Latent Variable Models and Iterative Refinement for **Non-Autoregressive** Neural Machine Translation

(= Parallel)

Jason Lee
New York University

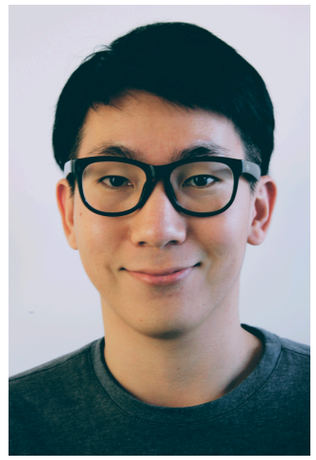
Based on joint work with Kyunghyun Cho, Elman Mansimov, Raphael Shu and Hideki Nakayama

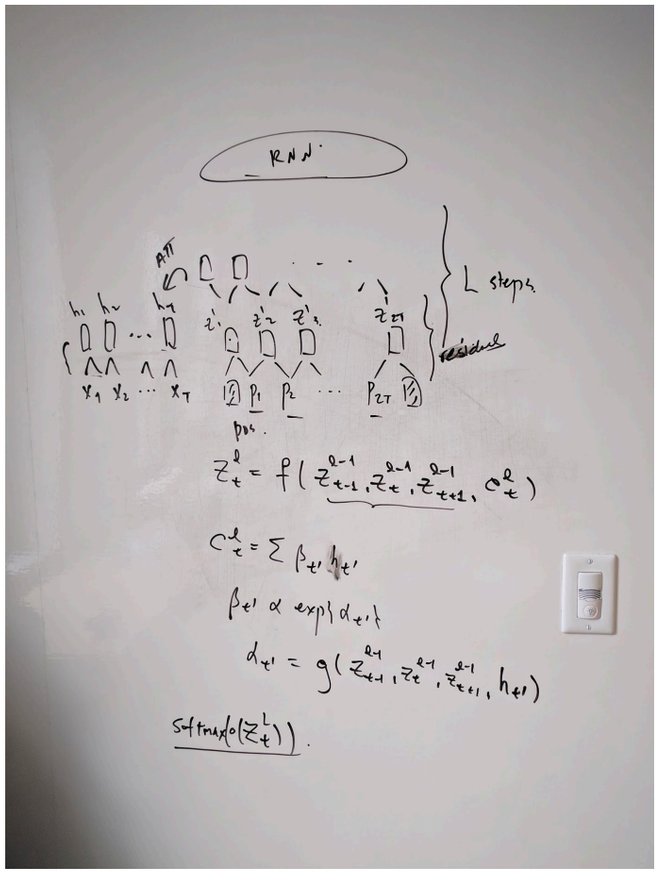
Fall 2017, when I started my PhD....

So Jason, what do you want to work on?

Slow decoding speed is a big problem in MT. Why don't you start looking into Neural GPU?

Umm… do you have any ideas?





Published as a conference paper at ICLR 2016

NEURAL GPUS LEARN ALGORITHMS

Łukasz Kaiser & Ilya Sutskever
Google Brain {lukaszkaiser,ilyasu}@google.com

Convolutional Sequence to Sequence Learning

Jonas Gehring
Michael Auli
David Grangier
Denis Yarats
Yann N. Dauphin
Facebook AI Research

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Model class, learning and parameterization

Inference algorithm

Latent Variable Models and Iterative Refinement for Non-Autoregressive Neural Machine Translation

Problem

Jason Lee
New York University

Based on joint work with Kyunghyun Cho, Elman Mansimov, Raphael Shu and Hideki Nakayama

Machine Translation

- Conditional sequence generation:
  - Given an input sequence $(x_1, \ldots, x_{t'})$, generate a corresponding output sequence $(y_1, \ldots, y_t)$.
- A broad range of tasks can be framed similarly:
  - Dialogue Modelling
  - Image denoising, Image inpainting
  - Automatic Speech Recognition / Text-to-speech

Sequence Learning

- Most neural sequence models are trained to maximize the log-likelihood of the training data

$$\hat{\theta} = \text{argmax}_\theta \sum_{n=1}^{N} \log p_\theta(y^n_{1:t} \mid x^n_{1:t'})$$

Autoregressive Sequence Models

- Learning: factorize the joint probability of a sequence into a product of conditionals. Exact likelihood.

$$\log p(y_{1:t} \mid x_{1:t'}) = \sum_{i=1}^{t} \log p(y_i \mid y_{<i}, x_{1:t'})$$

```
  p(I love BTS | 저는 방탄소년단을 좋아합니다)
= p(I          |                    저는 방탄소년단을 좋아합니다)
× p(love       | I,                 저는 방탄소년단을 좋아합니다)
× p(BTS        | I love,            저는 방탄소년단을 좋아합니다)
```

- Parameterization (conditional distribution):
  - RNNs (LSTMs, GRUs), ConvNet, Transformers

Autoregressive Models: Inference

$$\hat{y} = \text{argmax}_{y_1,\ldots,y_t} p(y_1, \ldots, y_t \,|\, x_{1:t'})$$

- Exact inference is intractable as the size of the search space grows exponentially w.r.t to t.

- Approximate inference:

  - Greedy search, beam search

Decoding latency grows linearly

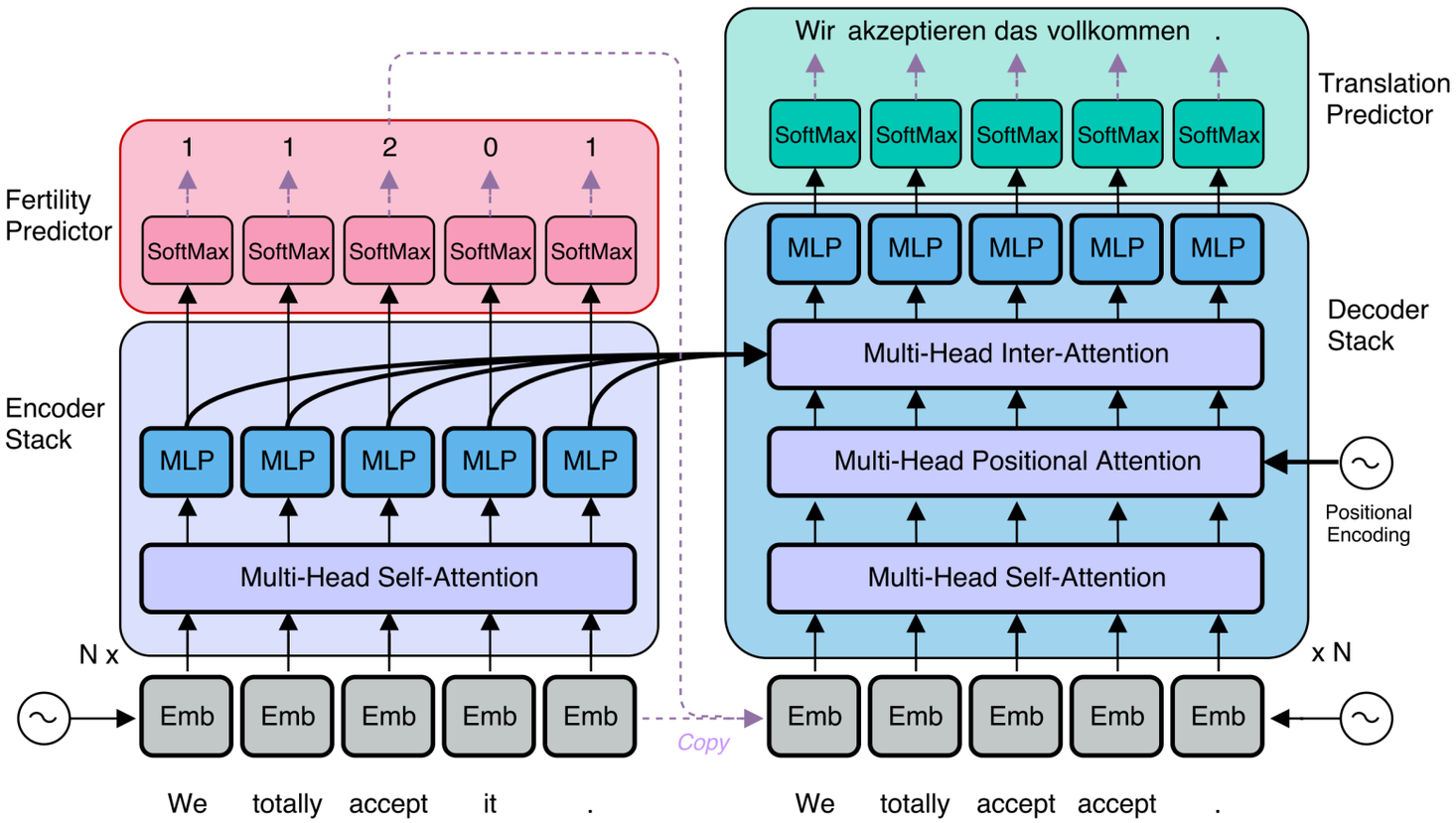
- Decoding from an autoregressive model is slow, i.e. latency scales w.r.t the output sequence length.

- Impossible to parallelize decoding at inference time, despite modern parallel hardware

- Presents significant challenges for:
  - Document-level machine translation (~1k subword segments, Junczys-Dowmunt, 2019)
  - Text-to-speech (32,000 samples/second)

Related work on Non-Autoregressive Sequence Models

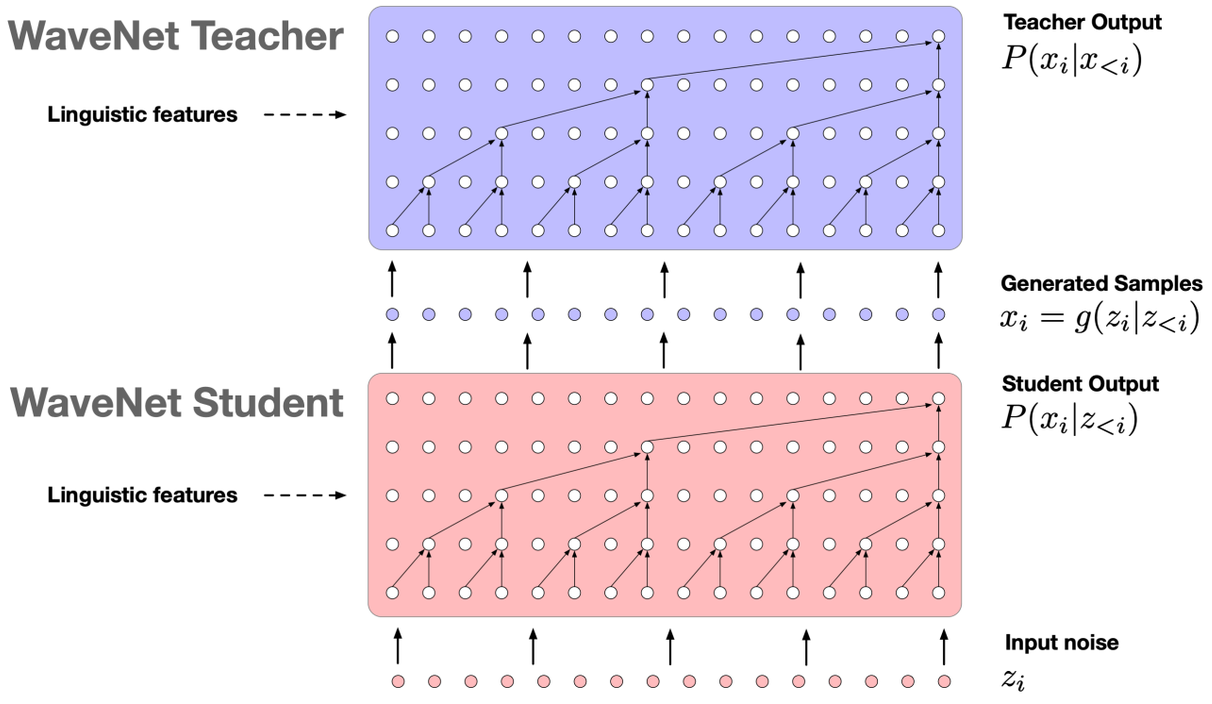
(1) Non-Autoregressive NMT



- Treat alignment (=fertility) as a latent variable
  - Supervise alignment prediction model using external model
- (1) Distillation, (2) Fertility as Latent Variable, (3) RL Finetuning + Reverse KL
- 3x speedup on WMT'14 En-De at the cost of 3-4 BLEU

