A Surprisingly Effective Fix for Deep Latent Variable Modeling of Text

Bohan Li*[1], Junxian He*[1], Graham Neubig[1], Taylor Berg-Kirkpatrick[2], Yiming Yang[1]

Core idea:

- **Pretraining encoder using AE objective**
- **Initializing VAE with pertained AE and training with FB objective**

Results:

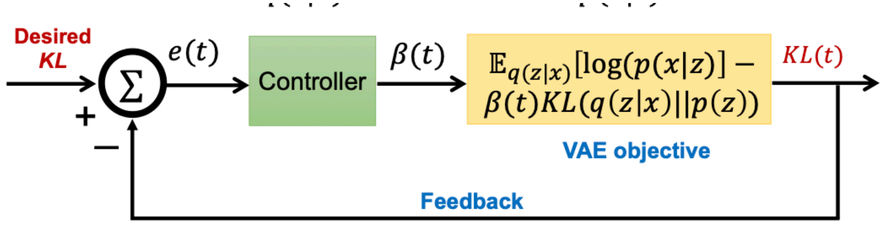
	Yahoo				
LSTM-LM	60.75	-	-	-	-
VAE	61.52	329.10	0	0.00	329.10
+ anneal	61.21	328.80	0	0.00	328.80
+ cyclic	66.93	333.80	4	2.83	336.63
+ aggressive	59.77	322.70	15	5.70	328.40
+ FBP ($\lambda = 9$)	62.59	322.91	6	9.08	331.99
+ FBP ($\lambda = 7$)	62.76	324.66	5	7.03	331.69
+ FBP ($\lambda = 5$)	62.78	326.26	3	5.07	331.32
+ FBP ($\lambda = 3$)	62.88	328.13	2	3.06	331.19
Ours ($\lambda = 6$)	59.23	317.39	32	12.09	329.48
Ours ($\lambda = 8$)	**59.51**	**315.31**	**32**	15.02	330.33
Ours ($\lambda = 9$)	59.60	315.09	32	15.49	330.58

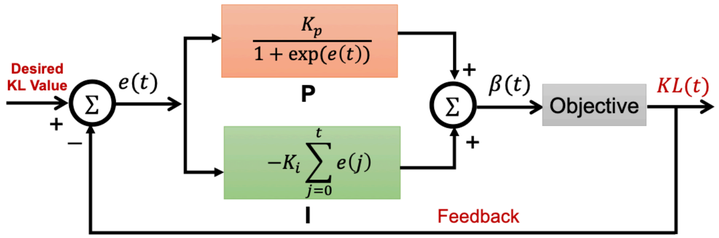
ControlVAE: Controllable Variational Autoencoder (ICML 2020)

Huajie Shao et al

Core ideas:

- Enhance β-VAE by controlling β with a new non-linear PI controller, β(t). The controller is not learnable, with Kp, Ki, Desired KL being hyperparameters.





- Intuition: Comparing error between Desired KL and Sampled KL (from inference output); if Sampled KL is too small, β(t) will be small so KL-divergence is encouraged to grow, and vice versa.

ControlVAE: Controllable Variational Autoencoder (ICML 2020)

Huajie Shao et al

Core ideas:

- PI Algorithm

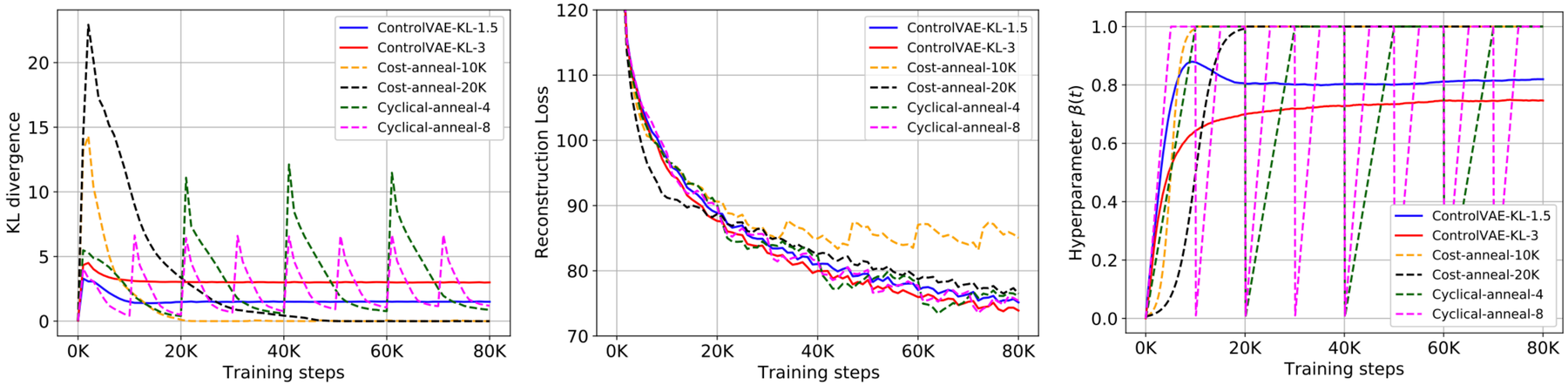
Algorithm 1 PI algorithm.

Input: desired KL v_{kl}, coefficients K_p, K_i, max/min value β_{max}, β_{min}, iterations N
Output: hyperparameter $\beta(t)$ at training step t
Initialization: $I(0) = 0$, $\beta(0) = 0$
for $t = 1$ **to** N **do**
  Sample KL-divergence, $\hat{v}_{kl}(t)$
  $e(t) \leftarrow v_{kl} - \hat{v}_{kl}(t)$
  $P(t) \leftarrow \frac{K_p}{1+\exp(e(t))}$
  if $\beta_{min} \leq \beta(t-1) \leq \beta_{max}$ **then**
    $I(t) \leftarrow I(t-1) - K_i e(t)$
  else
    $I(t) = I(t-1)$ // Anti-windup
  end if
  $\beta(t) = P(t) + I(t) + \beta_{min}$
  if $\beta(t) > \beta_{max}$ **then**
    $\beta(t) = \beta_{max}$
  end if
  if $\beta(t) < \beta_{min}$ **then**
    $\beta(t) = \beta_{min}$
  end if
  Return $\beta(t)$
end for

ControlVAE: Controllable Variational Autoencoder (ICML 2020)

Huajie Shao et al

Results:



(a) KL divergence (b) Reconstruction loss (c) $\beta(t)$

Methods/metric	Dis-1	Dis-2	self-BLEU-2	self-BLEU-3	PPL
ControlVAE-KL-35	**6.27K** $\pm$ 41	**95.86K** $\pm$ 1.02K	**0.663** $\pm$ 0.012	**0.447** $\pm$ 0.013	**8.81** $\pm$ 0.05
ControlVAE-KL-25	6.10K $\pm$ 60	83.15K $\pm$ 4.00K	0.698 $\pm$ 0.006	0.495 $\pm$ 0.014	12.47 $\pm$ 0.07
Cost anneal-KL-17	5.71K $\pm$ 87	69.60K $\pm$ 1.53K	0.721 $\pm$ 0.010	0.536 $\pm$ 0.008	16.82 $\pm$ 0.11
Cyclical (KL = 21.5)	5.79K $\pm$ 81	71.63K $\pm$ 2.04K	0.710 $\pm$ 0.007	0.524 $\pm$ 0.008	17.81 $\pm$ 0.33

Implicit Deep Latent Variable Models for Text Generation (EMNLP 2019)

Le Fang†, Chunyuan Li§, Jianfeng Gao§, Wen Dong†, Changyou Chen†

Core ideas:

- Attributing posterior collapse to the restrictive Gaussian assumption, and advocate more flexible sample-based posterior representation.
- Proposed iVAE:
  - Instead of assuming posterior as Gaussian, use sampling mechanism for $q_\phi(z|x)$. $z_{x,i} = G_\phi(x, \varepsilon_i)$, $\varepsilon_i \sim q(\varepsilon)$
  - How to evaluate KL on this? Use dual form of $KL(q_\phi(z|x) \parallel p(z))$:
  - New iVAE objective:

$$\begin{aligned} &\mathrm{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})\right) \qquad (7)\\ =&\max_{\nu} \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \nu_{\boldsymbol{\psi}}(\boldsymbol{x},\boldsymbol{z}) - \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})} \exp(\nu_{\boldsymbol{\psi}}(\boldsymbol{x},\boldsymbol{z})), \end{aligned}$$

$$\begin{aligned} \mathcal{L}_{\text{iVAE}} = &\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}} \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \\ &- \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}} \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \nu_{\boldsymbol{\psi}}(\boldsymbol{x},\boldsymbol{z}) \\ &+ \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}} \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})} \exp(\nu_{\boldsymbol{\psi}}(\boldsymbol{x},\boldsymbol{z})), \qquad (8) \end{aligned}$$

  - Here, ν_ψ is a MLP taking in (x, z).

