phoneset
  - 한국어: ETRI 46 phoneset
  - 영어: CMUDict(48), TIMIT(61) → CMU 39 phoneset
- Rule-based, Statistical Approach, Neural Approach

Acoustic Model



http://speech.cbnu.ac.kr/srhome/technology/index.html

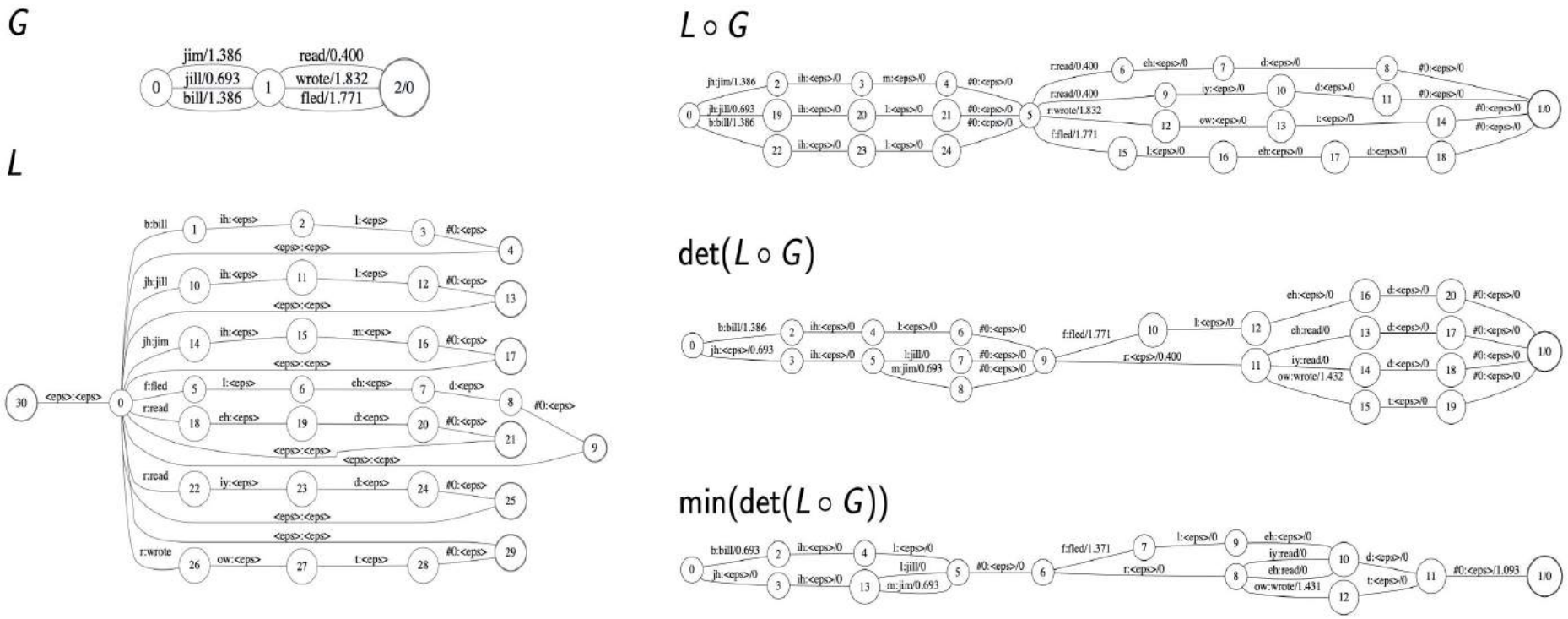
Search Network



http://speech.cbnu.ac.kr/srhome/technology/index.html

Search Network: wFST

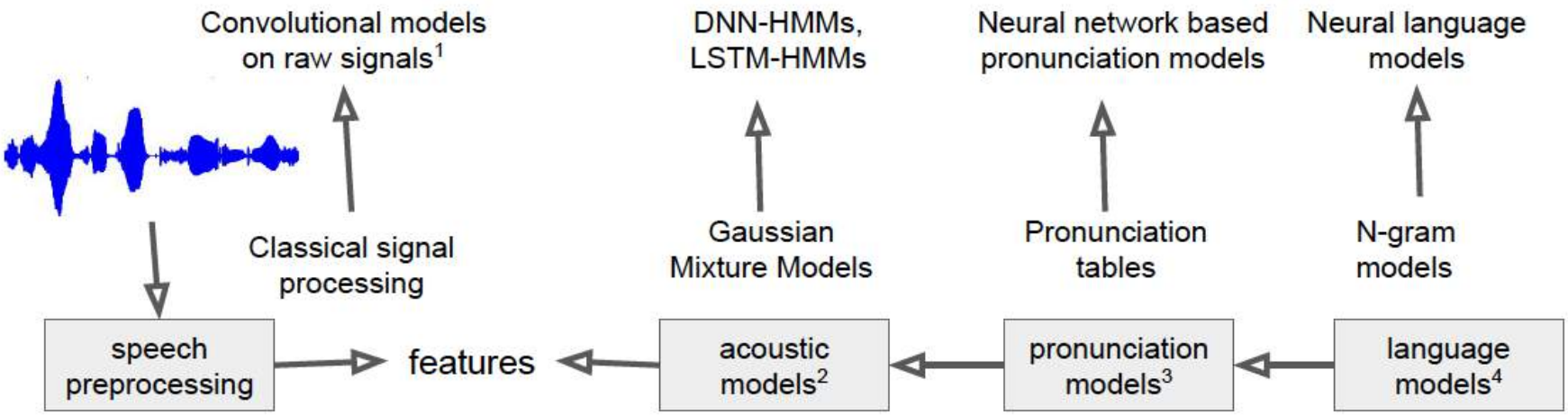
- Weighted Finite Statue Transducer



https://medium.com/@jonathan_hui/speech-recognition-weighted-finite-state-transducers-wfst-a4ece08a89b7

Deep Learning for ASR

Deep Learning for ASR



https://heartbeat.fritz.ai/the-3-deep-learning-frameworks-for-end-to-end-speech-recognition-that-power-your-devices-37b891ddc380

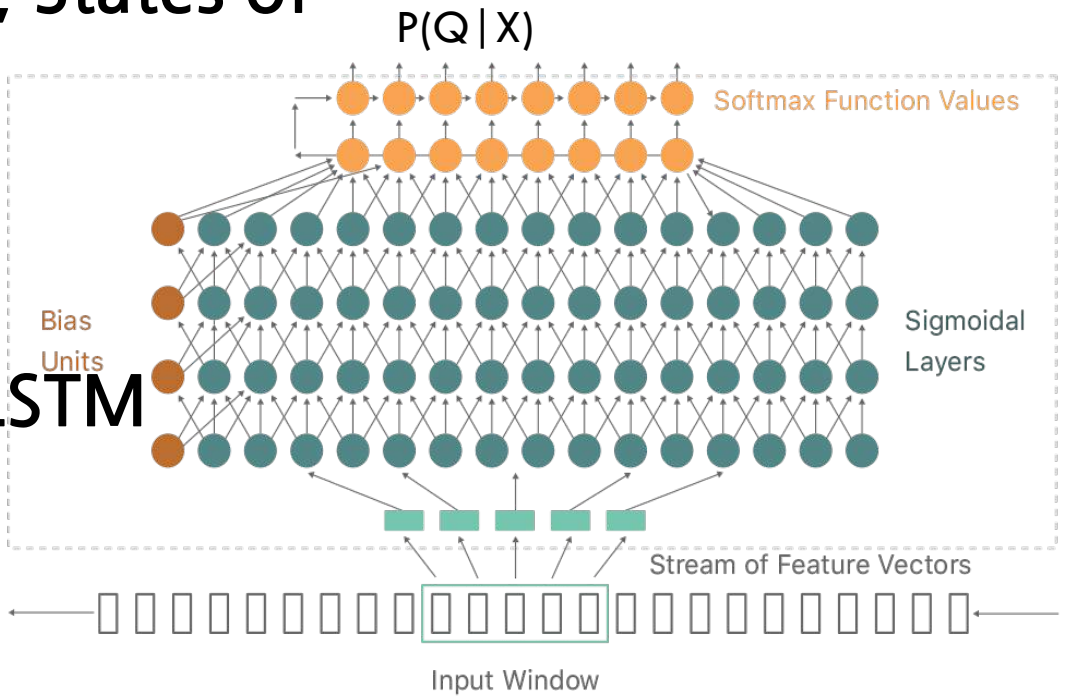
DNN-HMM (1)

- Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition, 2012, IEEE Trans. on Audio, Speech and Language Processing, Microsoft
- Large Vocabulary Continuous Speech Recognition With Context-dependent DBN-HMMS, 2011, ICASSP, Microsoft

- Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups, 2012, Hinton et al.

DNN-HMM (2)

- P(X|Q) = P(Q|X)P(X)/P(Q)
- Output Units: Senone, States of HMM(5k~20k)
- FC-DNN
- CNN
- RNN: GRU, LSTM, Bi-LSTM
- TDNN
- Longer Context Helps



TRAINING OF DNN-HMM

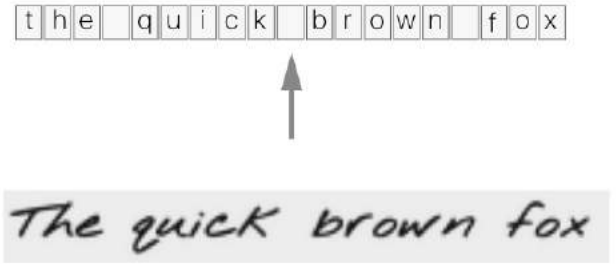
- Requires Frame-wise Label
- Forced-Alignment using Seed Model (Usually GMM-HMM Model)
  - Speech/Text Pair → g2p → State level alignment
- Kaldi Toolkit (2009~, JHU)
  - https://github.com/kaldi-asr/kaldi
- HTK Toolkit (1989~, Cambridge)
  - http://htk.eng.cam.ac.uk/
  - https://github.com/open-speech/HTK

Sequence Training

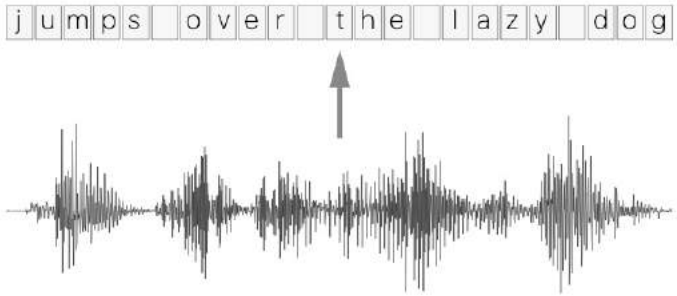
- Want to optimize sequence level objective function, not frame, phone or word
- ‘chain’ model in Kaldi
- CTC: Connectionist Temporal

CTC: Connectionist Temporal Classification

- Solving Sequence Labelling Problem



Handwriting recognition: The input can be (x, y) coordinates of a pen stroke or pixels in an image.



Speech recognition: The input can be a spectrogram or some other frequency based feature extractor.

- Connectionist Temporal Classification : Labelling Unsegmented Sequence Data with Recurrent Neural Networks , A. Graves, S. Fernandez, F. Gomez, J. Schmidhuber, Proceedings of the 23rd international conference on Machine Learning, pp. 369--376. 2006.

https://distill.pub/2017/ctc/

CTC

- Find mapping between input X and output Y

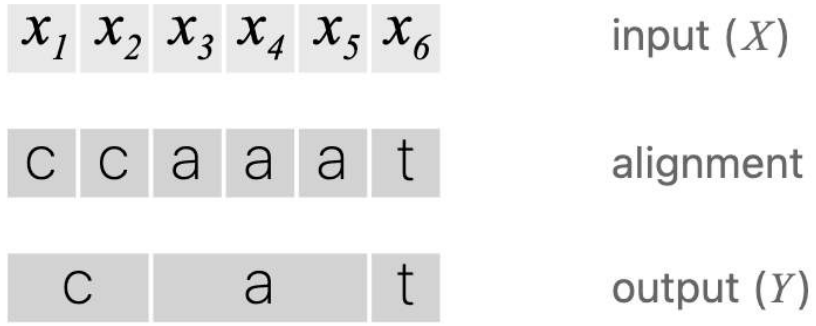
$$X = [x_1, x_2, \ldots, x_T] \qquad Y = [y_1, y_2, \ldots, y_U]$$

  - Both X and Y can vary in length
  - The ratio of the lengths of X and Y can vary.
  - We don't have an accurate alignment (correspondence of the elements) of X and Y.
- Training
  - maximize the probability it assigns to the right answer. Need to compute p(Y|X) efficiently and p(Y|X) should be differentiable
- Inference (Decoding) $Y^* = \underset{Y}{\operatorname{argmax}}\ p(Y \mid X)$

https://distill.pub/2017/ctc/

CTC Alignment

- Assume input length=6, Y=[c, a, t]

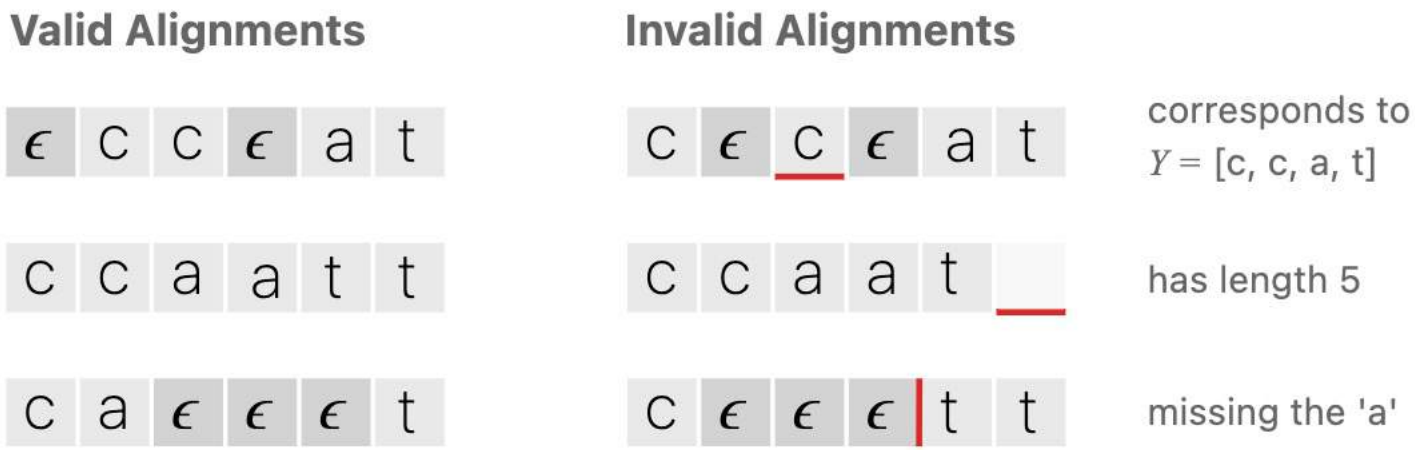


- Blank symbol, ϵ

https://distill.pub/2017/ctc/

CTC Alignment

- Assume input length=6, Y=[c, a, t]



