Variational Neural Machine Translation

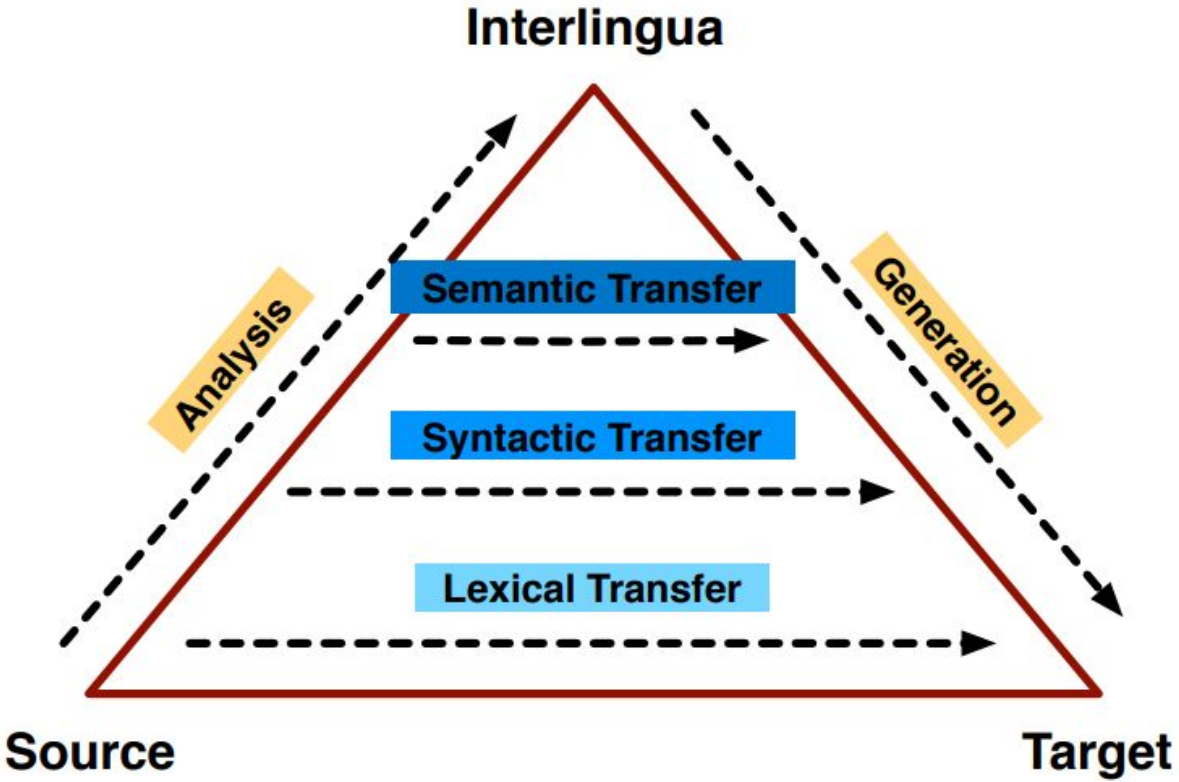
Zhang et. al.
Presenter: Michael Przystupa

Overview

1. Introduction
2. Background
3. Variational Neural Machine Translation
4. Experiments (Results for the most part)
5. ~~Related Work~~
6. Conclusion

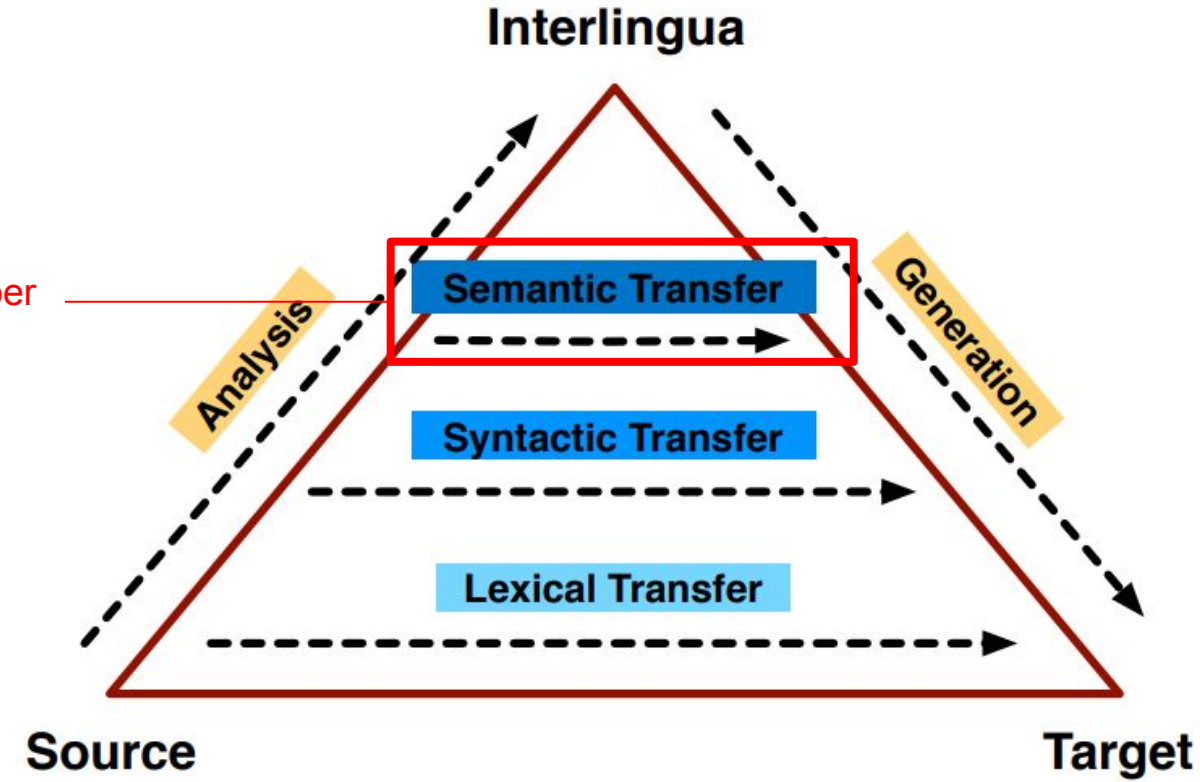
Disclaimer All code snippets are for illustrative purposes ONLY