Gu et al., 2017

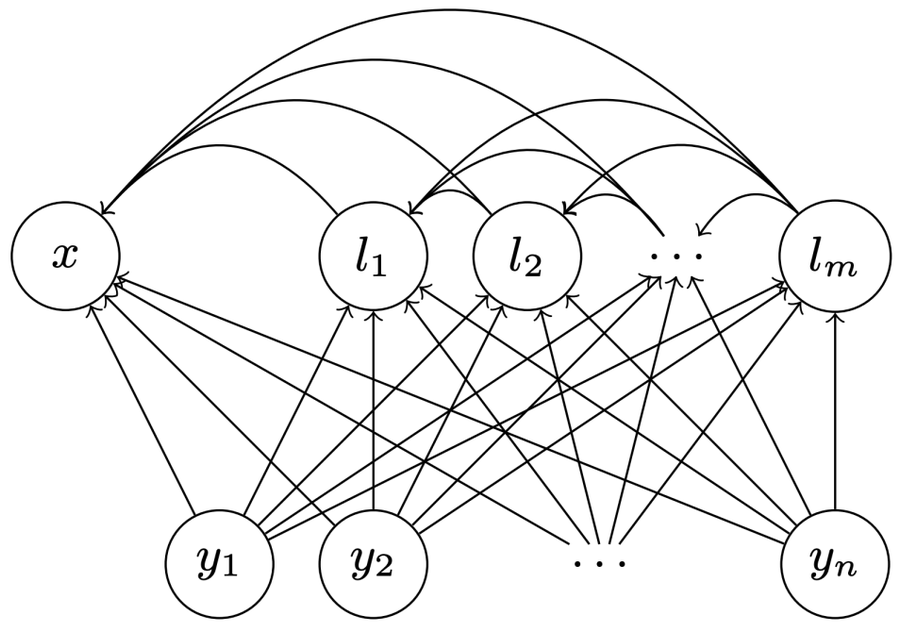
(2) Parallel WaveNet for TTS



- Non-autoregressive Inverse Autoregressive Flow
  - $\log p_x(x) = \log p_z(f(x)) + \log \left| \det\left(\frac{\partial f}{\partial x}\right) \right|, \qquad x_t = z_t \cdot s_\theta(\, z_{<t}\,) + \mu_\theta(\, z_{<t}\,)$
  - Trained with distillation from an autoregressive WaveNet
  - Auxiliary perceptual loss terms
- 1000x speedup vs. WaveNet

van den Oord et al., 2017

(3) Semi-autoregressive NMT



- **Discrete autoencoder (non-autoregressive)** for the target sentence
  - One discrete latent variable autoencodes 8 neighbouring target tokens
- **Autoregressive prior** from the source to the discrete latent variables.
- Discretization : VQ-VAE / semantic hashing
- 4x speedup on WMT'14 En-De at the cost of 1 BLEU
  - Using distillation (Roy et al., 2018)

Kaiser et al., 2018

(4) Non-autoregressive NMT with Masked Language Models

- $P(y_{\text{mask}} \mid y_{\text{obs}}, x)$: predict masked tokens from observed ones
  - Training: y_{mask} selected at random
  - Decoding: mask $1 - t/T$ of target tokens with the lowest probability scores (at the t-th refinement step).
- Length prediction can be batched with l different samples
  - $T = 10, l = 2$: **27.03 BLEU** on WMT'14 En->De (1.3 speedup)
  - $T = 4, l = 2$: **25.51 BLEU** on WMT'14 En->De (3x speedup)

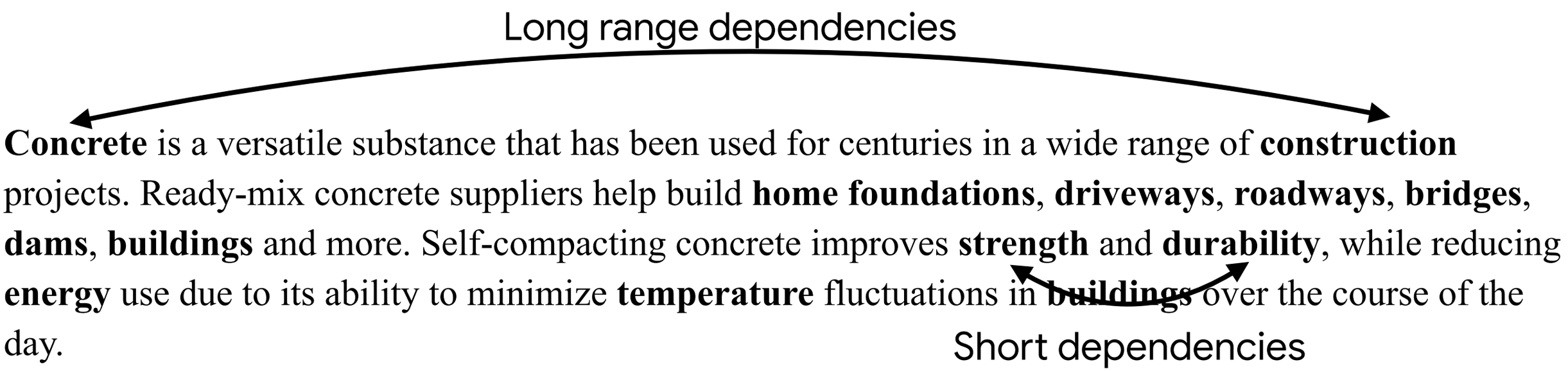
Ghazvininejad et al., 2019

Our Approach

- We'd like to factorize the output distribution:

$$p(y_1, \ldots, y_t | x) = \prod_{i=1}^{t} p(y_i | x)$$

- Main challenge: how do we capture the dependencies between the output variables y_i?



Research Questions

1. **Sequence Learning**

   - How can we learn the distribution of output sequences $(y_1, \ldots, y_t)$ non-autoregressively?

2. **Sequence Generation**

   - Having trained a model, how can we perform inference efficiently? (find the best output y given an input x)?

Research Questions

1. **Sequence Learning**

   - How can we learn the distribution of output sequences $(y_1, \ldots, y_t)$ non-autoregressively?

   - Let **latent variables** capture the dependencies, so the distribution of the output sequence can be factorized given the latent variables

   - $$\log p(y_{1:t}, z \mid x) = \sum_{i=1}^{t} \log p(y_i, z \mid x)$$

Research Questions

2. **Sequence Generation**

   - Having trained a model, how can we perform inference efficiently? (find the best output y given an input x)?

   - $\hat{y} = \text{argmax}_{y_{1:t}} \log \int_z p(y_{1:t}, z \mid x_{1:t'})$

   - Use **iterative refinement**

Latent Variable Models and Iterative Refinement

(x, y) : (input, output) sequence

z : latent variable capturing the output sequence

$p(y \mid x) = \sum_{z} p(y, z \mid x)$ for discrete latent variables z

1. Iterative Refinement in a Discrete Space (Lee, Mansimov and Cho, 2018).

$p(y \mid x) = \int p(y, z \mid x)\, dz$ for continuous latent variables z

2. Iterative Refinement in a Hybrid Space (Shu, Lee, Nakayama and Cho, 2020).

3. Iterative Refinement in a Continuous Space (Lee, Shu and Cho 2020).

Latent Variable Models and Iterative Refinement

(x, y) : (input, output) sequence

z : latent variable capturing the output sequence

$p(y \mid x) = \sum_z p(y, z \mid x)$ for discrete latent variables z

1. Iterative Refinement in a Discrete Space (Lee, Mansimov and Cho, 2018).

$p(y \mid x) = \int p(y, z \mid x)\, dz$ for continuous latent variables z

2. Iterative Refinement in a Hybrid Space (Shu, Lee, Nakayama and Cho, 2020).

3. Iterative Refinement in a Continuous Space (Lee, Shu and Cho 2020).

Discrete Latent Variable Models

We introduce L discrete latent variables $z_1, \ldots, z_L$ to capture the dependencies between the output variables.

$$p(y \mid x) = \sum_{z_{1:L}} p(y, z_{1:L} \mid x)$$

$$= \sum_{z_{1:L}} \prod_{l=1}^{L+1} p(z_l \mid z_{<l}, x), \text{ setting } y = z_{L+1}$$

Lee, Mansimov and Cho, 2018.

Discrete Latent Variable Models

We introduce L discrete latent variables $z_1, \ldots, z_L$ to capture the dependencies between the output variables.

$$p(y \mid x) = \sum_{z_{1:L}} p(y, z_{1:L} \mid x)$$

$$= \sum_{z_{1:L}} \prod_{l=1}^{L+1} p(z_l \mid z_{<l}, x), \text{ setting } y = z_{L+1}$$

We constrain z as discrete latent variables with the same dimensionality as the target sentence. $z \in \{0, \ldots, |V_{trg}|\}^T$

Lee, Mansimov and Cho, 2018.

Approximations to Sidestep the Marginalization

We introduce L discrete latent variables $z_1, \ldots, z_L$ to capture the dependencies between the output variables.

$$p(y \mid x) = \sum_{z_{1:L}} p(y, z_{1:L} \mid x)$$

$$= \sum_{z_{1:L}} \prod_{l=1}^{L+1} p(z_l \mid z_{<l}, x), \text{ setting } y = z_{L+1}$$

As this marginalization is intractable, we consider two approximations:

Lee, Mansimov and Cho, 2018.

(1) Deterministic Approximation

$$p(y \mid x) = \sum_{z_{1:L}} p(y, z_{1:L} \mid x)$$

$$= \sum_{z_{1:L}} \prod_{l=1}^{L+1} p(z_l \mid z_{<l}, x)$$

$$\geq \prod_{l=1}^{L+1} p(\hat{z}_l \mid \hat{z}_{<l}, x), \text{ where } \hat{z}_l = \text{argmax}_z p(z \mid \hat{z}_{<l}, x)$$

Lee, Mansimov and Cho, 2018.

(2) Stochastic Approximation

$$p(y \mid x) = \sum_{z_{1:L}} p(y, z_{1:L} \mid x)$$

$$= \sum_{z_{1:L}} \prod_{l=1}^{L+1} p(z_l \mid z_{<l}, x)$$

$$\approx \prod_{l=1}^{L+1} p(\hat{z}_l \mid \hat{z}_{<l}, x), \text{ where } \hat{z}_l \sim p(z \mid \hat{z}_{<l}, x)$$

This gives an unbiased estimate of the marginal.

Lee, Mansimov and Cho, 2018.

(2) Stochastic Approximation

$$p(y \mid x) = \sum_{z_{1:L}} p(y, z_{1:L} \mid x)$$

$$= \sum_{z_{1:L}} \prod_{l=1}^{L+1} p(z_l \mid z_{<l}, x)$$

$$\approx \prod_{l=1}^{L+1} p(\hat{z}_l \mid \hat{z}_{<l}, x), \text{ where } \hat{z}_l \sim p(z \mid \hat{z}_{<l}, x)$$

Can be viewed as a denoising autoencoder.

$$\prod_{l=1}^{L+1} p(y^* \mid \tilde{y}_l, x)$$

Lee, Mansimov and Cho, 2018.