- Property
  - Monotonic
  - X→Y is many-to-one
  - len(X) >= len(Y)

https://distill.pub/2017/ctc/

Loss Function

- P(Y|X)

https://distill.pub/2017/ctc/

Loss Function

- P(Y|X)

$$p(Y \mid X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p_t(a_t \mid X)$$

The CTC conditional **probability**

marginalizes over the set of valid alignments

computing the **probability** for a single alignment step-by-step.

https://distill.pub/2017/ctc/

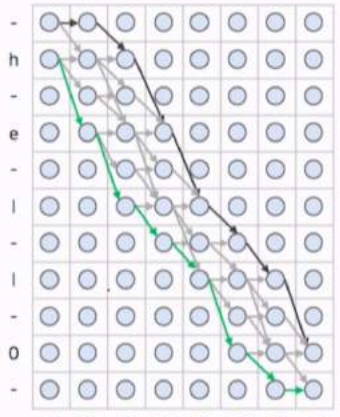
Efficient Computation

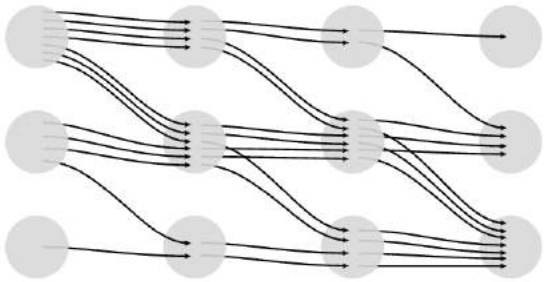
- All possible paths

CTC 입력 확률 벡터 시퀀스가 8개이고, 정답 레이블 시퀀스 l이 h, e, l, l, o일 때

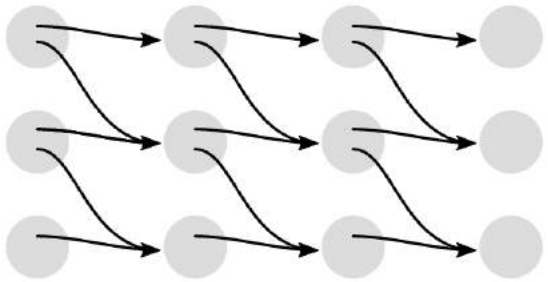
- - h - e - l - l - o -
- 검정색 실선 : ---hello
- 녹색 실선 : hello---

<출처 : ratsgo.github.io>





Summing over all alignments can be very expensive.

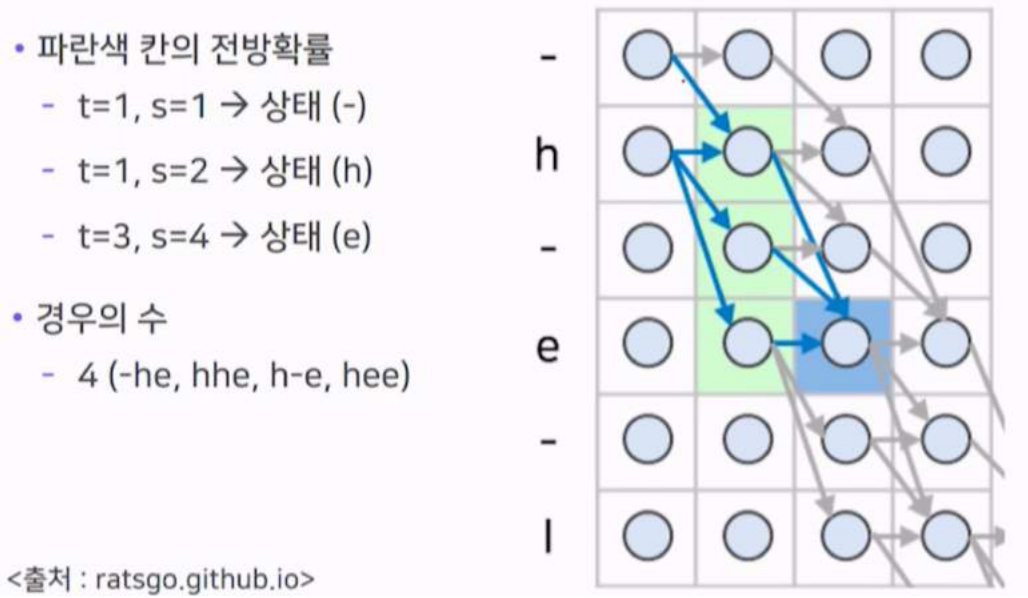


Dynamic programming merges alignments, so it's much faster.

https://distill.pub/2017/ctc/

CTC Forward Algorithm

- Let α be the score of the merged alignments at a given node



https://distill.pub/2017/ctc/

CTC Forward Algorithm

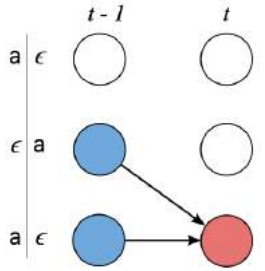
Let's let α be the score of the merged alignments at a given node. More precisely, $\alpha_{s,t}$ is the CTC score of the subsequence $Z_{1:s}$ after t input steps. As we'll see, we'll compute the final CTC score, $P(Y \mid X)$, from the α's at the last time-step. As long as we know the values of α at the previous time-step, we can compute $\alpha_{s,t}$. There are two cases.

- **Case 1:**
  - can't jump over previous token

$$\alpha_{s,t} = (\alpha_{s-1,t-1} + \alpha_{s,t-1}) \cdot p_t(z_s \mid X)$$

The CTC probability of the two valid subsequences after $t-1$ input steps.

The probability of the current character at input step t.

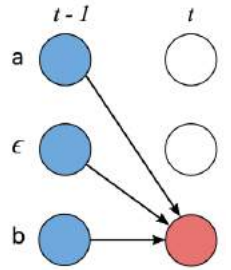


- **Case 2:**
  - can

$$\alpha_{s,t} = (\alpha_{s-2,t-1} + \alpha_{s-1,t-1} + \alpha_{s,t-1}) \cdot p_t(z_s \mid X)$$

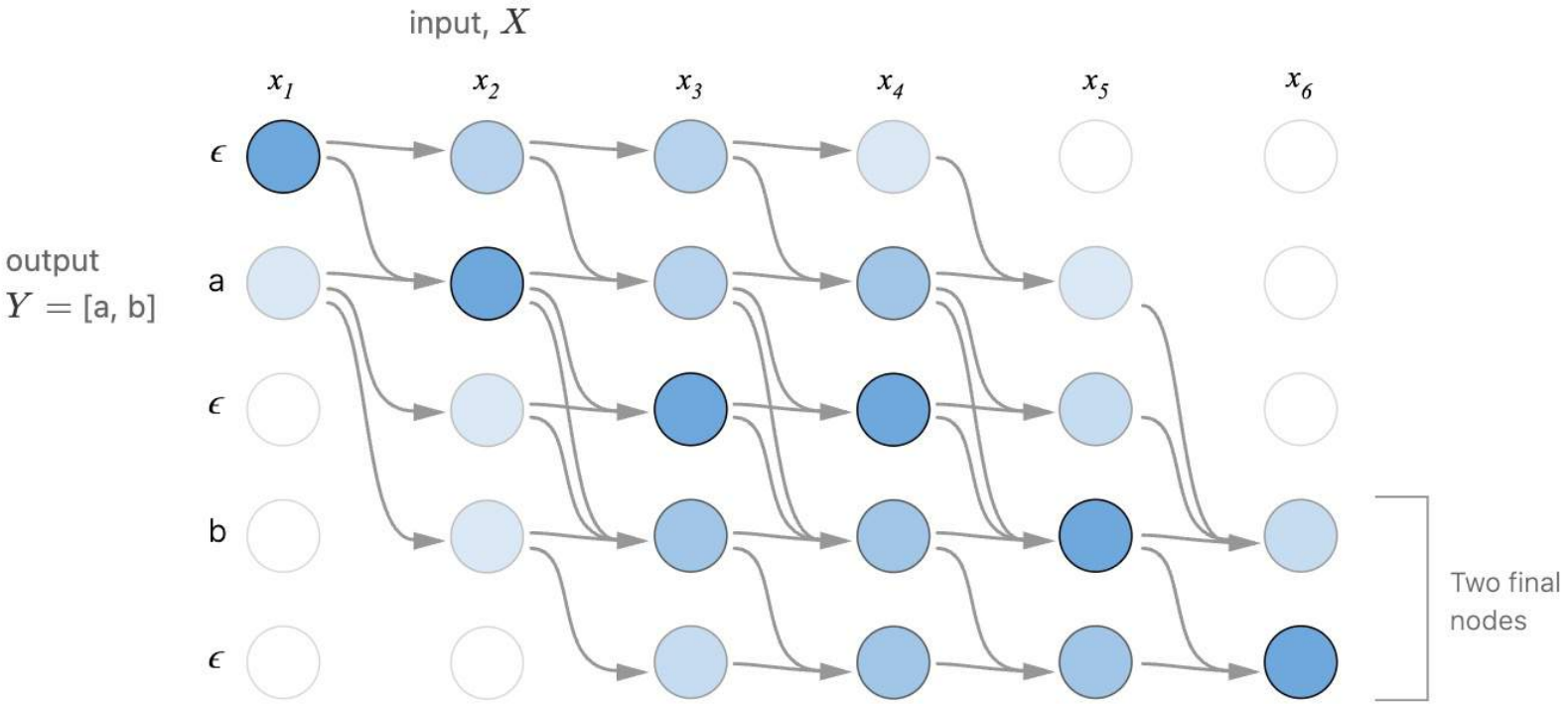
The CTC probability of the three valid subsequences after $t-1$ input steps.

The probability of the current character at input step t.



https://distill.pub/2017/ctc/

CTC Forward Algorithm



Node (s, t) in the diagram represents $\alpha_{s,t}$ – the CTC score of the subsequence $Z_{1:s}$ after t input steps.

https://distill.pub/2017/ctc/

CTC Training

- Compute the loss function and run the back propagation
- Minimize the negative log-likelihood

$$\sum_{(X,Y)\in\mathcal{D}} -\log\ p(Y \mid X)$$

https://distill.pub/2017/ctc/

CTC Inference

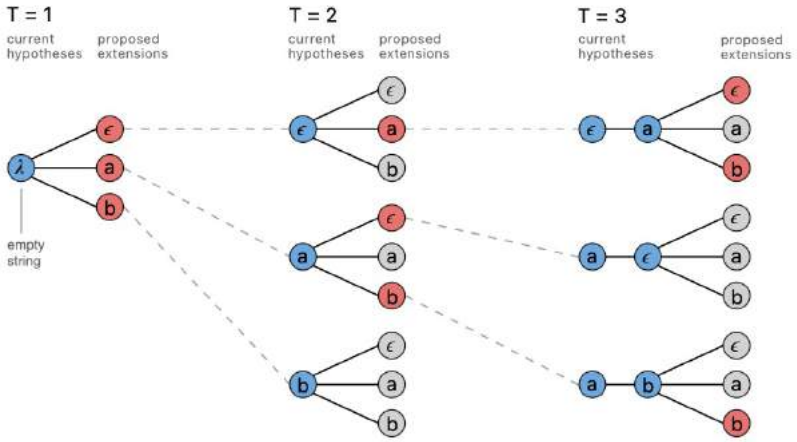
- Global Solution

$$Y^* = \underset{Y}{\operatorname{argmax}}\ p(Y \mid X)$$

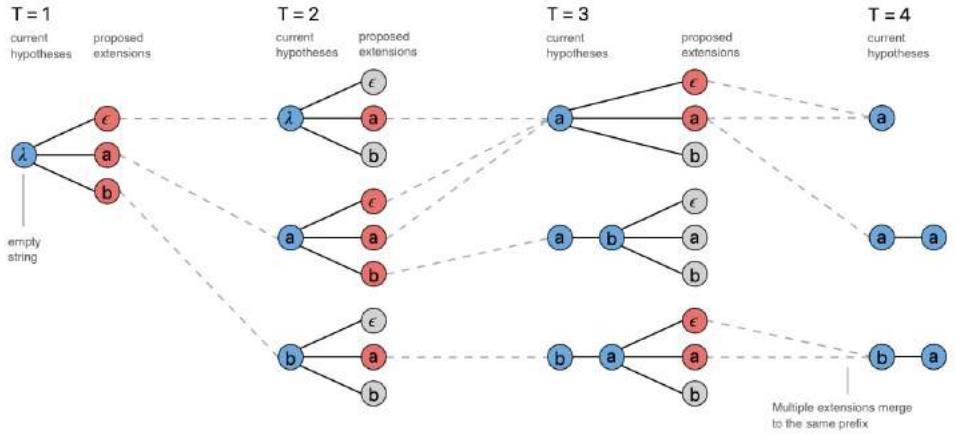
- Local Solution

$$A^* = \underset{A}{\operatorname{argmax}} \prod_{t=1}^{T} p_t(a_t \mid X)$$

- Modified beam-search



A standard beam search algorithm with an alphabet of $\{\epsilon, a, b\}$ and a beam size of three.



