

# Audio\_2

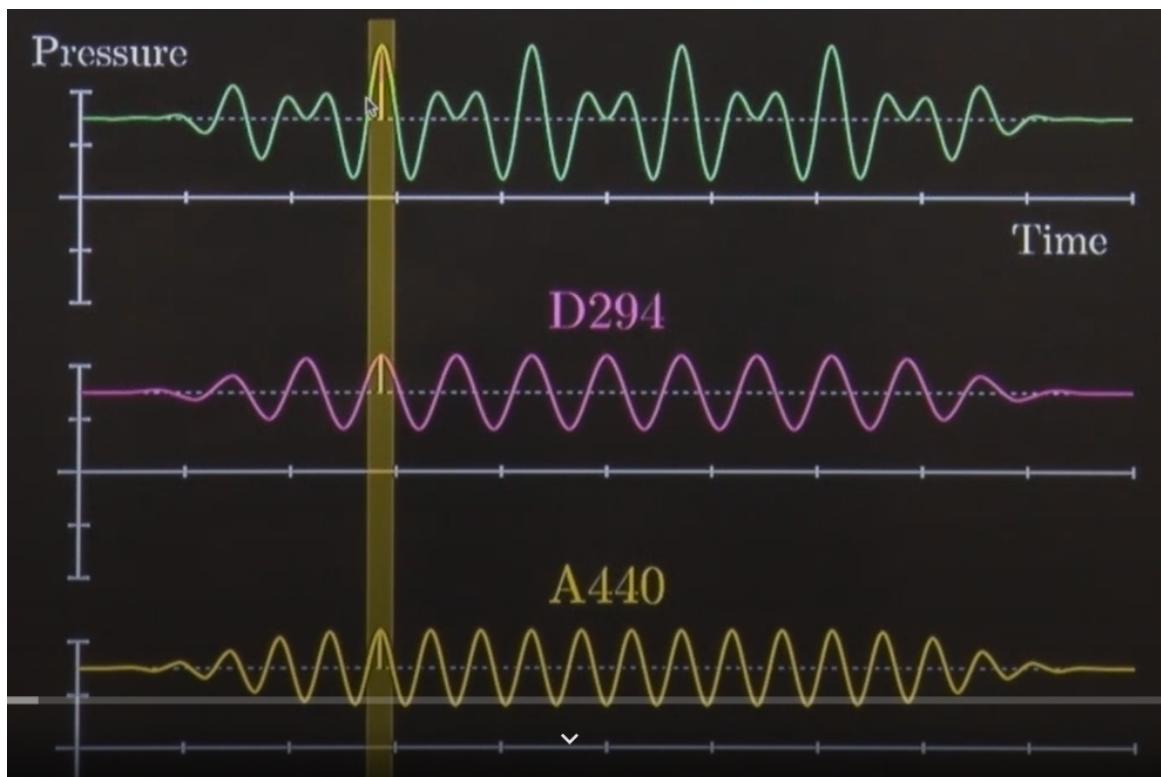
소리에서 얻을 수 있는 물리량

- Amplitude(Intensity) : 진폭
- Frequency : 주파수
- Phase(Degress of displacement) : 위상

## complex wave

우리가 사용하는 대부분의 소리들은 복합파이다.

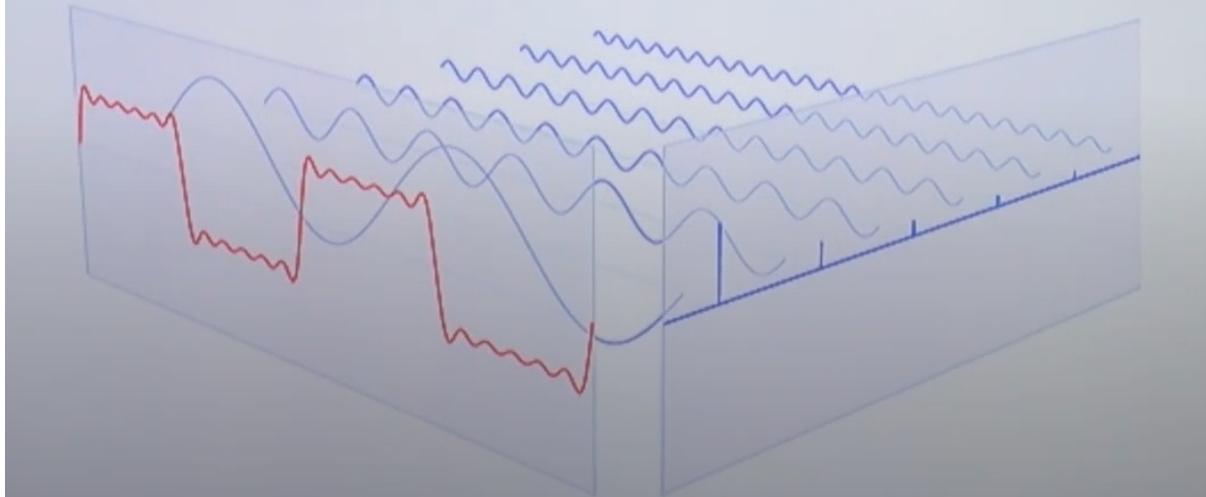
복합파는 복수의 서로 다른 정형파들의 합으로 이루어진 파형(Wave)이다.