Denoising Autoencoder

- $x \in \mathbb{R}^d$, L2 squared loss $l(\cdot)$, stochastic corruption function $c(\cdot)$, denoising function $d(\cdot)$

$$\mathscr{L}_{DAE} = \mathbb{E}[\ l(x, d(c(x)))\ ]$$

- Alain and Bengio showed that DAE estimates the score (gradient of the log density.)

- Our corruption operations: (1) swap neighbouring tokens, (2) replace token with a random token, (3) repeat tokens

Lee, Mansimov and Cho, 2018.

Joint Training Objective

$$J_{LVM}(\theta) = -\sum_{l=1}^{L+1} \log p_\theta(\hat{z}_l \,|\, \hat{z}_{<l}, x)$$

$$J_{DAE}(\theta) = -\sum_{l=1}^{L+1} \log p_\theta(y^* \,|\, \tilde{y}_l, x)$$

$$J(\theta) = \alpha_l \cdot J_{LVM}(\theta) + (1 - \alpha_l) \cdot J_{DAE}(\theta)$$

We train the length predictor (2-layer MLP from source hidden states) jointly with the rest of the model.

Lee, Mansimov and Cho, 2018.

Deterministic Inference

1. $\hat{T} = \text{argmax}_T \log p(T \mid X)$: predict the target length

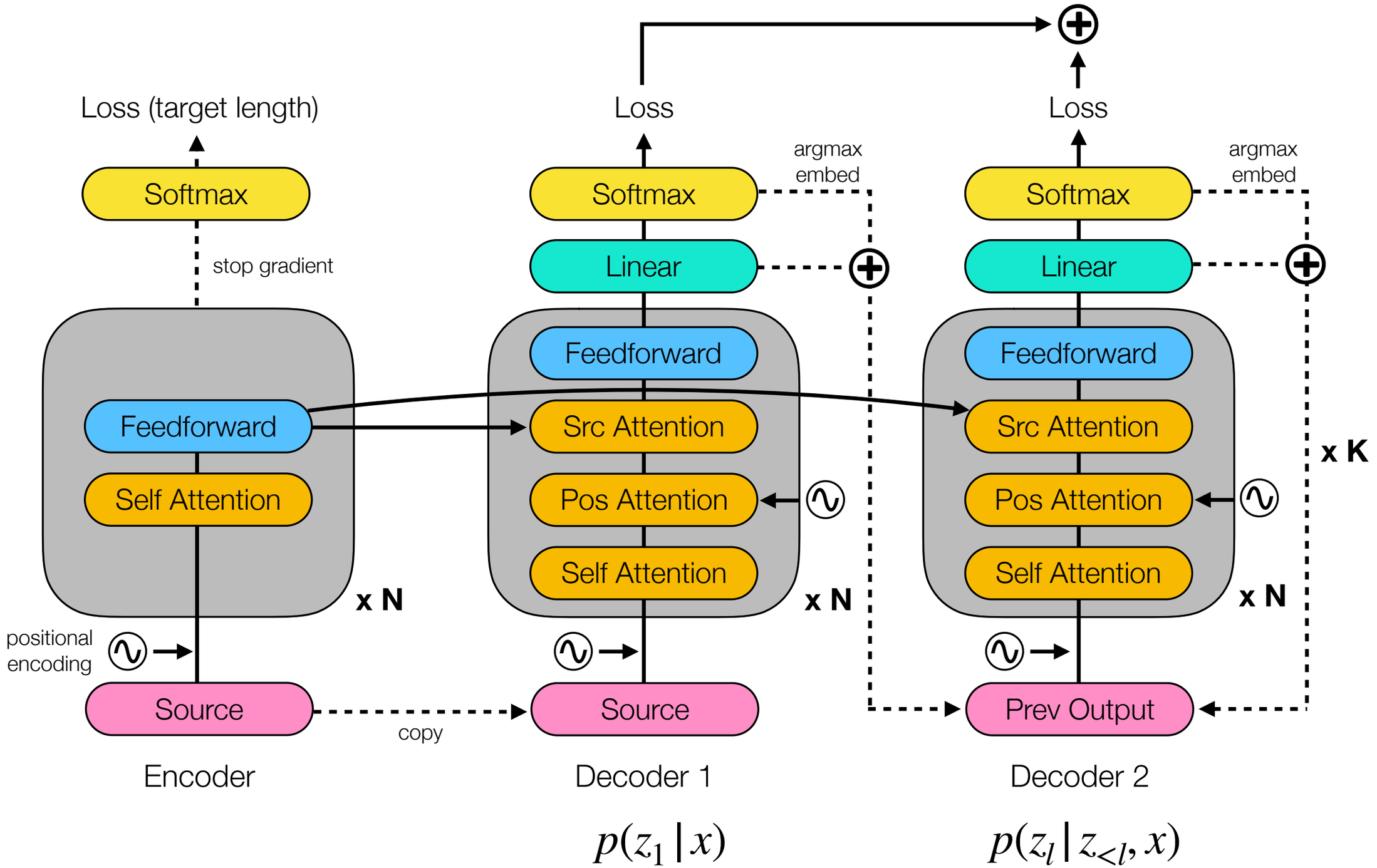
2. $\hat{t}_t^0 = \text{argmax}_{y_t} \log p(y_t^0 \mid x)$: predict the first latent variables

3. $\hat{t}_t^l = \text{argmax}_{y_t} \log p(y_t^l \mid \hat{Y}_l, x)$: refine the latent variables

4. Repeat (3) for either (1) K times, or (2) until a certain termination criterion has been met.

Lee, Mansimov and Cho, 2018.

Network Architecture

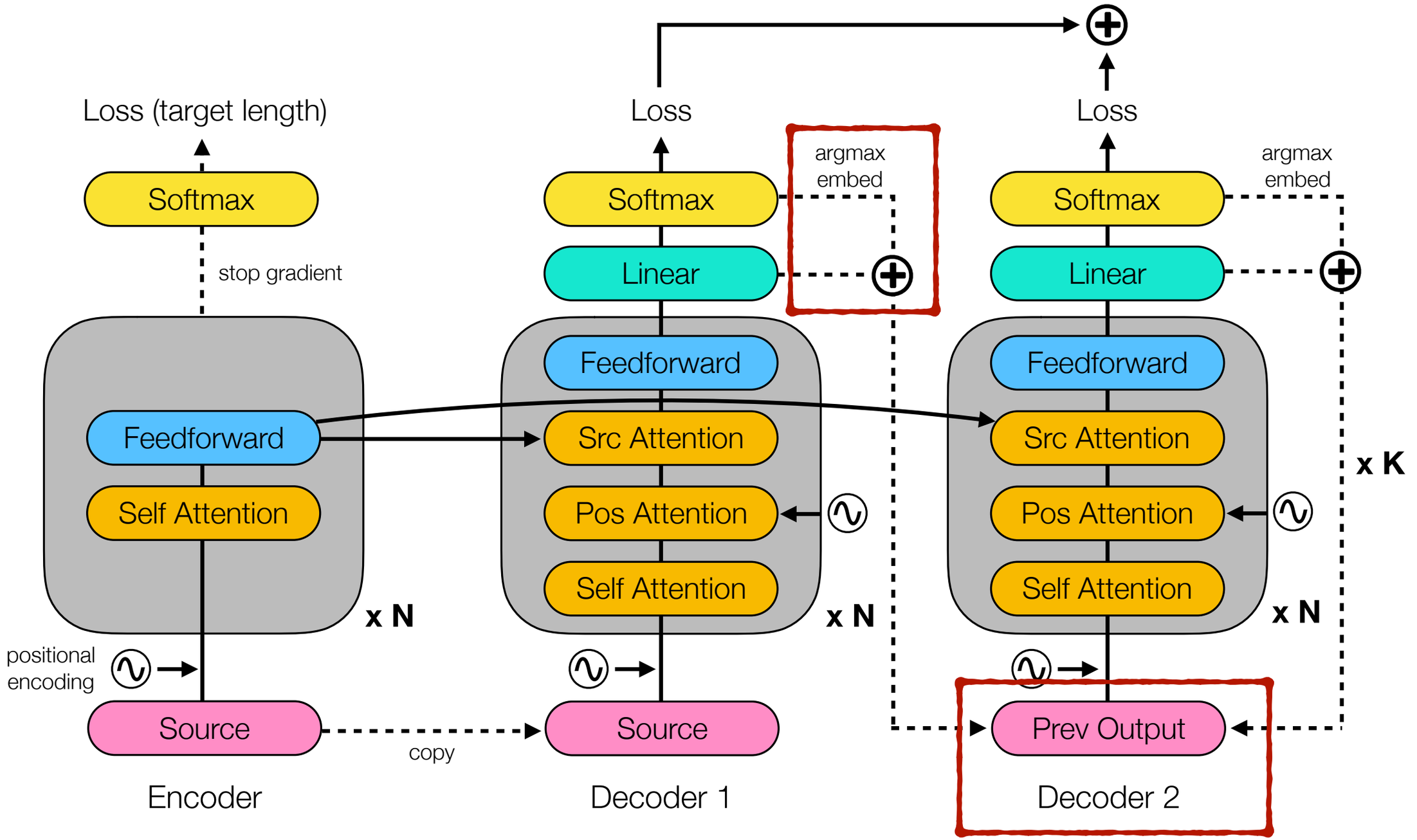


$p(z_1 \mid x)$

$p(z_l \mid z_{<l}, x)$

Lee, Mansimov and Cho, 2018.

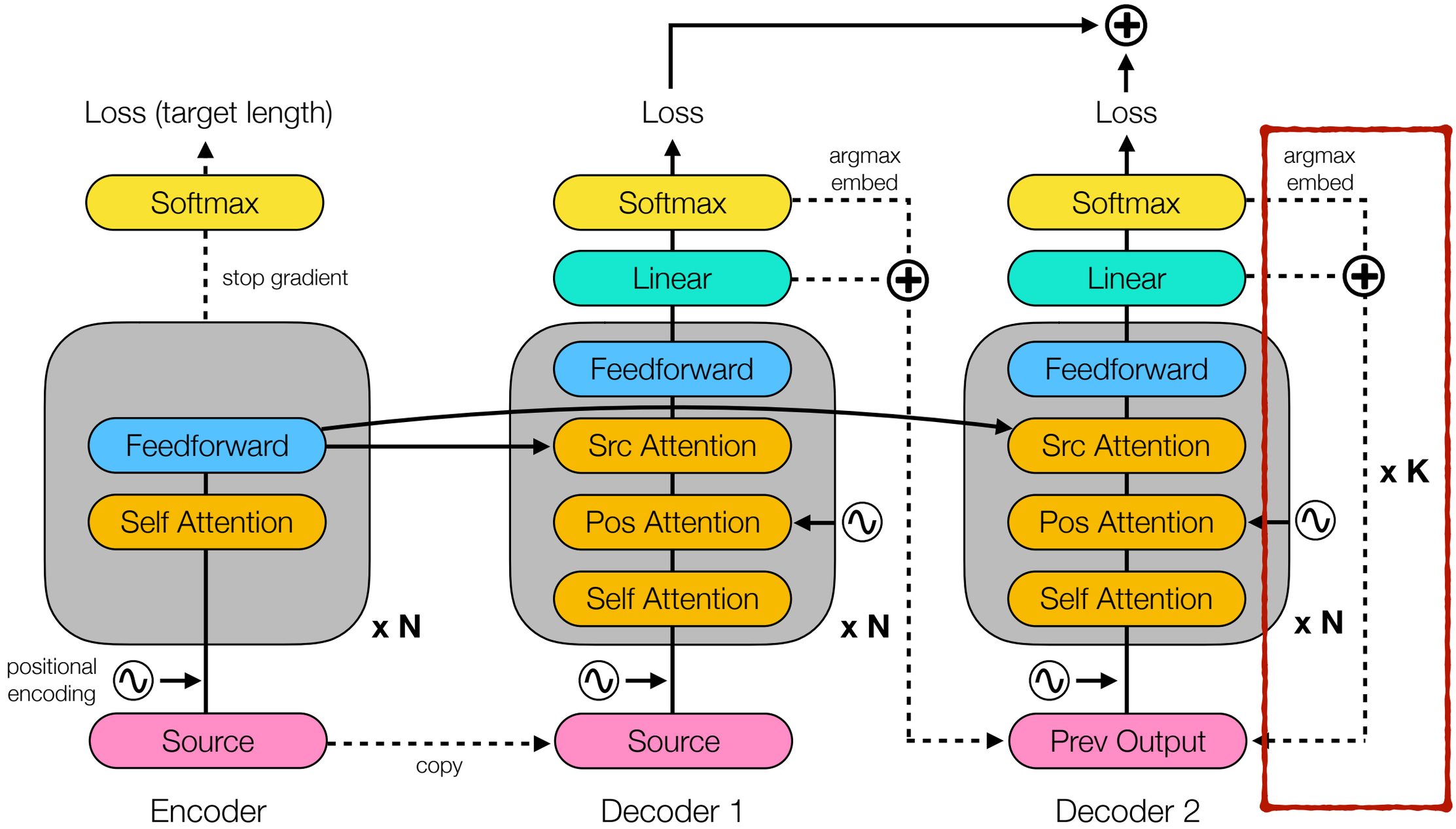
Network Architecture



- We use embeddings + hidden activations from the previous layer as input to the next

Lee, Mansimov and Cho, 2018.

