Outline for today

- Towards robustness via the the convex outer-polytope and/or sparsification of the network input or weights.
- Autoencoders
  - Structure and relation to PCA
  - k-sparse autoencoders
  - variational autoencoders (VAEs)
  - inference and β VAEs

Provable defense: convex polytope pt. 1 (Wong et al. 17[1])

Possible output of net $f_\theta(\cdot)$ from bounded perturbation is a non-convex set, say $\mathcal{Z}_\epsilon(x) = \{f_\theta(x+\delta) : \|\delta\|_\infty \leq \epsilon\}$. A convex outer-polytope of $\mathcal{Z}_\epsilon(x)$, say $\mathcal{Z}_\epsilon^{conv}(x)$, can be computed by replacing the input to each activation with a two dimensional convex set:

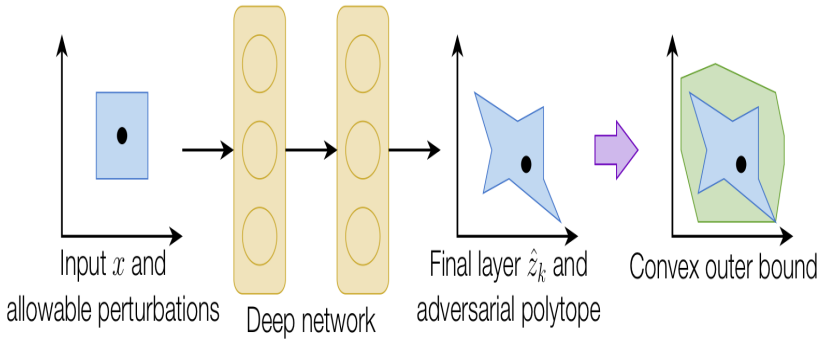


Figure 1. Conceptual illustration of the (non-convex) adversarial polytope, and an outer convex bound.

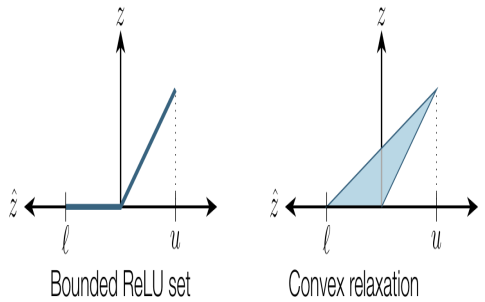


Figure 2. Illustration of the convex ReLU relaxation over the bounded set $[\ell, u]$.

Requires knowledge of lower and upper bound for each input to a nonlinear activation. Let $c = e_\ell - e_{\ell'}$ or $c = 2e_\ell - 1_{|class|}$ and solve:

$$\min_{\hat{z}_\ell \in \hat{\mathcal{Z}}_\epsilon(x)} c^T \hat{z}_\ell$$ and if nonnegative then robust to ϵ perturbation.

[1] https://arxiv.org/pdf/1711.00851.pdf

Provable defense: convex polytope pt. 2 (Wong et al. 17[2])

Algorithm 1 Computing Activation Bounds

input: Network parameters $\{W_i, b_i\}_{i=1}^{k-1}$, data point x, ball size ϵ

// initialization

$\hat{\nu}_1 := W_1^T$

$\gamma_1 := b_1^T$

$\ell_2 := x^T W_1^T + b_1^T - \epsilon \|W_1^T\|_{1,:}$

$u_2 := x^T W_1^T + b_1^T + \epsilon \|W_1^T\|_{1,:}$

// $\|\cdot\|_{1,:}$ for a matrix here denotes ℓ_1 norm of all columns

for $i = 2, \ldots, k-1$ **do**

  form $\mathcal{I}_i^-$, $\mathcal{I}_i^+$, $\mathcal{I}_i$; form D_i as in (10)

  // initialize new terms

  $\nu_{i,\mathcal{I}_i} := (D_i)_{\mathcal{I}_i} W_i^T$

  $\gamma_i := b_i^T$

  // propagate existing terms

  $\nu_{j,\mathcal{I}_j} := \nu_{j,\mathcal{I}_j} D_i W_i^T, \quad j = 2, \ldots, i-1$