## 푸리에변환(Fourier transform)

임의의 입력 신호를 다양한 주파수를 갖는 주기함수(복수 지수함수)들의 합으로 분해하여 표현하는 것

$$A_k = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) \exp\left(-i \cdot 2\pi \frac{k}{T} t\right) dt$$



정현파는 복소 주기함수이고, complex wave가 정현파가 모인 것이라고 했는데 complex wave는 주기함수(복소 지수함수)이다. 여기서 궁금한 점이 생길 수 있다. 이 관계를 나타내 주는 것이 **오일러 함수**이다.

지수함수와 주기함수의 관계가

$$e^{i\theta} = \cos \theta + i \sin \theta$$

**e**가 지수함수, **cos**과 **sin**이 주기함수라고 생각하면 이해가 편할 것이다.

푸리에 변환을 하면 실수부와 허수부를 가지는 복소수가 얻어진다. 보통 **실수부가 Frequency**영역을 나타내고, **허수부가 phase**를 나타낸다. 복소수의 절대값은 실수만 남게 되는 **Spectrum magnitude**가 있다.

## Audio는 채널이 어떤게 될까??

## 1. 1-D CNN

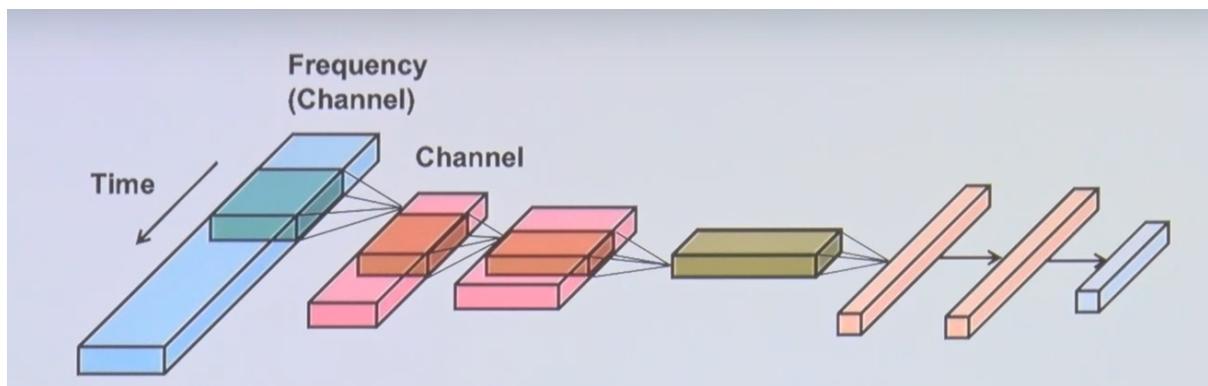
스팩토그램이 input으로 들어가니까 Convolution filter의 크기가 **frequency** 영역대는 고정되어 있으며, **Time**에 따라 진행된다