Implicit Deep Latent Variable Models for Text Generation (EMNLP 2019)

Le Fang†, Chunyuan Li§, Jianfeng Gao§, Wen Dong†, Changyou Chen†

Core ideas:

- Training Scheme:
  - Sample a mini-batch of $\boldsymbol{x}_i \sim \mathcal{D}$, $\epsilon_i \sim q(\epsilon)$, and generate $\boldsymbol{z}_{\boldsymbol{x}_i,\epsilon_i} = G(\boldsymbol{x}_i, \epsilon_i; \boldsymbol{\phi})$; Sample a mini-batch of $\boldsymbol{z}_i \sim p(\boldsymbol{z})$.
  - Update $\boldsymbol{\psi}$ in $\nu_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{z})$ to maximize

$$\sum_i \nu_{\boldsymbol{\psi}}(\boldsymbol{x}_i, \boldsymbol{z}_{\boldsymbol{x}_i,\epsilon_i}) - \sum_i \exp(\nu_{\boldsymbol{\psi}}(\boldsymbol{x}_i, \boldsymbol{z}_i)) \quad (9)$$

  - Update parameters $\{\boldsymbol{\phi}, \boldsymbol{\theta}\}$ to maximize

$$\sum_i \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i|\boldsymbol{z}_{\boldsymbol{x}_i,\epsilon_i}) - \sum_i \nu_{\boldsymbol{\psi}}(\boldsymbol{x}_i, \boldsymbol{z}_{\boldsymbol{x}_i,\epsilon_i}) \quad (10)$$

Implicit Deep Latent Variable Models for Text Generation (EMNLP 2019)

Le Fang†, Chunyuan Li§, Jianfeng Gao§, Wen Dong†, Changyou Chen†

Core ideas:

- Proposed Mutual Information Regularized iVAE
  - Replace –KL ($q_\phi(z|x) \parallel p(z)$) with –KL ($q_\phi(z) \parallel p(z)$), where $q_\phi(z)=\int q(x)q_\phi(z|x)dx$, estimated by ancestral sampling in practice.
  - This objective also maximize the mutual information I(x,z), as they claimed.
  - The new objective:

$$\begin{aligned} &\mathrm{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}) \parallel p(\boldsymbol{z})\right) \qquad (13)\\ =&\max_{\nu} \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z})} \nu_{\boldsymbol{\psi}}(\boldsymbol{z}) - \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})} \exp(\nu_{\boldsymbol{\psi}}(\boldsymbol{z})). \end{aligned}$$

$$\begin{aligned} \mathcal{L}_{\mathrm{iVAE_{MI}}} = &\mathbb{E}_{\boldsymbol{x}\sim D} \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \qquad (14)\\ &- \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z})} \nu_{\boldsymbol{\psi}}(\boldsymbol{z}) + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})} \exp(\nu_{\boldsymbol{\psi}}(\boldsymbol{z})), \end{aligned}$$

  - The only difference with iVAE is that ν_ψ is a MLP taking in only z.

Implicit Deep Latent Variable Models for Text Generation (EMNLP 2019)

Le Fang†, Chunyuan Li§, Jianfeng Gao§, Wen Dong†, Changyou Chen†

Results:

Methods	-ELBO↓	PPL↓	KL↑	MI↑	AU↑
Dataset: PTB					
VAE	102.6	108.26	1.08	0.8	2
β(0.5)-VAE	104.5	117.92	**7.50**	3.1	5
SA-VAE	102.6	107.71	1.23	0.7	2
Cyc-VAE	103.1	110.50	3.48	1.8	5
iVAE	**87.6**	**54.46**	6.32	**3.5**	**32**
iVAE$_{\text{MI}}$	**87.2**	**53.44**	**12.51**	**12.2**	**32**
Dataset: Yahoo					
VAE	328.6	61.21	0.0	0.0	0
β(0.4)-VAE	328.7	61.29	6.3	2.8	8
SA-VAE	327.2	60.15	5.2	2.7	10
Lag-VAE	326.7	59.77	5.7	2.9	15
iVAE	**309.5**	**48.22**	**8.0**	**4.4**	**32**
iVAE$_{\text{MI}}$	**309.1**	**47.93**	**11.4**	**10.7**	**32**
Dataset: Yelp					
VAE	357.9	40.56	0.0	0.0	0
β(0.4)-VAE	358.2	40.69	4.2	2.0	4
SA-VAE	355.9	39.73	2.8	1.7	8
Lag-VAE	355.9	39.73	3.8	2.4	11
iVAE	**348.2**	**36.70**	**7.6**	**4.6**	**32**
iVAE$_{\text{MI}}$	**348.7**	**36.88**	**11.6**	**11.0**	**32**

FlowPrior: Learning Expressive Priors for Latent Variable Sentence Models (NAACL 2021)

Xiaoan Ding, Kevin Gimpel

Core ideas:

- Using normalizing flows (NF) for the prior distribution. In this paper, using *real-valued non-volume preserving* (real NVP) transformations (Dinh et al., 2016)
  - $z_L = f_L \circ f_{L-1} \circ \ldots \circ f_1(z_0)$, $z_0 \sim N(0, I)$, z_L is the sentence latent variable.

$$z_0 = f_1^{-1} \circ \ldots \circ f_{L-1}^{-1} \circ f_L^{-1}(z_L) \tag{5}$$

$$\log p_\psi(z_L) = \log p_0(z_0) - \sum_{l=1}^{L} \log \left| \det \left(\frac{\partial f_l(z_{l-1})}{\partial z_{l-1}} \right) \right| \tag{6}$$

  - Inspired by Real-NVP, the choice of f is an affine transformation, which is f: $z_{i-1} \rightarrow z_i$

$$z_i^{(1:d)} = z_{i-1}^{(1:d)}$$
$$z_i^{(d+1:D)} = z_i^{(d+1:D)} \odot \exp(s(z_{i-1}^{(1:d)})) + t(z_{i-1}^{(1:d)})$$

    - 1) easily invertible
    - 2) its Jacobian determinant is easy to compute. (Do not require inverse/Jacobian of s and t)
  - So we can model s and t using NNs, denoted by $p_\psi(z)$

$$\mathbf{J} = \begin{bmatrix} \mathbb{I}_d & \mathbf{0}_{d \times (D-d)} \\ \frac{\partial \mathbf{y}_{d+1:D}}{\partial \mathbf{x}_{1:d}} & \operatorname{diag}(\exp(s(\mathbf{x}_{1:d}))) \end{bmatrix}$$

FlowPrior: Learning Expressive Priors for Latent Variable Sentence Models (NAACL 2021)

Xiaoan Ding, Kevin Gimpel

Core ideas:

- New Objective: Using importance weighting + Monte Carlo for KL

$$\mathcal{L}(\theta, \phi, \psi; x) = \log \frac{1}{N} \sum_{i=1}^{N} \frac{p_\theta(x|z^{(i)}) p_\psi(z^{(i)})}{q_\phi(z^{(i)}|x)}$$

$$+ \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(x|z^{(i)}) - \mathrm{KL}_{\phi,\psi}(x, \{z^{(i)}\}_{i=1}^{N})$$

$$\text{s.t. } z^{(i)} \sim q_\phi(z|x) \quad (7)$$

FlowPrior: Learning Expressive Priors for Latent Variable Sentence Models (NAACL 2021)

Xiaoan Ding, Kevin Gimpel

Core ideas:

- Training Scheme

1. Draw N samples $z_L^{(1)}, z_L^{(2)}, ..., z_L^{(N)}$ from the inference network using the reparameterization trick.
2. Perform the inverse transformation to get the image of each point under the base distribution: $z_0^{(1)}, z_0^{(2)}, ..., z_0^{(N)}$.
3. Compute the exact log likelihood of the sample prior with change of variable theorem (Eq. 6).
4. Compute and backpropagate the loss (Eq. 7).

FlowPrior: Learning Expressive Priors for Latent Variable Sentence Models (NAACL 2021)

Xiaoan Ding, Kevin Gimpel

Results:

Model	PPL(↓)	Recon(↓)	KL	AU(↑)	MI(↑)
VAE	101.40	101.28	0.00	0	0.00
Cyc-VAE	107.73	101.17	2.01	5	1.24
Lag-VAE	100.25	100.41	1.04	3	0.79
VAE + FB	101.56	99.84	4.46	32	0.90
Pre-VAE + FB	96.35	94.52	8.15	32	6.30
MoG-VAE	98.22	100.54	0.00	0	0.00
MoG-VAE + FB	97.50	99.44	2.35	32	0.68
Vamp-VAE	98.27	100.56	0.00	0	0.00
Vamp-VAE + FB	97.83	99.53	2.31	32	0.72
FlowPrior	94.72	98.46	3.28	2	2.25
FlowPrior + FB	93.58	99.20	7.21	31	2.83

Table 1: Language modeling results on PTB dataset.