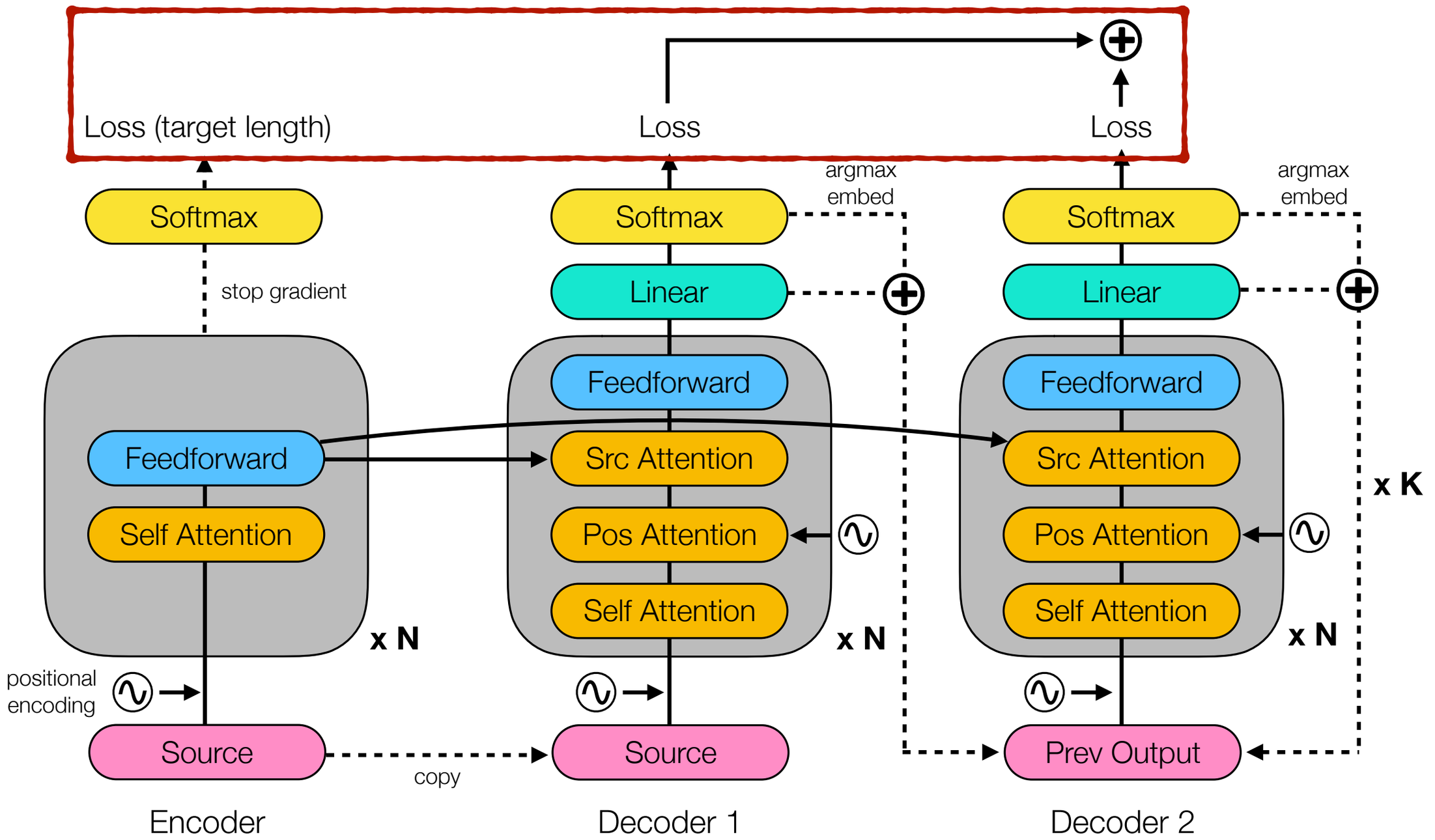
Network Architecture



- Refine K (=4) times during training, potentially use > K steps during inference.
- **Adaptive** decoding : refine until output converges

Lee, Mansimov and Cho, 2018.

Network Architecture



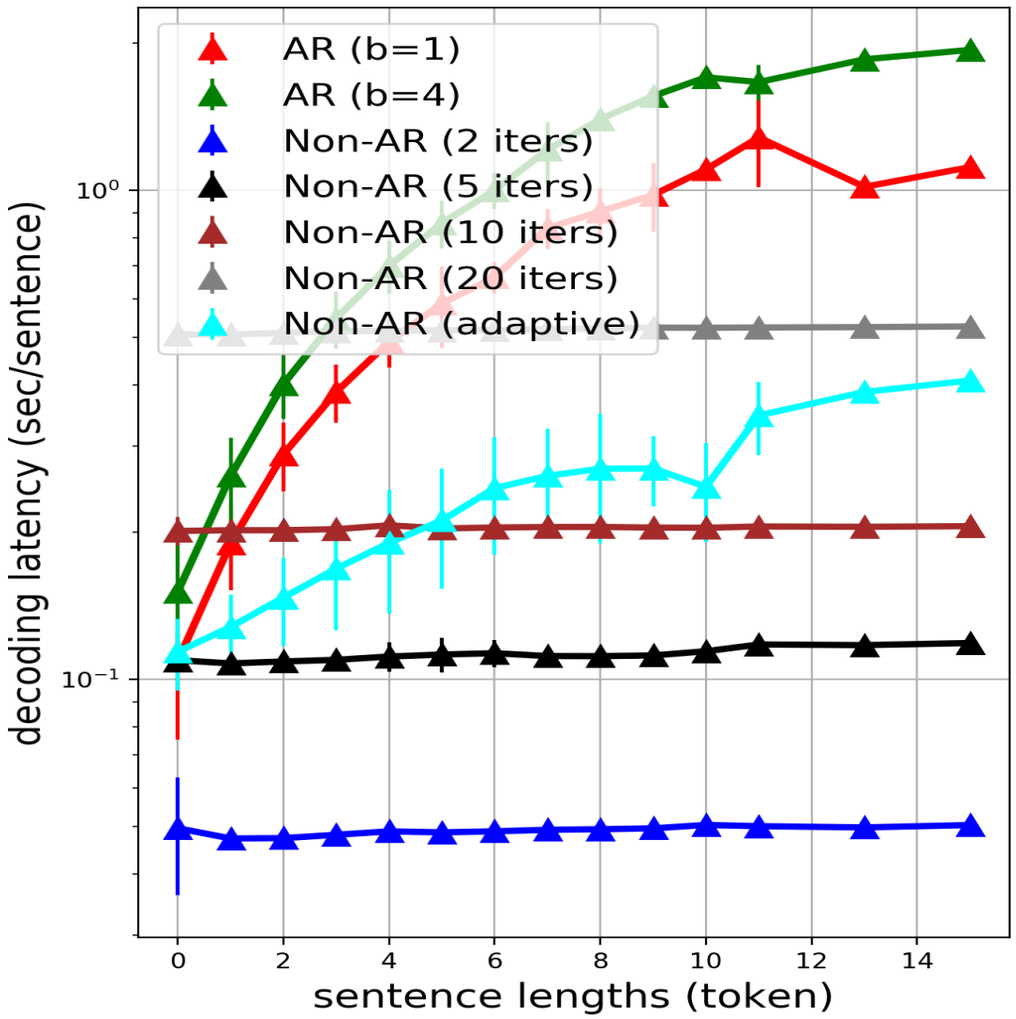
- Loss from all iterations and the length prediction model are summed

Lee, Mansimov and Cho, 2018.

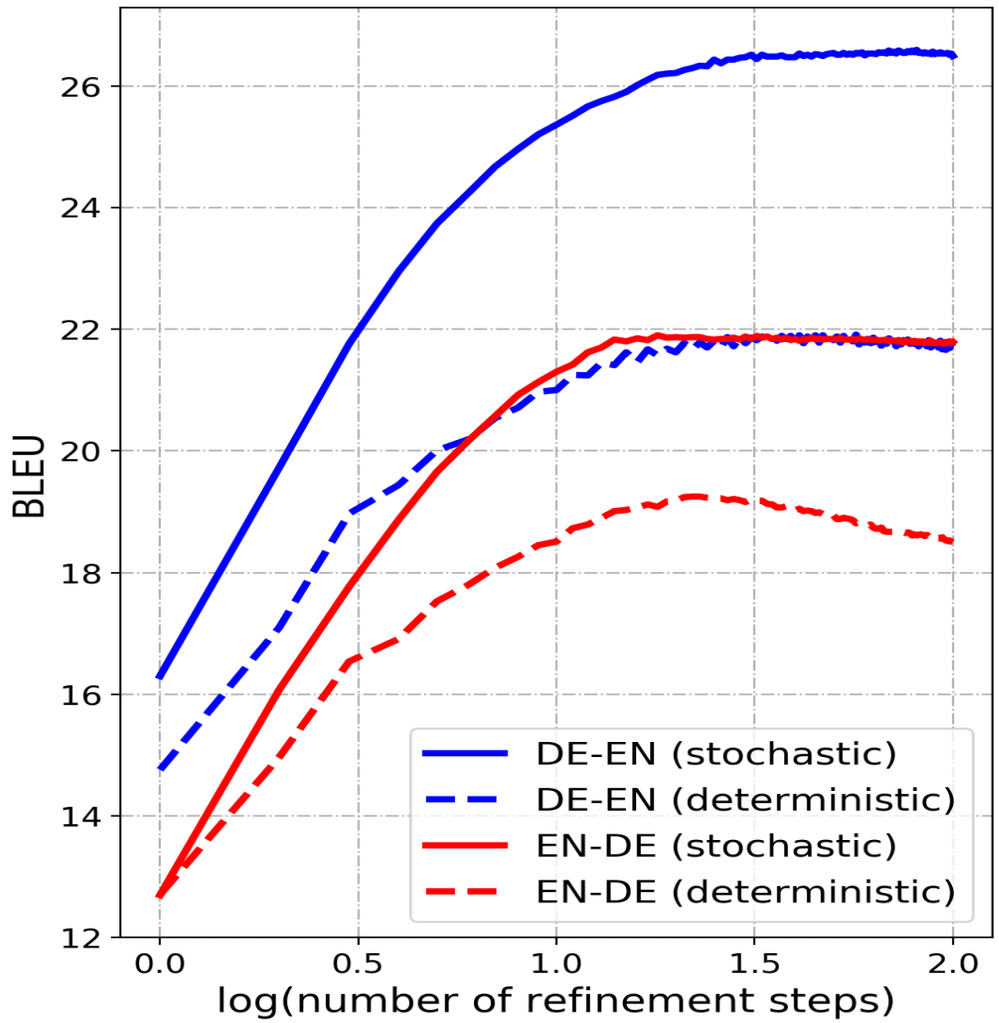
Non-autoregressive NMT on WMT'14 En-De

		BLEU(%)	Gap	Speed
(Gu et al., 2018)	Baseline	23.4		
	Gu's Model (NAT + NPD S=100)	19.1	-4.3	2.3x
(Lee et al., 2018)	Baseline	24.5		
	Lee's Model (Adaptive NAR)	21.5	-3.0	1.9x
(Kaiser et al., 2018)	Kaiser's baseline	23.5		
	Kaiser's Model (LT, Improved semhash)	19.8	-3.7	3.8x

Latency vs. Sequence Length



Refinement Steps vs. Performance



Lee, Mansimov and Cho, 2018.

