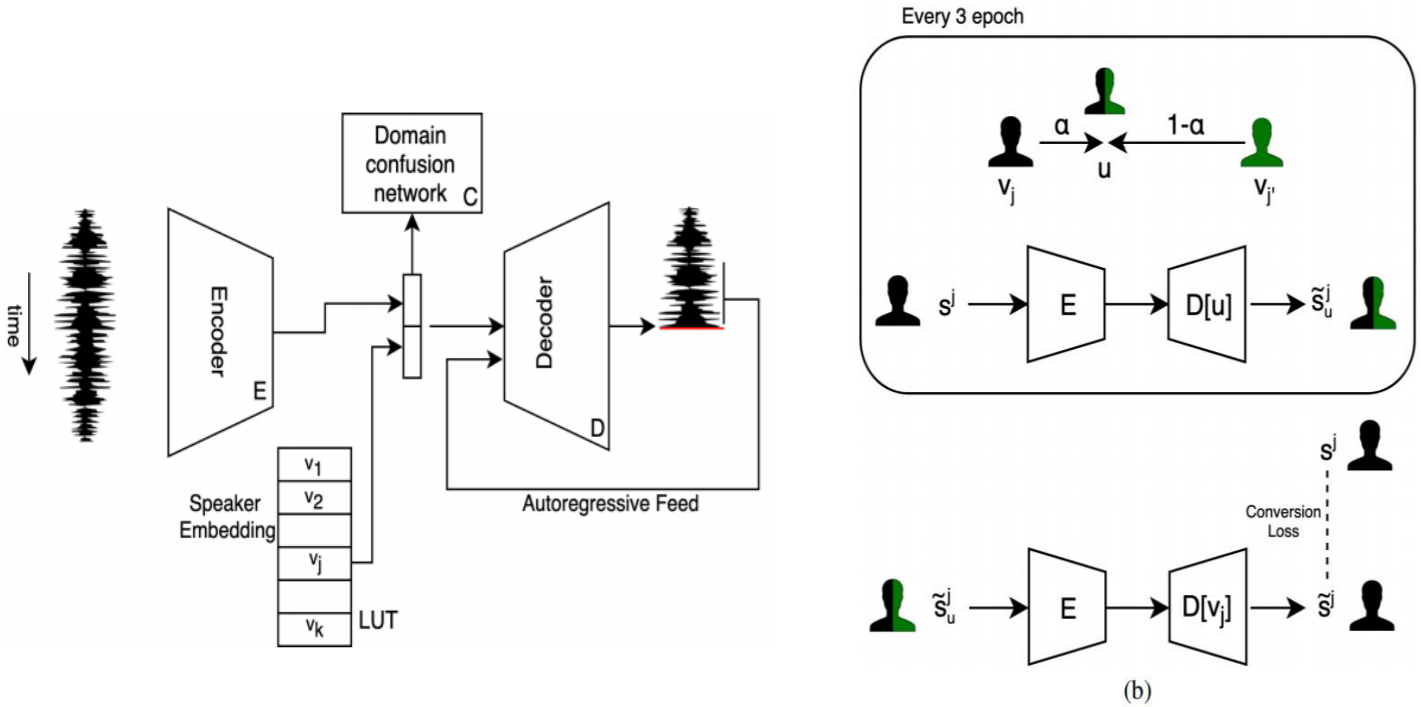
이거부터 알아야 해요~! 헷갈리면 요 페이지 와서 다시 보고 가기!



s^j : j번째 가수의 목소리(이것이 인풋!)

-> 이 때, 가수의 목소리는 원본 샘플일 수도 있고 3.2파트에 명시한 augmented된 샘플일 수도 있음.