The CTC beam search algorithm with an output alphabet $\{\epsilon, a, b\}$ and a beam size of three.

https://distill.pub/2017/ctc/

End-to-end ASR

Contents

- How (Ancient) ASR Works: Recap
- How ASR Works: To-Be
- Introduction to
  - RNN, Attention, Encoder-Decoder, Word Embedding
  - Sequence-to-Sequence Model
- Transformer
- Transformer for ASR
- End-to-End ASR in Practice
- Q&A

Guess who?

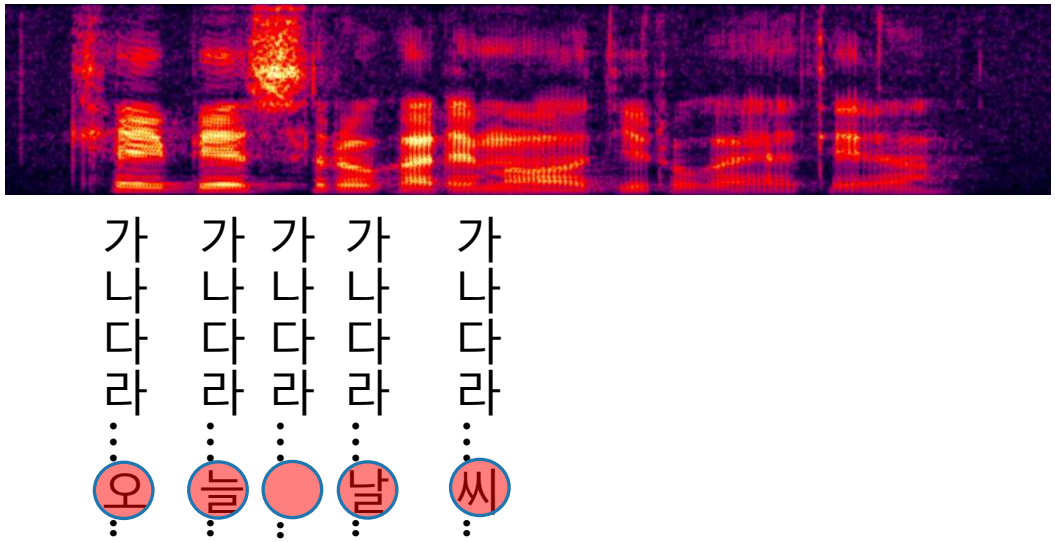
- Find A Criminal Among Suspects Given Evidence
- Criminal = argmax P(Suspect|Evidence)
- Criminal = argmax P(Evidence|Suspect)

  = argmax

  P(Evidence|Behavior)P(Behavior|Suspect)P(Suspect)

How It works: REVISITED

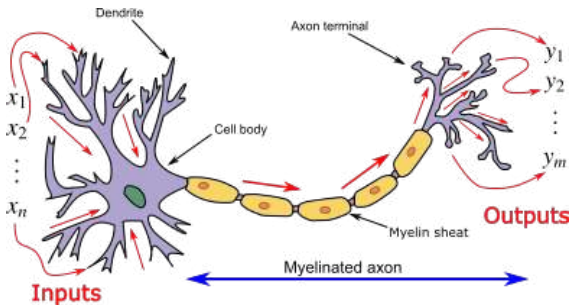
- W* = argmax P(W|X)
  - To Find Most Probable Word Sequence Given Input Signal/Feature



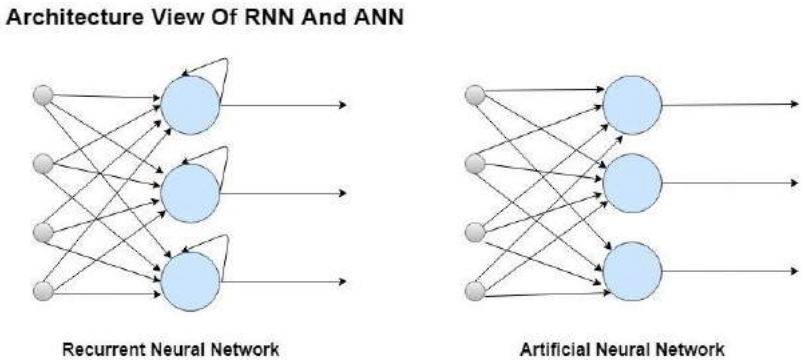
- Considerations
  - Boundary? Segmentation?
  - Output Units? Words, Characters, Phoneme, …
  - Classification Accuarcy? Unit Accuracy vs. Sentence Accuracy

RNN: Recurrent neural network

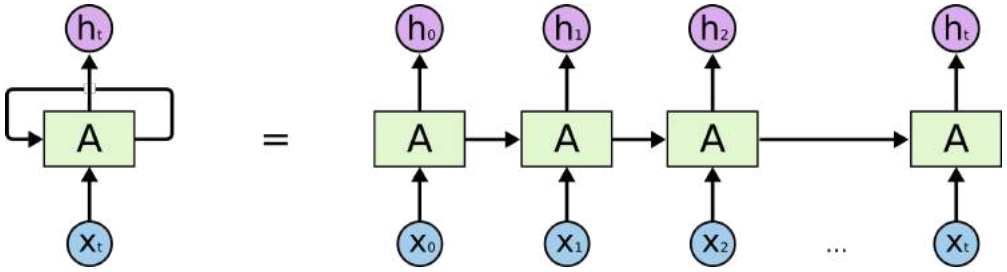
- Neural Networks
  - Mimic human brain: Neuron, Synapse



https://en.wikipedia.org/wiki/Nervous_system



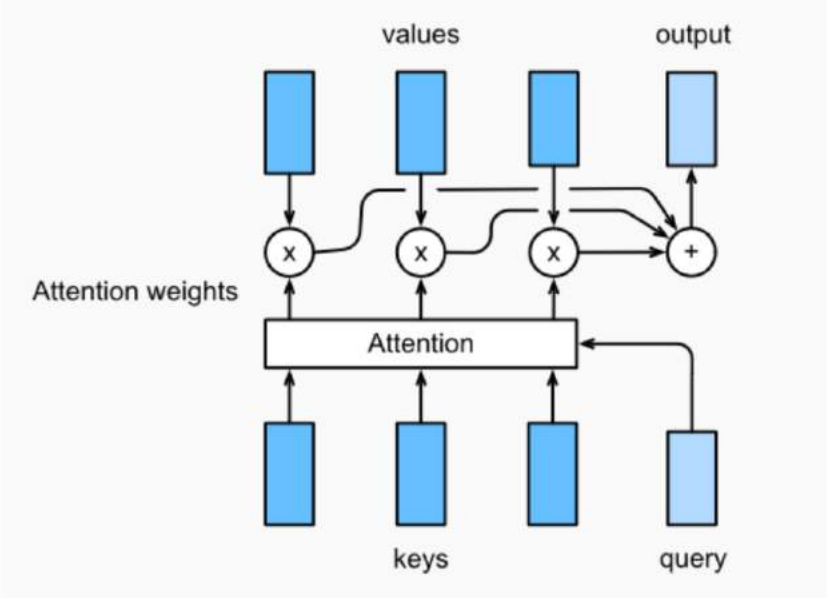
https://medium.com/datadriveninvestor/recurrent-neural-networks-in-deep-learning-part-1-df3c8c9198ba



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Attention

- Query, Key, Value
- Memory = Dictionary(Key,Value)
- Output = Weighted Sum of Value
- Weight = Similarity Between Query and Key



https://programming.vip/docs/5e4cadd75dc1d.html

Word Embedding

- Word2Vec, …
- Sparse Representation vs. Dense Representation
- Preserve Meaning
  - 한국 - 서울 + 파리 = 프랑스
  - 어머니 - 아버지 + 여자 = 남자
  - 아버지 + 여자 = 어머니

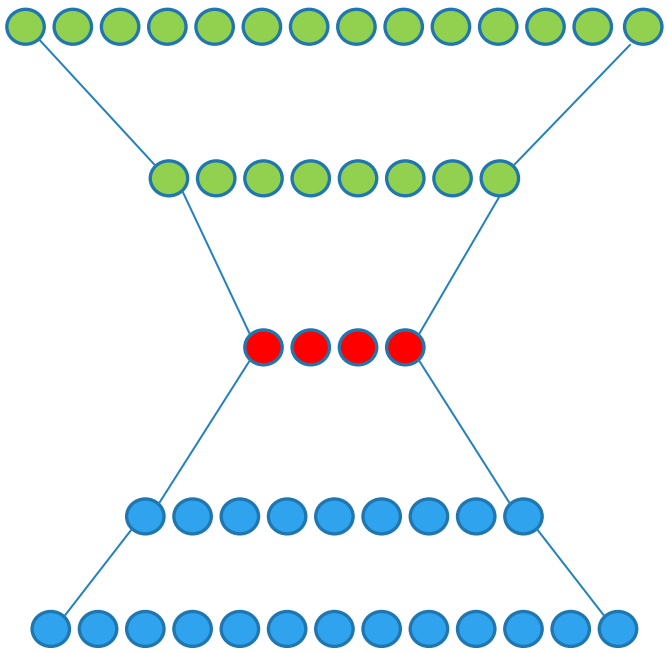
Index	Words	One-hot
1	aaron	000…00001
2	aback	000…00010
3	abacus	000…00100
…	…	…
15439	macaroni	00…010…0
	…	…
29500	zulu	1000…0000

Tokenizing

- Byte Pair Encoding
  - Neural Machine Translation of Rare Words with Subword Units, ACL, 2016
- Token?
  - Char vs Word vs Subword
- How to?
  - example: https://wikidocs.net/22592

encoder-decoder

- Auto Encoder



Y: Output

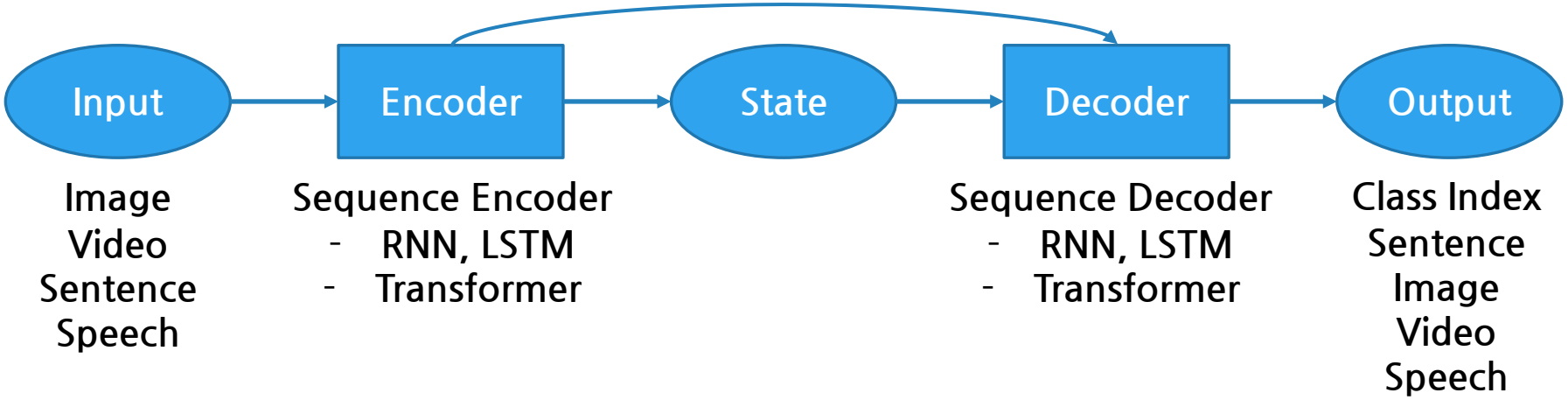
Decoder

Z: Latent Variable, Hidden Representation
Context

Encoder

X: Input

Encoder-Decoder for Sequence

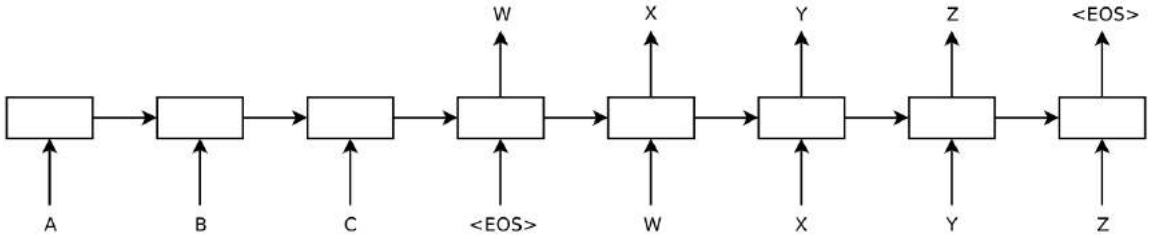


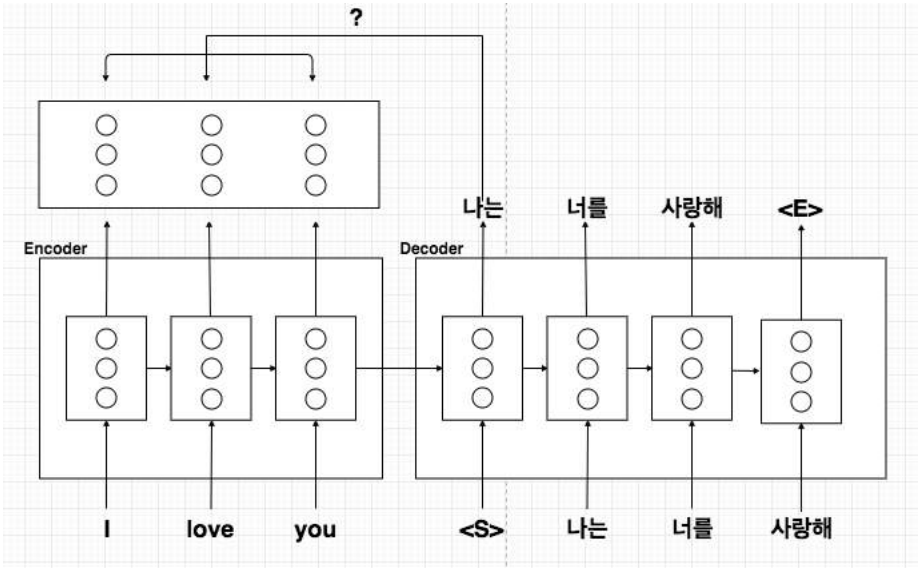
- Translation
- Image/Video Captioning
- Q&A, Document Summarization
- Speech
  - Recognition, Synthesis, Translation, Dialog System(Google Duplex, 2018)