Introduction



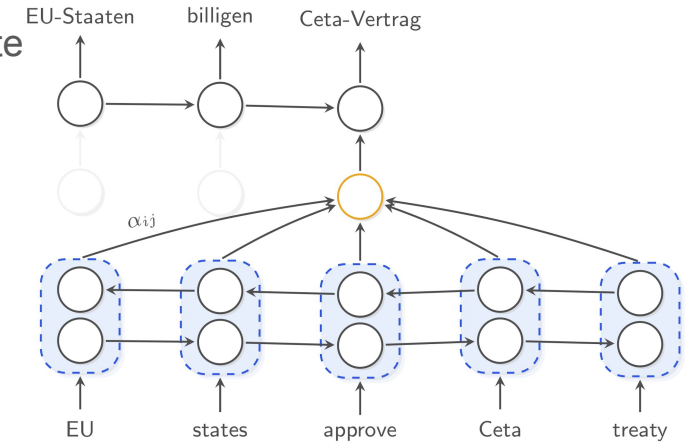
Introduction

Focus of this Paper



Introduction: Motivation

- S.O.T.A. Neural Machine Translation:
  - Seq2Seq model with attention performs well on translation
  - Models the following distribution P(Y | X) i.e. a discriminative approach
- Current Limitations of S.O.T.A:
  - Semantic meaning is not explicitly modeled
    - Relies on attention mechanism to do this
  - Attention mechanisms are not perfect
    - Tu et.al 2016 showed they can under/over translate
- Solution:
  - Let’s explicitly model semantics



Introduction: Assume Semantic Variable Z

$$p(\mathbf{y}|\mathbf{x}) = \int_z p(\mathbf{y}, z|\mathbf{x}) d_z = \int_z p(\mathbf{y}|z, \mathbf{x}) p(z|\mathbf{x}) d_z \quad (1)$$

Original Probability

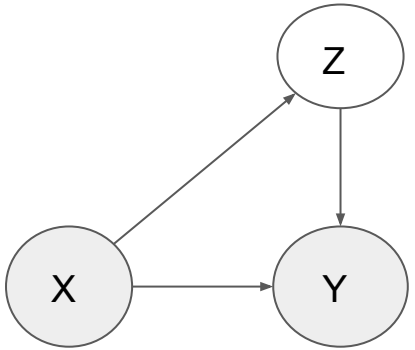
Introduce Semantic variable **Z**

Proposed Model

Problems with the model:

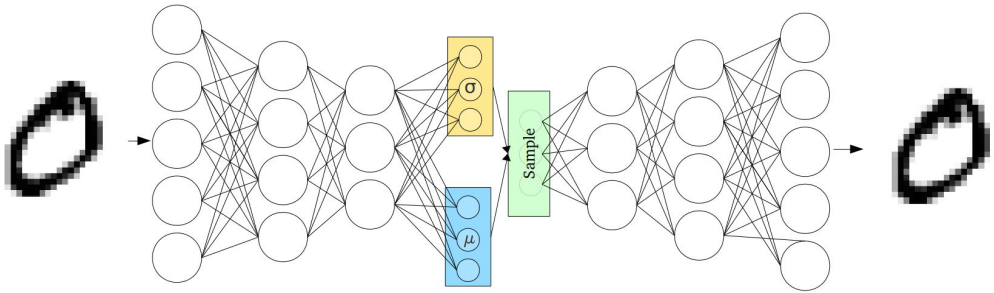
1. Posterior inference Intractable
2. Large scale learning is thus also difficult to do

Solution: Variational Autoencoders!



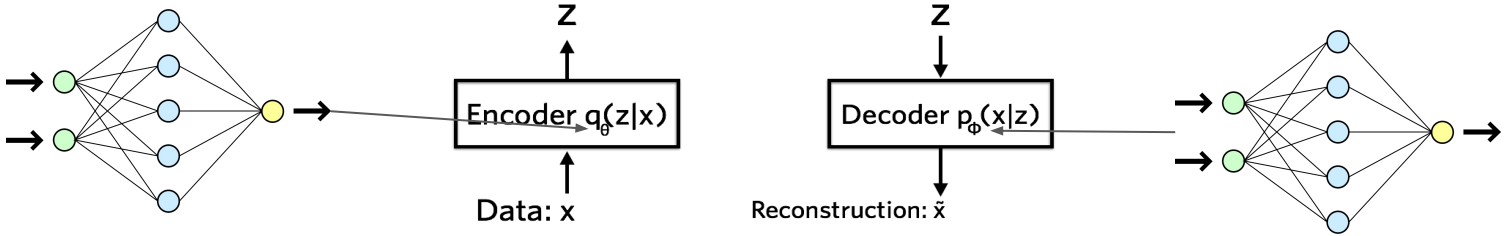
Background: VAEs

- You have data X and introduced latent variable Z
- Assumes your latent variable Z generated X
  - P(x,z) = P(x | z)P(z)
- Same intractability problems with posterior distribution
  - We will approximate it with a variational distribution



Background: VAEs

- Neural Approximation (Amortized inference)



- Reparameterization

$$\tilde{z} = \mu + \sigma \odot \epsilon \qquad (3)$$

$\epsilon \sim N(0, 1)$

- ELBO:

$$\mathcal{L}_{\text{VAE}}(\theta, \phi; \mathbf{x}) = -\,\mathbf{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \leq \log p_\theta(\mathbf{x}) \qquad (4)$$

Definition of KL

$$D_{\mathrm{KL}}(P \parallel Q) = -\sum_{x \in \mathcal{X}} P(x) \log\left(\frac{Q(x)}{P(x)}\right)$$