장점

- 적은 수의 파라미터
- 빠른 학습
- 작은 데이터셋

단점

- pitch shifting은 변할 수 없다

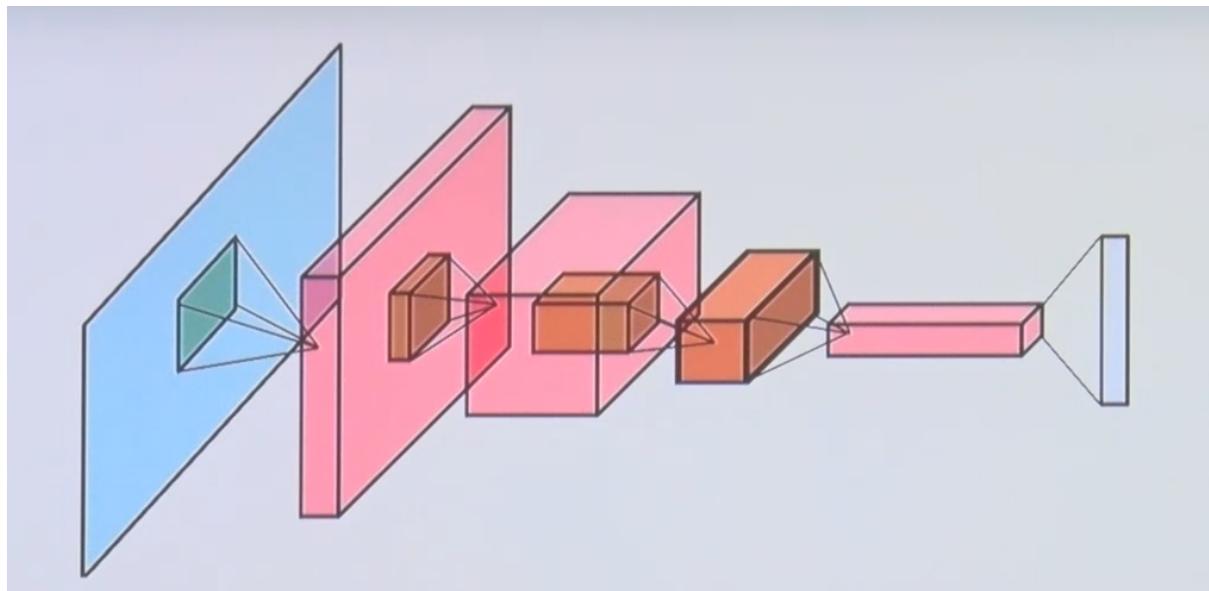


## 2. 2-D CNN

Convolution filter의 크기가 **frequency**과 **Time**에 따라 진행된다.

장점

- 두 가지 영역에서 pattern을 찾는다.
- 1D CNN에 비해 큰 데이터셋에서 높은 성능을 가짐



## Sample CNN

Sample level CNN의 가장 큰 특징은 **input데이터를 waveform 그 자체로 사용**할 수 있다 는 점

장점

- ‘phase-invariant’ representation을 반영한다
- 커널이 input signal에 대한 spectral bandwidth를 계산해준다

### ▼ phase invariant

일반적으로 **classification**을 하면 **phase-invariant**를 고려하지 않고 학습을 진행하게 된다. **Sample CNN**을 활용할 경우 이를 반영하게 되는데, 이는 classification이 아닌 **화자인식**과 같은 사람의 **음색**을 고려해야하는 문제에서 유용하게 사용된다

(우리 컴퓨터 사용하면 좋을 듯)