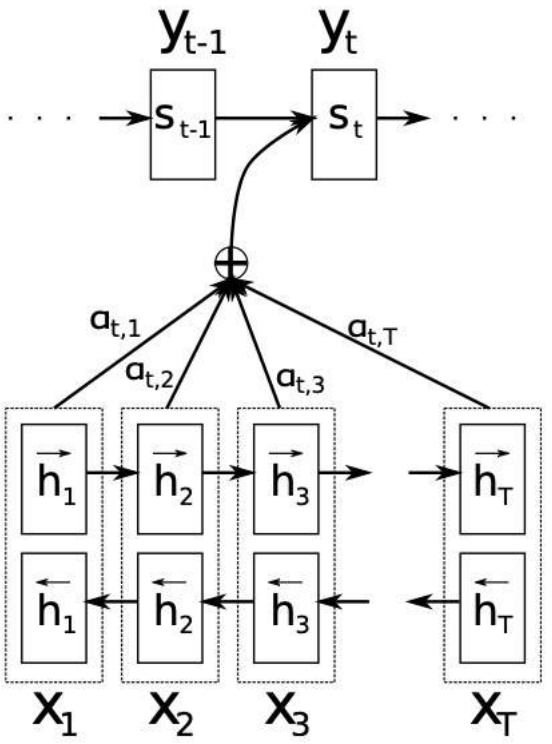
Era of Sequence-to-Sequence

- Natural Language Processing
- Sequence to Sequence Leanring with Neural Networks, NeurIPS, 2015
- Neural Machine Translation By Jointly Learning To Align And Translate, ICLR, 2016
- Attention Is All You Need, NuerIPS, 2017
- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, ACL, 2019