E : 인코더(Encoder)

D[u] : 벡터 u에 대한 디코더(Decoder)

C : 가수 분류(판별) 네트워크

-> 이 때, C는 confusion network임. 이건 뭐냐면 하나의 가설만 세우고 돌리는 머신러닝이 아니라, 오류를 허용하는 multiple 가설로 돌리는 음성/자연어 처리 툴

$[v_j]$: j번째 가수에 대해 학습된 임베딩 벡터

LUT : 1,..., j, ..., k번째 가수 각각의 임베딩 벡터가 모두 저장된 look-up table

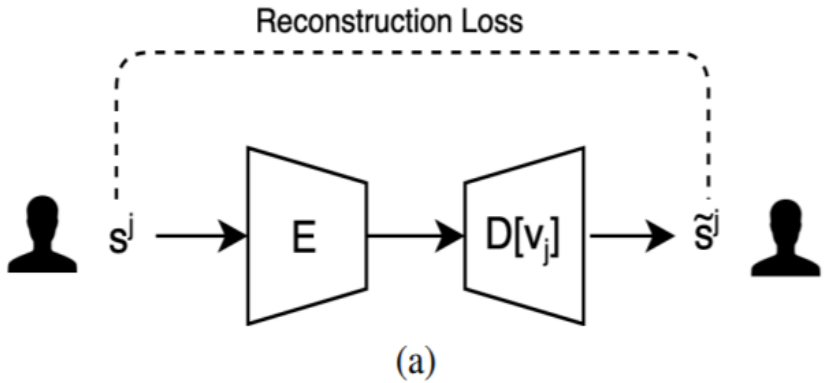
-> 이 때, 매번 training 반복할 때마다 1보다 큰 각 임베딩 벡터는 normalize된다.

학습이 어떻게 진행되냐!!?!?!?!?

"C는 input으로 받은 샘플 s가 들어왔을 때 어떤 가수인지 예측한다."
-> How? "E(s)"를 기반으로!
-> E(s): 인풋 s에 대해 인코더 E가 만든 latent vectors!

예측 모델링을 위한 training은 2단계로 이루어진다.

<1단계>



(a)

① C가 아래의 loss function을 최소화한다.

$$\sum_j \sum_{s^j} \mathcal{L}(C(E(s^j)), j)$$

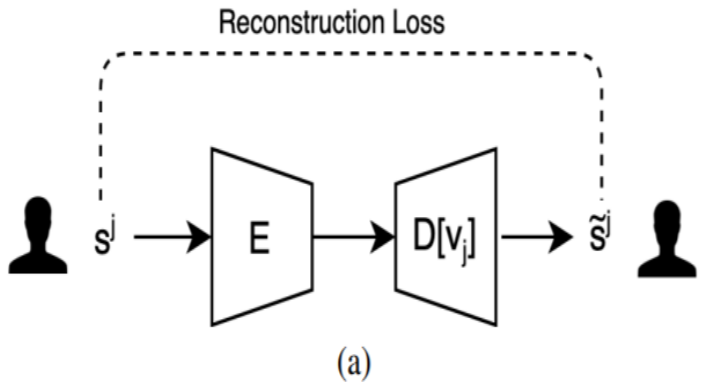
//이게 무슨 loss function인지 직관적으로 이해하자!//

-> 왼쪽 시그마 두 개 빼고 L(A,B)만 일단 보자.
-> L(A,B)에서 A는 위의 [ ] 에서 설명한 예측값.
즉, j번째 가수 목소리 샘플에 대해 인코더 E가 만든 latent vectors를 기반으로 C가 예측한 결과!
((말이 길어서 모르겠다면 A자리에 위치한 C(E(s))를 스스로 해석해보면 된다.))
-> 결국, 예측한 output과 실제 정답 j와의 차이를 전부 합한 값!

학습이 어떻게 진행되냐!!?!?!?!?

예측 모델링을 위한 training은 2단계로 이루어진다.