  $\gamma_j := \gamma_j D_i W_i^T, \quad j = 1, \ldots, i-1$

  $\hat{\nu}_1 := \hat{\nu}_1 D_i W_i^T$

  // compute bounds

  $\psi_i := x^T \hat{\nu}_1 + \sum_{j=1}^{i} \gamma_j$

  $\ell_{i+1} := \psi_i - \epsilon \|\hat{\nu}_1\|_{1,:} + \sum_{j=2}^{i} \sum_{i' \in \mathcal{I}_i} \ell_{j,i'} [-\nu_{j,i'}]_+$

  $u_{i+1} := \psi_i + \epsilon \|\hat{\nu}_1\|_{1,:} - \sum_{j=2}^{i} \sum_{i' \in \mathcal{I}_i} \ell_{j,i'} [\nu_{j,i'}]_+$

end for

output: bounds $\{\ell_i, u_i\}_{i=2}^{k}$

[2] https://arxiv.org/pdf/1711.00851.pdf

Provable defense: convex polytope pt. 3 (Wong et al. 17[3])

Table 1. Error rates for various problems and attacks, and our robust bound for baseline and robust models.

PROBLEM	ROBUST	ϵ	TEST ERROR	FGSM ERROR	PGD ERROR	ROBUST ERROR BOUND
MNIST	X	0.1	1.07%	50.01%	81.68%	100%
MNIST	√	0.1	1.80%	3.93%	4.11%	5.82%
FASHION-MNIST	X	0.1	9.36%	77.98%	81.85%	100%
FASHION-MNIST	√	0.1	21.73%	31.25%	31.63%	34.53%
HAR	X	0.05	4.95%	60.57%	63.82%	81.56%
HAR	√	0.05	7.80%	21.49%	21.52%	21.90%
SVHN	X	0.01	16.01%	62.21%	83.43%	100%
SVHN	√	0.01	20.38%	33.28%	33.74%	40.67%

[3] `https://arxiv.org/pdf/1711.00851.pdf`

Robustness via sparsification (Gopalakrishnan et al. 18[4])

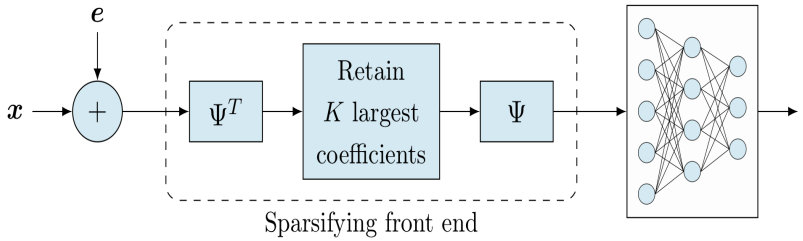


Figure 1: Sparsifying front end defense: For a basis in which the input is sparse, the input is projected onto the subspace spanned by the K largest basis coefficients. This attenuates the impact of the attack by K/N, where N is the input dimension.

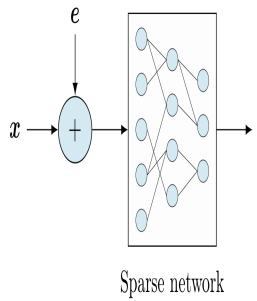


Figure 2: Network sparsity defense: Imposing sparsity *within* the neural network attenuates the worst-case growth of the attack as it flows up the network.