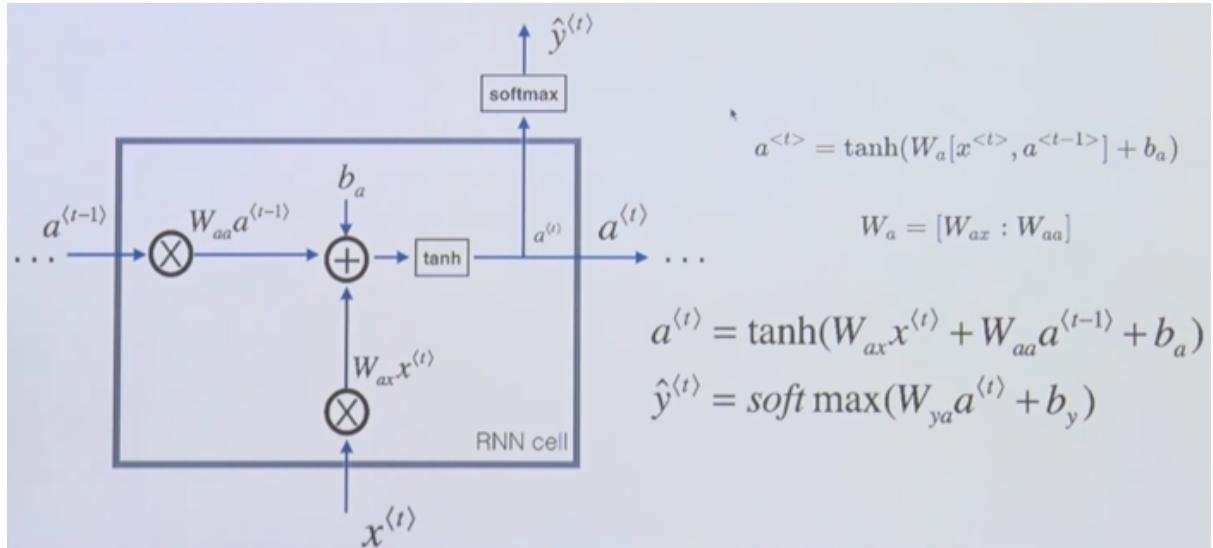
## RNN

RNN은 Hidden State를 유지하면서 이전 출력을 입력으로 사용할 수 있는 신경망

장점

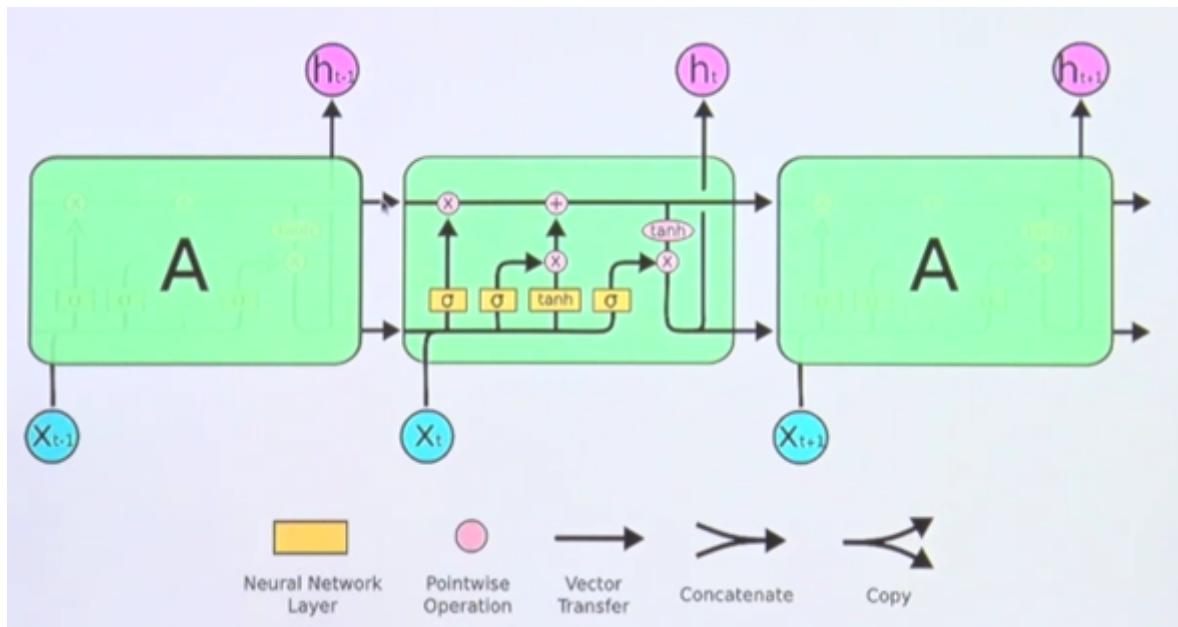
- 어떠한 input length든 커버 가능
- 입력 크기에 따라 모델 Size가 증가하지 않음
- Historical Information을 잘 활용