<1단계>



② $D[v_j] \circ E, j = 1, 2, \ldots, k$은 아래의 loss function을 최소화한다.

$$\sum_j \sum_{s^j} \mathcal{L}(D[v_j](E(s^j)), s^j) - \lambda \sum_j \sum_{s^j} \mathcal{L}(C(E(s^j)), j)$$

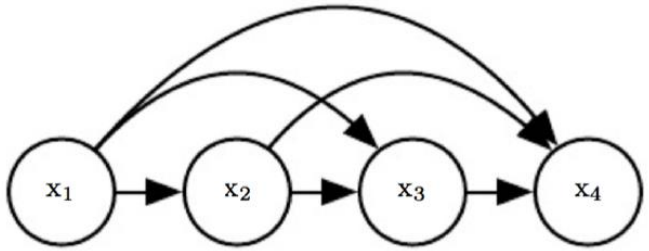
L(C, D): C와 D의 elementwise cross entropy
gamma: 가중치

$D[v_j]$는 autoregressive model로, 학습된 가수 벡터 v_j와 인코더 E의 output에 의해 도출되는 decoder.(완벽이해 x)

training동안 이 autoregressive model은 이전 time-step의 target output인 s^j 로 fed된다. (번역 자체가 어렵다.. fed?)
-> 이러한 training을 "teacher forcing" 이라고 한다.

잠깐잠깐... 무슨 말인지 아직 감이 잘 안 온다. autoregressive model과 teacher forcing을 알아보기로 하자!

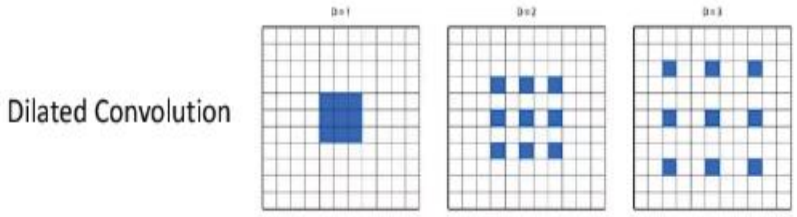
<Autoregressive model>

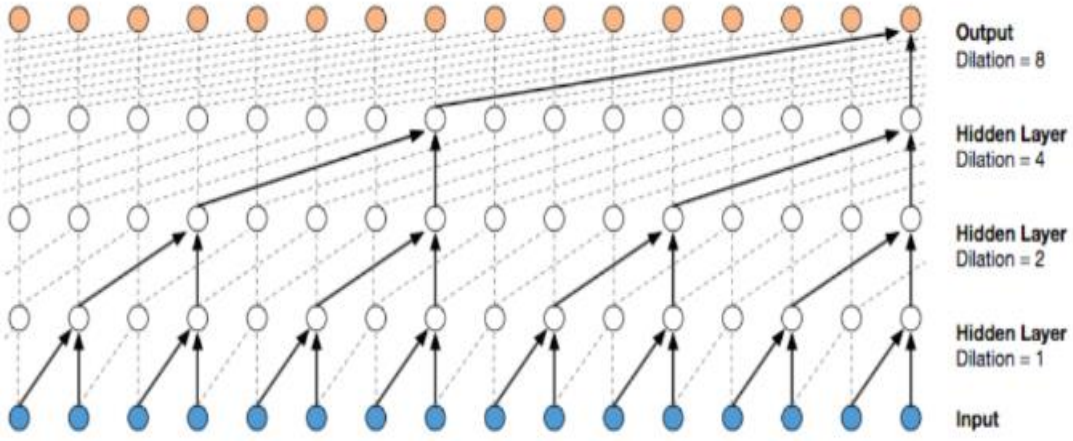


$$p(x) = \prod_i p(x_i | x_1, \ldots, x_{i-1})$$
$$= p(x_1)\, p(x_2|x_1) \ldots p(x_i|x_1, \ldots, x_{i-1})$$

Autoregressive model이란?