Qualitative Results : IWSLT'16 De -> En

Src	seitdem habe ich sieben Häuser in der Nachbarschaft mit den Lichtern versorgt und sie funktionierenen wirklich gut .
Iter 1	and I 've been seven homes since in neighborhood with the lights and they 're really functional .
Iter 2	and I 've been seven homes in the neighborhood with the lights , and they 're a really functional .
Iter 4	and I 've been seven homes in neighborhood with the lights , and they 're a really functional .
Iter 8	and I 've been providing seven homes in the neighborhood with the lights and they 're a really functional .
Iter 20	and I 've been providing seven homes in the neighborhood with the lights , and they 're a very good functional .
Ref	since now , I 've set up seven homes around my community , and they 're really working .

Src	er sah sehr glücklich aus , was damals ziemlich ungewöhnlich war , da ihn die Nachrichten meistens deprimierten .
Iter 1	he looked very happy , which was pretty unusual the , because the news was were usually depressing .
Iter 2	he looked very happy , which was pretty unusual at the , because the news was s depressing .
Iter 4	he looked very happy , which was pretty unusual at the , because news was mostly depressing .
Iter 8	he looked very happy , which was pretty unusual at the time because the news was mostly depressing .
Iter 20	he looked very happy , which was pretty unusual at the time , because the news was mostly depressing .
Ref	there was a big smile on his face which was unusual then , because the news mostly depressed him .

Src	furchtlos zu sein heißt für mich , heute ehrlich zu sein .
Iter 1	to be , for me , to be honest today .
Iter 2	to be fearless , me , is to be honest today .
Iter 4	to be fearless for me , is to be honest today .
Iter 8	to be fearless for me , me to be honest today .
Iter 20	to be fearless for me , is to be honest today .
Ref	so today , for me , being fearless means being honest .

Table 4: Three sample De→En translations from the non-autoregressive sequence model. Source sentences are from the dev set of IWSLT'16. The first iteration corresponds to Decoder 1, and from thereon, Decoder 2 is repeatedly applied. Sub-sequences with changes across the refinement steps are underlined.

Lee, Mansimov and Cho, 2018.

Qualitative Results: MS COCO

Generated Caption

Iter 1	a yellow bus parked on parked in of parking road .
Iter 2	a yellow and black on parked in a parking lot .
Iter 3	a yellow and black bus parked in a parking lot .
Iter 4	a yellow and black bus parked in a parking lot .

Reference Captions

a tour bus is parked on the curb waiting
city bus parked on side of hotel in the rain .
bus parked under an awning next to brick sidewalk
a bus is parked on the curb in front of a building .
a double decked bus sits parked under an awning

Generated Caption

Iter 1	a woman standing on playing tennis on a tennis racquet .
Iter 2	a woman standing on a tennis court a tennis racquet .
Iter 3	a woman standing on a tennis court a a racquet .
Iter 4	a woman standing on a tennis court holding a racquet .

Reference Captions

a female tennis player in a black top playing tennis
a woman standing on a tennis court holding a racquet .
a female tennis player preparing to serve the ball .
a woman is holding a tennis racket on a court
a woman getting ready to reach for a tennis ball on the ground

Lee, Mansimov and Cho, 2018.

Latent Variable Models and Iterative Refinement

(x, y) : (input, output) sequence

z : latent variable capturing the output sequence

$p(y \mid x) = \sum_z p(y, z \mid x)$ for discrete latent variables z

1. Iterative Refinement in a Discrete Space (Lee, Mansimov and Cho, 2018).

$p(y \mid x) = \int p(y, z \mid x)\, dz$ for continuous latent variables z

2. Iterative Refinement in a Hybrid Space (Shu, Lee, Nakayama and Cho, 2020).

3. Iterative Refinement in a Continuous Space (Lee, Shu and Cho 2020).

Continuous Latent Variable Models

$$\log p_\theta(y \mid x) = \log \int p_\theta(y, z \mid x) dz$$

Shu, Lee, Nakayama and Cho, 2020.

Continuous Latent Variable Models

$$\log p_\theta(y \mid x) = \log \int p_\theta(y, z \mid x) dz$$

$$= \log \int \frac{p_\theta(y \mid z, x)\, p_\theta(z \mid x)}{q_\phi(z \mid y, x)}\, q_\phi(z \mid y, x) dz$$

$$= \log \mathbb{E}_{z \sim q_\phi}\left[\frac{p_\theta(y \mid z, x)\, p_\theta(z \mid x)}{q_\phi(z \mid y, x)}\right]$$

$$\geq \mathbb{E}_{z \sim q_\phi}[\log p_\theta(y \mid z, x) + \log p_\theta(z \mid x) - \log q_\phi(z \mid y, x)]$$

Decoder **Prior** **Approximate Posterior**

$$= \mathbb{E}_{z \sim q_\phi}[\log p_\theta(y \mid z, x)] - \mathrm{KL}[q_\phi(z \mid y, x) \,||\, p_\theta(z \mid x)]$$

$$= \mathrm{ELBO}(x, y, \theta, \phi)$$

Shu, Lee, Nakayama and Cho, 2020.

Continuous Latent Variable Models

$$\log p_\theta(y \mid x) = \log \int p_\theta(y, z \mid x) dz$$

$$= \log \int \frac{p_\theta(y \mid z, x)\, p_\theta(z \mid x)}{q_\phi(z \mid y, x)}\, q_\phi(z \mid y, x) dz$$

$$= \log \mathbb{E}_{z \sim q_\phi}\left[\frac{p_\theta(y \mid z, x)\, p_\theta(z \mid x)}{q_\phi(z \mid y, x)}\right]$$

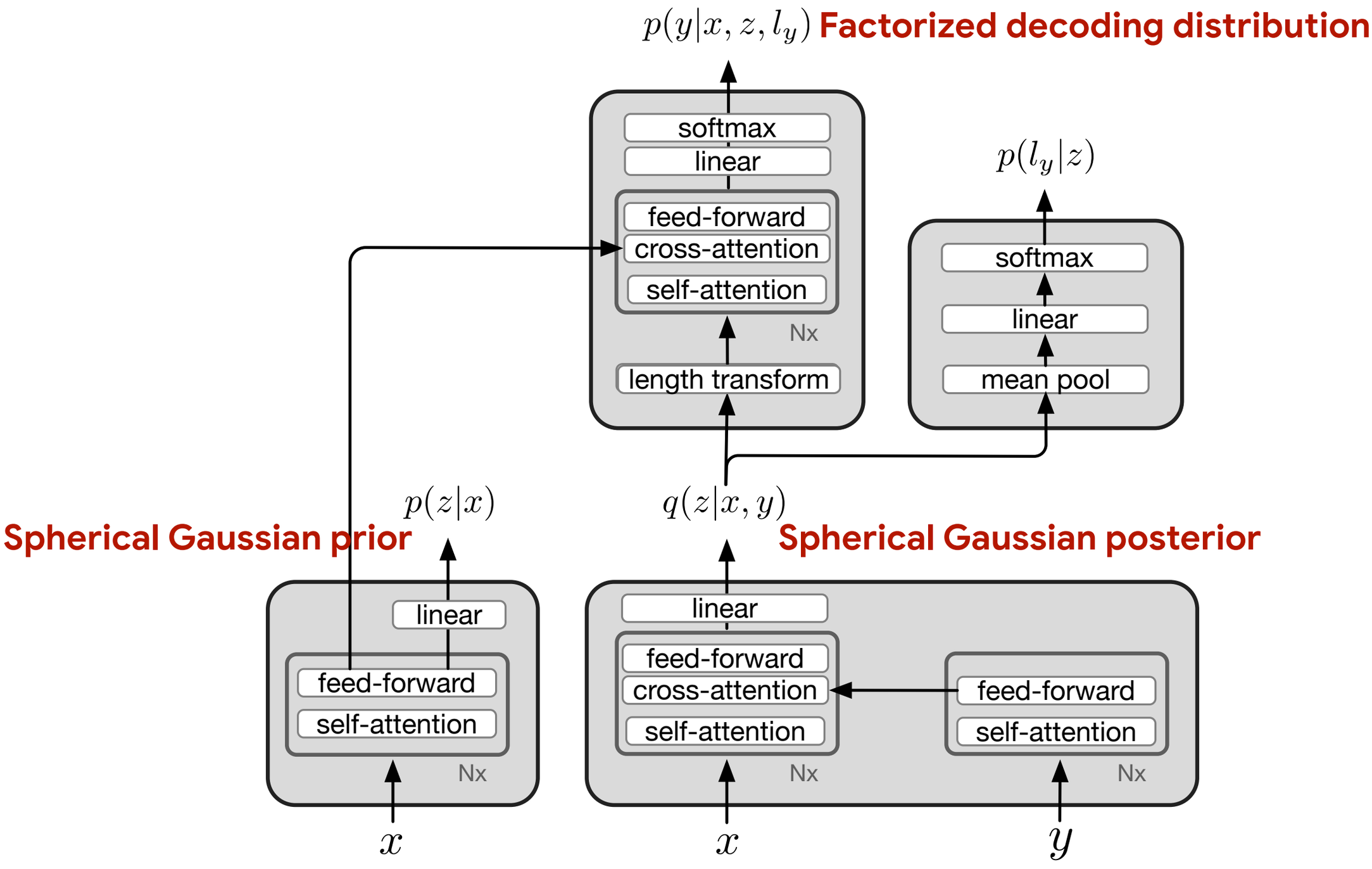
$$\geq \mathbb{E}_{z \sim q_\phi}[\log p_\theta(y \mid z, x) + \log p_\theta(z \mid x) - \log q_\phi(z \mid y, x)]$$

$$= \mathbb{E}_{z \sim q_\phi}[\log p_\theta(y \mid z, x)] - \mathrm{KL}[q_\phi(z \mid y, x) \,||\, p_\theta(z \mid x)]$$

$$= \mathrm{ELBO}(x, y, \theta, \phi) \qquad J(x, y, \theta, \phi) = \mathrm{ELBO}(x, y, \theta, \phi) + \log p(l_y \mid z)$$

Shu, Lee, Nakayama and Cho, 2020.

Network Architecture



Shu, Lee, Nakayama and Cho, 2020.