- 시간축에 따른 Wights Sharing이 진행

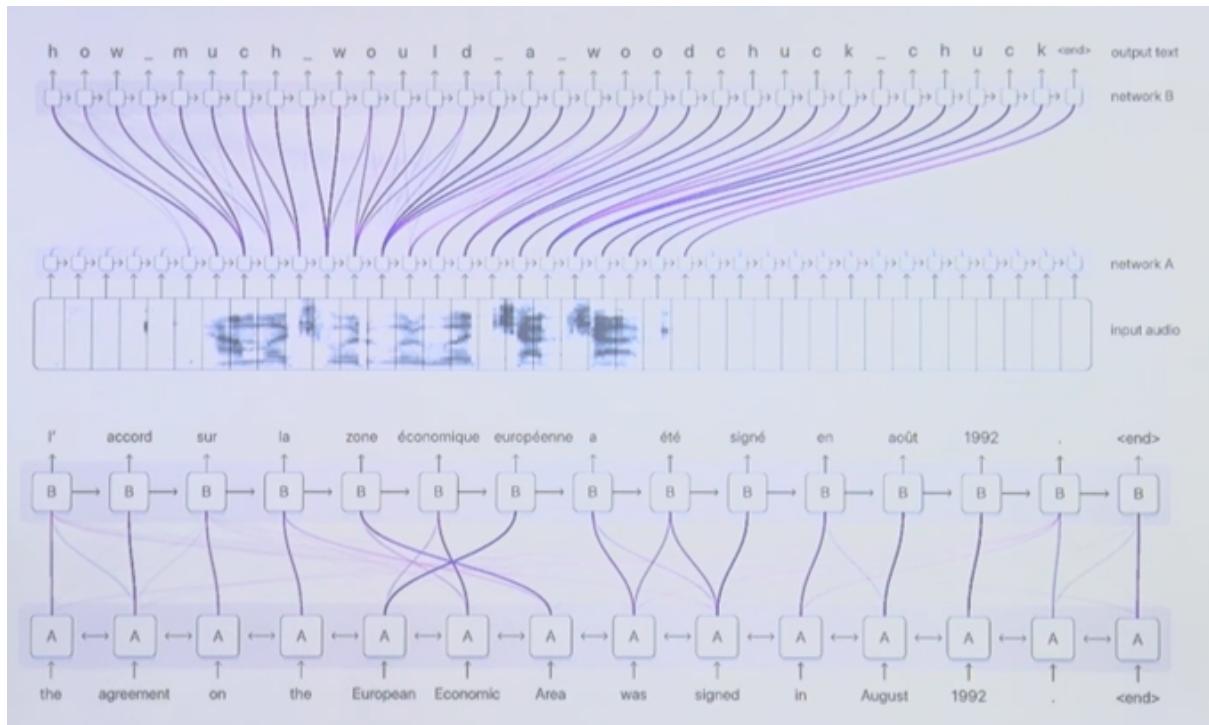


## LSTM

GRU 및 LSTM은 기존 RNN에서 vanishing Gradient를 처리하며 LSTM은 GRU의 일반화된 모델



## Attention

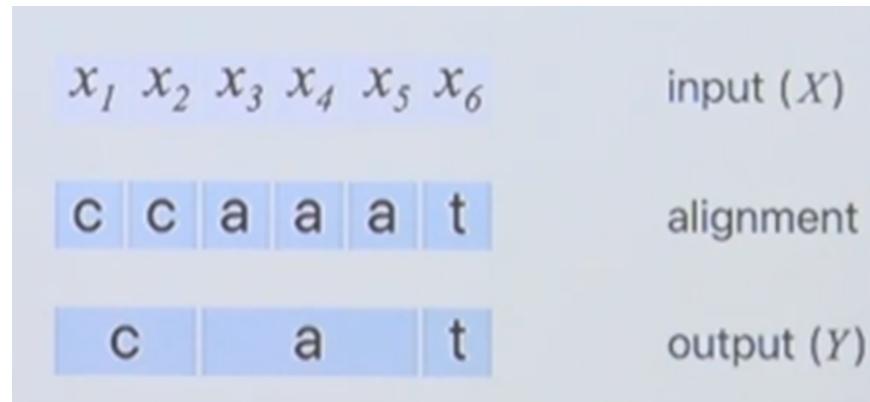


RNN의 encoding 방식에서는 계속 마지막 hidden state까지 학습하면서 연산을 해야했다. 이 문제를 해결하는 것이 Attention이다. 이는 input source와 hidden state의 관계를 학습시키는 추가적인 network를 만든다. 이 attention은 output에 의해 weight를 학습하게 된다.

## CTC(Connectionist Temporal Classification)

일반적인 Speech Recognition에서 우리는 데이터셋으로 오디오 클립과 Transcript를 받게 된다. 하지만 우리는 어떤 단어의 Character가 Audio와 Alignment가 맞는지 알 수 없다. 이러한 Alignment 없이 어떤 Audio와 Text 사이의 규칙을 정의하기 힘들다. 또한 사람들의 언어 사용을 하는 것은 다양하기 때문에 단일한 Rule로 그들을 정의하기는 쉽지 않다.

CTC는 input과 output 사이의 정확한 Alignment가 labeling이 되어있지 않은 데이터에 대해 최대한으로 input에 대해 output의 확률값(Alignment)을 맞춰보자는 것. **CTC는 둘 사이의 가능한 모든 Alignment의 가능성을 합산하여 작용한다.** 이 말은 즉 input도 output의 가능한 Align을 모두 뽑아 Marginalize하자라는 뜻이다.

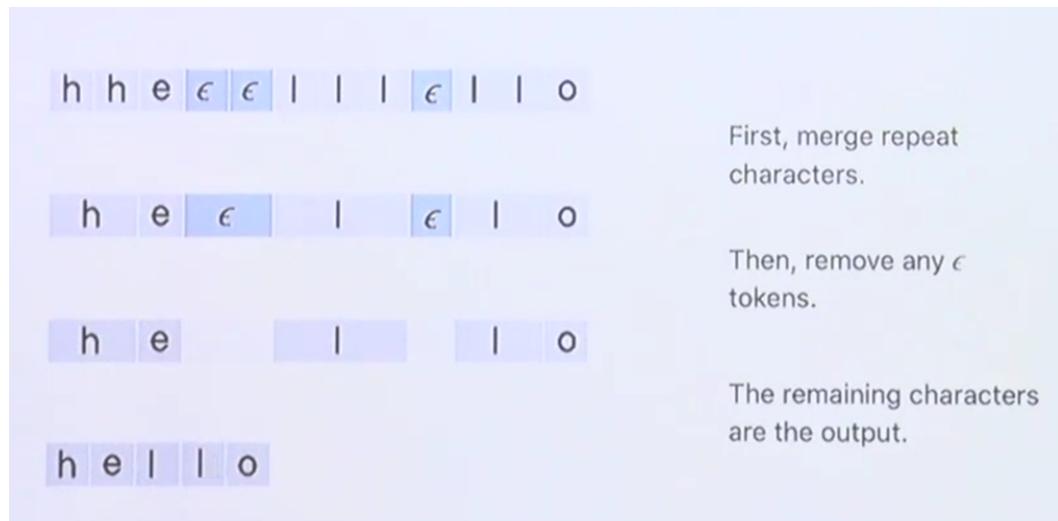


이러한 접근은 2가지 문제가 있다

**모든  $x$ 에  $y$ 가 할당되는 것은 옳지 않다!!** 예를 들어 Silence(쉼음)가 존재하는 부분 또한 글자가 할당되는 문제가 발생할 수 있다.

이러한 문제를 해결하기 위해 CTC는 허용된 output에 새로운 토큰을 도입한다. 이 토큰을 **epsilon**이라고 한다.