Variational Neural Machine Translation

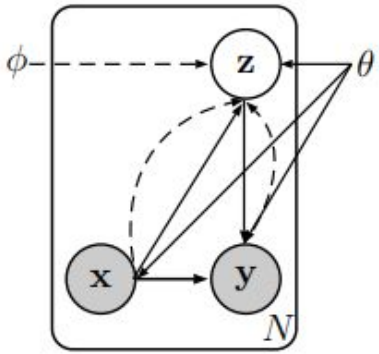
Using previous defined Model and The VAE loss we get following:

$$p(\mathbf{y}|\mathbf{x}) = \int_z p(\mathbf{y}, z|\mathbf{x})d_z = \int_z p(\mathbf{y}|z, \mathbf{x})p(z|\mathbf{x})d_z \tag{1}$$

$$\mathcal{L}_{\text{VAE}}(\theta, \phi; \mathbf{x}) = -\,\text{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \leq \log p_\theta(\mathbf{x}) \tag{4}$$

$$\mathcal{L}_{\text{VNMT}}(\theta, \phi; \mathbf{x}, \mathbf{y}) = -\text{KL}(q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})||p_\theta(\mathbf{z}|\mathbf{x})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p_\theta(\mathbf{y}|\mathbf{z}, \mathbf{x})] \tag{5}$$

We introduce a variational distribution $q_\phi(z \mid x, y)$ to approximate posterior



Variational Neural Machine Translation

$$p(\mathbf{y}|\mathbf{x}) = \int_z p(\mathbf{y}, z|\mathbf{x}) d_z = \int_z p(\mathbf{y}|z, \mathbf{x}) p(z|\mathbf{x}) d_z \quad (1)$$

$$+$$

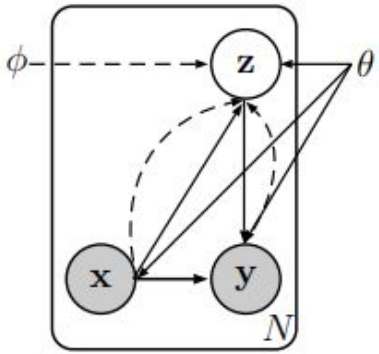
$$\mathcal{L}_{\text{VAE}}(\theta, \phi; \mathbf{x}) = -\,\text{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \leq \log p_\theta(\mathbf{x}) \quad (4)$$

$$=$$

$$\mathcal{L}_{\text{VNMT}}(\theta, \phi; \mathbf{x}, \mathbf{y}) = -\text{KL}(q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})||p_\theta(\mathbf{z}|\mathbf{x})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p_\theta(\mathbf{y}|\mathbf{z}, \mathbf{x})] \quad (5)$$

Normal Objective of neural machine translation

(Without KL term almost regular NMT)



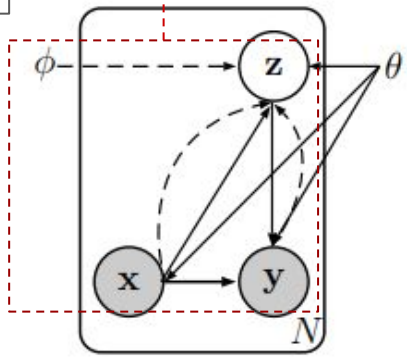
Variational Neural Machine Translation

$$p(\mathbf{y}|\mathbf{x}) = \int_z p(\mathbf{y}, z|\mathbf{x})dz = \int_z p(\mathbf{y}|z, \mathbf{x})p(z|\mathbf{x})dz \quad (1)$$

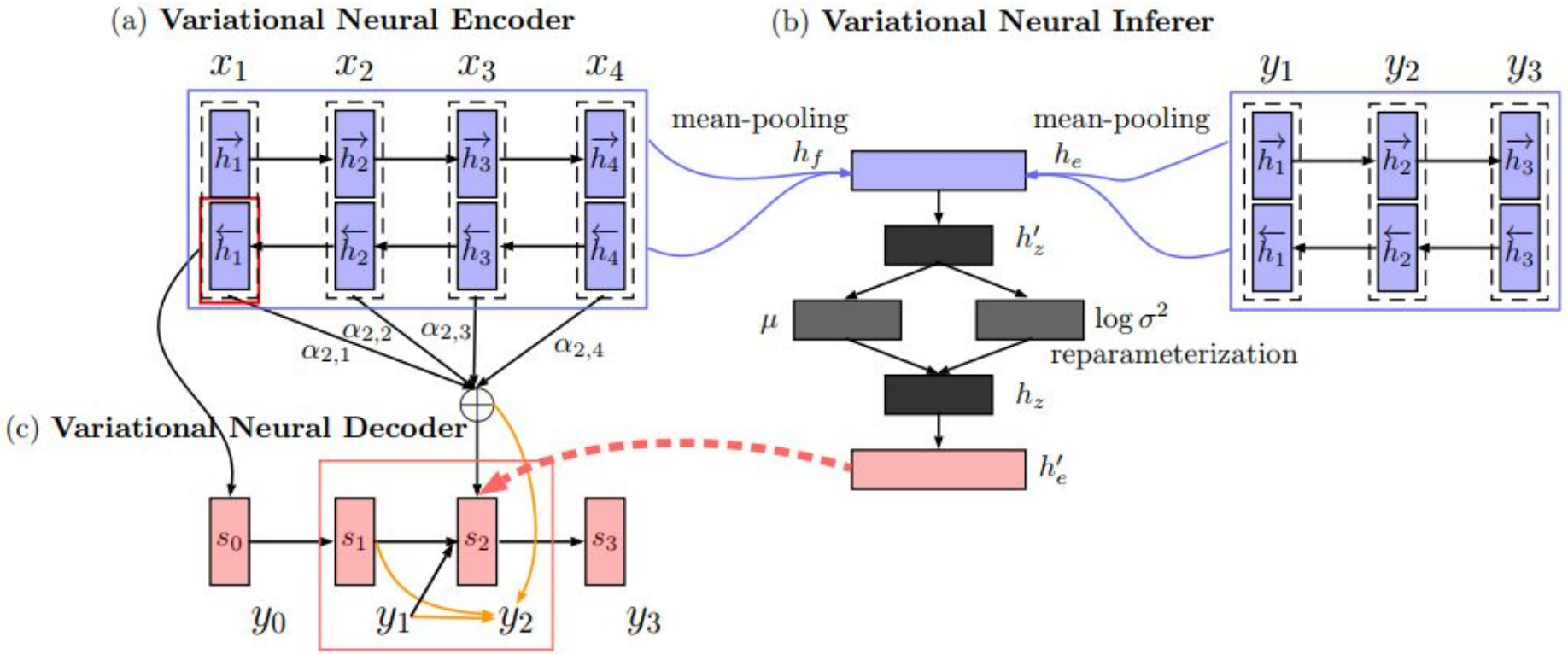
$$\mathcal{L}_{\text{VAE}}(\theta, \phi; \mathbf{x}) = -\text{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \leq \log p_\theta(\mathbf{x}) \quad (4)$$