Inference with continuous LVM

Given x, inference is the search problem:

$$\hat{y} = \text{argmax}_y \int_z p(y, z \mid x)\, dz$$

$$= \text{argmax}_y \int_z p(y \mid z, x)\, p(z \mid x)\, dz$$

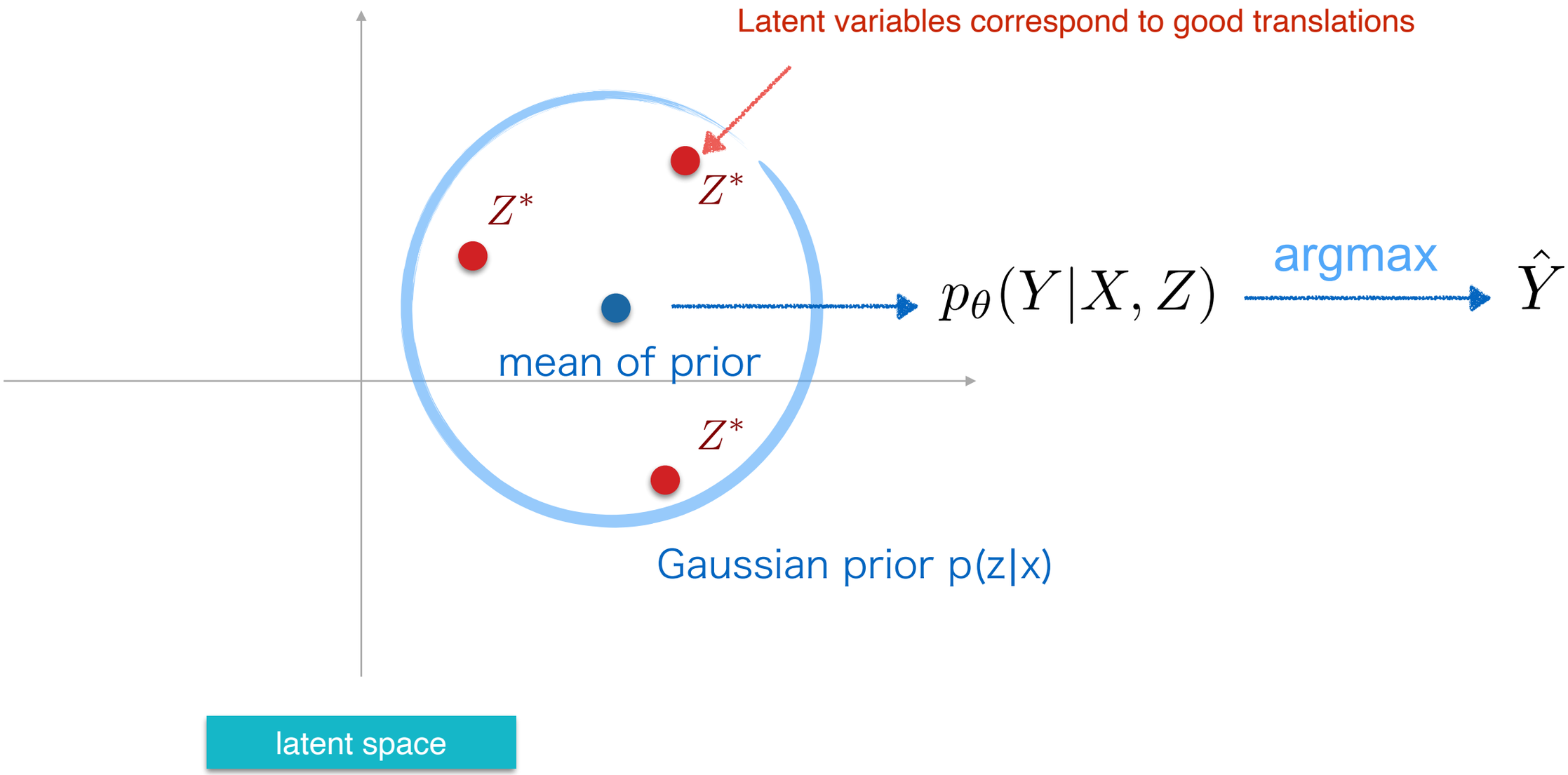
Spherical Gaussian Prior

$\neq \text{argmax}_y\, p(y \mid \mu, x)$, where μ is the mean of the prior

Simply decoding from the mean of the Gaussian prior may not lead to the most likely output sequence y

Shu, Lee, Nakayama and Cho, 2020.

Inference with continuous LVM



Shu, Lee, Nakayama and Cho, 2020.

Iterative Inference with a Delta Posterior

Given x, inference is the search problem:

$$\hat{y} = \text{argmax}_y \text{ELBO}(x, y, \theta, \phi)$$

$$= \text{argmax}_y \mathbb{E}_{z \sim q_\phi}[\log p_\theta(y \mid z, x) + \log p_\theta(z \mid x) - \log q_\phi(z \mid y, x)]$$

The expectation over z makes solving this intractable.

Shu, Lee, Nakayama and Cho, 2020.

Given x, inference is the search problem:

$$\hat{y} = \text{argmax}_y \text{ELBO}(x, y, \theta, \phi)$$

By introducing a Dirac delta posterior:

$$\delta(z \mid \mu) = 1 \text{ if } z = \mu$$
$$= 0 \text{ otherwise}$$

$$\text{ELBO}(x, y, \theta, \phi) = \mathbb{E}_{z \sim q_\phi}[\log p_\theta(y \mid z, x) + \log p_\theta(z \mid x) - \log q_\phi(z \mid y, x)]$$

$$= 0$$

$$= \log p_\theta(y \mid \mu, x) + \log p_\theta(\mu \mid x)$$

We perform approximate maximization of this **proxy lowerbound**, using a iterative procedure reminiscent to the EM algorithm.

Shu, Lee, Nakayama and Cho, 2020.

Iterative Inference with a Delta Posterior

We can approximately maximize $\log p_\theta(y \mid \mu, x) + \log p_\theta(\mu \mid x)$ with an EM-like procedure : we call this **Delta Inference**

1. Initialization : $z_0 := \mathbb{E}_{z \sim p_\theta(z|x)}[z]$ (mean of the prior)

2. First output : $\hat{y}_0 = \text{argmax}_y p(y \mid z_0, x)$

3. Repeat until done:

1. E-Step: match (proxy) delta posterior to the approximate posterior.

$$\hat{\mu}_i = \text{argmin}_\mu \, \text{KL}\left[\delta(z \mid \mu) \,\|\, q_\phi(z \mid \hat{y}_{i-1}, x)\right] \quad = \mathbb{E}_{z \sim q_\phi(z|y,x)}\left[z\right]$$

Continuous optimization

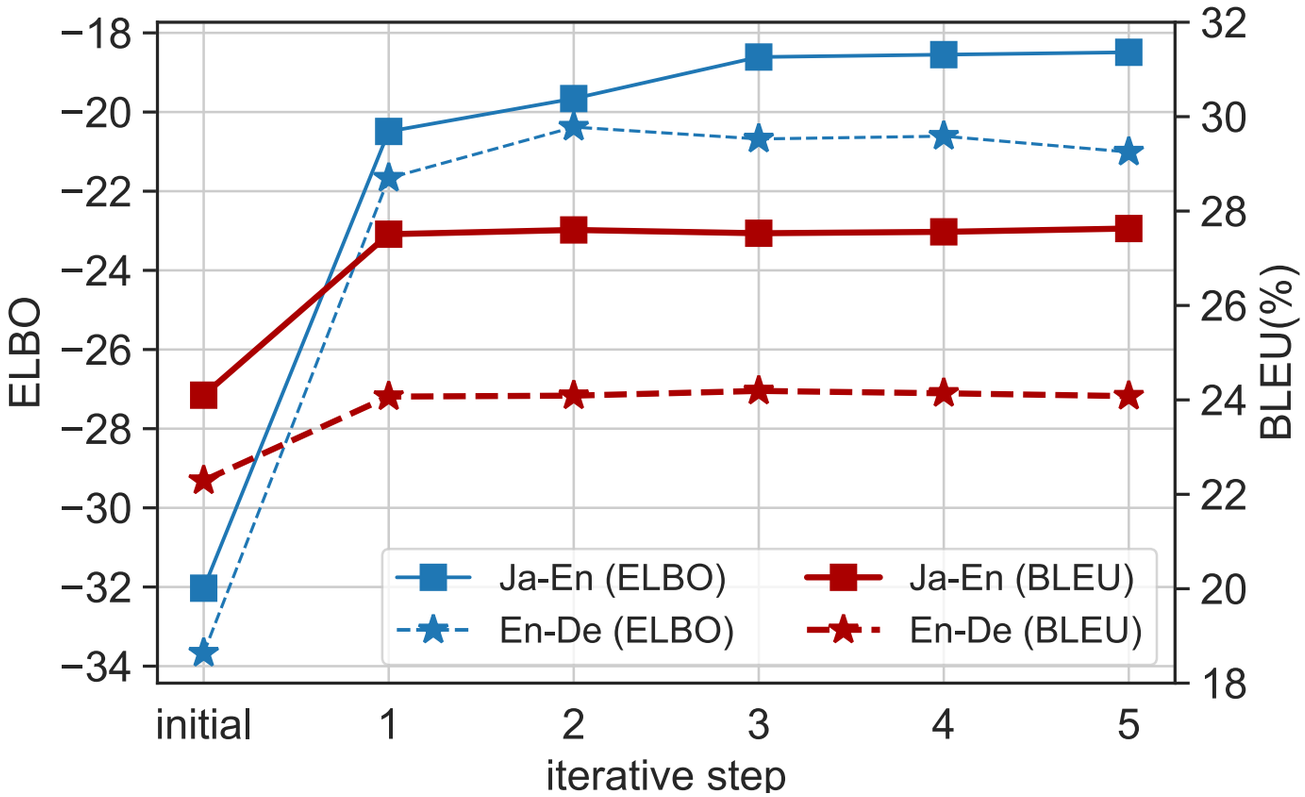
2. M-Step: maximize the lowerbound w.r.t y:

$$\hat{y}_i = \text{argmax}_y p(y \mid \hat{\mu}_i, x)$$

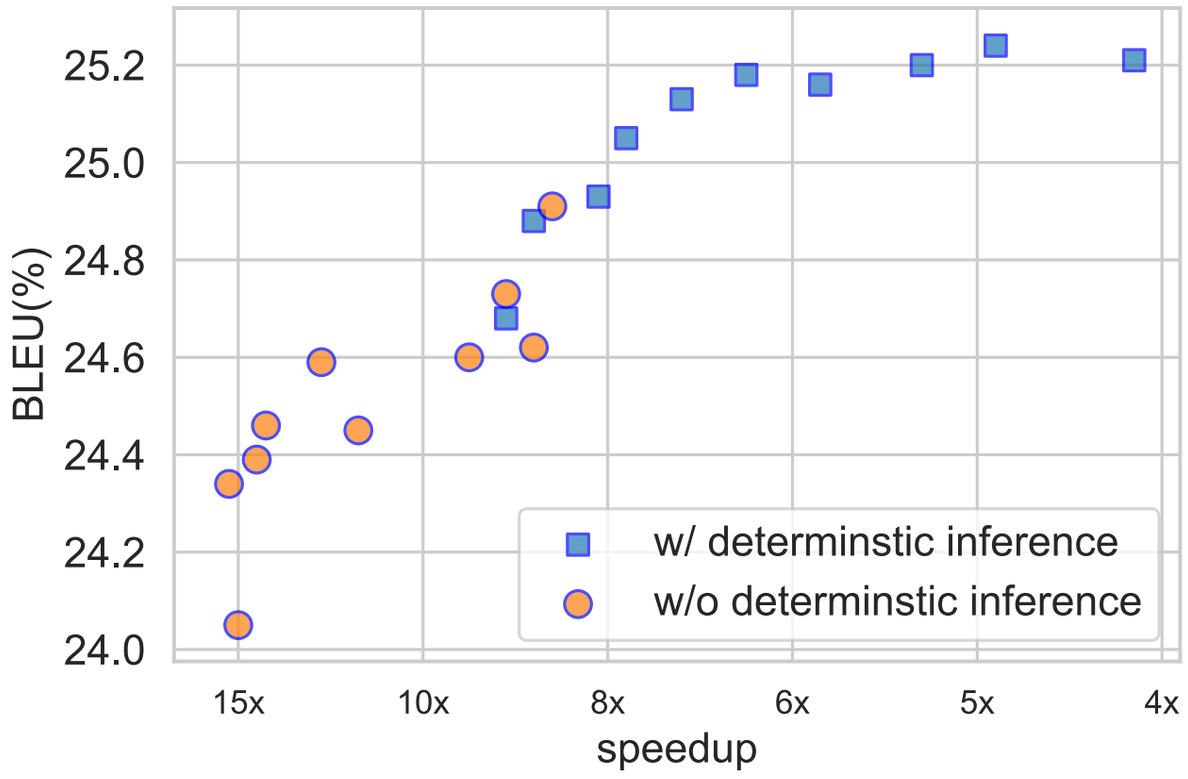
Discrete search

Shu, Lee, Nakayama and Cho, 2020.