-> 자기 자신을 입력으로 하여 자기 자신을 예측하는 모형



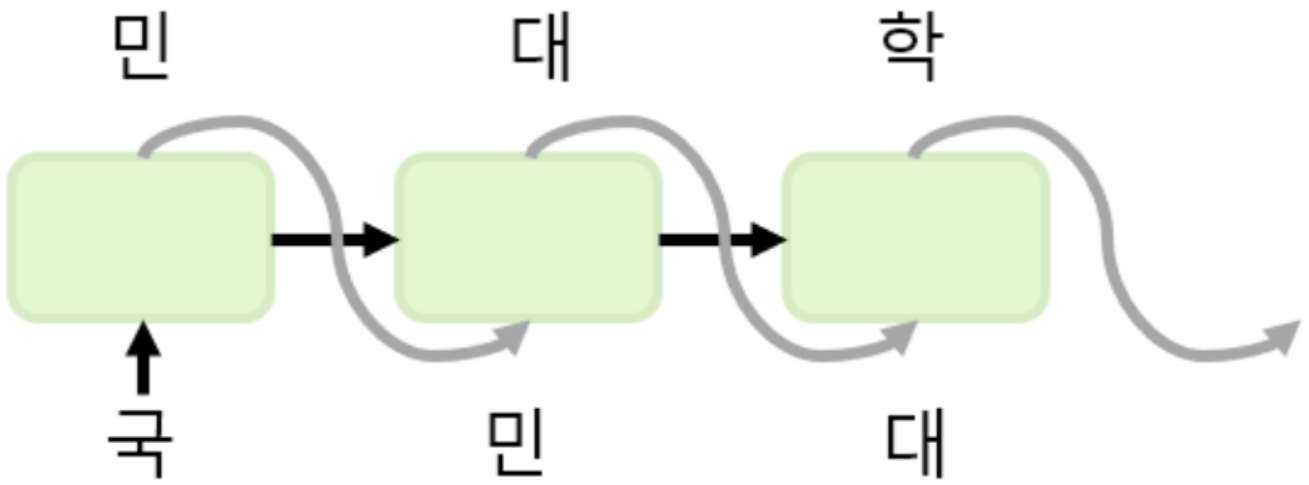


<WaveNet 굴러가는 모양>

Q. WaveNet이 Autoregressive model?

A. 현 시점 예측 때 과거 다양한 시점의 데이터를 사용하기 때문에 autoregressive model이라고 함!

<Teacher Forcing>



순환신경망에서 위와 같이 입력과 출력이 같은 경우! ex) 입력: (국)민대학... 출력: 민대학...

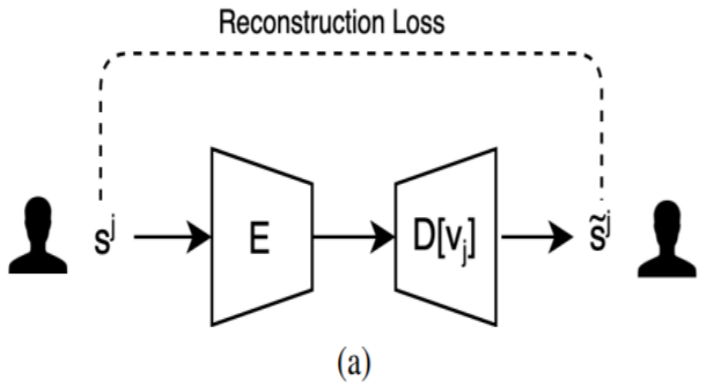
학습시키는 과정에서 뒤로 갈수록 실제 데이터와 입력이 달라져버리는 경우가 생긴다.
ex) 학습 도중 두 번째 노드에서 '민'을 입력으로 넣었을 때 '대'가 아닌 '의'를 예측했다면
그 다음 세 번째 노드에서 입력하게 되는 '의' 실제 데이터와 다르다.

이와 같은 문제를 막기 위해 입력에는 실제 데이터를 넣어주는 방법을 쓰고,
이것을 teacher forcing이라고 부른다!

학습이 어떻게 진행되냐!!?!?!?!?

예측 모델링을 위한 training은 2단계로 이루어진다.