$$\mathcal{L}_{\text{VNMT}}(\theta, \phi; \mathbf{x}, \mathbf{y}) = -\text{KL}(q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})||p_\theta(\mathbf{z}|\mathbf{x})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p_\theta(\mathbf{y}|\mathbf{z}, \mathbf{x})] \quad (5)$$

Regularize objective with approximate Posterior ϕ to prior Θ in model

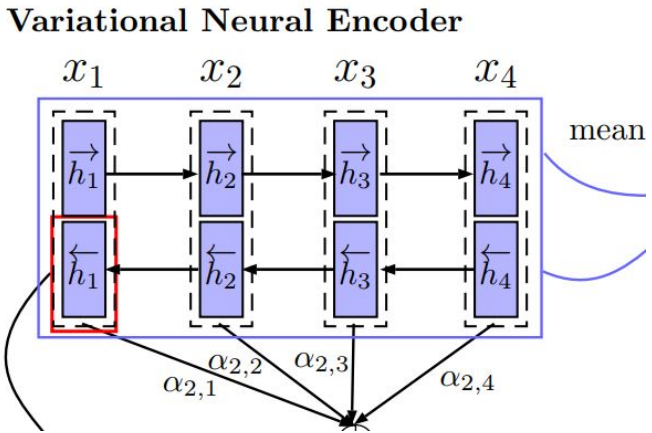


Variational Neural Machine Translation



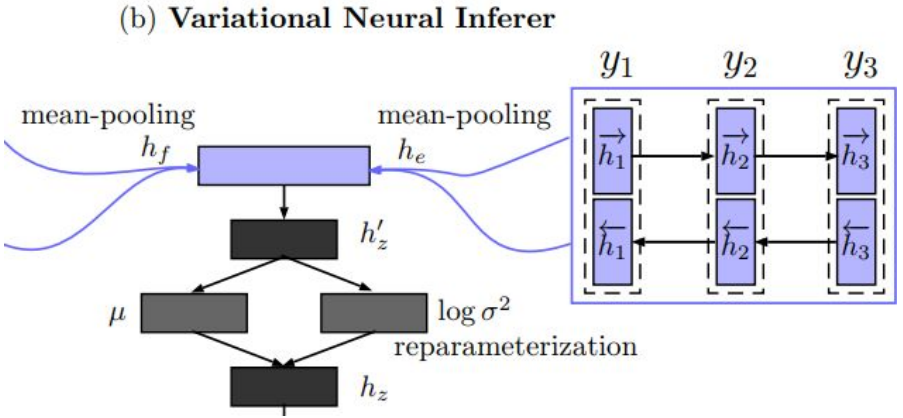
VNMT: Variational Neural Encoder

- Convert sentences into a continuous vector
- Follows largely approach of Bahdanau et al 2014:
  - Encode Sentences into annotation vectors $\mathbf{h}_j$, j ϵ(1, T)
  - Chose Bidirectional GRU for experiments
- Key difference:
  - Our encoder runs over both Source and Target!



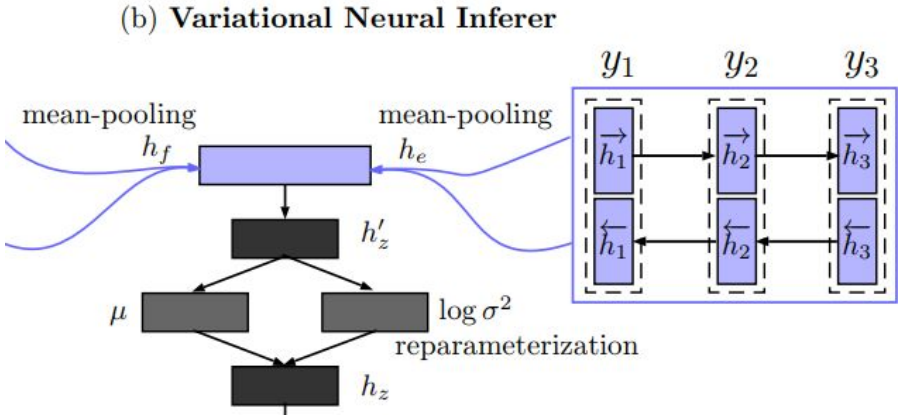
VNMT: Variational Neural Inferer

- Use annotation vectors to learn the distribution parameters
- We need to model 2 distributions:
  - Prior: $P_\theta(Z \mid X)$
  - Posterior: $P_\phi(Z \mid X, Y)$
- We use gaussian for both:
  - $N(\mathbf{z} ; \boldsymbol{\mu}, \boldsymbol{\sigma}^2 \mathbf{I})$