Sequence to Sequence with Attention





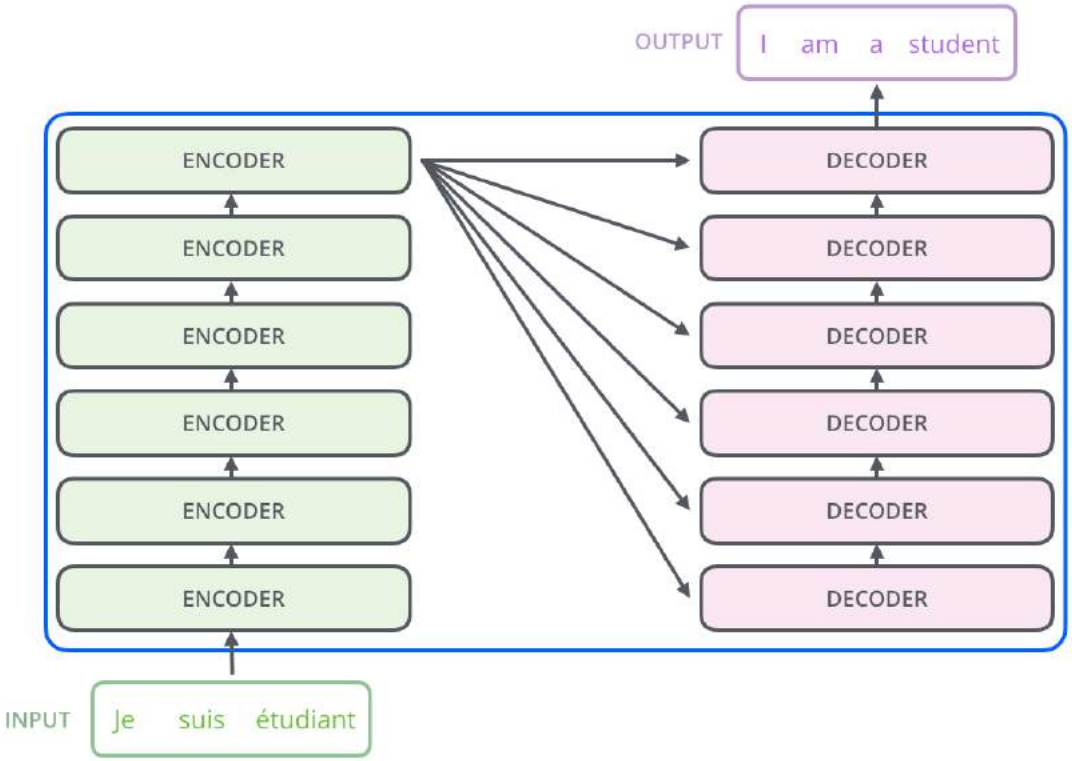


https://medium.com/platfarm어텐션-메커니즘과-transfomer-self-attention-842498fd3225

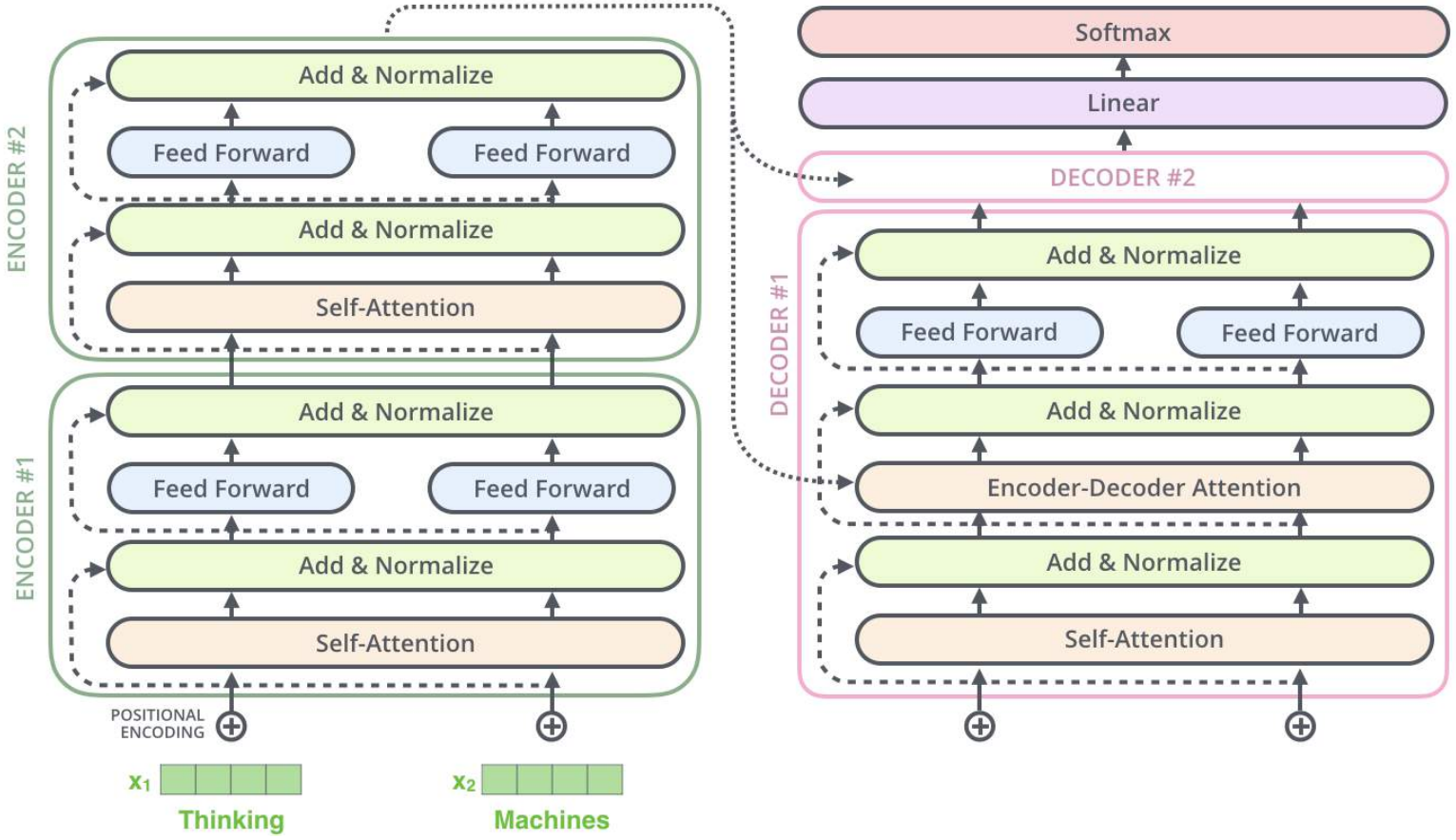
Transformer

Transformer: Overall STructure

- https://jalammar.github.io/illustrated-transformer/

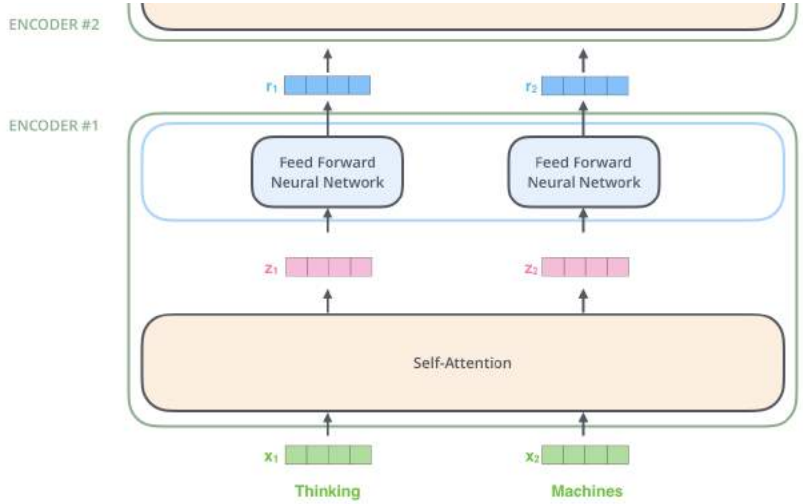


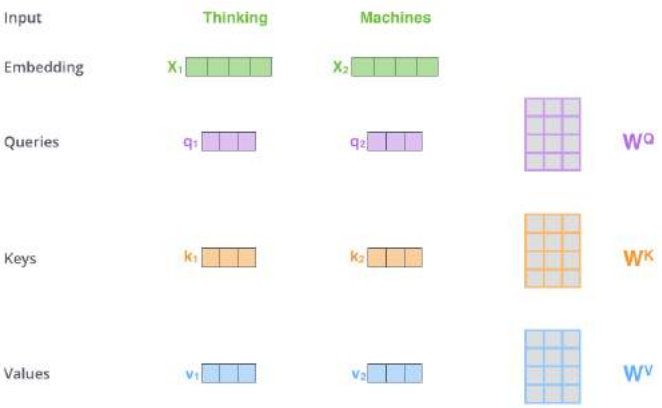
Transfomer: Detailed STructure

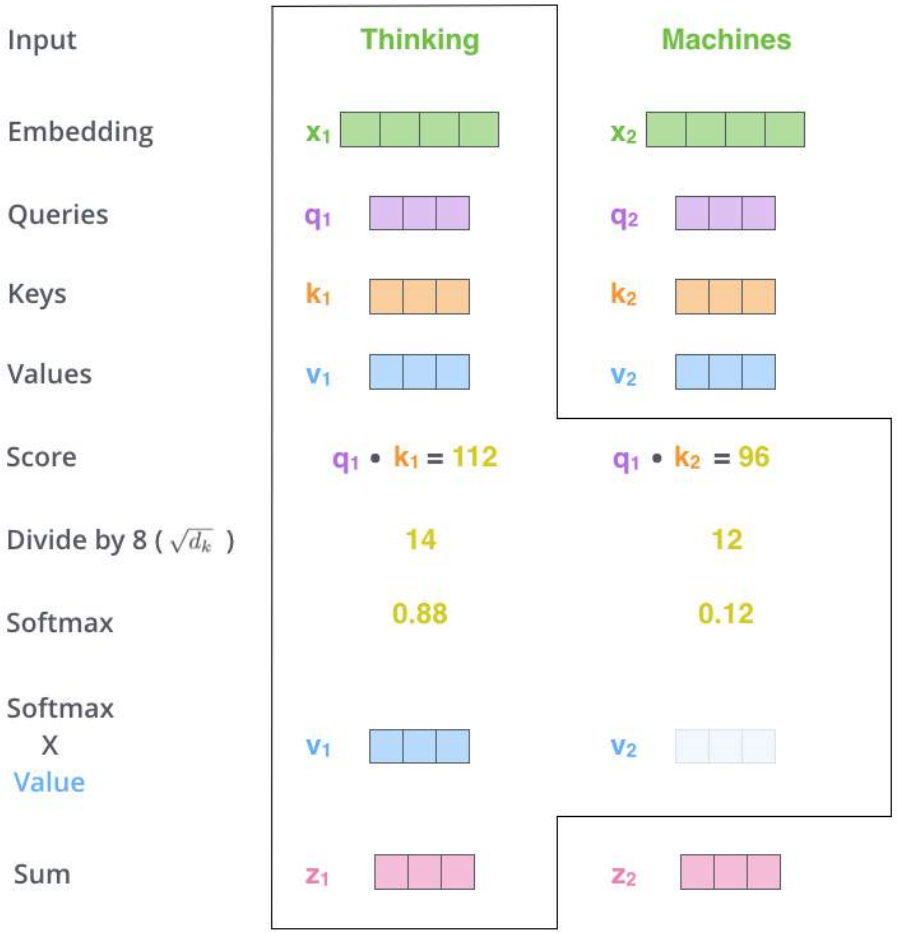


https://jalammar.github.io/illustrated-transformer/

Self Attention

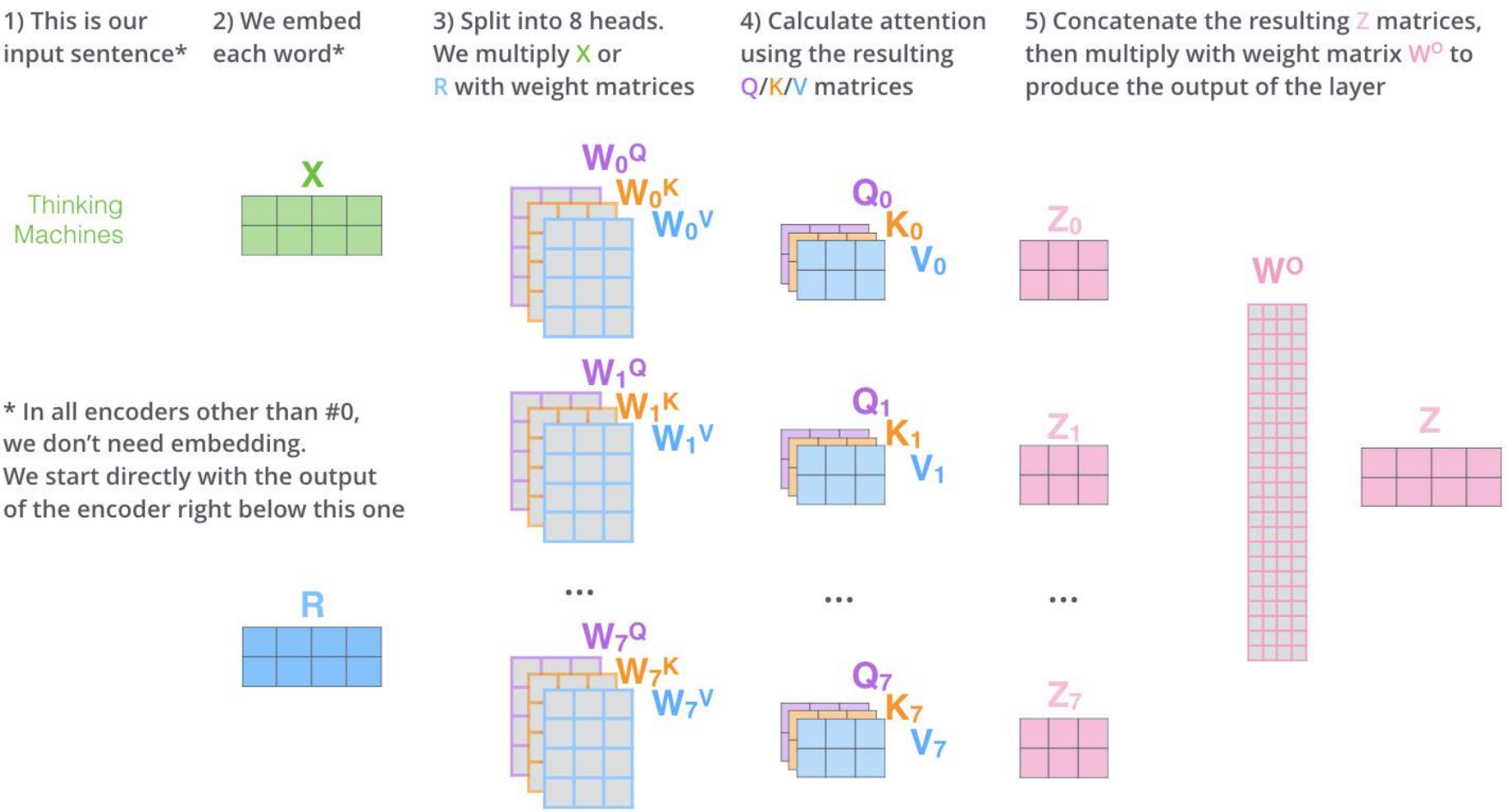






https://jalammar.github.io/illustrated-transformer/

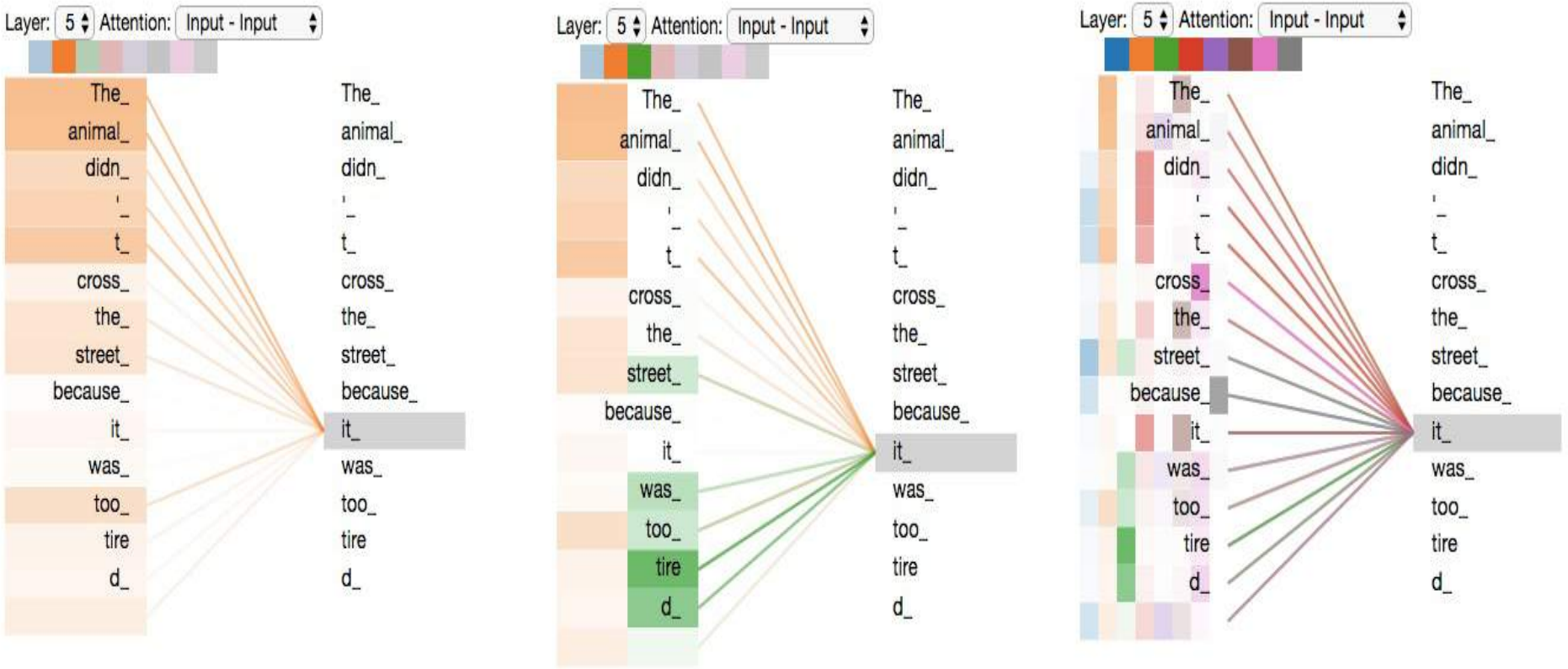
Multihead attention



https://jalammar.github.io/illustrated-transformer/

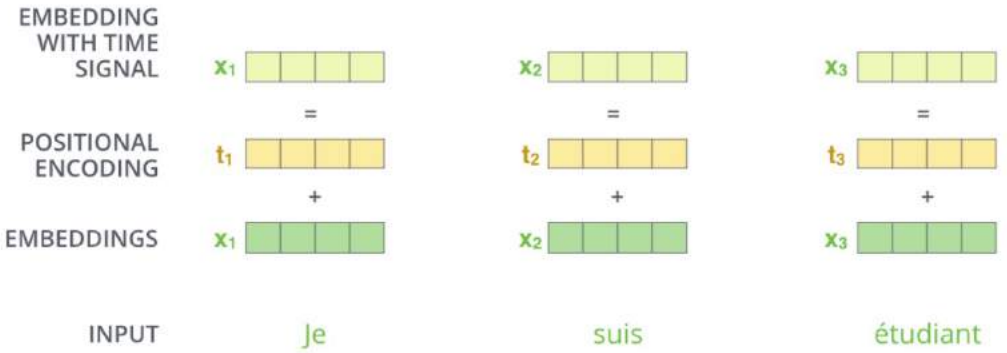
Effect of Self Attention

The animal didn't cross the street because it was too tired



Positional Encoding

- No Position Depenent Computation in Transfomer



0 : 0 0 0 0
1 : 0 0 0 1
2 : 0 0 1 0
3 : 0 0 1 1
4 : 0 1 0 0
5 : 0 1 0 1
6 : 0 1 1 0
7 : 0 1 1 1

- Absolute/Relative Position Encoding
  - Sinusoidal Positional Encoding

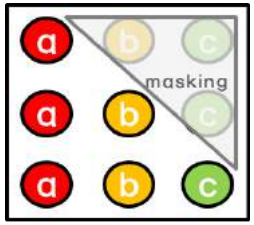
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

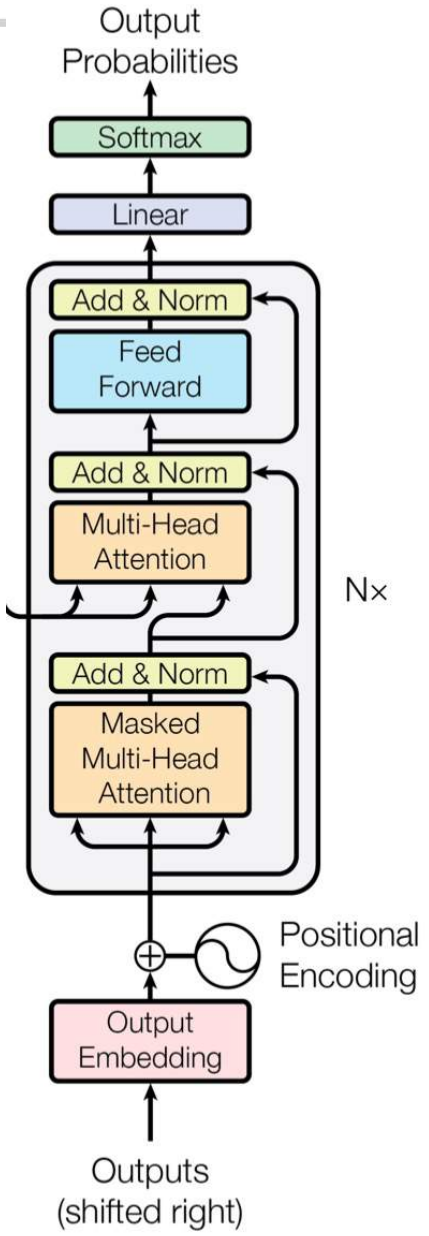
https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Decoder

- Masked Multi-Head Self Attention

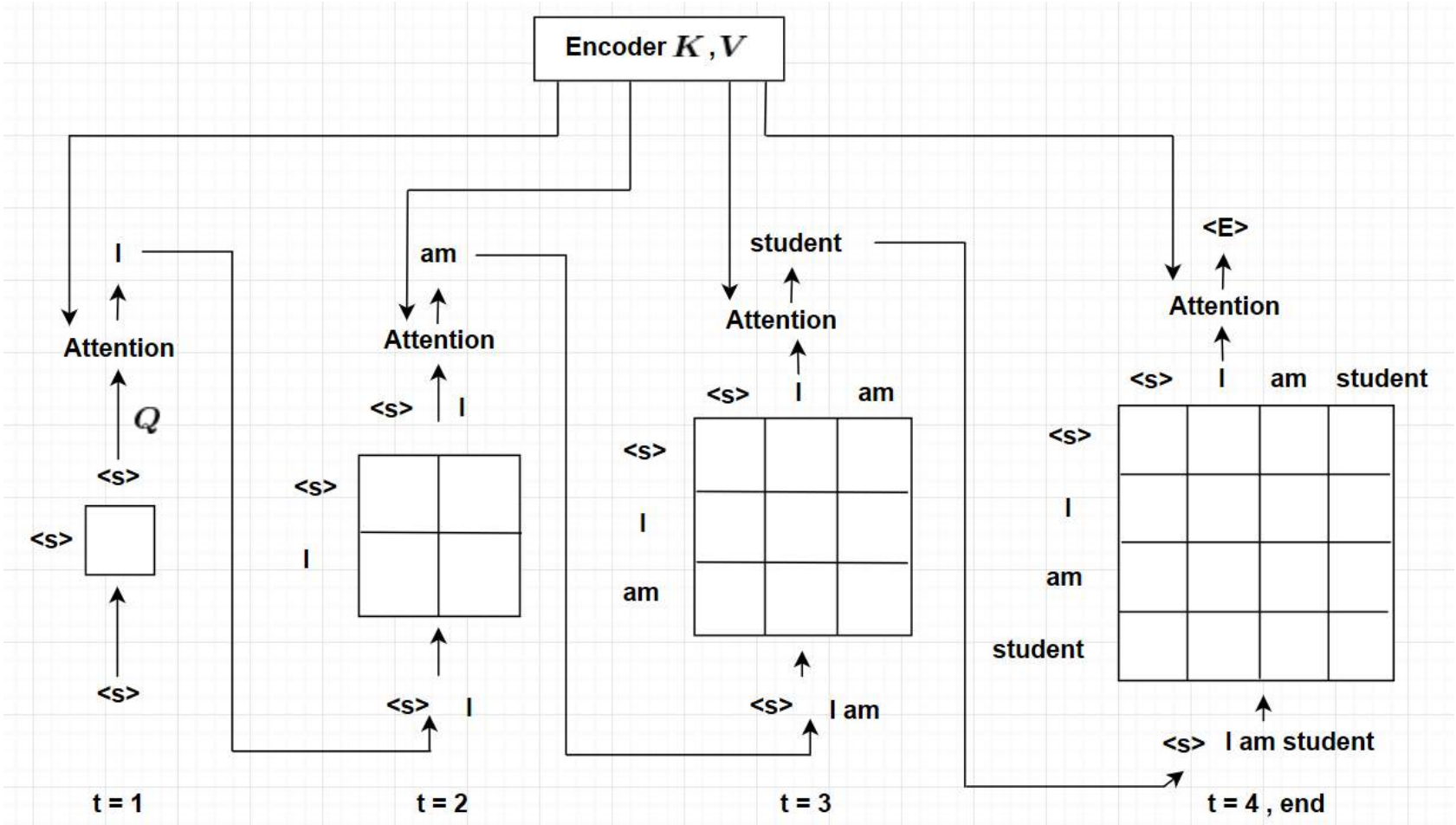


- Encoder-Decoder Attention
  - K, V from Encoder Last Layer
  - Q from Self Attention
- Beam Search