Delta inference gives higher BLEU and ELBO



- Both ELBO and BLEU improve with more refinement steps



- Quality vs. Speed tradeoff.
- Iterative inference improves translation quality in all settings.

Shu, Lee, Nakayama and Cho, 2020.

Non-autoregressive NMT on WMT'14 En-De

		BLEU(%)	Gap	Speed
(Gu et al., 2018)	Baseline	23.4		
	Gu's Model (NAT + NPD S=100)	19.1	-4.3	2.3x
(Lee et al., 2018)	Baseline	24.5		
	Lee's Model (Adaptive NAR)	21.5	-3.0	1.9x
(Kaiser et al., 2018) (Roy et al., 2018)	Kaiser's baseline	23.5		
	Kaiser's Model (LT, Improved semhash)	19.8	-3.7	3.8x
	Roy's baseline	28.1		
	Roy's Model (VQVAE + EM + distillation)	26.7	-1.4	4.1x
(Ghaz et al., 2019)	Baseline	27.8		
	Ghaz's Model (CMLM with 4 iterations)	26.0	-1.8	
	Ghaz's Model (CMLM with 10 iterations)	26.9	-0.9	2~3x
(Shu et al., 2020)	Baseline	28.3		
	LANMT + Delta Inference	26.1	-2.2	6.3x

Latent Variable Models and Iterative Refinement

(x, y) : (input, output) sequence

z : latent variable capturing the output sequence

$p(y \mid x) = \sum_z p(y, z \mid x)$ for discrete latent variables z

1. Iterative Refinement in a Discrete Space (Lee, Mansimov and Cho, 2018).

$p(y \mid x) = \int p(y, z \mid x)\, dz$ for continuous latent variables z

2. Iterative Refinement in a Hybrid Space (Shu, Lee, Nakayama and Cho, 2020).

3. Iterative Refinement in a Continuous Space (Lee, Shu and Cho 2020).

Latent Variable Models and Iterative Refinement

(x, y) : (input, output) sequence

z : latent variable capturing the output sequence

$p(y \mid x) = \sum_{z} p(y, z \mid x)$ for discrete latent variables z

1. Iterative Refinement in a Discrete Space (Lee, Mansimov and Cho, 2018).

$p(y \mid x) = \int p(y, z \mid x)\, dz$ for continuous latent variables z

2. Iterative Refinement in a Hybrid Space (Shu, Lee, Nakayama and Cho, 2020).

3. Iterative Refinement in a Continuous Space (Lee, Shu and Cho 2020).

Motivation

- Delta inference is **unsatisfactory** as the M-step requires computing the softmax (slow operation).

2. M-Step: update our parameters to maximize the lowerbound:

$$\hat{y}_i = \text{argmax}_y p(y \mid \hat{\mu}_i, x)$$

Discrete search

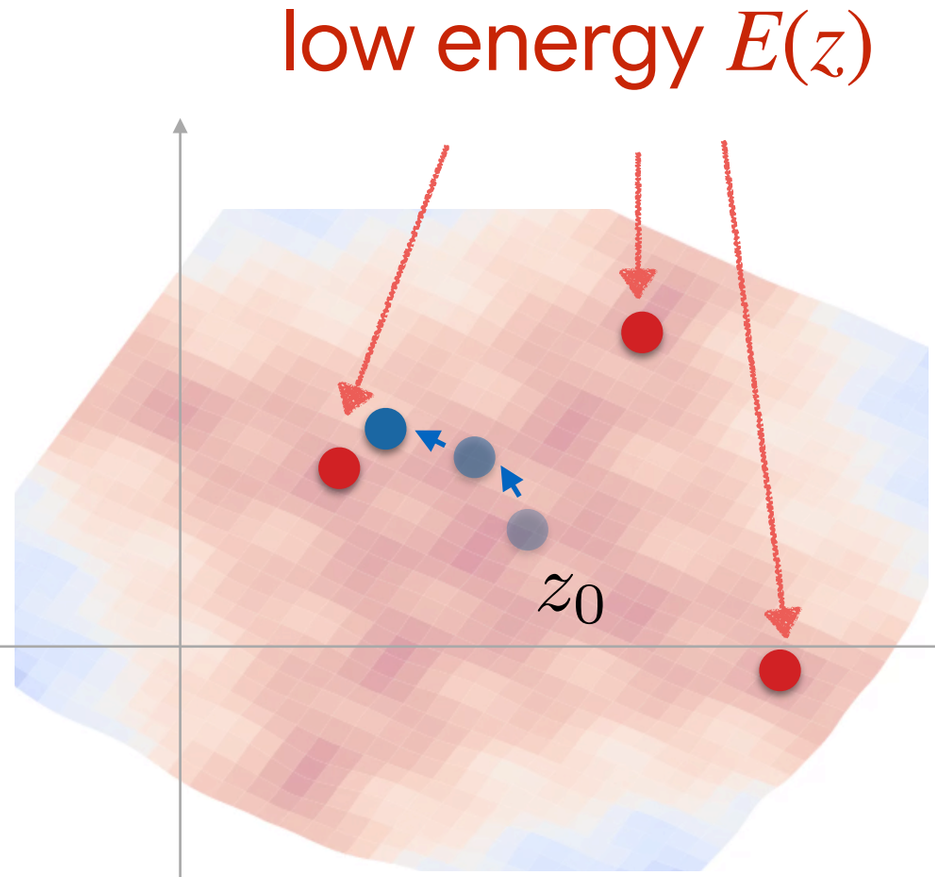
- If iterative refinement was performed in a continuous space only, we'd see big gains in speed.

Lee, Shu and Cho, 2020.

Faster Inference with Energy/Score Model

- We can imagine an energy-based model that assigns low energies to local optimas of log-likelihood

- If we can train such an energy-based model $E(z)$, then we can cast a discrete combinatorial search problem into a **continuous optimization** problem.



$$z_{t+1} = z_t - \alpha \cdot \nabla_z E(z)$$

Lee, Shu and Cho, 2020.

Objective for Inference Network

- Given a latent variable model $p_\theta(y, z \mid x)$ (trained, fixed):
- Let $\tau_\theta(z; x)$ be a function that approximates the quality of any latent variable z w.r.t the source sentence x.
  - $\tau_\theta(z; x) := \log p_\theta(\hat{y} \mid x), \quad \hat{y} = \text{argmax}_y \, p_\theta(y \mid z, x)$
- Our goal : Find a function $E_\psi(z; x)$ where

$$\text{argmin}_z E_\psi(z; x) \approx \text{argmax}_z(\tau_\theta(z; x))$$

Shu, Lee, Nakayama and Cho, 2020.

Learning

$$\tau_\theta(z; x) := \log p_\theta(\hat{y} \mid x), \quad \hat{y} = \text{argmax}_y \, p_\theta(y \mid z, x)$$

$$\text{argmin}_z E_\psi(z; x) \approx \text{argmax}_z(\tau_\theta(z; x))$$

Instead of directly approximating τ_θ , we instead train $-E_\psi$ to learn the difference of τ_θ between a pair of latent variable configurations (easier task, this is all we need for inference).

We solve the following for a pair of latent variables $z \neq \bar{z}$:

$$\min_\psi \left\| \left(-E_\psi(\bar{z}) + E_\psi(z) \right) - \left(\tau(\bar{z}) - \tau(z) \right) \right\|^2$$

1st order
Taylor expansion

$$\approx \min_\psi \left\| \nabla_z E_\psi(z) \right\|^2 + 2 \left(\left(\nabla_z E_\psi(z) \right)^\intercal \cdot \nabla_z \tau(z) \right)$$

Shu, Lee, Nakayama and Cho, 2020.

Learning

$$\tau_\theta(z; x) := \log p_\theta(\hat{y} \mid x), \quad \hat{y} = \text{argmax}_y \, p_\theta(y \mid z, x)$$

$$\text{argmin}_z E_\psi(z; x) \approx \text{argmax}_z(\tau_\theta(z; x))$$

Instead of directly approximating τ_θ , we instead train $-E_\psi$ to learn the difference of τ_θ between a pair of latent variable configurations (easier task, this is all we need for inference).

We solve the following for a pair of latent variables $z \neq \bar{z}$:

$$\min_\psi \left\| \left(-E_\psi(\bar{z}) + E_\psi(z) \right) - \left(\tau(\bar{z}) - \tau(z) \right) \right\|^2$$

1st order
Taylor expansion

$$\approx \min_\psi \left\| \nabla_z E_\psi(z) \right\|^2 + 2 \left(\left(\nabla_z E_\psi(z) \right)^\intercal \cdot \nabla_z \tau(z) \right)$$

Shu, Lee, Nakayama and Cho, 2020.

Learning

$$\tau_\theta(z; x) := \log p_\theta(\hat{y} \mid x), \quad \hat{y} = \boxed{\text{argmax}_y} p_\theta(y \mid z, x)$$

$$\text{argmin}_z E_\psi(z; x) \approx \text{argmax}_z(\tau_\theta(z; x))$$

As $\tau_\theta(z; x)$ is not differentiable with respect to z due to the argmax operation in $\hat{y}$, $\nabla_z \tau_\theta(z; x)$ is not defined.

We therefore use proxy gradient from delta inference, and our final training objective is the following:

$$\mathbb{E}_{z \sim p_\theta(z|x)} \left[\left\| \nabla_z E_\psi(z; x) \right\|^2 + 2 \left(\left(\nabla_z E_\psi(z; x) \right)^\intercal \cdot (\tilde{z} - z) \right) \right]$$

where $\tilde{z}$ is the output of applying k steps of delta inference on z

Shu, Lee, Nakayama and Cho, 2020.

Parameterization

$$\mathbb{E}_{z\sim p_\theta(z|x)}\left[\left\|\nabla_z E_\psi(z;x)\right\|^2 + 2\Big(\big(\nabla_z E_\psi(z;x)\big)^\intercal \cdot (\tilde{z}-z)\Big)\right]$$

We either parameterize $\nabla_z E_\psi(z;x)$ as:

1. Gradient of a scalar-valued function E (**Energy** function; LeCun et al., 2006).

2. Train a model $S_\psi(z;x)$ to directly approximate the gradients (**Score** function; Hyvärinen, 2005), bypassing energy estimation.

Shu, Lee, Nakayama and Cho, 2020.