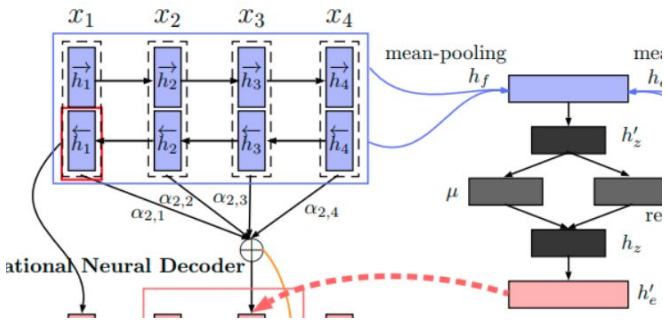
VNMT: Neural Posterior Approximator

- $q_\phi(z \mid x, y) = N(\mathbf{z} ; \boldsymbol{\mu}(x, y), \sigma^2(x, y)\, \mathbf{I})$
- Mean pool our annotation vectors for both source and target:
  - $\hat{\boldsymbol{h}}_{\mathbf{src}} = (1 / T_{input}) \sum_{i=1}^{T} \mathbf{h}_i$
- Project into a new “latent space”:
  - $\mathbf{h}_z\text{'} = g(\mathbf{W}_z^{(1)} [\hat{\boldsymbol{h}}_{src} ; \hat{\boldsymbol{h}}_{tgt}] + \mathbf{b}_z^{(1)})$
  - g is some activation function e.g. tanh
  - $[\hat{\boldsymbol{h}}_{src} ; \hat{\boldsymbol{h}}_{tgt}]$ concatenation of embeddings
- Our Normal Distribution:
  - $\boldsymbol{\mu} = W_\mu \mathbf{h}_z\text{'} + b_\mu$
  - $\log(\boldsymbol{\sigma}^2) = W_\sigma \mathbf{h}_z\text{'} + \mathbf{b}_\sigma$



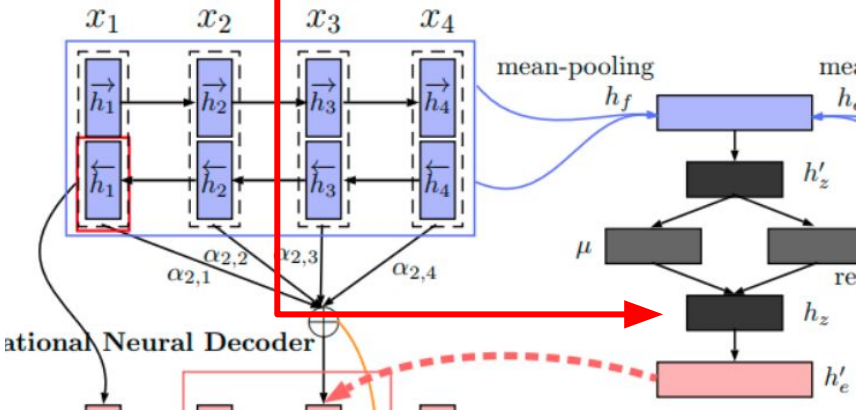
VNMT: Neural Prior Model

- Largely Same procedure as Neural Posterior Approximator
  - $\mathbf{h}_z = \boldsymbol{\mu}'(x) + \exp(\log\boldsymbol{\sigma}^2) * \epsilon, \epsilon \sim N(0, 1)$
    - Note that $\boldsymbol{\mu}'$ and $\log\boldsymbol{\sigma}^2$ are independent of y
  - Do one extra projection step
    - $\mathbf{h}_e' = g(W_z^{(2)}\mathbf{h}_z + b_z^{(2)})$
- During training we sample $\mathbf{h}_z$
  - Help prevent overfitting
- During inference $\mathbf{h}_z$ is deterministic
  - i.e. $\mathbf{h}_z = \boldsymbol{\mu}'(x)$



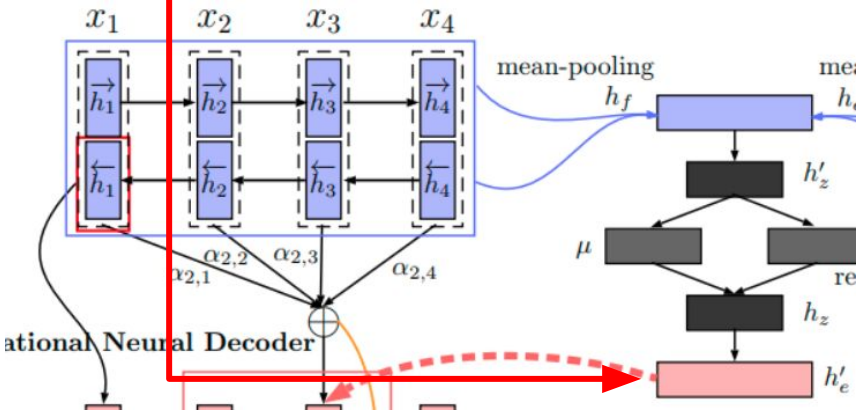
VNMT: Neural Prior Model

- Largely Same procedure as Neural Posterior Approximator
  - $\boldsymbol{\mu}' = W_{\mu}\mathbf{h_z}' + b_{\mu}$, $\log(\boldsymbol{\sigma}'^2) = W_{\sigma}\mathbf{h_z}' + \mathbf{b_{\sigma}}$
  - $\mathbf{h}_z = \boldsymbol{\mu}'(x) + \exp(\log\boldsymbol{\sigma}'^2) * \epsilon$, $\epsilon \sim N(0, 1)$
    - Note that $\boldsymbol{\mu}'$ and $\log\boldsymbol{\sigma}^2$ are independent of y
  - Do one extra projection step
    - $\mathbf{h}_e' = g(W_z^{(2)}\mathbf{h}_z + b_z^{(2)})$
- During Training we sample $\mathbf{h}_z$
  - Help prevent overfitting
- During Inference $\mathbf{h}_z$ is deterministic
  - i.e. $\mathbf{h}_z = \boldsymbol{\mu}'(x)$