Attention Is All You Need, NuerIPS, 2017

Decoder in action



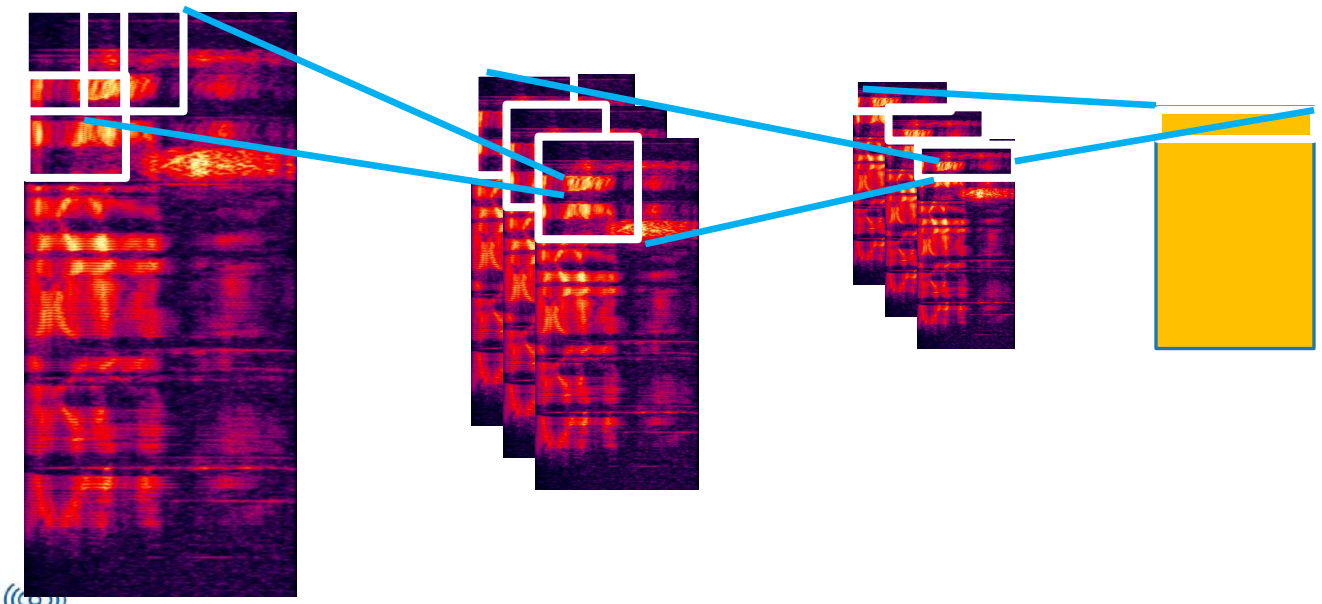
https://medium.com/platfarm/어텐션-메커니즘과-transfomer-self-attention-842498fd3225

End-to-end For ASR

- ESPNet: End-to-end Speech Processing Toolkit
  - ASR, TTS, Speech Translation
  - https://github.com/espnet/espnet
- CTC Hybrid*: Connectionist Temporal Classification
  - Multi-task training with CTC Criteiron
  - Increase Stability while Training
  - Hybrid CTC/Attention Architecture for End-to-End Speech Recognition, IEEE Journal of Selected Topics in Signal Processing, 2018
- Input Embedding

Input embedding

- TEXT: Input = Word: One-hot → Vector
- ASR: Input = MELFB: Vector → Vector
- 2x Conv2d layer, 3x3 kernel with stride=2
  - TxF → adim x T/2 x F/2 → adim x T/4 x F/4 → T/4 x adim



Python Code

espnet2/asr/espnet_model.py (class ESPnetASRModel(AbsESPnetModel))
espnet2/asr/encoder/transformer_encoder.py (class TransformerEncoder(AbsEncoder))

def __init__()

def forward()

End-to-End ASR In Practice

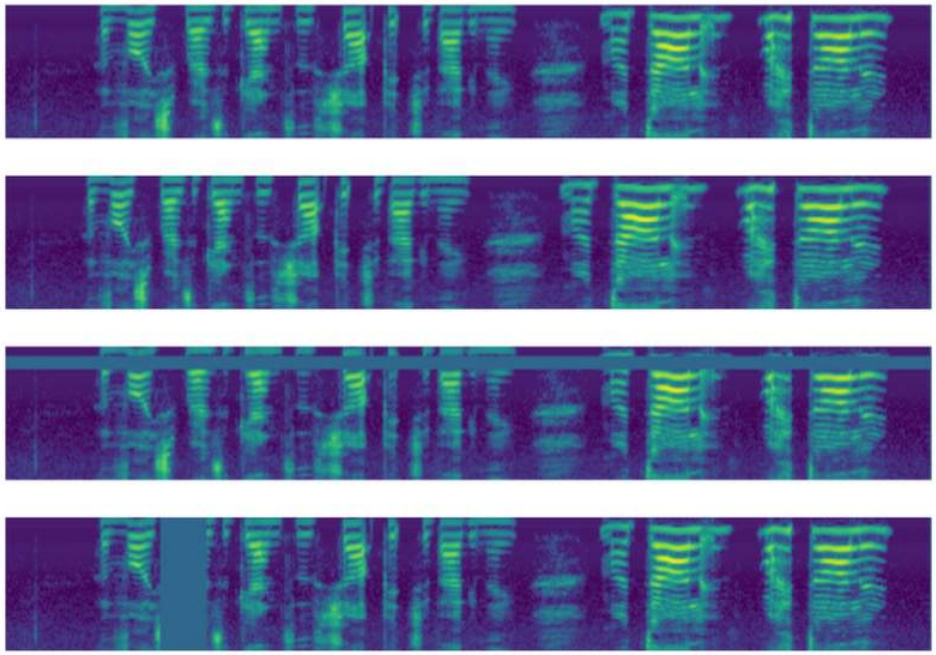
- Output Units
  - 영어: Alphabet, BPE(Byte Pair Encoding), Word
  - 한국어: Char(음절~2500), BPE(~5000), 형태소분석기
- Relative Performance
  - WER/CER
  - 25% (GMM-HMM) → 15% (DNN-HMM) → 10% (LSTM-HMM)
  - 7% Transformer
- Limitation
  - Process Whole Sentence → Streaming ASR

성능개선방안

- 데이터!
  - 실환경 데이터수집: 적응훈련/연결학습
  - 음향모델/언어모델?
- 데이터!!
  - 데이터 증강
  - SpecAug, Speed/Volume perturbation, Noise addition, Simulated data
- 모델 파라미터
  - Number of epoch
  - Number of paramerters: layers, dimension etc
  - Gradient scale: batchsize, learning rate etc
  - Robustness: dropout rate,

SpecAug

- SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition (2019)
- https://ai.googleblog.com/2019/04/specaugment-new-data-augmentation.html



Hyperparameters (1)

- Hyperparameter experiments on end-to-end automatic speech recognition (말소리와 음성과학, 2021)

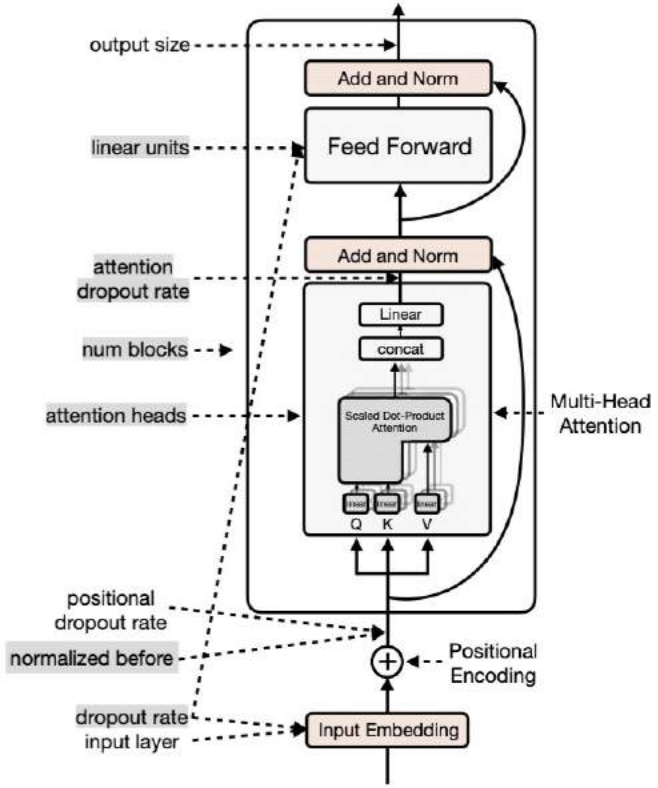


Table 1. The range of hyperparameters in the transformer encoder network

Hyperparameters	Values				
output size	**256**				
input layer	**2d conv**				
normalized before	**True**		false		
attention heads	1	**2**	4	8	
linear units	512	1,024	**2,048**	4,096	
num blocks	2	4	6	8	**12**
dropout rate	0.0	**0.1**	0.2	0.3	0.4
positional dropout rate	**0.1**				
attention dropout rate	**0.0**	0.1	0.2	0.3	0.4

Table 2. The range of transformer decoder network hyperparameters

Hyperparameters	Values				
attention heads	1	**2**	4	8	
linear units	512	1,024	**2,048**	4,096	
num blocks	2	4	**6**	8	12
dropout rate	0.0	**0.1**	0.2	0.3	0.4
positional dropout rate	**0.1**				
self attention dropout rate	**0.0**	0.1	0.2	0.3	0.4
src attention dropout rate	**0.0**				

Hyperparameters (2)

Table 3. The range of the model hyperparameters

Hyperparameters	Values				
batch type	**folded**				
batch size	**32**				
accum grad	**8**				
max epoch	**50**				
patience	**none**				
init	chainer	**xavier uniform**	xavier normal	kai-minguni-form	kaiming normal
optim	**adam**				
lr	**0.005**				
scheduler	**warmuplr**				
warmup steps	10,000	20,000	**30,000**	40,000	
keep nbest model	5	**10**	15	20	
ctc weight	0.0	0.1	0.2	**0.3**	0.4
lsm weight	0.0	**0.1**	0.2	0.3	0.4
length normalized loss	true		**false**		

Hyperparameters (3)

- Impact on WER (Example Only)

Table 4. WER from each hyperparameter model on WSJ dev93 and eval92

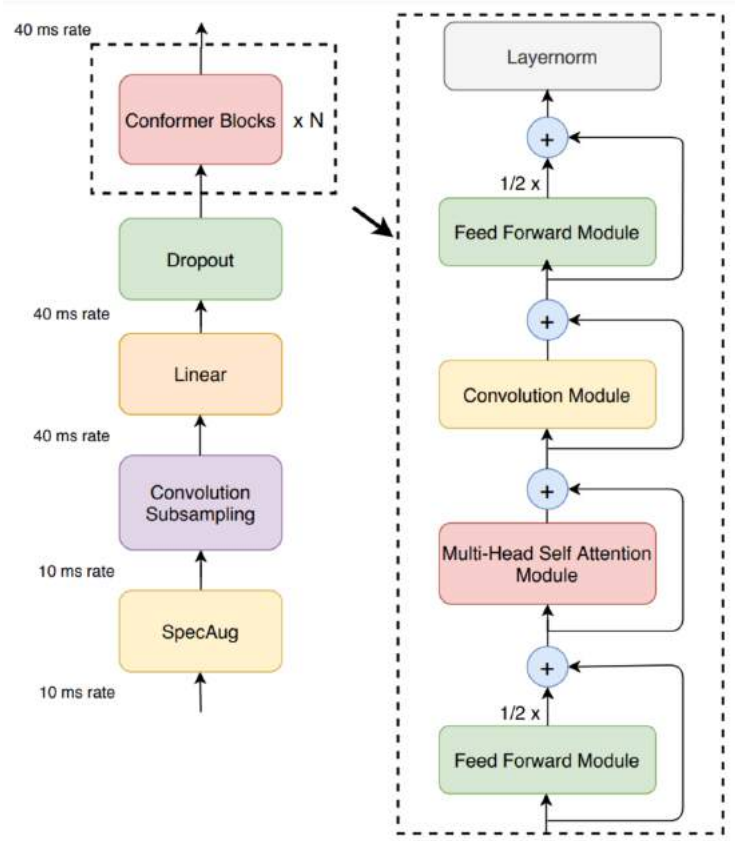
Hyperparameters		Values / WER				
MODEL	init	chainer	xavier uniform	xavier normal	kaiming uniform	kaiming normal
		42.0/35.1	**17.0/12.7**	17.3/14.0	17.7/13.1	17.6/13.4
	warmup steps	10,000	20,000	30,000	40,000	
		15.6/12.4	16.0/12.8	17.3/12.7	17.3/13.6	
	keep nbest model	5	10	15	20	
		17.0/12.8	17.3/12.7	16.9/13.0	16.9/13.4	
	ctc weight		0.1	0.2	0.3	0.4
			17.3/13.6	17.0/13.1	**17.3/12.7**	16.5/13.0
	lsm weight		0.1	0.2	0.3	0.4
			17.3/12.7	17.3/13.2	17.5/12.9	18.0/13.3
	length normalized loss	true	false			
		17.7/14.0	**17.3/12.7**			

ENCODER	attention heads	1	2	4	8	
		17.6/13.3	**16.7/13.0**	17.3/12.7	17.6/13.4	
	linear units	512	1,024	2,048	4,096	
		18.9/14.8	18.0/14.0	17.3/12.7	**16.3/12.3**	
	num blocks	2	4	6	8	12
		24.5/19.7	20.2/16.0	18.5/15.0	17.3/13.5	**17.3/12.7**
	dropout rate	0.0	0.1	0.2	0.3	0.4
		17.4/14.4	**17.3/12.7**	17.7/13.4	17.8/13.7	20.4/15.7
	attention dropout rate	0.0	0.1	0.2	0.3	0.4
		17.3/12.7	16.5/13.0	15.6/12.8	**15.8/12.6**	15.9/12.7
	normalized before	true	false			
		17.3/12.7	14.1			
DECODER	attention heads	1	2	4	8	
		17.3/13.0	17.1/12.9	**17.3/12.7**	17.6/12.8	
	linear units	512	1,024	2,048	4,096	
		17.7/13.5	17.5/13.4	17.2/12.7	**16.9/12.9**	
	num blocks	2	4	6	8	12
		19.7/16.0	17.2/13.6	17.3/12.7	**16.3/12.5**	16.3/12.5
	dropout rate	0.0	0.1	0.2	0.3	0.4
		16.5/13.3	**17.3/12.7**	16.9/13.8	16.3/13.6	17.5/13.5
	self attention dropout rate	0.0	0.1	0.2	0.3	0.4
		17.3/12.7	16.5/13.8	17.0/13.9	16.6/13.8	16.5/13.5

WER, word error rate.