Gradient-based Inference

Energy-based Inference

1. Initialize: $z = \mathbb{E}_{p_\theta(z|x)}[z]$

2. Repeat:

Backprop

$$z = z - \alpha \cdot \nabla_z E_\psi(z; x)$$

3. Output:

$$\hat{y} = \text{argmax}_y p(y \mid z, x)$$

Score-based Inference

1. Initialize: $z = \mathbb{E}_{p_\theta(z|x)}[z]$

2. Repeat:

$$z = z - \alpha \cdot S_\psi(z; x)$$

3. Output:

$$\hat{y} = \text{argmax}_y p(y \mid z, x)$$

Softmax only computed once at the end.
Not on every iteration

Results

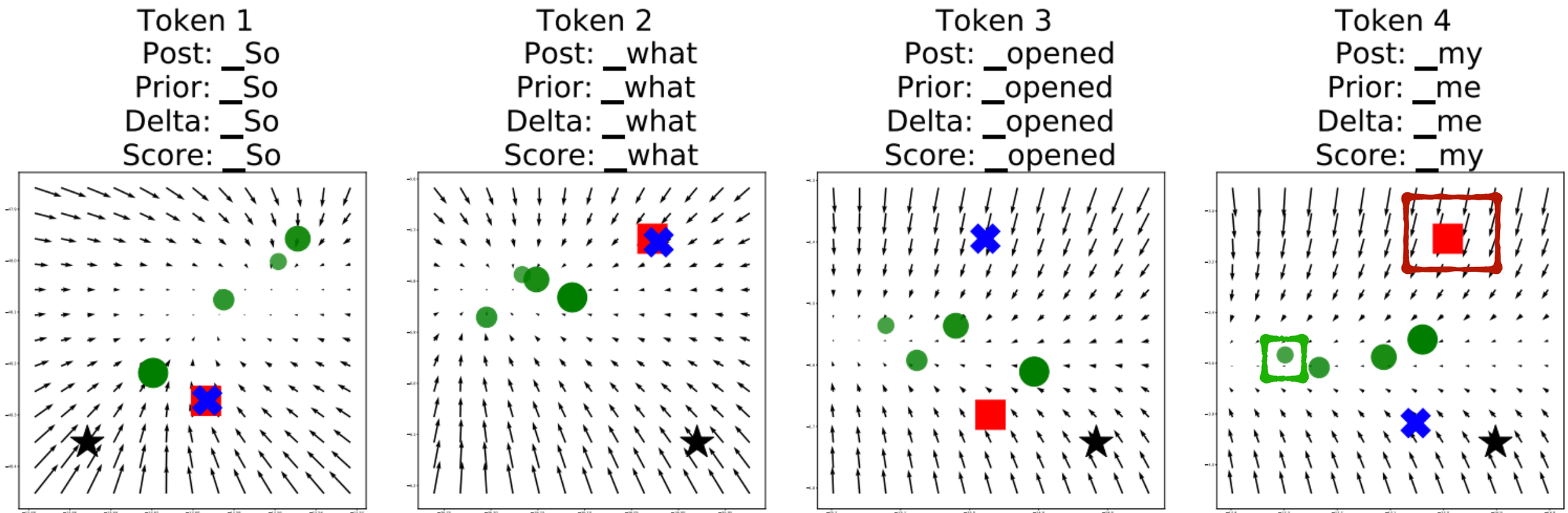
	WMT'14 En-De		WMT'16 Ro -> En		IWSLT'16 De -> En	
	BLEU	Speed	BLEU	Speed	BLEU	Speed
Transformer baseline, beam = 3	28.3	1x	31.5	1x	31.5	1x
Transformer baseline, beam = 1	27.5	1.1x	30.9	1.1x	31.1	1.1x
Latent-variable non-autoregressive	25.7	15x	28.4	34x	27.0	19x
+ Delta Inference	26.1	6.3x	29.0	19x	28.3	11x
+ Energy Inference	26.1	5.8x	28.8	17x	28.6	9.5x
+ Score Inference	26.3	10x	29.1	24x	28.8	13x
+ Score Inference + Latent Search	27.4	6.2x	30.4	15x	30.2	6.3x

- Score inference gives slightly better results than energy inference
  - While being much faster, as it avoids backprop.

Non-autoregressive NMT on WMT'14 En-De

		BLEU(%)	Gap	Speed
(Gu et al., 2018)	Baseline	23.4		
	Gu's Model (NAT + NPD S=100)	19.1	-4.3	2.3x
(Lee et al., 2018)	Baseline	24.5		
	Lee's Model (Adaptive NAR)	21.5	-3.0	1.9x
(Kaiser et al., 2018) (Roy et al., 2018)	Kaiser's baseline	23.5		
	Kaiser's Model (LT, Improved semhash)	19.8	-3.7	3.8x
	Roy's baseline	28.1		
	Roy's Model (VQVAE + EM + distillation)	26.7	-1.4	4.1x
(Ghaz et al., 2019)	Baseline	27.8		
	Ghaz's Model (CMLM with 4 iterations)	26.0	-1.8	
	Ghaz's Model (CMLM with 10 iterations)	26.9	-0.9	2~3x
(Shu et al., 2020) (Lee et al., 2020)	Baseline	28.3		
	LANMT + Delta Inference	26.1	-2.2	6.3x
	LANMT + Score Inference + Latent Search	27.4	-0.9	6.2x

Gradient and latent space visualization



★ : prior mean

● : score inference

✖ : delta inference

■ : posterior mean

Source	Was öffnete mir also die Augen?
Reference	So what opened my eyes ?
Posterior	So what opened my eyes ?
Prior	So what opened me eyes ?
Delta	So what opened me eyes ?
Score	So what opened my eyes ?

Shu, Lee, Nakayama and Cho, 2020.

Non-trivial, non-local changes

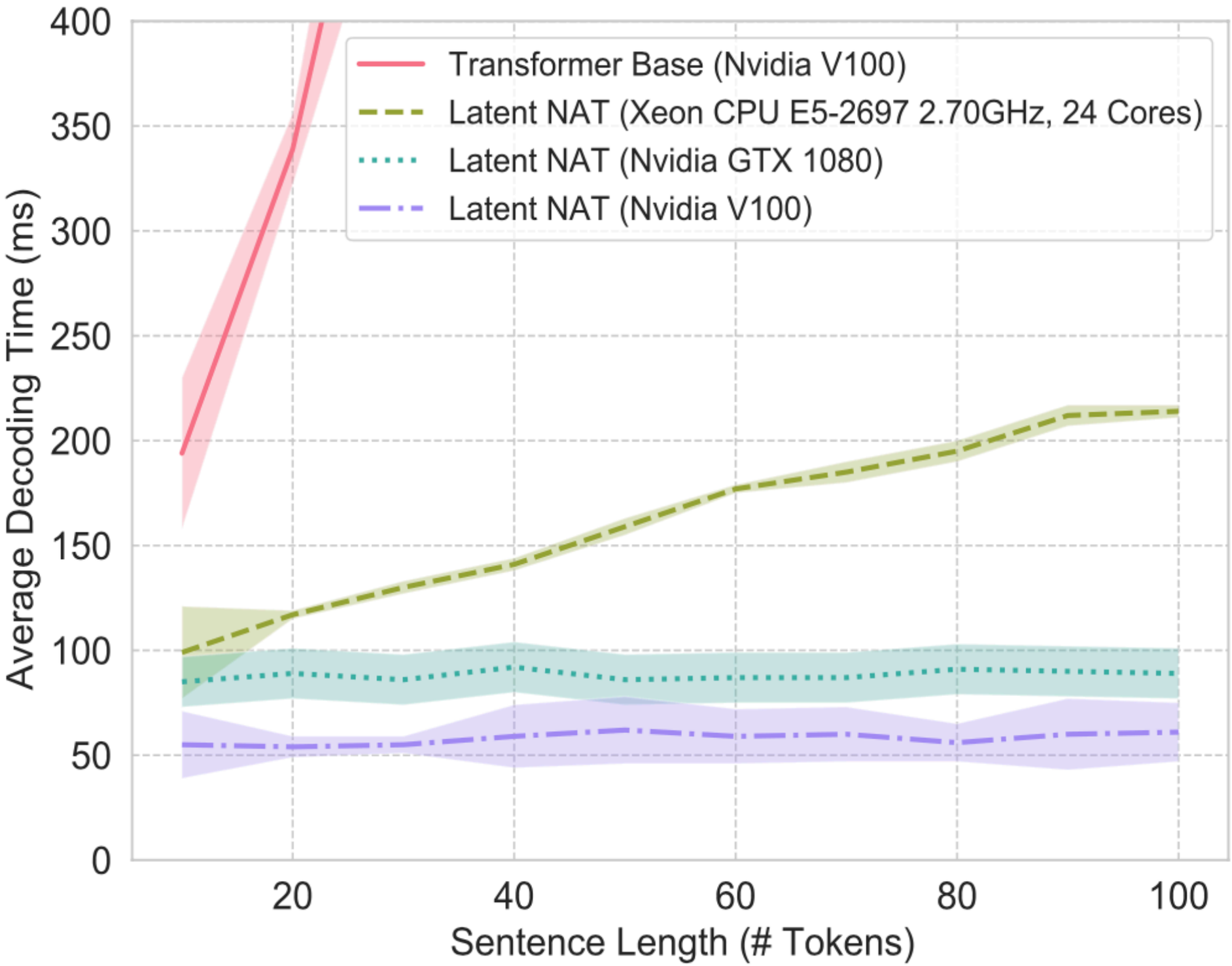
Example 1	
Source	There aren ’t many doctors in the west African country ; just one for every 5,000 people
Reference	In dem westafrikanischen Land gibt es nicht viele rzte, nur einen fr 5.000 Menschen
Original	Es gibt nicht viele rzte im westafrikanischen Land, nur eine fr 5.000 Menschen.
Refined	Im westafrikanischen Land gibt es nicht viele rzte, nur eine fr 5.000 Menschen.

Example 2	
Source	Costumes are expected to account for $ 1.2 billion dollars out of the $ 6.9 billion spent , according to the NRF .
Reference	Die Kostme werden etwa 1,2 Milliarden der 6,9 Milliarden ausgegebenen US-Dollar ausmachen, so der NRF.
Original	Es wird von, Kostme, dass sie die dem NRF ausgegebenen 6,9 Milliarden Dollar 1,2 Milliarden Dollar ausmachen.
Refined	Es wird erwartet, dass die Kostme nach Angaben des NRF 1,2 Milliarden Dollar aus den 6,9 Milliarden Dollar ausmachen.

Example 3	
Source	It was with this piece of Bedouin wisdom that the first ever chairman Wolfgang Henne described the history and fascination behind the “Helping Hands” society .
Reference	Mit dieser Beduinenweisheit beschrieb der erste Vorsitzende Wolfgang Henne die Geschichte und Faszination des Vereins “Helfende Hnde”.
Original	Der erste Vorsitzende Wolfgang Henne beschrieb mit dieser erste Weisheit in Bedouin” die Geschichte und Faszination hinter der “Helenden Hands” Gesellschaft
Refined	Mit diesem Stck Bedouin-Weisheit beschrieb der erste Vorsitzende Wolfgang Henne jemals die Geschichte und Faszination hinter der “Heling Hands” Gesellschaft

Shu, Lee, Nakayama and Cho, 2020.

Inference Latency (lower = better)
w.r.t Length and Computational Capacity



Is speed the only reason for going non-auto?

- Autoregressive models suffer from:
  - Exposure bias (due to teacher forcing)
  - Pathological degenerate generations (infinite repetitions)

```
response = openai.Completion.create(engine="davinci", prompt="roses are red, elephants are gray,
print(response["choices"][0]["text"])
```

```
 gray, and so on.

The color of an object is determined by the wavelength of light that it reflects. The color of a
rbs.

The color of an object is not determined by the wavelength of light that it emits.

The color of an object is not determined by the wavelength of light that it transmits.

The color of an object is determined by the wavelength of light that it reflects.
```

Degenerate GPT-3 generations found by Cho

Collaborators

Kyunghyun

Raphael

Elman

Hideki

Thank you. Questions?

jasonleeinf.github.io