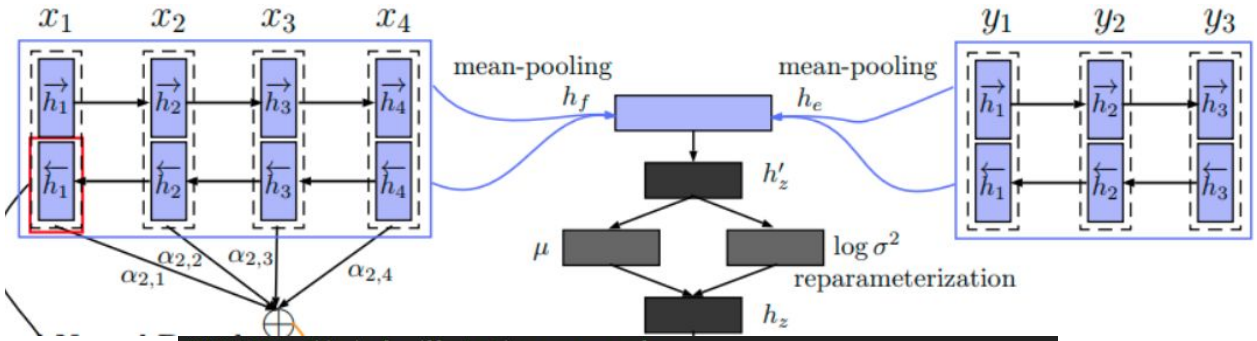


VNMT: Neural Prior Model

- Largely Same procedure as Neural Posterior Approximator
  - $\boldsymbol{\mu}' = W_{\mu}\mathbf{h}_z' + b_{\mu}$, $\log(\boldsymbol{\sigma}^2) = W_{\sigma}\mathbf{h}_z' + \mathbf{b}_{\sigma}$
  - $\mathbf{h}_z = \boldsymbol{\mu}'(x) + \exp(\log\boldsymbol{\sigma}^2) * \epsilon$, $\epsilon \sim N(0, 1)$
    - Note that $\boldsymbol{\mu}'$ and $\log\boldsymbol{\sigma}^2$ are independent of y
  - Do one extra projection step
    - $\mathbf{h}_e' = g(W_z^{(2)}\mathbf{h}_z + b_z^{(2)})$
- During Training we sample $\mathbf{h}_z$
  - Help prevent overfitting
- During Inference $\mathbf{h}_z$ is deterministic
  - i.e. $\mathbf{h}_z = \boldsymbol{\mu}'(x)$



VNMT: Pseudo Code for Neural Posterior



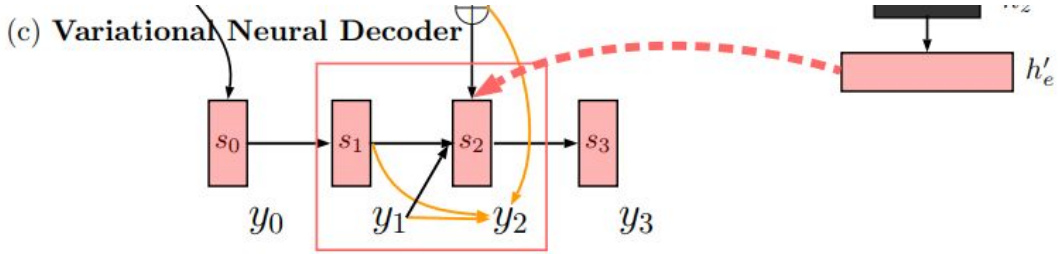

```
#DISCLAIMER: This is for illustration purposes only
#Sentences ==> word Embeddings
x_embeds, x_len, x_mask, y_sent = self.x_embed(x_sent, y=y_sent)
y_embeds, y_len, y_mask, y_indices = self.y_embed(y_sent, y=range(len(y_sent)), pack_seq=True)
#Encode SRC X and TGT Y
x_out, _ = self.encoder(x_embeds)
X, x_len = pad_packed_sequence(x_out, batch_first=self.b_f)

y_out, _ = self.encoder(y_embeds)
Y, y_len = pad_packed_sequence(y_out, batch_first=self.b_f)
#Mean pooling over Hidden vectors
X = torch.sum(X, dim=1) / x_len.unsqueeze(1).float()
Y = torch.sum(Y, dim=1) / y_len.unsqueeze(1).float()
# Sorting your targets to align with X again
Y = zip([r for r in Y], y_indices)
Y = sorted(Y, key=lambda x: x[1], reverse=False)
Y = torch.stack([y[0] for y in Y])

#Mean pool over the hidden states to
z_input = torch.cat([X, Y],dim=1)
z_mean, z_sig = self.posterior(z_input)
#sample from our variational distribution P(Z | X, Y)
with pyro.plate('z_minibatch'):
        semantics = pyro.sample('z_semantics', dist.Normal(z_mean, z_sig))
```

VNMT: Variational Neural Decoder

- How do we actually use $\mathbf{h}_e'$ for anything?
- We will include it as an additional input to the decoder
  - Remember that $\mathbf{h}_z$ encodes global semantic information
  - The meaning of the sentence is relevant throughout the whole sentence
- Other than that, it's same as Bahdanau et. al. 2014
  - i.e. they still use a context vector



VNMT: Variational Neural Decoder Change

Original Gated Recurrent Unit

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$
$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)$$

VNMT Gated Recurrent Unit (sorry for notation swap)

$$s_j = (1 - u_j) \odot s_{j-1} + u_j \odot \tilde{s}_j,$$
$$\tilde{s}_j = \tanh(W E_{y_j} + U[r_j \odot s_{j-1}] + C c_j + V \mathbf{h}'_e)$$
$$u_j = \sigma(W_u E_{y_j} + U_u s_{j-1} + C_u c_j + V_u \mathbf{h}'_e)$$
$$r_j = \sigma(W_r E_{y_j} + U_r s_{j-1} + C_r c_j + V_r \mathbf{h}'_e)$$

What's the difference?