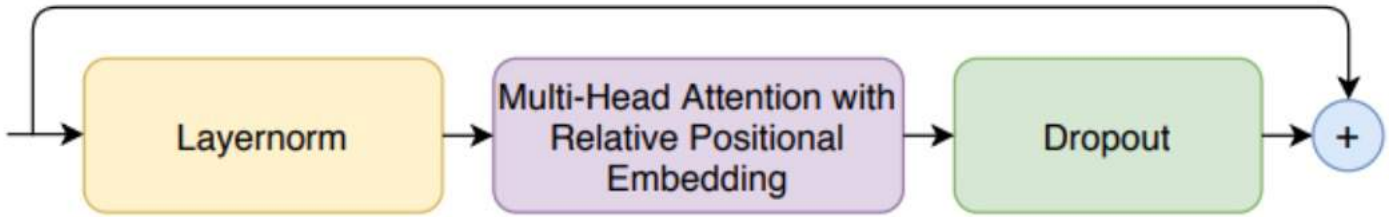
Conformer

- Conformer: Convolution-augmented Transformer for Speech Recognition, 2020, Interspeech, Google



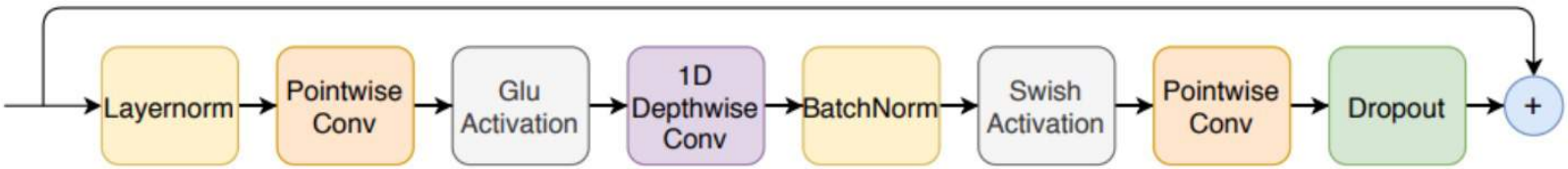
Conformer: Multi-Headed Self Attention Module

- Relative Sinusoidal Positional Encoding
- Pre-norm Residual Units with Dropout

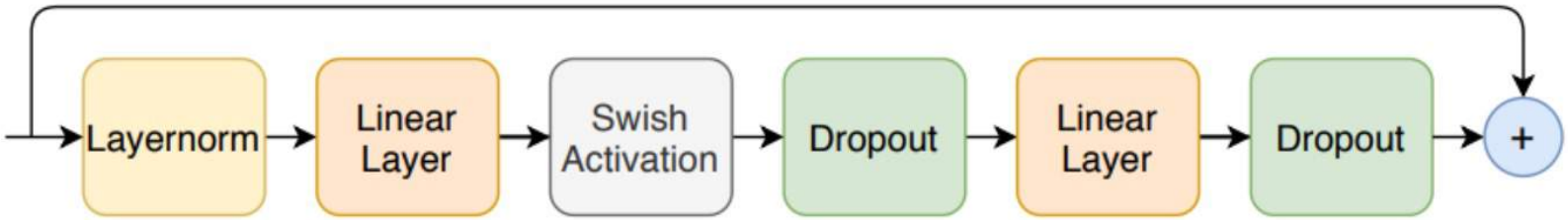


Conformer: Convolution Module

- Pointwise Convolution
- Glu Acctivation
- Depthwise Convolution
- Swish Activation
- Batch Normalization

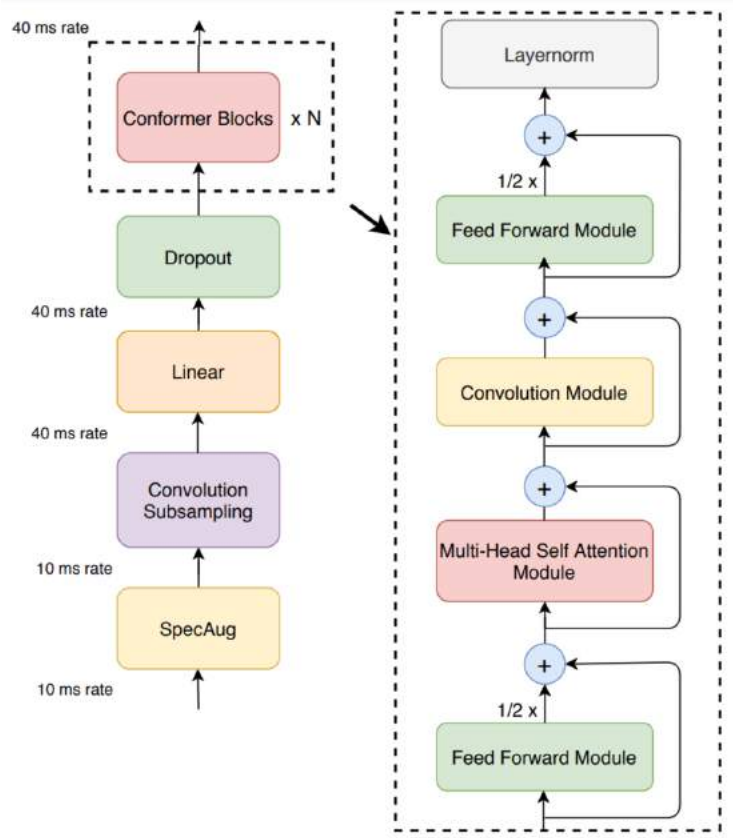


Conformer: Feed Forward Module



Conformer

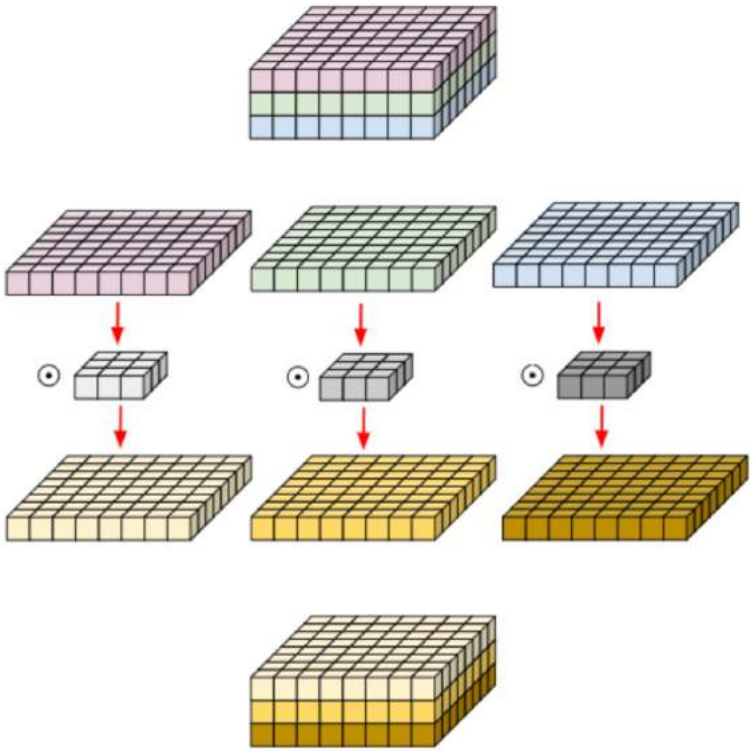
$$\tilde{x}_i = x_i + \frac{1}{2}\mathrm{FFN}(x_i)$$
$$x'_i = \tilde{x}_i + \mathrm{MHSA}(\tilde{x}_i)$$
$$x''_i = x'_i + \mathrm{Conv}(x'_i)$$
$$y_i = \mathrm{Layernorm}(x''_i + \frac{1}{2}\mathrm{FFN}(x''_i))$$

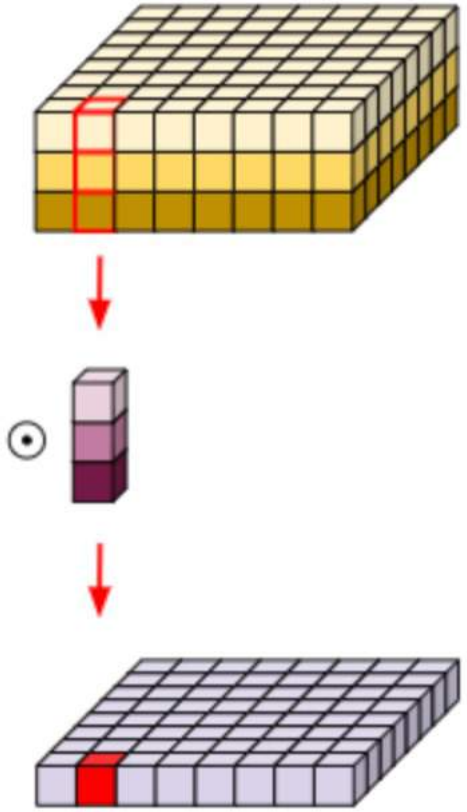


espnet2/asr/encoder/conformer_encoder.py
espnet/nets/pytorch_backend/conformer/encoder_layer.py
espnet/nets/pytorch_backend/conformer/convolution.py

Depth/point-wise convolution

- depthwise vs pointwise





Activation & Norm

- GLU Activation

$$h_l(\mathbf{X}) = (\mathbf{X} * \mathbf{W} + \mathbf{b}) \otimes \sigma(\mathbf{X} * \mathbf{V} + \mathbf{c})$$

```python
class GLU(nn.Module):
    def __init__(self, in_size):
        super().__init__()
        self.linear1 = nn.Linear(in_size, in_size)
        self.linear2 = nn.Linear(in_size, in_size)

    def forward(self, X):
        return self.linear1(X) * self.linear2(X).sigmoid()
```

```python
class FastGLU(nn.Module):
    def __init__(self, in_size):
        super().__init__()
        self.in_size = in_size
        self.linear = nn.Linear(in_size, in_size*2)

    def forward(self, X):
        out = self.linear(X)
        return out[:, :self.in_size] * out[:, self.in_size:].sigmoid()
```

https://medium.com/deeplearningmadeeasy/glu-gated-linear-unit-21e71cd52081

- Swish Activation

Normalization

- LayerNormalization
- BatchNormalization

Conformer: Ablation Study

Table 3: ***Disentangling Conformer.*** *Starting from a Conformer block, we remove its features and move towards a vanilla Transformer block: (1) replacing SWISH with ReLU; (2) removing the convolution sub-block; (3) replacing the Macaron-style FFN pairs with a single FFN; (4) replacing self-attention with relative positional embedding [20] with a vanilla self-attention layer [6]. All ablation study results are evaluated without the external LM.*

Model Architecture	dev clean	dev other	test clean	test other
Conformer Model	1.9	4.4	2.1	4.3
– SWISH + ReLU	1.9	4.4	2.0	4.5
– **Convolution Block**	2.1	4.8	2.1	4.9
– Macaron FFN	2.1	5.1	2.1	5.0
– Relative Pos. Emb.	2.3	5.8	2.4	5.6

Table 4: ***Ablation study of Conformer Attention Convolution Blocks.*** *Varying the combination of the convolution block with the multi-headed self attention: (1) Conformer architecture; (2) Using Lightweight convolutions instead of depthwise convolution in the convolution block in Conformer; (3) Convolution before multi-headed self attention; (4) Convolution and MHSA in parallel with their output concatenated [17].*

Model Architecture	dev clean	dev other
Conformer	1.9	4.4
– Depthwise conv + Lightweight convolution	2.0	4.8
Convolution block before MHSA	1.9	4.5
Parallel MHSA and Convolution	2.0	4.9

Table 5: ***Ablation study of Macaron-net Feed Forward modules.*** *Ablating the differences between the Conformer feed forward module with that of a single FFN used in Transformer models: (1) Conformer; (2) Conformer with full-step residuals in Feed forward modules; (3) replacing the Macaron-style FFN pair with a single FFN.*

Model Architecture	dev clean	dev other	test clean	test other
Conformer	1.9	4.4	2.1	4.3
Single FFN	1.9	4.5	2.1	4.5
Full step residuals	1.9	4.5	2.1	4.5

Table 6: *Ablation study on the attention heads in multi-headed self attention.*

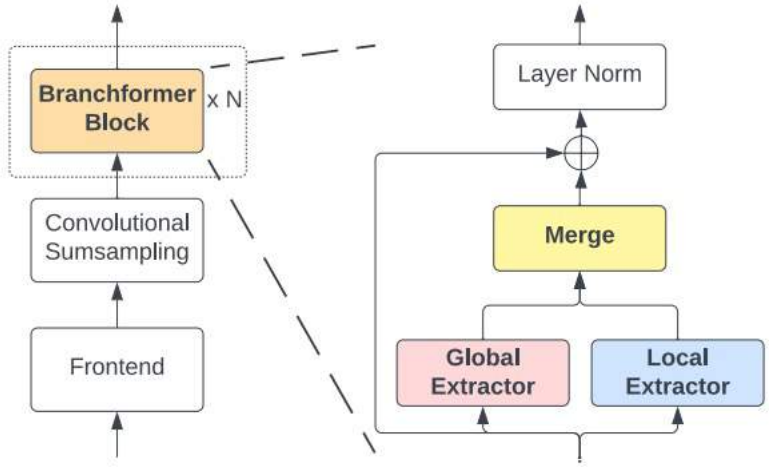
Attention Heads	Dim per Head	dev clean	dev other	test clean	test other
4	128	1.9	4.6	2.0	4.5
8	64	1.9	4.4	2.1	4.3
16	32	2.0	4.3	2.2	4.4
32	16	1.9	4.4	2.1	4.5