CTC의 경우 input과 같은 길이로 Alignment를 진행한다. 그 다음에 Y로 Mapping하는 단계에서 epsilon을 제거하게 된다.



CTC Alignment에 재밌는 속성이 있다.

하나의 input에서 next input으로 진행하면서 output을 동일하게 유지하거나 next output으로 assign한다. (c다음에 c나 a는 올 수 있지만 t는 올 수 없다!!)

순서가 중요하다는 소리!!

두번째 요소는 x와 y의 정렬이 many-to-one 함수라는 것이다. 하나 이상의 input이나 output에 들어갈 수 있지만 반대는 성립하지 않는다.

Valid Alignments	Invalid Alignments	
$\epsilon \text{ } c \text{ } c \text{ } \epsilon \text{ } a \text{ } t$	$c \text{ } \epsilon \text{ } \underline{c} \text{ } \epsilon \text{ } a \text{ } t$	corresponds to $Y = [c, c, a, t]$
$c \text{ } c \text{ } a \text{ } a \text{ } t \text{ } t$	$c \text{ } c \text{ } a \text{ } a \text{ } t \text{ } \underline{\quad}$	has length 5
$c \text{ } a \text{ } \epsilon \text{ } \epsilon \text{ } \epsilon \text{ } t$	$c \text{ } \epsilon \text{ } \epsilon \text{ } \epsilon \text{ }   t \text{ } t$	missing the 'a'

Invalid Alignments하다고 여겨지는 것은 불가능한 것.



1. input sequence의 **spectrogram**으로 시작하게 된다.
2. input은 RNN계열의 Layer에 input으로 들어가게 된다.
3. Network는  $p_t(\alpha|X)$ 를 반환한다. 이  $\in$  output인 {h,e,l,o,\epsilon}의 각 input time step별의 확률분포이다.  
(매 타임마다 모든 알파벳에 대한 확률분포를 가지게 된다!)
4. 각 Time step별로 output을 가지고, 우리는 다른 시퀀스간의 확률을 구할 수 있다.
5. Alignment에 대해 Marginalize를 진행하면, 우리는 output의 확률분포를 구할 수 있다.

CTC는 조건부 확률 분포이다  $p(Y|X)$

$(X, Y)$  [X: input(Spectrogram), Y: sentence transcript] 의 pair dataset에 대해 각각의 x의 input step을 따라가면서 step-by-step으로 single alignment의 확률을 계산한다. 그 다음 Validation alignment에 대해 Marginalize를 진행한다.

CTC는 Time Step별 확률 분포를 얻기 위해, 그리고 input sequence의 context를 고려하기 위해 RNN기반의 모델을 통해 학습한다.

하지만 RNN은 input sequence가 fixed size splice 때 더 활용하기 좋다.

$$\hat{\theta} = \arg \max_{\theta} \prod_{i=1}^N p(z^{(i)} | x^{(i)}; \theta)$$
$$p(z | x; \theta) = \sum_{\pi \in B^{-1}(z)} p(\pi | x; \theta)$$

$$B(A\_A\_\underline{AAAA}\_BBBCCCC) = B(A\_\underline{A}\_BBBB\_\underline{CC}) = AABC$$

위의 식을 보면

$L$ 을 label set,  $L'$ 를 epsilon이 있는 label set으로 둔다.

길이가 T인 시퀀스의 가능한 경로 집합(valid path)  $L'T$ 를  $\pi$ 로 칭하겠다(가능한  $\pi$ 에 대해 확률값을 sum으로 mapping하여 가장 probability가 높은 path를 찾고자 하는 것)