(besides notation & format)

VNMT: Variational Neural Decoder Change

Original Gated Recurrent Unit

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$
$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)$$

VNMT Gated Recurrent Unit (sorry for notation swap)

$$s_j = (1 - u_j) \odot s_{j-1} + u_j \odot \tilde{s}_j,$$
$$\tilde{s}_j = \tanh(W E_{y_j} + U[r_j \odot s_{j-1}] + C c_j + V \mathbf{h}'_e)$$
$$u_j = \sigma(W_u E_{y_j} + U_u s_{j-1} + C_u c_j + V_u \mathbf{h}'_e)$$
$$r_j = \sigma(W_r E_{y_j} + U_r s_{j-1} + C_r c_j + V_r \mathbf{h}'_e)$$

We just include more inputs!

VNMT: Variational Neural Decoder Code Snippet

```
def forward(self, x_t, h_t_1, context, h_e):
    x_t, h_t_1, = x_t.squeeze(0), h_t_1.squeeze(0)
    #Calculate Update & Reset Gates
    r_t = sigmoid(self.W_r(x_t) + self.U_r(h_t_1) + self.C_r(context) + self.V_r(h_e))
    z_t = sigmoid(self.W_z(x_t) + self.U_z(h_t_1) + self.C_z(context) + self.V_z(h_e))
    # Produce new hidden state
    h_s = tanh(self.W_h(x_t) + self.U_h(r_t * h_t_1) + self.C_h(context) + self.V_h(h_e))
    h_t = (1 - z_t) * h_t_1 + z_t * h_s
    return h_t.unsqueeze(0), h_t.unsqueeze(0)
```

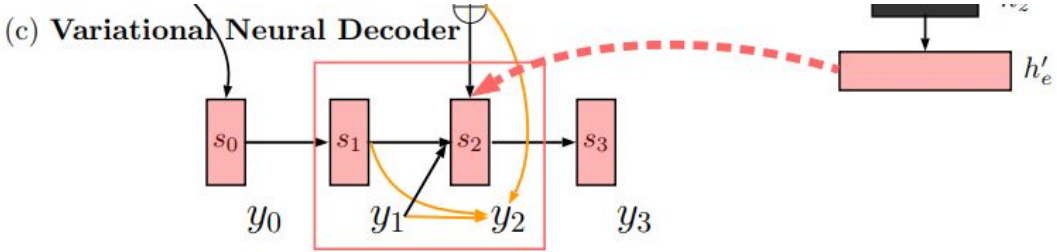
$$r_j = \sigma(W_r E_{y_j} + U_r s_{j-1} + C_r c_j + V_r \mathbf{h}'_e)$$
$$u_j = \sigma(W_u E_{y_j} + U_u s_{j-1} + C_u c_j + V_u \mathbf{h}'_e)$$
$$\tilde{s}_j = \tanh(W E_{y_j} + U[r_j \odot s_{j-1}] + C c_j + V \mathbf{h}'_e)$$
$$s_j = (1 - u_j) \odot s_{j-1} + u_j \odot \tilde{s}_j,$$

VNMT: Variational Neural Decoder

- How do we actually get the probability of each word?
- Lets use amortized inference:
  - You've seen this already because VAEs already do this
  - Learning free parameters from each data point is expensive
  - Let's learn a function instead to do that instead

$$p(\mathbf{y}|\mathbf{z},\mathbf{x}) = \prod_{j=1}^{T_e} p(y_j|y_{<j},\mathbf{z},\mathbf{x}) \qquad (12)$$