<1단계>



② $D[v_j] \circ E, j = 1, 2, \ldots, k$은 아래의 loss function을 최소화한다.

$$\sum_j \sum_{s^j} \mathcal{L}(D[v_j](E(s^j)), s^j) - \lambda \sum_j \sum_{s^j} \mathcal{L}(C(E(s^j)), j)$$

L(C, D): C와 D의 elementwise cross entropy
gamma: 가중치

$D[v_j]$는 autoregressive model로, 학습된 가수 벡터 v_j와 인코더 E의 output에 의해 도출되는 decoder.

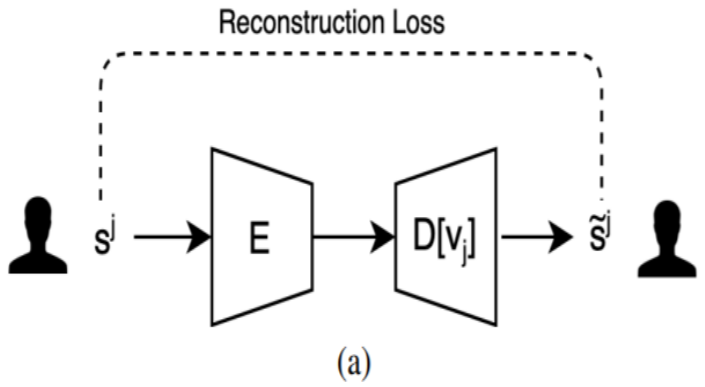
training동안 이 autoregressive model은 이전 time-step의 target output인 s^j로 fed된다.
-> 이러한 training을 "teacher forcing" 이라고 한다.

//아까 이해 못했던 이 부분을 다시 이해해보면!//
$D[v_j]$는 이전 시점의 결과인 s^j를 입력으로 받아 train하는 decoder이다! 이 때, 이 디코더는 인코더 E의 output을 전달받아 학습된 벡터 v_j에 대한 결과를 생성해낸다.

학습이 어떻게 진행되냐!!?!?!?!?

예측 모델링을 위한 training은 2단계로 이루어진다.

<1단계>



② $D[v_j] \circ E, j = 1, 2, \ldots, k$은 아래의 loss function을 최소화한다.

$$\sum_j \sum_{s^j} \mathcal{L}(D[v_j](E(s^j)), s^j) - \lambda \sum_j \sum_{s^j} \mathcal{L}(C(E(s^j)), j)$$

L(C, D): C와 D의 elementwise cross entropy
gamma: 가중치

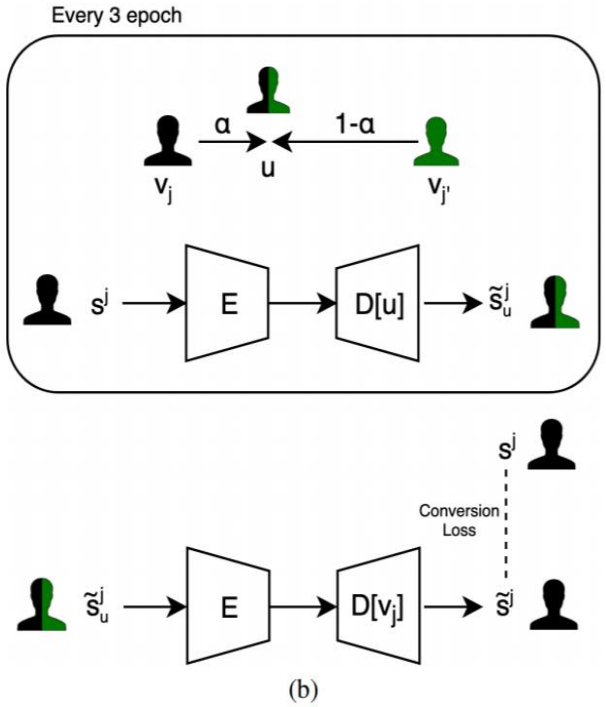
다시 돌아와서, 이렇게 train된 네트워크는 original signal을 재구성하고, 이것의 인코더는 singer-agnostic한 임베딩을 만들어낸다. 하지만, 이것은 singing voice conversion을 직접적으로 하게 만드는 train은 아니기 때문에 이것으로 완성되지는 않는다.