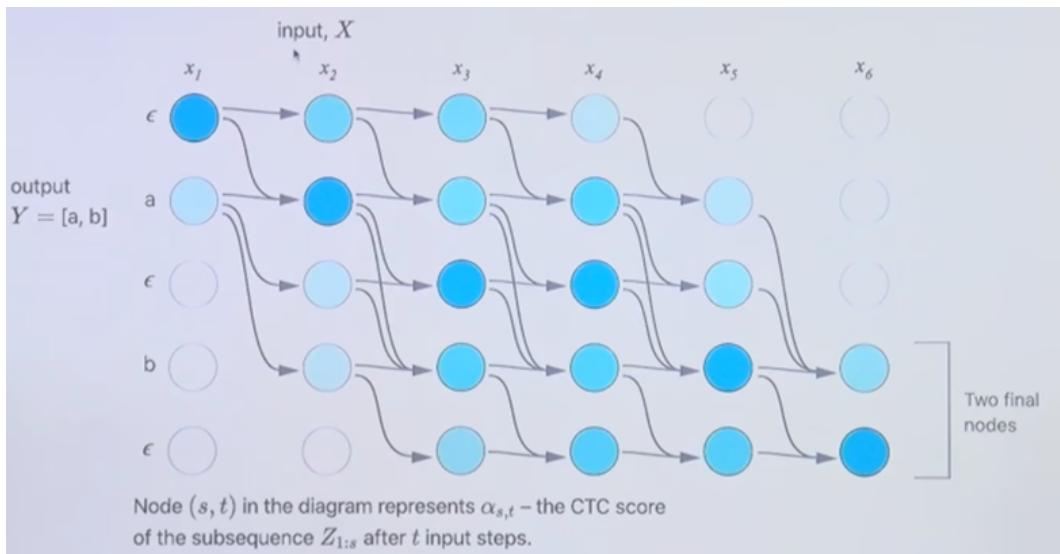
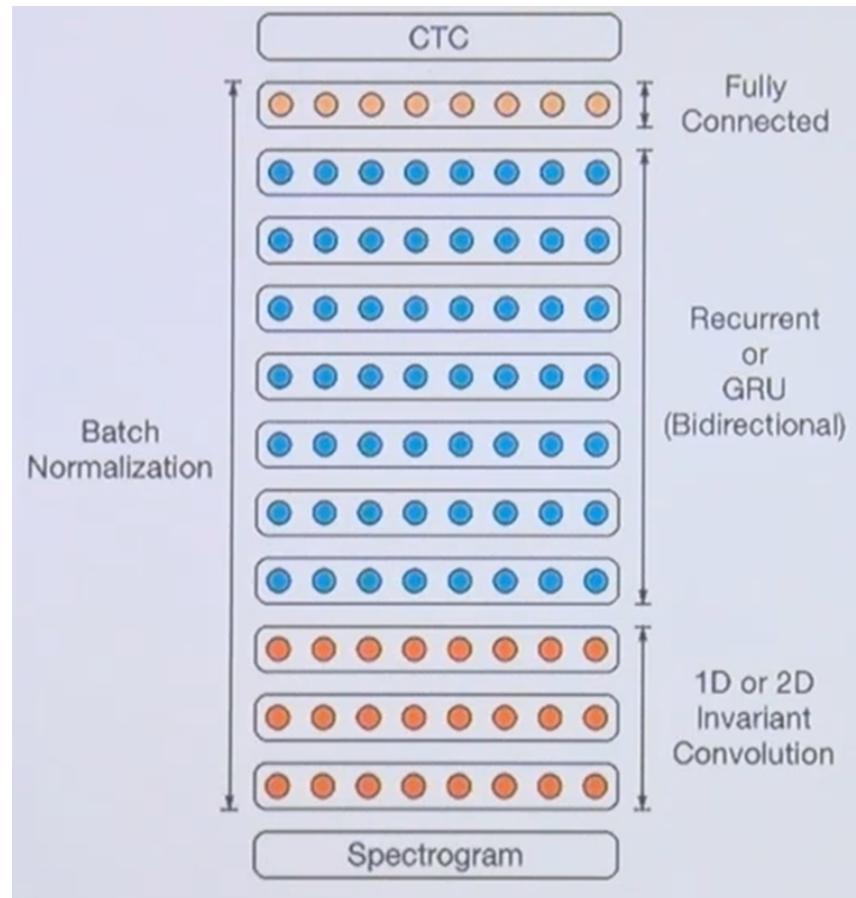
여기서 **B는 marginalize map** 이라고 생각하면 된다.

그래서 CTC는 가능한 경로 집합을 뽑기위해 확률값을 활용하는 loss function이라고 생각하면 된다.

모든 가능한 path를 찾게 되면 input sequence가 길어지면 길어질수록 커버해야하는 알파벳의 영역이 커지면 커질수록 가능한 경우의 수가 많아지고 연산량이 커지는 문제가 있다.