where $p(y_j|y_{<j},\mathbf{z},\mathbf{x}) = g'(y_{j-1}, s_{j-1}, c_j)$



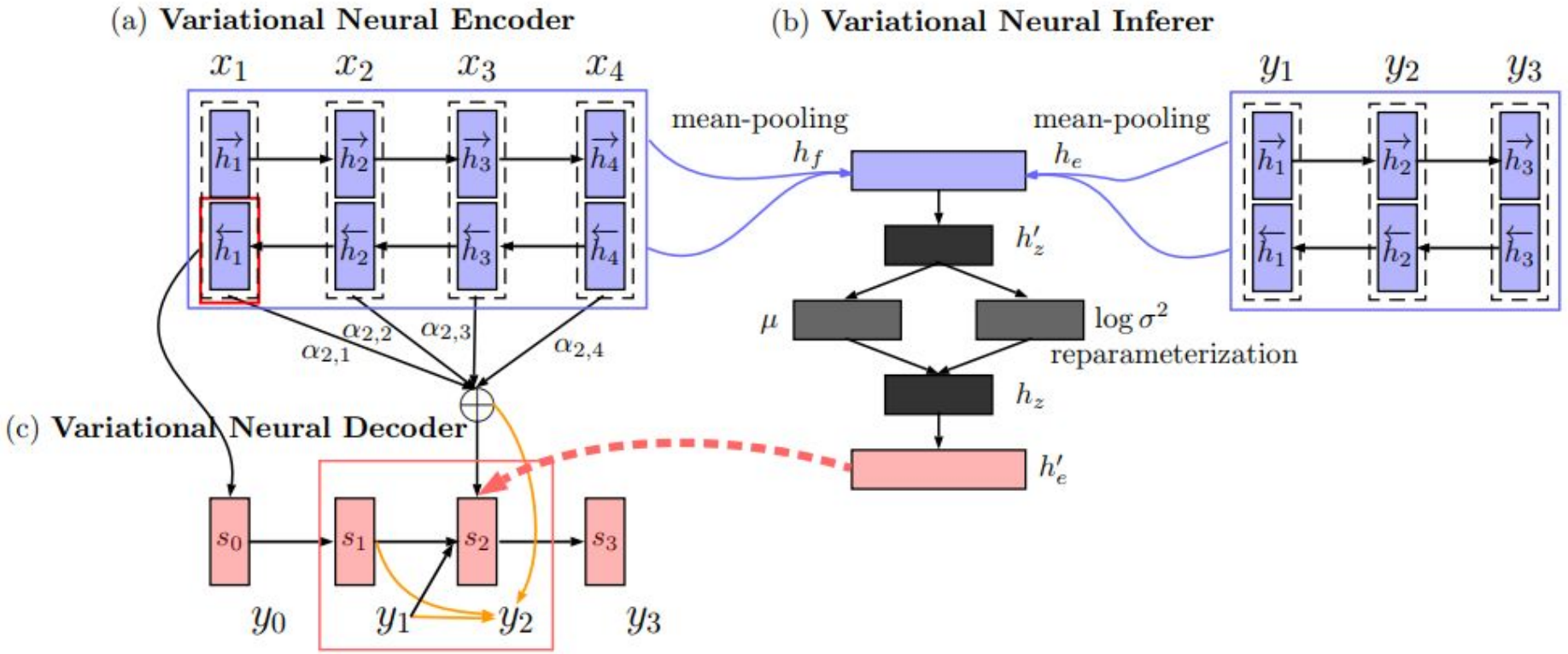
VNMT: Model Training

- Approximate Posterior with “Monte Carlo method”
  - i.e. sample L trajectories to approximate gradient
  - Authors Use L = 1
- With L = 1 we have original NMT objective regularized by KL term
  - This is just the ELBO in regular VAEs

$$\mathcal{L}(\theta, \phi) \simeq -\mathbf{KL}(q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})||p_\theta(\mathbf{z}|\mathbf{x})) + \frac{1}{L}\sum_{l=1}^{L}\sum_{j=1}^{T_e} \log p_\theta(y_j|y_{<j}, \mathbf{x}, \mathbf{h}_z^{(l)}) \quad (13)$$

$$\text{where } \mathbf{h}_z^{(l)} = \mu + \sigma \odot \epsilon^{(l)} \text{ and } \epsilon^{(l)} \sim \mathcal{N}(0, \mathbf{I})$$

VNMT: Review of Diagram



Experiments: Overview

- Did experiments on following language pairs:
  - Chinese - English
  - English - German
- Compared to GroundHog and Moses systems
- Experiments
  - Compare BLEU-4 scores to other systems
  - Long sentence generation
  - Ablation study (does the KL term help?)

Experiments: Results on Chinese-English Translation

System	MT05	MT02	MT03	MT04	MT06	MT08	AVG
Moses	33.68	34.19	34.39	35.34	29.20	22.94	31.21
GroundHog	31.38	33.32	32.59	35.05	29.80	22.82	30.72
VNMT w/o KL	31.40	33.50	32.92	34.95	28.74	22.07	30.44
VNMT	32.25	**34.50**$^{++}$	33.78^{++}	**36.72**$^{\Uparrow ++}$	**30.92**$^{\Uparrow ++}$	**24.41**$^{\uparrow ++}$	**32.07**

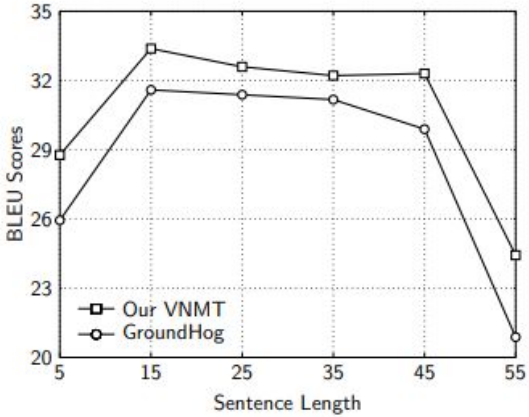
Points about VNMT w/o KL:

1. Modeling semantics explicitly DOES make a difference
2. The VNMT model is larger than GroundHog's model yet underperforms w/o KL term

Results on Long Sentences

- Graph shows BLEU score on disjoint sets of different sentence lengths
- Table shows results on synthetic long sentences (> 50)

System	MT05	MT02	MT03	MT04	MT06	MT08
GroundHog	18.23	22.20	20.19	21.67	19.11	13.41
VNMT	**21.31**	**26.02**	**23.78**	**25.81**	**21.81**	**15.59**



Results on English-German Translation

System	Architecture	BLEU
Existing end-to-end NMT systems		
Jean et al. (2015)	RNNSearch	16.46
Jean et al. (2015)	RNNSearch + unk replace	18.97
Jean et al. (2015)	RNNsearch + unk replace + large vocab	19.40
Luong et al. (2015a)	LSTM with 4 layers + dropout + local att. + unk replace	20.90
Our end-to-end NMT systems		
this work	RNNSearch	16.40
	VNMT	17.13^{++}
	VNMT + unk replace	19.58^{++}

Translation Analysis

Source	两国官员确定了今后会谈的日程和模式,建立起进行持续对话的机制,此举标志着巴印对话进程在中断两年后重新启动,为两国逐步解决包括克什米尔争端在内的所有悬而未决的问题奠定了基础,体现了双方可贵的和平诚意。
Reference	*the officials of the two countries have established the mechanism for continued dialogue down the road, including a confirmed schedule and model of the talks. this symbolizes the restart of the dialogue process between pakistan and india after an interruption of two years and has paved a foundation for the two countries to sort out gradually all the questions hanging in the air, including the kashmir dispute. it is also a realization of their precious sincerity for peace.*
Moses	*officials of the two countries set the agenda for future talks , and the pattern of a continuing dialogue mechanism . this marks a break in the process of dialogue between pakistan and india , two years after the restart of the two countries including kashmir dispute to gradually solve all the outstanding issues have laid the foundation of the two sides showed great sincerity in peace .*
GroundHog	*the two countries have decided to set up a mechanism for conducting continuous dialogue on the agenda and mode of the talks . this indicates that the ongoing dialogue between the two countries has laid the foundation for the gradual settlement of all outstanding issues including the dispute over kashmir .*
VNMT	*the officials of the two countries set up a mechanism for holding a continuous dialogue on the agenda and mode of the future talks, and this indicates that the ongoing dialogue between pakistan and india has laid a foundation for resolving all outstanding issues , including the kashmir disputes , and this serves as a valuable and sincere peace sincerity .*

Conclusion

- VAE + NMT = Great Success!
- Longer Sentence Translation is more robust
- Regularization helps robustness of model
- Future work:
  - Lexical latent variables
    - Spoiler: Somebody already did this