Adversarial Poincaré Variational Autoencoder (APo- VAE) (NAACL 2021)

Shuyang Dai, Zhe Gan, Yu Cheng, Chenyang Tao, Lawrence Carin, Jingjing Liu,

Core ideas:

- Natural languages have latent hierarchy, but prior distribution from Euclidean space couldn't capture it, so resorting to Riemannian Geometry.
- Proposing a prior based on Poincaré Ball Model in Hyperbolic space, with the following advantages:

$$\mathbb{B}_c^n := \{\boldsymbol{z} \in \mathbb{R}^n \,|\, c\|\boldsymbol{z}\|^2 < 1\}$$

  - It implies significant representation ability.
  - It's a well defined compact metric space, with well defined algebraic operations that allows back-propagation.

$$\boldsymbol{z} \oplus_c \boldsymbol{z}' := \frac{(1 + 2c\langle \boldsymbol{z}, \boldsymbol{z}' \rangle + c\|\boldsymbol{z}'\|^2)\boldsymbol{z} + (1 - c\|\boldsymbol{z}\|^2)\boldsymbol{z}'}{1 + 2c\langle \boldsymbol{z}, \boldsymbol{z}' \rangle + c^2\|\boldsymbol{z}\|^2\|\boldsymbol{z}'\|^2}. \tag{4}$$

$$\exp_{\boldsymbol{\mu}}^c(\boldsymbol{u}) := \boldsymbol{\mu} \oplus_c (\tanh(\sqrt{c}\frac{\lambda_{\boldsymbol{\mu}}^c\|\boldsymbol{u}\|}{2})\frac{\boldsymbol{u}}{\sqrt{c}\|\boldsymbol{u}\|}),$$

$$\log_{\boldsymbol{\mu}}^c(\boldsymbol{y}) := \frac{2}{\sqrt{c}\lambda_{\boldsymbol{\mu}}^c}\tanh^{-1}(\sqrt{c}\|\boldsymbol{\kappa}_{\boldsymbol{\mu},\boldsymbol{y}}\|)\frac{\boldsymbol{\kappa}_{\boldsymbol{\mu},\boldsymbol{y}}}{\|\boldsymbol{\kappa}_{\boldsymbol{\mu},\boldsymbol{y}}\|}, \tag{5}$$

$$P_{\mathbf{0}\to\boldsymbol{\mu}}^c(\boldsymbol{v}) = \log_{\boldsymbol{\mu}}^c(\boldsymbol{\mu} \oplus_c \exp_{\mathbf{0}}^c(\boldsymbol{v})) = \frac{\lambda_{\mathbf{0}}^c}{\lambda_{\boldsymbol{\mu}}^c}\boldsymbol{v}. \tag{6}$$

where $\boldsymbol{\kappa}_{\boldsymbol{\mu},\boldsymbol{y}} := (-\boldsymbol{\mu}) \oplus_c \boldsymbol{y}$.

Adversarial Poincaré Variational Autoencoder (APo- VAE) (NAACL 2021)

Shuyang Dai, Zhe Gan, Yu Cheng, Chenyang Tao, Lawrence Carin, Jingjing Liu,

Core ideas:

- How does the defined compact metric space contribute to the formulation of a prior?
  - Poincaré ball based normal prior $z \sim N_{B^n_c}(\mu, \Sigma)$:

$$\boldsymbol{z} = \exp_{\boldsymbol{\mu}}^{c}\left(\frac{\lambda_{\mathbf{0}}^{c}}{\lambda_{\boldsymbol{\mu}}^{c}}\boldsymbol{v}\right), \boldsymbol{v} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}). \qquad (7)$$

  - Inspired by iVAE, we can enhance $\boldsymbol{v} := G(\boldsymbol{x}, \boldsymbol{\xi}; \boldsymbol{\phi}_1)$ and $\boldsymbol{\mu} := F(\boldsymbol{x}; \boldsymbol{\phi}_2)$, as outputs of encoder (ϕ). Then we get z from the above formula.
- Same as iVAE, they use dual form to evaluate $KL(q_\phi(z|x) \parallel p(z))$:

$$\mathbb{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})) = \max_{\boldsymbol{\psi}} \qquad (9)$$
$$\left\{\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\nu_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{z}) - \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\exp\nu_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{z})\right\},$$

- Same as iVAE, ν_ψ is optimized by the following:

$$\mathcal{L}_1 = \mathbb{E}_{\boldsymbol{x}\sim\boldsymbol{X}}[\,\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\nu_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{z})$$
$$- \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\exp\nu_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{z})\,], \qquad (10)$$

Adversarial Poincaré Variational Autoencoder (APo- VAE) (NAACL 2021)

Shuyang Dai, Zhe Gan, Yu Cheng, Chenyang Tao, Lawrence Carin, Jingjing Liu,

Core ideas:

- Different from iVAE, here the real prior is not assumed Gaussian. It's estimated by sampling scheme used in VampPrior (Tomczak and Welling, 2018)

$$p_{\boldsymbol{\delta}}(\boldsymbol{z}) = \frac{1}{K}\sum_{k=1}^{K} q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s}_k), \qquad (12)$$

- $\delta := \{s_k\}_{k=1}^K$ is now learnable pseudo inputs. Replacing $p(z)$ with $p_\delta(z)$ seeks to match the aggregated posterior $q(z) = \Sigma\, q_\phi(z|x_i) / N$.

Adversarial Poincaré Variational Autoencoder (APo- VAE) (NAACL 2021)

Shuyang Dai, Zhe Gan, Yu Cheng, Chenyang Tao, Lawrence Carin, Jingjing Liu,

Core ideas:

- For the geometry-aware decoder, they use a deterministic (and learnable) hyperbolic linear function f to extract feature, then pass to LSTM decoder. (a, b, together with the LSTM, are trainable parameters of decoder θ)

$$f^c_{\boldsymbol{a},\boldsymbol{b}}(\boldsymbol{z}) = \operatorname{sign}(\langle \boldsymbol{a}, \log^c_{\boldsymbol{b}}(\boldsymbol{z})\rangle_{\boldsymbol{b}})\|\boldsymbol{a}\|_{\boldsymbol{b}}\, d^{\mathbb{B}}_c(\boldsymbol{z}, H^c_{\boldsymbol{a},\boldsymbol{b}}), \tag{8}$$

$$\text{where } H^c_{\boldsymbol{a},\boldsymbol{b}} = \{\boldsymbol{z} \in \mathbb{B}^n_c \,|\, \langle \boldsymbol{a}, \log^c_{\boldsymbol{b}}(\boldsymbol{z})\rangle_{\boldsymbol{b}} = 0\},$$

$$d^{\mathbb{B}}_c(\boldsymbol{z}, H^c_{\boldsymbol{a},\boldsymbol{b}}) = \frac{1}{\sqrt{c}} \sinh^{-1}\left(\frac{2\sqrt{c}|\langle \boldsymbol{\kappa}_{\boldsymbol{b},\boldsymbol{z}}, \boldsymbol{a}\rangle|}{(1-c\|\boldsymbol{\kappa}_{\boldsymbol{b},\boldsymbol{z}}\|^2)\|\boldsymbol{a}\|}\right)$$

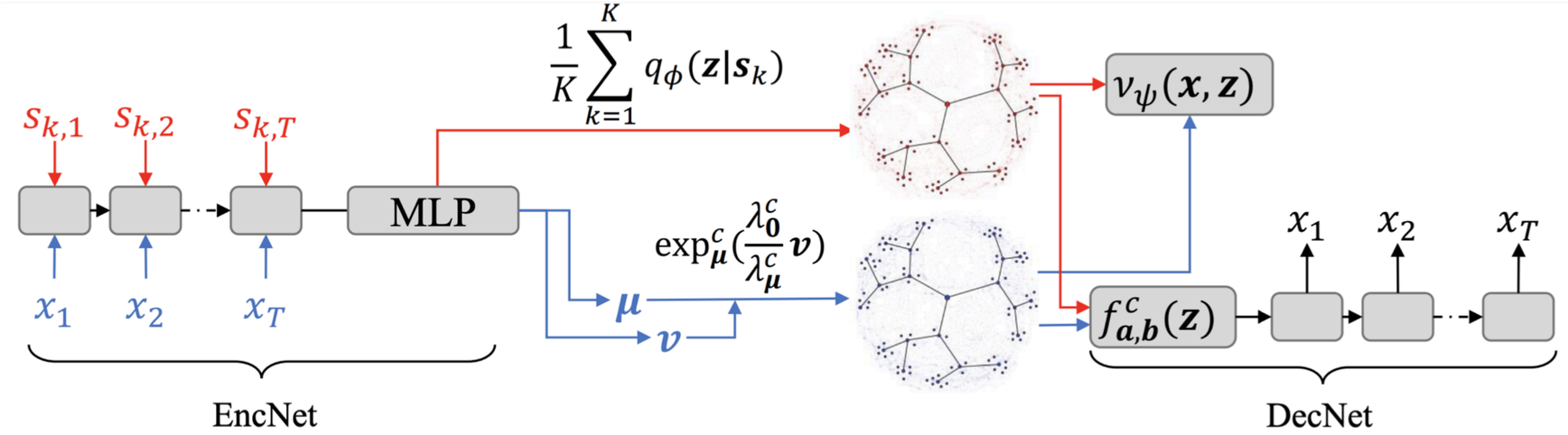
- θ and ϕ are optimized by the following objective

$$\mathcal{L}_2 = \mathbb{E}_{\boldsymbol{x}\sim\boldsymbol{X}}\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\, \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) - \nu_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{z})\,]. \tag{11}$$