→ Q. original signal=원래 가수의 목소리? / signer-agnostic 임베딩: 그 singer가 맞는지 아닌지 알 수 없는 임베딩? 어렵다..

학습이 어떻게 진행되냐!!?!?!?!?

예측 모델링을 위한 training은 2단계로 이루어진다.

<2단계>



- parallel 샘플로 train 하기 위해 backtranslation과 mixup 기법을 사용한다. (2-3 설명자료 by 미노민호 참고)
- 모든 mixup sample은 mixup된 가수 임베딩 벡터 u를 기반으로 한다. 여기서 u는 두 명의 다른 가수 j와 j'를 mix해서 만들어진 벡터!
- 어떻게?

-> mixup sample: $s_u^j = D[u](E(s^j)$

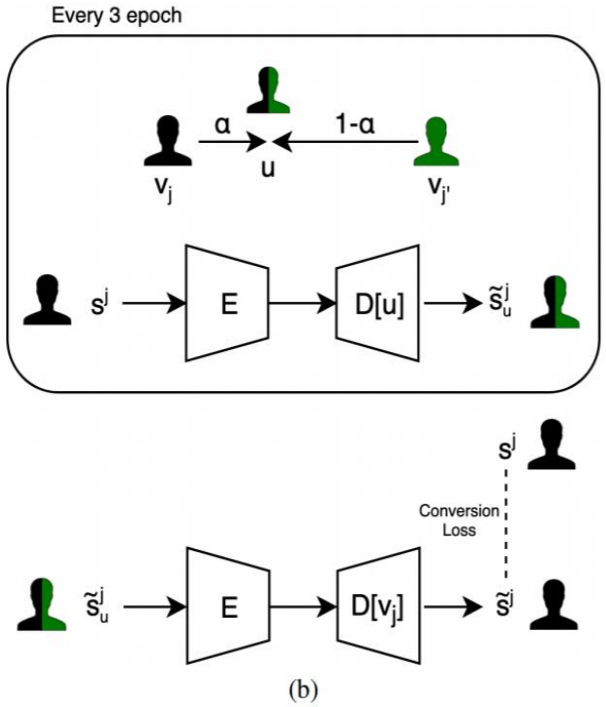
-> mixup sample 이 기반으로 하는 벡터 u: $u = \alpha v_j + (1-\alpha)v_{j'}$

(이 때, alpha는 균일분포를 따른다.)

학습이 어떻게 진행되냐!!?!?!?!?

예측 모델링을 위한 training은 2단계로 이루어진다.

<2단계>



2단계 training이 시작되면, 3 epoch마다 새로운 mixup sample을 만들게 된다.
-> why?
-> 디코더 D와 인코더 E를 train하는 과정 중 아래 loss function에 쓰기 위해!

$$\sum_{s_u^j} \mathcal{L}(D[v_j](E(s_u^j)), s^j),$$

(이 때, s^j 는 s_u^j 를 만들 때 사용한 audio clip으로 계산된다.)

학습이 어떻게 진행되냐!!?!?!?!?

<Test>

인풋으로 들어갔던 한 singer의 목소리 sample ‘s’를 다른 singer j의 목소리로 바꾸기 위해, test단계에서 j의 autoencoder pathway를 적용한다. (음...완벽 이해 x..)

이를 통해 새로운 sample인 $D[v_j](E(s_u^j))$ 를 얻게 된다.

문제는 실시간으로 돌릴 때, autoregressive decoding 과정에서 CUDA kernel에서도 힘들다는 것.

<참고 자료>

Confusion Network https://en.wikipedia.org/wiki/Confusion_network

Autoregressive model https://ratsgo.github.io/generative%20model/2018/01/31/AR/

Teacher Forcing http://doc.mindscale.kr/km/data_mining/08.html