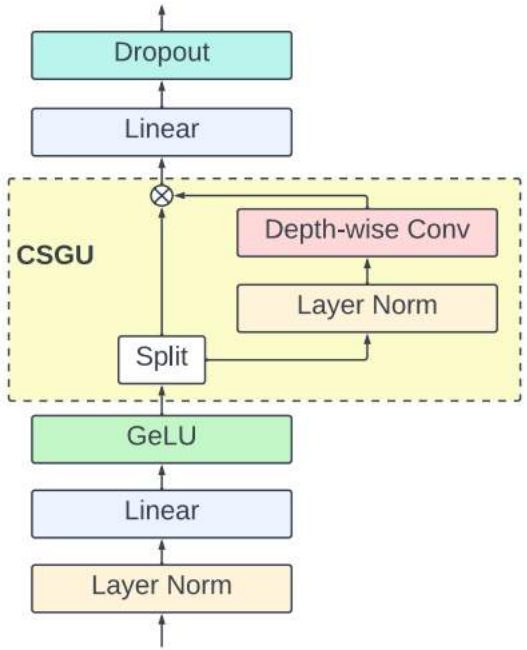
Table 7: *Ablation study on depthwise convolution kernel sizes.*

Kernel size	dev clean	dev other	test clean	test other
3	1.88	4.41	1.99	4.39
7	1.88	4.30	2.02	4.44
17	1.87	4.31	2.04	4.38
32	1.83	4.30	2.03	4.29
65	1.89	4.47	1.98	4.46

e-branchformer

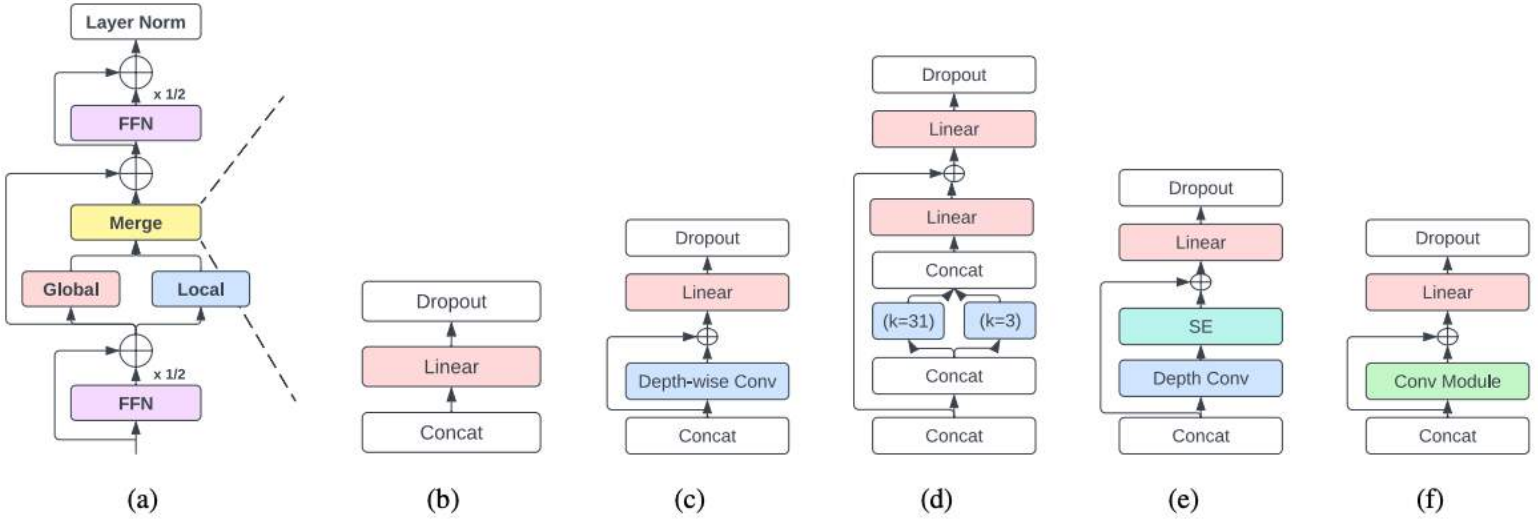
- E-Branchformer: Branchformer with Enhanced Merging for Speech Recognition," SLT, 2022
- Branchformer





e-branchformer

- Merge



Using Whisper as Pretrained Model

- Building robust Korean speech recognition model by fine-tuning large pretrained model (말소리와 음성과학, 2023)

모델		파라미터 개수	모델 크기
Whisper	large-v2	1.54 B	6.17 GB
	medium	762.32 M	3.05 GB
	small	240.58 M	962.33 MB
	base	71.83 M	287.3 MB
	tiny	37.18 M	148.74 MB
Baseline Conformer		116.15 M	464.59 MB

테스트 데이터셋	large-v2	medium	base	tiny
kspon-evalclean	10.83	11.94	22.85	33.57
kspon-evalother	10.83	11.96	21.12	30.90
spon-bcast	13.14	14.03	21.30	28.49
spon-debate	10.25	11.48	18.65	25.97
spon-present	11.14	11.49	16.76	21.91
libri-testother	6.31	6.74	11.95	15.79

Whisper 적응 훈련

Table 5. CER (%) for each size of the Whisper model after fine-tuning and error reduction ratio (ERR) compared to zero-shot inference (%)

테스트 데이터셋	CER (%)			ERR (%)		
	medi	base	tiny	medi	base	tiny
kspon-evalclean	7.61	12.99	15.94	36.26	43.15	52.52
kspon-evalother	8.36	13.68	17.67	30.10	35.23	42.82
spon-bcast	14.75	20.91	29.44	–5.13	1.83	–3.33
spon-debate	7.42	16.01	22.64	35.37	14.16	12.82
spon-present	4.74	9.35	12.87	58.75	44.21	41.26

Table 7. CER for full trained baseline transformer model and comparison to fine-tuned Whiser medium model (CER, %)

테스트 데이터셋	1,000시간 전체학습	6,500시간 전체학습	7,500시간 전체학습	파인튜닝 medium 모델
kspon-evalclean	8.80	9.84	7.60	7.61
kspon-evalother	9.74	9.44	8.29	8.36
spon-bcast	27.04	11.85	11.53	14.75
spon-debate	13.43	7.12	6.85	7.42
spon-present	6.49	7.81	6.98	4.74

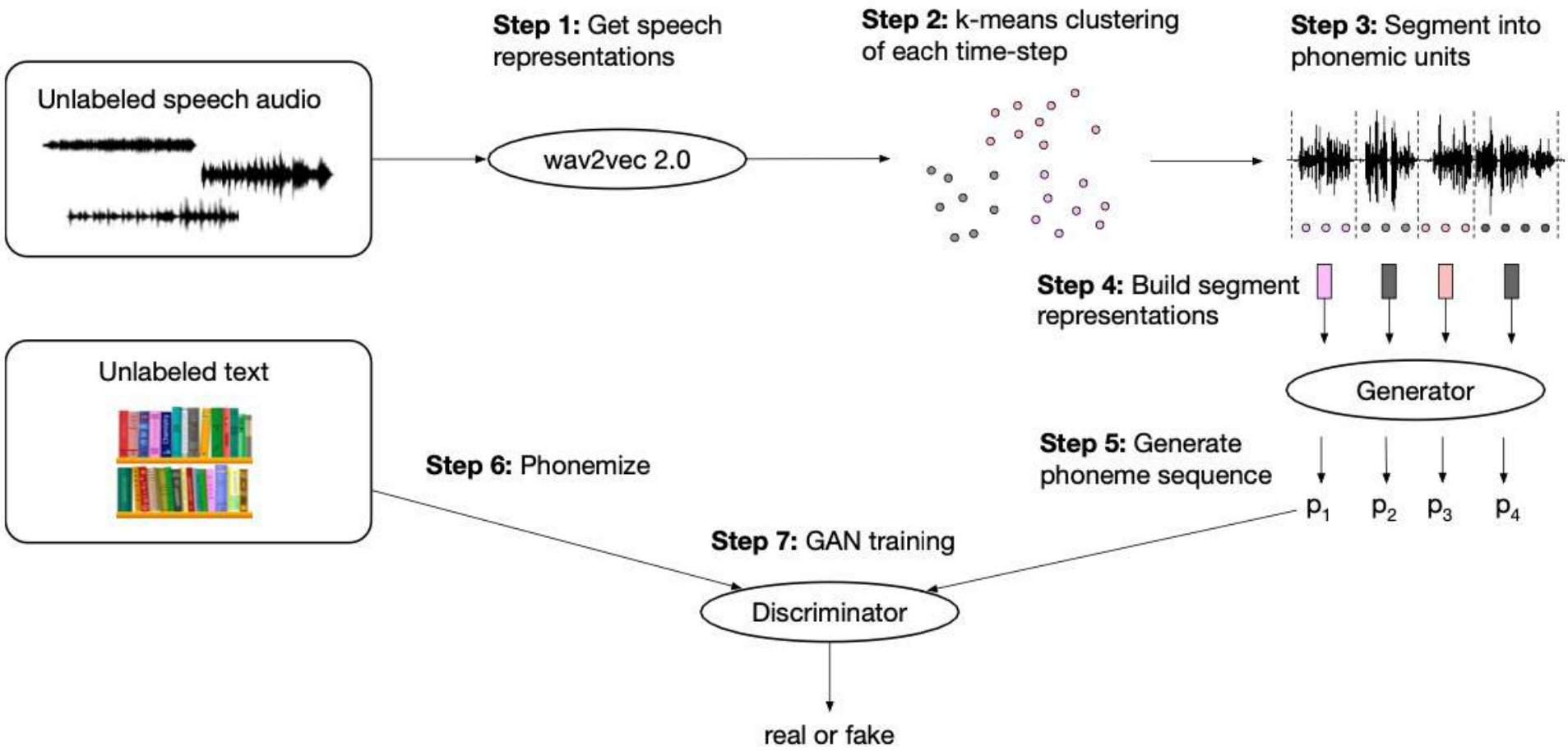
Ongoing Researches

- Semi/Un-Supervised Training
  - Training without labeled data
  - wav2vec 2.0, …
- Data augmentation
  - Generative models
- Transfer learning
  - Domain transfer
- Domain adaptation
- Streaming Transformer

Unsupervised Speech Recognition

- NeurIPS, 2021, Alexei Baevski, Wei-Ning Hsu, Alexis CONNEAU, Michael Auli
- Motivation
  - 7,000 Languages > 125 STT Lang
  - Unsupervised NMT: Conneau et al 2018, …
  - A Framework for unsupervised learning of speech recognition

Architecture



Unsupervised Learning

- Objective
  - GAN loss
  - Gradient penalty
  - Segment smoothness penalty
  - Phoneme diversity loss

$$\min_{\mathcal{G}} \max_{\mathcal{C}} \mathop{\mathbb{E}}_{P^r \sim \mathcal{P}^r} [\log \mathcal{C}(P^r)] - \mathop{\mathbb{E}}_{S \sim \mathcal{S}} [\log (1 - \mathcal{C}(\mathcal{G}(S)))] - \lambda \mathcal{L}_{gp} + \gamma \mathcal{L}_{sp} + \eta \mathcal{L}_{pd}$$

$$\mathcal{L}_{gp} = \mathop{\mathbb{E}}_{\tilde{P} \sim \tilde{\mathcal{P}}} \left[\left(\|\nabla \mathcal{C}(\tilde{P})\| - 1 \right)^2 \right]$$

$$\mathcal{L}_{sp} = \sum_{(p_t, p_{t+1}) \in \mathcal{G}(S)} \|p_t - p_{t+1}\|^2 \qquad \mathcal{L}_{pd} = \frac{1}{|B|} \sum_{S \in B} -H_{\mathcal{G}}(\mathcal{G}(S))$$

Experimental Results

- Comparison to Supervised Speech Recognition on Librispeech

Model	Unlabeled data	LM	dev		test	
			clean	other	clean	other
960h - Supervised learning						
DeepSpeech 2 [Amodei et al., 2016]	-	5-gram	-	-	5.33	13.25
Fully Conv [Zeghidour et al., 2018]	-	ConvLM	3.08	9.94	3.26	10.47
TDNN+Kaldi [Xu et al., 2018]	-	4-gram	2.71	7.37	3.12	7.63
SpecAugment [Park et al., 2019]	-	RNN	-	-	2.5	5.8
ContextNet [Han et al., 2020]	-	LSTM	1.9	3.9	1.9	4.1
Conformer [Gulati et al., 2020]	-	LSTM	2.1	4.3	1.9	3.9
960h - Self and semi-supervised learning						
Transf. + PL [Synnaeve et al., 2020]	LL-60k	CLM+Transf.	2.00	3.65	2.09	4.11
IPL [Xu et al., 2020b]	LL-60k	4-gram+Transf.	1.85	3.26	2.10	4.01
NST [Park et al., 2020]	LL-60k	LSTM	1.6	3.4	1.7	3.4
wav2vec 2.0 [Baevski et al., 2020c]	LL-60k	Transf.	1.6	3.0	1.8	3.3
wav2vec 2.0 + NST [Zhang et al., 2020b]	LL-60k	LSTM	1.3	2.6	1.4	2.6
Unsupervised learning						
wav2vec-U LARGE	LL-60k	4-gram	13.3	15.1	13.8	18.0
wav2vec-U LARGE + ST	LL-60k	4-gram	3.4	6.0	3.8	6.5
	LL-60k	Transf.	3.2	5.5	3.4	5.9

Discussion and Q&A