Rnn 구조의 arg max를 찾을 때 가장 간단하다고 말한다. (하지만 이 방식은 좋은 방식이 아니다)

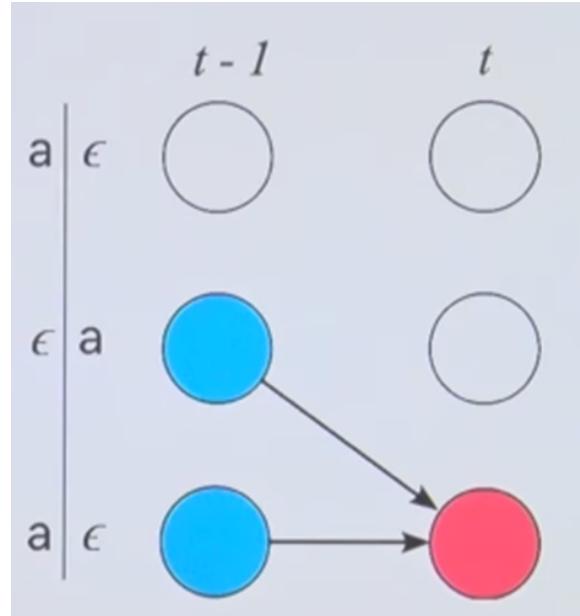
논문마지막에서는 이것을 beam search를 하는 아이디어를 제안한다.



모든 output sequence 앞에 epsilon이 있다고 가정한 그림을 보겠다.

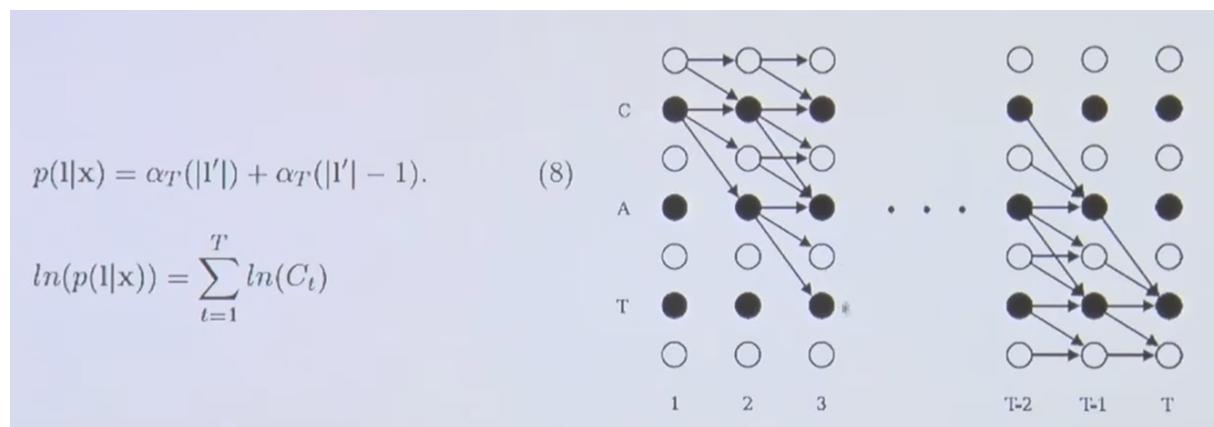
앞서 언급했던 재밌는 속성에 따라 시간의 순서가 중요하고, invalid Alignment에 따라 위와 같은 형태로 나오는 것을 알 수 있다.

output이 a,b이라고 했을 때, 모든 가능한 path으로 보지 않고 데이터의 형태로 봤을 때 가능한 4번째에서는 a가 나와야 한다는 것이다.



$$\alpha_{s,t} = (\alpha_{s-1,t-1} + \alpha_{s,t-1})$$

$t-1$  입력 단계 후 두 개의 유효한 하위 시퀀스의 CTC 확률.



연결이 반드시 왼쪽 위 끝의 blank 혹은 왼쪽 2번째 행의 c에서부터 우측 끝의 T나 오른쪽 아래 끝의 blank까지 연결되어야 한다.

그렇기 때문에 왼쪽 아래와 오른쪽 위의 노드들은 연결이 생길 수 없다.

## 전체 flow

CNN 쌓고 RNN 쌓아서 encoding이 잘된 애를 FC layer를 통과한 다음 거기에 맞춰 softmax를 취했더니 sequence별로 probability distribution이 나오고 이걸 CTC에 의해 relevant한 path를 찾을 수 있다.

## LAS(Listen, Attend and Spell)

이후 더욱 발전하기 위한 것으로 LAS가 나옴

CTC모델의 기본은 RNN기반의 Sequence모델을 여러층 쌓고, 인코더로 들어오는 입력 데 이터는 최정적으로 softmax를 통과해서 출력된다.

반면 LAS는 input X와 이전 Sequence를 Auto-regressive방식으로 Given으로 받고, 그 다음 Y를 예측한다.

### Attention

먼저 Listener는 Encoder로 음성 신호를 Feature Extraction하는 역할을 하고 있다. Listener는 BiLSTM을 Pyramid 형식으로 3개를 붙여 사용한다.

이를 pBLSTM 으로 부르고, 사용 이유는 pBLSTM 하나당 연산속도를 2배로 줄여준다고 한다.

결국 LAS는 encoder, decoder를 활용하는데 이때 사이에 Attention값을 활용하고, decoder의 output이 다음번의 decoder를 예측하는데 사용하는 Auto-regressive방식을 활용한다고 알면 될 것 같다!!!