Adversarial Poincaré Variational Autoencoder (APo- VAE) (NAACL 2021)

Shuyang Dai, Zhe Gan, Yu Cheng, Chenyang Tao, Lawrence Carin, Jingjing Liu

Core ideas:



Adversarial Poincaré Variational Autoencoder (APo- VAE) (NAACL 2021)

Shuyang Dai, Zhe Gan, Yu Cheng, Chenyang Tao, Lawrence Carin, Jingjing Liu,

Core ideas:

- Training Procedure

Algorithm 1 Training procedure of APo-VAE.

Input: Data samples $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N$, Poincaré curvature c, and number of pseudo-input K.
Initialize $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, $\boldsymbol{\psi}$, and $\boldsymbol{\delta}$.
for $iter$ from 1 to max_iter **do**
  Sample a mini-batch $\{\boldsymbol{x}_m\}_{m=1}^M$ from $\boldsymbol{X}$ of size M.
  # Sampling in the Hyperbolic Space.
  Obtain $\boldsymbol{\mu}_m$ and $\boldsymbol{v}_m$ from $\text{EncNet}_{\boldsymbol{\phi}}(\boldsymbol{x}_m)$.
  Move $\boldsymbol{v}_m$ to $\boldsymbol{u}_m = P^c_{\mathbf{0} \to \boldsymbol{\mu}_m}(\boldsymbol{v}_m)$ by (6).
  Map $\boldsymbol{u}_m$ to $\boldsymbol{z}_m = \exp^c_{\boldsymbol{\mu}_m}(\boldsymbol{u}_m)$ by (5).
  # Update the dual function and the pseudo-input.
  Sample $\tilde{\boldsymbol{z}}_m$ by (12).
  Update $\boldsymbol{\psi}$ and $\boldsymbol{\delta}$ by gradient ascent on (10)
  # Update the encoder and decoder networks.
  Update $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ by gradient ascent on (11).
end for

Adversarial Poincaré Variational Autoencoder (APo- VAE) (NAACL 2021)

Shuyang Dai, Zhe Gan, Yu Cheng, Chenyang Tao, Lawrence Carin, Jingjing Liu,

Results:

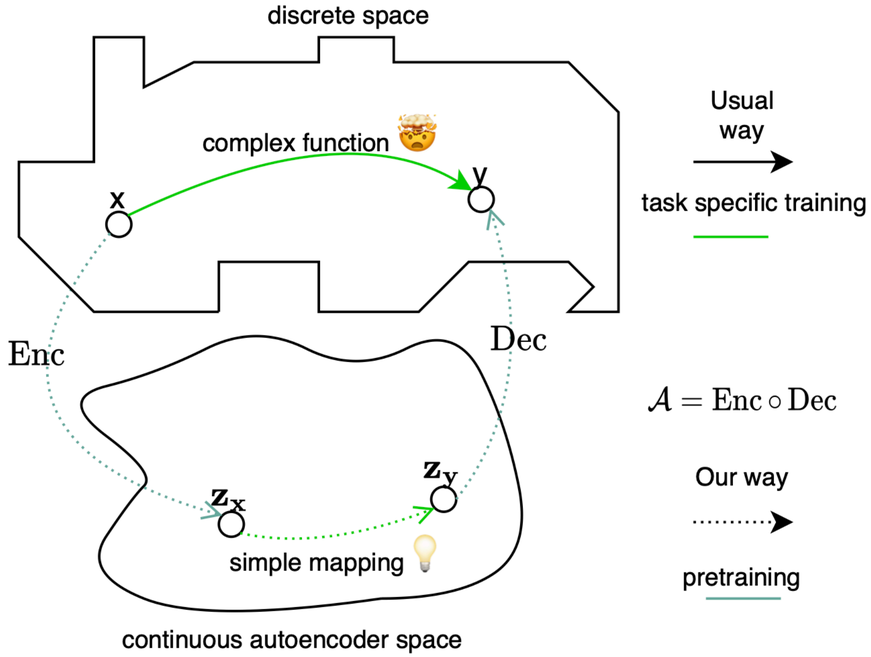
Model	-ELBO	PPL	KL	MI	AU
	PTB				
VAE	102.6	108.26	1.1	0.8	2
β-VAE	104.5	117.92	7.5	3.1	5
SA-VAE	102.6	107.71	1.2	0.7	2
vMF-VAE	95.8	93.70	2.9	3.2	21
$\mathcal{P}$-VAE	91.4	76.13	4.5	2.9	23
iVAE	87.2	53.44	**12.5**	**12.2**	**32**
APo-VAE	87.2	53.32	8.4	4.8	**32**
APo-VAE+VP	**87.0**	**53.02**	8.9	4.5	**32**
	Yahoo				
VAE	328.6	61.21	0.0	0.0	0
β-VAE	328.7	61.29	6.3	2.8	8
SA-VAE	327.2	60.15	5.2	2.9	10
LAG-VAE	326.7	59.77	5.7	2.9	15
vMF-VAE	318.5	53.92	6.3	3.7	23
$\mathcal{P}$-VAE	313.4	50.57	7.2	3.3	27
iVAE	309.1	47.93	**11.4**	**10.7**	**32**
APo-VAE	286.2	47.00	6.9	4.1	**32**
APo-VAE+VP	**285.6**	**46.61**	8.1	4.9	**32**
	Yelp				
VAE	357.9	40.56	0.0	0.0	0
β-VAE	358.2	40.69	4.2	2.0	4
SA-VAE	357.8	40.51	2.8	1.7	8
LAG-VAE	355.9	39.73	3.8	2.4	11
vMF-VAE	356.2	51.03	4.1	3.9	13
$\mathcal{P}$-VAE	355.4	50.64	4.3	4.8	19
iVAE	348.7	36.88	11.6	**11.0**	**32**
APo-VAE	319.7	34.10	12.1	7.5	**32**
APo-VAE+VP	**316.4**	**32.91**	**12.7**	6.2	**32**

Plug and Play Autoencoders for Conditional Text Generation

Florian Mai et al

Core ideas:

- Learning mappings from continuous space is easier than from discrete space.

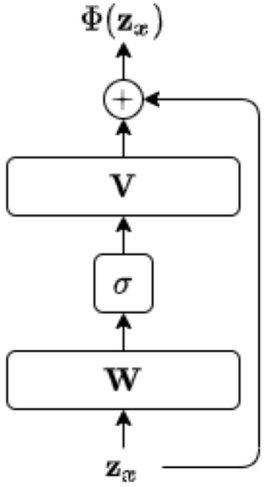


Plug and Play Autoencoders for Conditional Text Generation

Florian Mai et al

Core ideas:

- Encoder and Decoder are pretrained and fixed during training.
- Adding a mapping function Φ, parametrized by a OffsetNet
    - Different from DAAE, where operations in latent space is just adding and subtracting. Here the operation is conditioned on the input, and during inference, using FGIM to fulfill style transfer.



(b) OffsetNet

Plug and Play Autoencoders for Conditional Text Generation

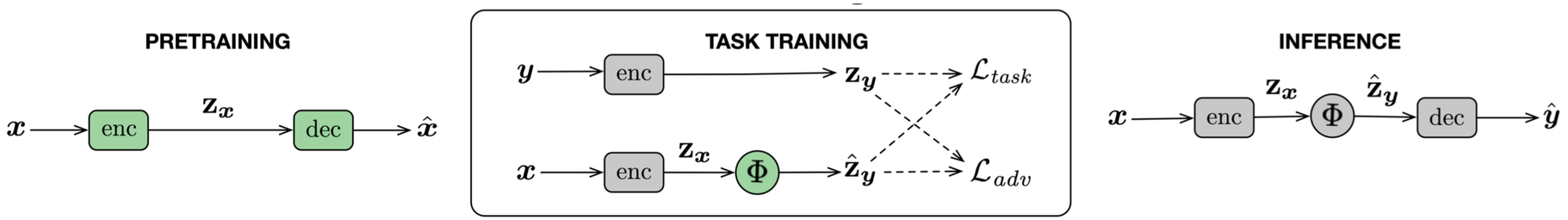
Florian Mai et al

Core ideas:

- Adversarial learning to ensure the manifold of the mapping Φ resembles the original latent space.

$$\mathcal{L}_{adv}(\Phi(\mathbf{z}_{\boldsymbol{x}_i}); \boldsymbol{\theta}) = -\log(\text{disc}(\Phi(\mathbf{z}_{\boldsymbol{x}_i}); \theta)) \quad (11)$$

- For supervised tasks:



$$\mathcal{L} = \mathcal{L}_{task} + \lambda_{adv}\mathcal{L}_{adv}. \quad (1)$$

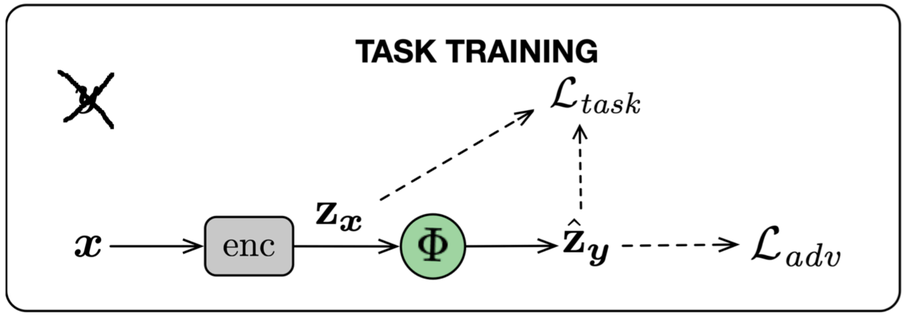
$$\mathcal{L}_{task} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_{emb}(\Phi(\mathbf{z}_{\boldsymbol{x}_i}; \boldsymbol{\theta}), \mathbf{z}_{\boldsymbol{y}_i}). \quad (8)$$

Plug and Play Autoencoders for Conditional Text Generation

Florian Mai et al

Core ideas:

- For unsupervised tasks:



$$\mathcal{L}_{task}(\hat{\mathbf{z}}_{\mathbf{y}}, \mathbf{z}_{\mathbf{x}}) = \lambda_{sty}\mathcal{L}_{sty}(\hat{\mathbf{z}}_{\mathbf{y}}) + (1 - \lambda_{sty})\mathcal{L}_{cont}(\hat{\mathbf{z}}_{\mathbf{y}}, \mathbf{z}_{\mathbf{x}})$$

$$\mathcal{L}_{sty}(\Phi(\mathbf{z}_{\boldsymbol{x}_i}; \boldsymbol{\theta}), \mathbf{z}_{\boldsymbol{x}_i}) = -\log(\mathbf{c}(\Phi(\mathbf{z}_{\boldsymbol{x}_i}; \boldsymbol{\theta}))).$$

Here c is a pretrained classifier for style. They freeze c when training and encourage Φ to produce outputs of the target attribute (y=1).

Plug and Play Autoencoders for Conditional Text Generation

Florian Mai et al

Core ideas:

- For unsupervised tasks, how to better manipulate latent space (i.e to invert sentiment)? Use FGIM until the confidence of the sentiment classifier is greater than a threshold.

$$\hat{\hat{\mathbf{z}}}_{\boldsymbol{x}_i} = \hat{\mathbf{z}}_{\boldsymbol{x}_i} + \omega \nabla_{\hat{\mathbf{z}}_{\boldsymbol{x}_i}} \mathcal{L}(\hat{\mathbf{z}}_{\boldsymbol{x}_i}, \mathbf{z}_{\boldsymbol{x}_\mathbf{i}}),$$

Plug and Play Autoencoders for Conditional Text Generation

Florian Mai et al

Results:

Model	BLEU	SARI	Time
S2S-Scratch	3.6	15.6	3.7×
S2S-Pretrain	5.4	16.2	3.7×
S2S-MLP	10.5	17.7	3.7×
S2S-Freeze	23.3	22.4	2.2×
Emb2Emb	**34.7**	**25.4**	1.0×

Table 1: Text simplification performance of model variants of end2end training on the test set. "Time" is wall time of one training epoch, relative to our model, Emb2Emb.

Model	Acc.	s-BLEU	+Time
Shen et al.	96.8	6.5	0.5×
FGIM	94.9	10.8	70.0×
Emb2Emb + FGIM	93.1	18.1	2820.0×
Emb2Emb	87.1	22.1	1.0×

Table 2: Self-BLEU ("s-BLEU") on the Yelp sentiment transfer test set for the configurations in Figure 5 with highest transfer accuracy ("Acc."). "+Time" reports the inference-time slowdown factor due to each model's additional computation (relative to our method).

Wasserstein Auto-Encoders

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Scho ̈lkopf

Core ideas:

- Minimizing $W_c(P_X,P_G)$ between the true (but unknown) data distribution P_X and a latent variable model P_G

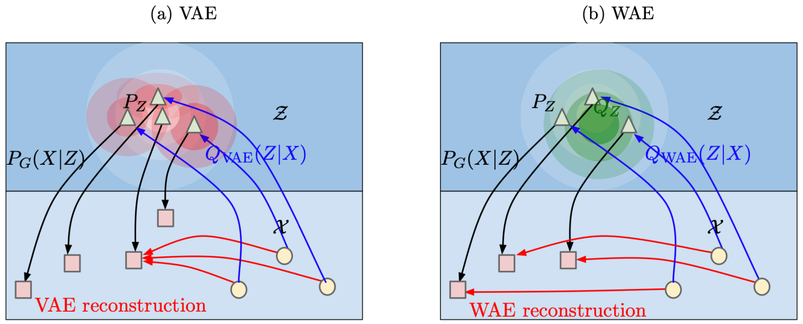
$$W_c(P_X, P_G) := \inf_{\Gamma \in \mathcal{P}(X \sim P_X, Y \sim P_G)} \mathbb{E}_{(X,Y)\sim\Gamma}\big[c(X,Y)\big]\,,$$

Theorem 1. *For P_G as defined above with deterministic $P_G(X|Z)$ and any function $G\colon \mathcal{Z} \to \mathcal{X}$*

$$\inf_{\Gamma \in \mathcal{P}(X \sim P_X, Y \sim P_G)} \mathbb{E}_{(X,Y)\sim\Gamma}\left[c\big(X,Y\big)\right] = \inf_{Q\colon\, Q_Z = P_Z} \mathbb{E}_{P_X}\mathbb{E}_{Q(Z|X)}\left[c\big(X,G(Z)\big)\right],$$

where Q_Z is the marginal distribution of Z when $X \sim P_X$ and $Z \sim Q(Z|X)$.

- Enforcing aggregated posterior ($Q_Z := \int Q(Z|X)dP_X$) to match the Prior P_z



$$D_{\text{WAE}}(P_X, P_G) := \inf_{Q(Z|X)\in\mathcal{Q}} \mathbb{E}_{P_X}\mathbb{E}_{Q(Z|X)}\big[c\big(X,G(Z)\big)\big] + \lambda \cdot \mathcal{D}_Z(Q_Z, P_Z),$$

Wasserstein Auto-Encoders

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Scho ̈lkopf

Core ideas:

- Two proposed penalties for D_Z
  - GAN-based D_Z

$$\mathcal{D}_Z(Q_Z, P_Z) = D_{\mathrm{JS}}(Q_Z, P_Z)$$

Algorithm 1 Wasserstein Auto-Encoder with GAN-based penalty (WAE-GAN).

Require: Regularization coefficient $\lambda > 0$.

Initialize the parameters of the encoder Q_ϕ, decoder G_θ, and latent discriminator D_γ.

while (ϕ, θ) not converged **do**

Sample $\{x_1, \dots, x_n\}$ from the training set

Sample $\{z_1, \dots, z_n\}$ from the prior P_Z

Sample $\tilde{z}_i$ from $Q_\phi(Z|x_i)$ for $i = 1, \dots, n$

Update D_γ by ascending:

$$\frac{\lambda}{n} \sum_{i=1}^{n} \log D_\gamma(z_i) + \log\big(1 - D_\gamma(\tilde{z}_i)\big)$$

Update Q_ϕ and G_θ by descending:

$$\frac{1}{n} \sum_{i=1}^{n} c\big(x_i, G_\theta(\tilde{z}_i)\big) - \lambda \cdot \log D_\gamma(\tilde{z}_i)$$

end while

Wasserstein Auto-Encoders

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Scho ̈lkopf

Core ideas:

- Two proposed penalties for D_z
  - MMD-based D_z

$$\mathrm{MMD}_k(P_Z, Q_Z) = \left\| \int_{\mathcal{Z}} k(z, \cdot) dP_Z(z) - \int_{\mathcal{Z}} k(z, \cdot) dQ_Z(z) \right\|_{\mathcal{H}_k},$$

Algorithm 2 Wasserstein Auto-Encoder with MMD-based penalty (WAE-MMD).

Require: Regularization coefficient $\lambda > 0$, characteristic positive-definite kernel k.

Initialize the parameters of the encoder Q_ϕ, decoder G_θ, and latent discriminator D_γ.

while (ϕ, θ) not converged **do**

Sample $\{x_1, \dots, x_n\}$ from the training set

Sample $\{z_1, \dots, z_n\}$ from the prior P_Z

Sample $\tilde{z}_i$ from $Q_\phi(Z|x_i)$ for $i = 1, \dots, n$

Update Q_ϕ and G_θ by descending:

$$\frac{1}{n} \sum_{i=1}^{n} c\big(x_i, G_\theta(\tilde{z}_i)\big) + \frac{\lambda}{n(n-1)} \sum_{\ell \neq j} k(z_\ell, z_j) + \frac{\lambda}{n(n-1)} \sum_{\ell \neq j} k(\tilde{z}_\ell, \tilde{z}_j) - \frac{2\lambda}{n^2} \sum_{\ell, j} k(z_\ell, \tilde{z}_j)$$

end while

Wasserstein Auto-Encoders

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Scho¨lkopf

Core ideas:

- Difference with WGAN
    - Claiming the intractability of infimum, WGAN uses dual form of 1-Wasserstein Distance (W_1), while WAE could use any cost function c (W_c)

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\, \|x - y\| \,\right],$$

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

Wasserstein Auto-Encoders

Ilya Tolstikhin[*], Olivier Bousquet[*], Sylvain Gelly[*], and Bernhard Scho ̈lkopf[†]

Core ideas:

- Relation to other models:
    - AAE: When c equals to L2 norm, WAE-GAN becomes AAE

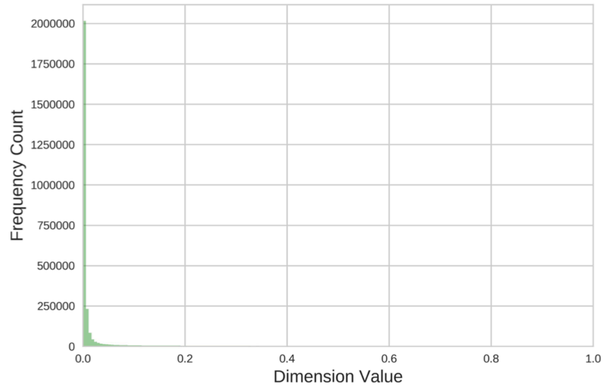
Stochastic Wasserstein Autoencoder for Probabilistic Sentence Generation (NAACL 2019)

Hareesh Bahuleyan, Lili Mou, Hao Zhou, Olga Vechtomova

Core ideas:

- Theoretically and empirically, the encoded Gaussian of the original WAE would be Dirac-delta ($\sigma \to 0$)

Theorem 1. *Suppose we have a Gaussian family* $\mathcal{N}(\boldsymbol{\mu}, \operatorname{diag}\boldsymbol{\sigma}^2)$*, where* $\boldsymbol{\mu}$ *and* $\boldsymbol{\sigma}$ *are parameters. The covariance is diagonal, meaning that the variables are independent. If the gradient of* $\boldsymbol{\sigma}$ *completely comes from sample gradient and* $\boldsymbol{\sigma}$ *is small at the beginning of training, then the Gaussian converges to a Dirac delta function with stochastic gradient descent, i.e.,* $\boldsymbol{\sigma} \to \mathbf{0}$*. (See Appendix A for the proof.)* □



(a) $\lambda_{\mathrm{KL}} = 0$

- Add a stochastic term to original WAE-MMD, yielding a relaxed optimization of WAE loss with a constraint on σ

$$J = J_{\mathrm{rec}} + \lambda_{\mathrm{WAE}} \cdot \widehat{\mathrm{MMD}} + \lambda_{\mathrm{KL}} \sum\nolimits_n \mathrm{KL}\left(\mathcal{N}(\boldsymbol{\mu}_{\mathrm{post}}^{(n)}, \operatorname{diag}(\boldsymbol{\sigma}_{\mathrm{post}}^{(n)})^2) \middle\| \mathcal{N}(\boldsymbol{\mu}_{\mathrm{post}}^{(n)}, \mathbf{I})\right) \tag{5}$$

Stochastic Wasserstein Autoencoder for Probabilistic Sentence Generation (NAACL 2019)

Hareesh Bahuleyan, Lili Mou, Hao Zhou, Olga Vechtomova

Results:

- For generation, WAE keeps continuity and smoothness as VAE, while have a much higher reconstruction performance. WAE-S further encourages the stochasticity.

	BLEU$^{\uparrow}$	PPL$^{\downarrow}$	UniKL$^{\downarrow}$	Entropy	AvgLen
Corpus	-	-	-	$\rightarrow 5.65$	$\rightarrow 9.6$
DAE	**86.35**	146.2	0.178	6.23	11.0
VAE (KL-annealed)	43.18	79.4	0.081	5.04	8.8
WAE-D $\lambda_{\text{WAE}}=3$	86.03	113.8	0.071	5.59	10.0
WAE-D $\lambda_{\text{WAE}}=10$	84.29	104.9	0.073	5.57	9.9
WAE-S $\lambda_{\text{KL}}=0.0$	75.66	115.2	0.069	**5.61**	9.9
WAE-S $\lambda_{\text{KL}}=0.01$	82.01	84.9	**0.058**	5.26	**9.4**
WAE-S $\lambda_{\text{KL}}=0.1$	47.63	**62.5**	0.150	4.65	8.7

Table 1: Results of SNLI-style sentence generation, where WAE is compared with DAE and VAE. **D** and **S** refer to the deterministic and stochastic encoders, respectively. $^{\uparrow/\downarrow}$The larger/lower, the better. For **Entropy** and **AvgLen**, the closer to corpus statistics, the better (indicated by the $\rightarrow$ arrow).

Reproducible SOTA VAEs for reference

From all text VAEs (with code) after 2020 in https://github.com/matthewvowels1/Awesome-VAEs

Results (ppl computed from nll):

- BN-VAE: (reproduced)

Algorithm 1 BN-VAE training.

Initialize $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$.
for $i = 1, 2, \cdots$ Until Convergence **do**
  Sample a mini-batch $\mathbf{x}$.
  μ, $\log \sigma^2 = f_\phi(\mathbf{x})$.
  $\mu' = BN_{\gamma,\beta}(\mu)$.
  Sample $\mathbf{z} \sim \mathcal{N}(\mu', \sigma^2)$ and reconstruct $\mathbf{x}$ from $f_\theta(\mathbf{z})$.
  Compute gradients $\mathbf{g}_{\phi,\theta} \leftarrow \nabla_{\phi,\theta}\mathcal{L}(\mathbf{x}; \phi, \theta)$.
  Update ϕ, θ using $\mathbf{g}_{\phi,\theta}$.
end for

	Yahoo				Yelp			
Model	**NLL**	**KL**	**MI**	**AU**	**NLL**	**KL**	**MI**	**AU**
Without a pretrained AE encoder								
CNN-VAE	≤332.1	10.0	-	-	≤359.1	7.6	-	-
LSTM-LM	328	-	-	-	351.1	-	-	-
VAE	328.6	0.0	0.0	0.0	357.9	0.0	0.0	0.0
β-VAE (0.4)	328.7	6.3	2.8	8.0	358.2	4.2	2.0	4.2
cyclic *	330.6	2.1	2.0	2.3	359.5	2.0	1.9	4.1
Skip-VAE *	328.5	2.3	1.3	8.1	357.6	1.9	1.0	7.4
SA-VAE	327.2	5.2	2.7	9.8	**355.9**	2.8	1.7	8.4
Agg-VAE	**326.7**	5.7	2.9	15.0	**355.9**	3.8	2.4	11.3
FB (4)	331.0	4.1	3.8	3.0	359.2	4.0	1.9	32.0
FB (5)	330.6	5.7	2.0	3.0	359.8	4.9	1.3	32.0
δ-VAE (0.1) *	330.7	3.2	0.0	0.0	359.8	3.2	0.0	0.0
vMF-VAE (13) *	327.4	2.0	-	32.0	357.5	2.0	-	32.0
BN-VAE (0.6) *	**326.7**	6.2	5.6	32.0	356.5	6.5	5.4	32.0
BN-VAE (0.7) *	327.4	8.8	7.4	32.0	**355.9**	9.1	7.4	32.0
With a pretrained AE encoder								
cyclic *	333.1	25.8	9.1	32.0	361.5	20.5	9.3	32.0
FB (4) *	**326.2**	8.1	6.8	32.0	356.0	7.6	6.6	32.0
δ-VAE (0.15) *	331.0	5.6	1.1	11.2	359.4	5.2	0.5	5.9
vMF-VAE (13) *	328.4	2.0	-	32.0	357.0	2.0	-	32.0
BN-VAE (0.6) *	326.7	6.4	5.8	32.0	**355.5**	6.6	5.9	32.0
BN-VAE (0.7) *	326.5	9.1	7.6	32.0	355.7	9.1	7.5	32.0

- DAVAM (IJCNN 2021): VQ-VAE (learning discrete latent space) with attention mechanism, and apply to text data. (Under experiment, close to finish)

#	Methods	Yahoo			PTB			SNLI		
		Rec↓	PPL↓	KL	Rec↓	PPL↓	KL	Rec↓	PPL↓	KL
1	LSTM-LM	-	60.75	-	-	100.47	-	-	21.44	-
2	VAE	329.10	61.52	0.00	101.27	101.39	0.00	33.08	21.67	0.04
3	+anneal	328.80	61.21	0.00	101.28	101.40	0.00	31.66	21.50	1.42
4	+cyclic	333.80	66.93	2.83	101.85	108.97	1.37	30.69	23.67	3.63
5	+aggressive	322.70	59.77	5.70	100.26	99.83	0.93	31.53	21.16	1.42
6	+FBP	322.91	62.59	9.08	98.52	99.62	2.95	25.26	22.05	8.99
7	+pretraining+FBP	315.09	59.60	15.49	96.91	96.17	4.99	22.30	22.33	13.40
8	GAVAM	350.14	79.28	0.00	102.20	105.94	0.00	30.90	17.68	0.38
9	DAVAM-q (K=512)	323.10	57.14	0.33	95.83	79.24	0.27	28.16	13.71	0.12
10	DAVAM (K=128)	303.65	45.07	1.88	83.57	50.15	2.23	16.11	5.58	2.38
11	DAVAM (K=512)	**259.68**	**26.61**	2.60	**60.16**	**17.94**	3.12	**10.85**	**3.52**	2.69

Reproducible SOTA VAEs for reference

From all text VAEs (with code) after 2020 in https://github.com/matthewvowels1/Awesome-VAEs

Results (ppl computed from nll):

- TWR-VAE: (reproduced)

Model	PTB				Yelp15				Yahoo			
	NLL↓	PPL↓	MI↑	KL	NLL↓	PPL↓	MI↑	KL	NLL↓	PPL↓	MI↑	KL
VAE-LSTM	101.2	101.4	0.0	0.0	357.9	40.6	0.0	0.0	328.6	61.2	0.0	0.0
SA-VAE	101.0	100.7	0.8	1.3	355.9	39.7	2.8	1.7	327.2	60.2	2.7	5.2
Cyc-VAE	102.8	109.0	1.3	1.4	359.5	41.3	1.0	2.0	330.6	65.3	2.0	2.1
Lag-VAE	100.9	99.8	0.8	0.9	355.9	39.7	2.4	3.8	326.7	59.8	2.9	5.7
BN-VAE (0.7)	100.2	96.9	**5.5**	7.2	355.9	39.7	**7.4**	9.1	327.4	60.2	**7.4**	8.8
TWR-VAE$_{sum}$	96.7	63.2	3.7	5.9	378.3	47.4	3.8	3.9	345.6	71.1	3.7	3.8
TWR-VAE$_{mean}$	95.6	60.4	3.9	4.9	361.7	40.0	3.9	3.5	324.8	55.0	4.1	4.8
TWR-VAE	**86.6**	**40.9**	4.1	5.0	**344.3**	**33.5**	4.1	3.1	**317.3**	**50.2**	4.1	3.3

Equivariant

2D Pixel domain

Group Equivariant Convolutional Networks

Taco S. Cohen, Max Welling

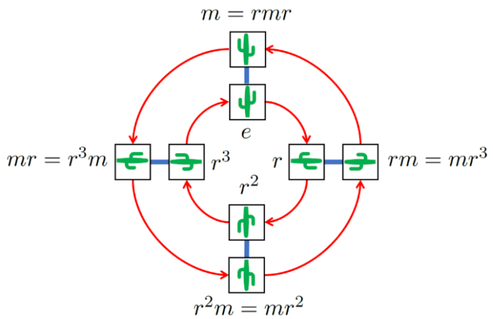
Core ideas:

- Equivariance: How one should transform the latent feature space if the input transforms.

$$\Phi(T_g\, x) = T'_g\, \Phi(x), \qquad (1)$$

- Considering a special group of transformation p4m applied to images, the feature map from original convolution wouldn't be equivariant (original convolution is only equivariant to translation).

$$g(m, r, u, v) = \begin{bmatrix} (-1)^m \cos(\frac{r\pi}{2}) & -(-1)^m \sin(\frac{r\pi}{2}) & u \\ \sin(\frac{r\pi}{2}) & \cos(\frac{r\pi}{2}) & v \\ 0 & 0 & 1 \end{bmatrix},$$

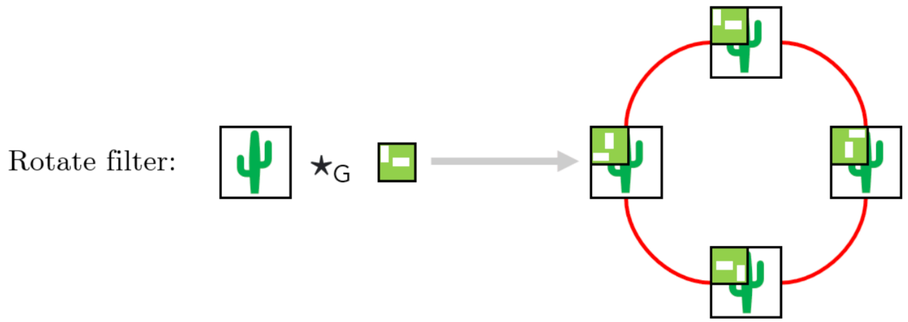


Group Equivariant Convolutional Networks

Taco S. Cohen, Max Welling

Core ideas:

- Proposed Group Equivariant Convolution
    - First Layer (P4ConvZ2): Transform the input to a structured object

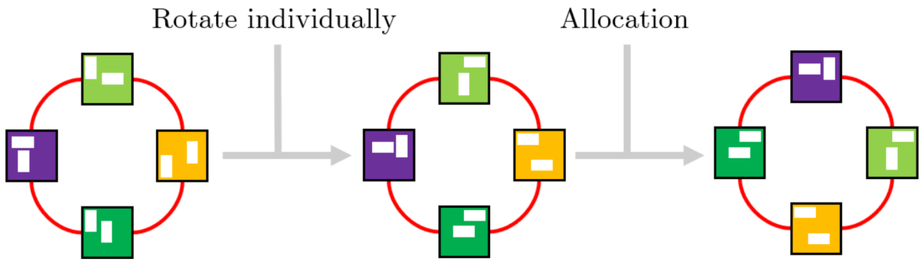


Group Equivariant Convolutional Networks

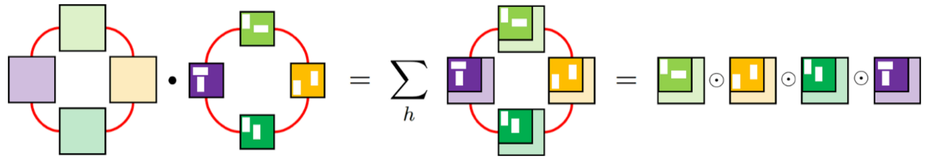
Taco S. Cohen, Max Welling

Core ideas:

- Proposed Group Equivariant Convolution
    - Intermediate Layer : Structured object to Structured object using two tools
        - Tool 1: transformation of a structured object



        - Tool 2: Dot product of two structured objects

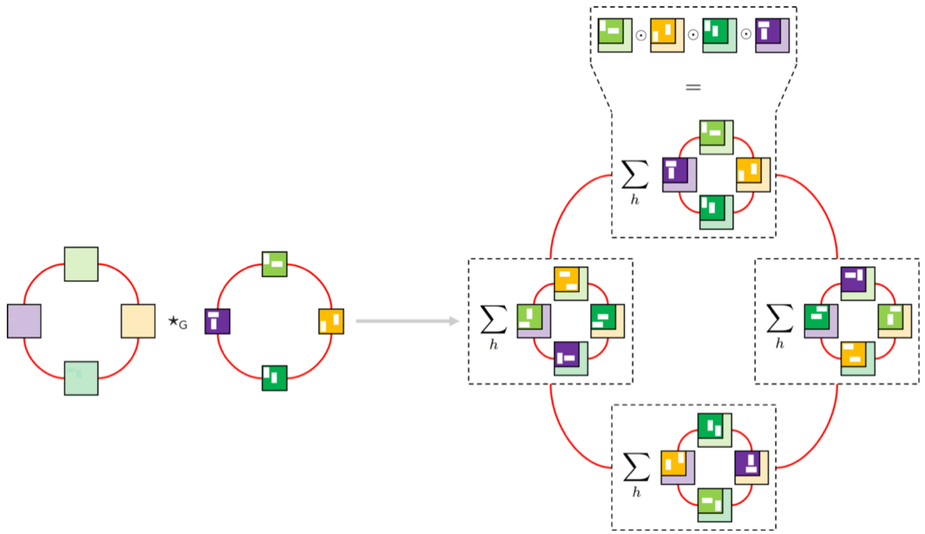


Group Equivariant Convolutional Networks

Taco S. Cohen, Max Welling

Core ideas:

- Proposed Group Equivariant Convolution
    - Intermediate Layer (P4ConvP4): Structured object to Structured object using two tools



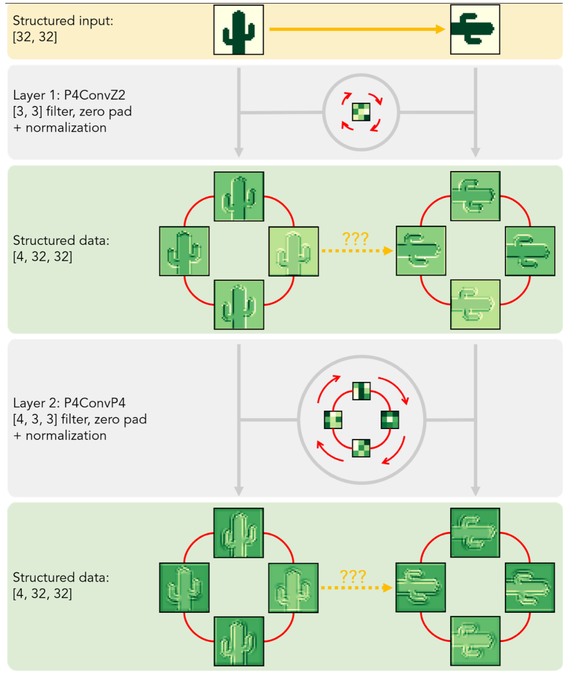
G-convolution on a structured input using the group p4 specifically (called the P4ConvP4). It creates differently located stacks of feature maps that preserve the structure of the groups graph.

Group Equivariant Convolutional Networks

Taco S. Cohen, Max Welling

Core ideas:

- Proposed Group Equivariant Convolution reflected the transformation of input in its feature stacks, with Pointwise nonlinearity and special equivariant pooling.



Group Equivariant GAN

Neel Dey, Antong Chen & Soheil Ghafuria

Core ideas:

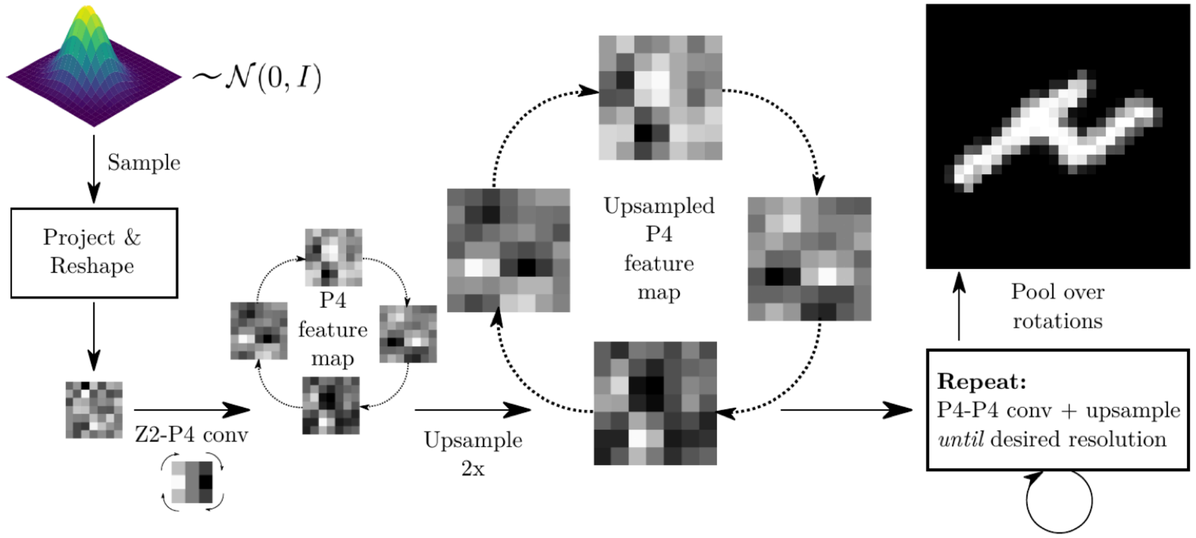


Figure 2: An abbreviated illustration of group-convolutions used in our generator networks.

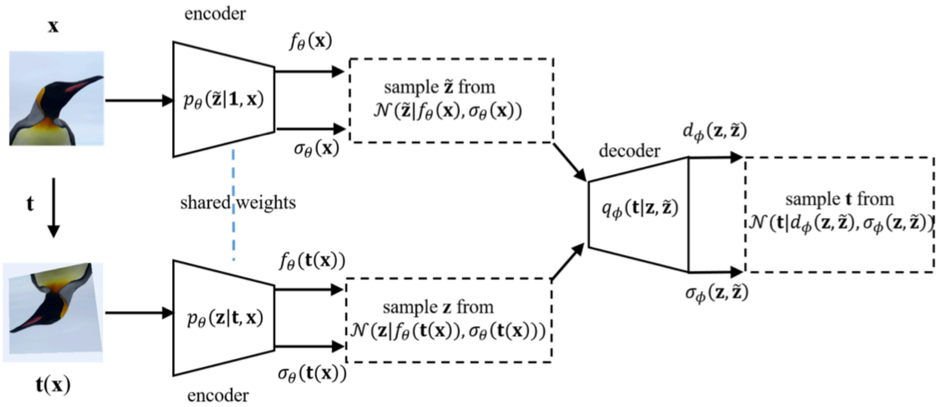
AVT: Unsupervised Learning of Transformation Equivariant Representations by Autoencoding Variational Transformations

Huawei

Core ideas:

- Maximize mutual information between the representation and the transformation. More meaningful latent representation.

$$\max_{\theta} I(\mathbf{t}; \mathbf{z}|\tilde{\mathbf{z}})$$



$$\max_{\theta,\phi} \frac{1}{n} \sum_{i=1}^{n} \log \mathcal{N}(\mathbf{t}^i | d_\phi(\mathbf{z}^i, \tilde{\mathbf{z}}^i), \sigma_\phi(\mathbf{z}^i, \tilde{\mathbf{z}}^i)) \qquad (3)$$

Equivariant

3D Graph/Mesh/Point Cloud domain

SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks (NEURIPS 20)

Fabian B. Fuchs, Daniel E. Worrall, Volker Fischer, Max Welling

Core ideas:

- Self-Attention that is equivariant each layer.

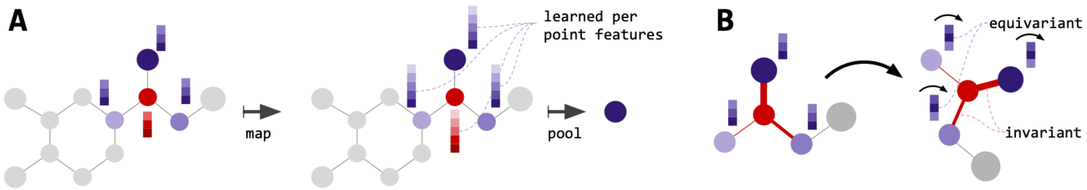


Figure 1: **A**) Each layer of the SE(3)-Transformer maps from a point cloud to a point cloud (or graph to graph) while guaranteeing equivariance. For classification, this is followed by an invariant pooling layer and an MLP. **B**) In each layer, for each node, attention is performed. Here, the red node attends to its neighbours. Attention weights (indicated by line thickness) are invariant w.r.t. input rotation.

Equivariant

Text Domain? Maybe need to learn a more meaningful embedding or latent space, on which certain transformations are defined, then use equivariant for better performance.