Theorem 2. *Consider an ℓ_∞-constrained input perturbation $\boldsymbol{e}_0 = \boldsymbol{e}$, with $\|\boldsymbol{e}\|_\infty \leq \epsilon$. Suppose that we impose ℓ_1 constraints on the weights at each layer as follows:*

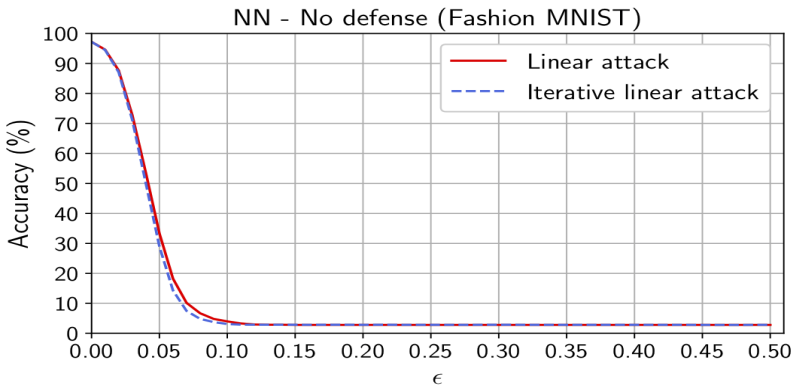
$$\|\boldsymbol{w}_{ij}\|_1 \leq \gamma_j \quad \forall\ i$$

Then the effect of the perturbation is ℓ_∞-bounded at each layer:

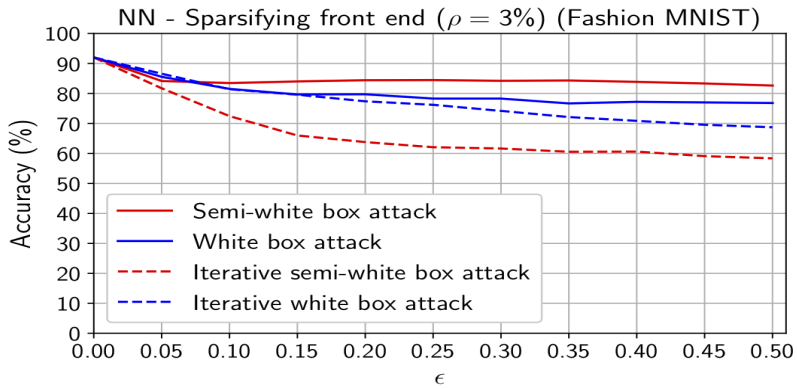
$$\|\boldsymbol{e}_j\|_\infty \leq \epsilon \prod_{l=1}^{j} \gamma_l \tag{2}$$

[4] https://arxiv.org/abs/1810.10625

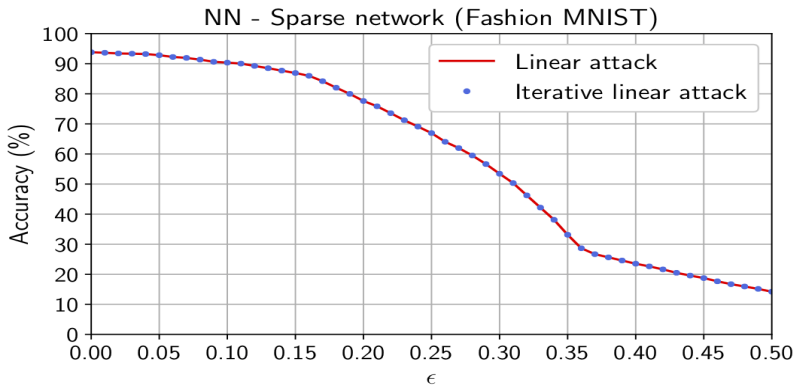
Robustness via sparsification (Gopalakrishnan et al. 18[5])



(a) No defense



(b) With sparsifying front end



(c) With network sparsity

Figure 6: Fashion-MNIST: Binary classification accuracies as a function of ϵ

[5] https://arxiv.org/abs/1810.10625

Generative deep nets (Goodfellow et al. 14'[7])

Example of a deep convolutional generator:

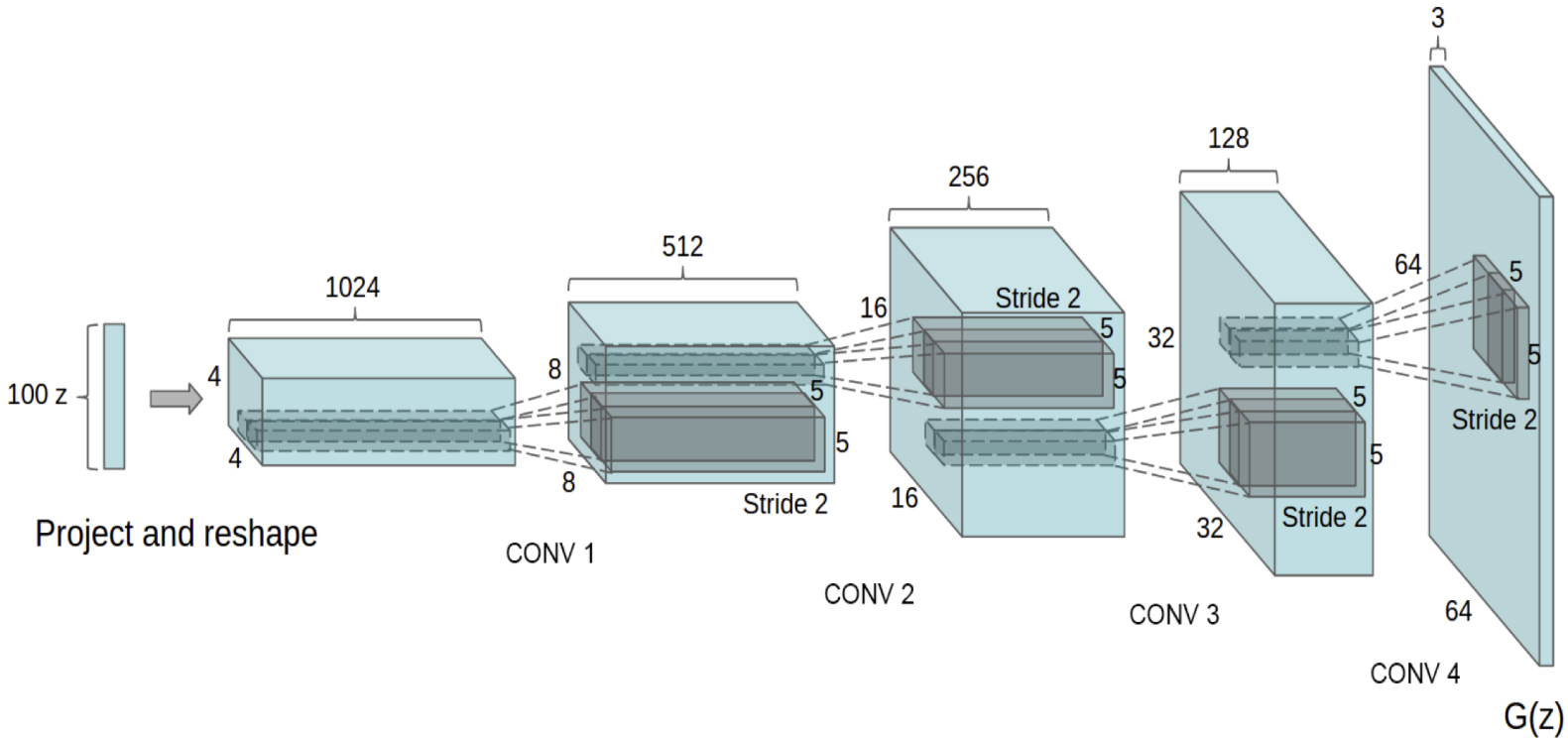
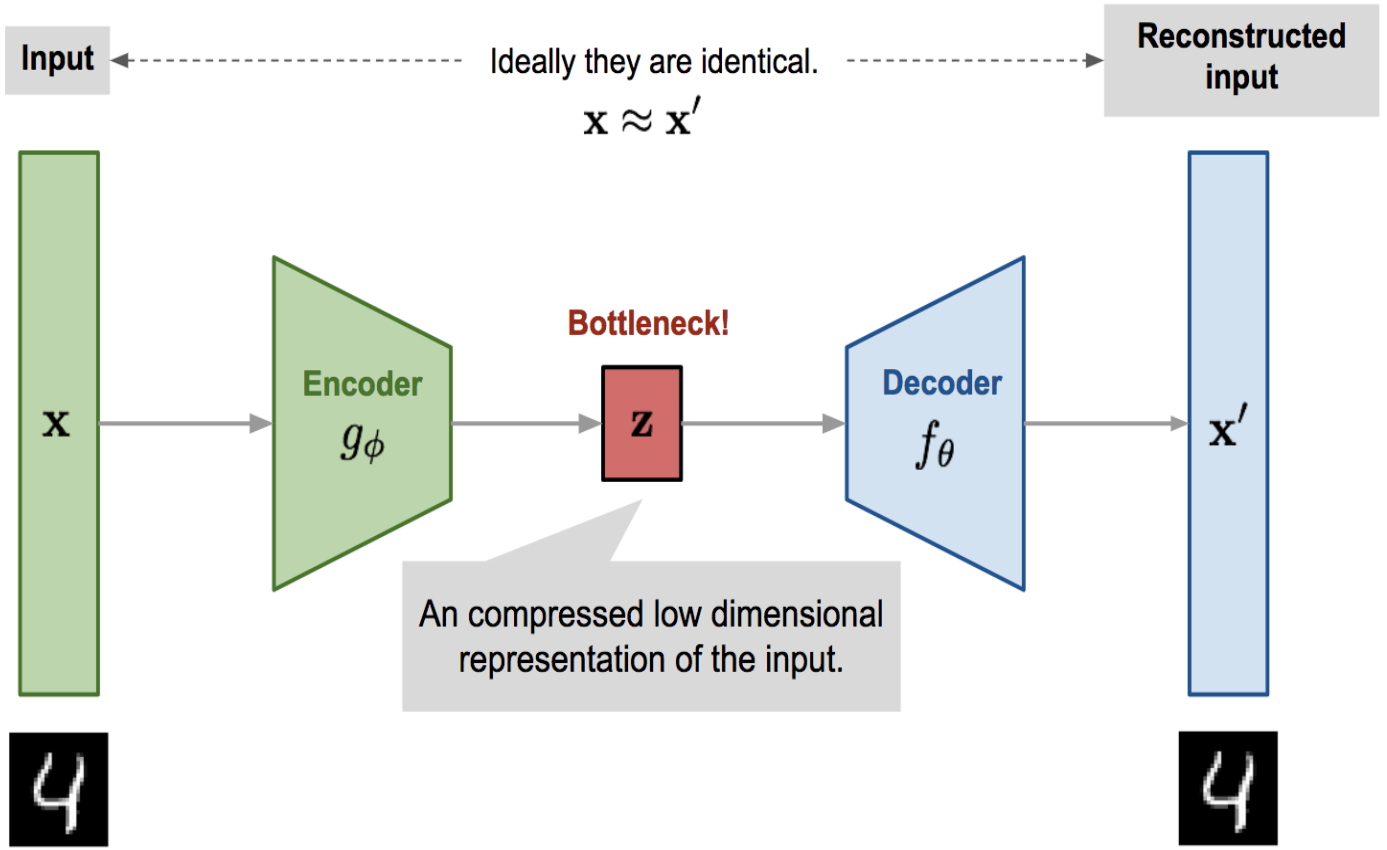


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. [6]

[6] https://arxiv.org/pdf/1511.06434.pdf

[7] https://arxiv.org/pdf/1406.2661.pdf

Autoencoder Illustration



The parameters, (θ, ϕ), of the autoencoder are then learned:

$$\mathcal{L}(\theta, \phi) = n^{-1} \sum_{\mu=1}^{n} l(x_\mu, f_\theta(g_\phi(x_\mu)))$$

[8] https://lilianweng.github.io/lil-log/2018/08/12/

Principal component analysis as an AE

The parameters, (θ, ϕ), of the autoencoder are then learned:

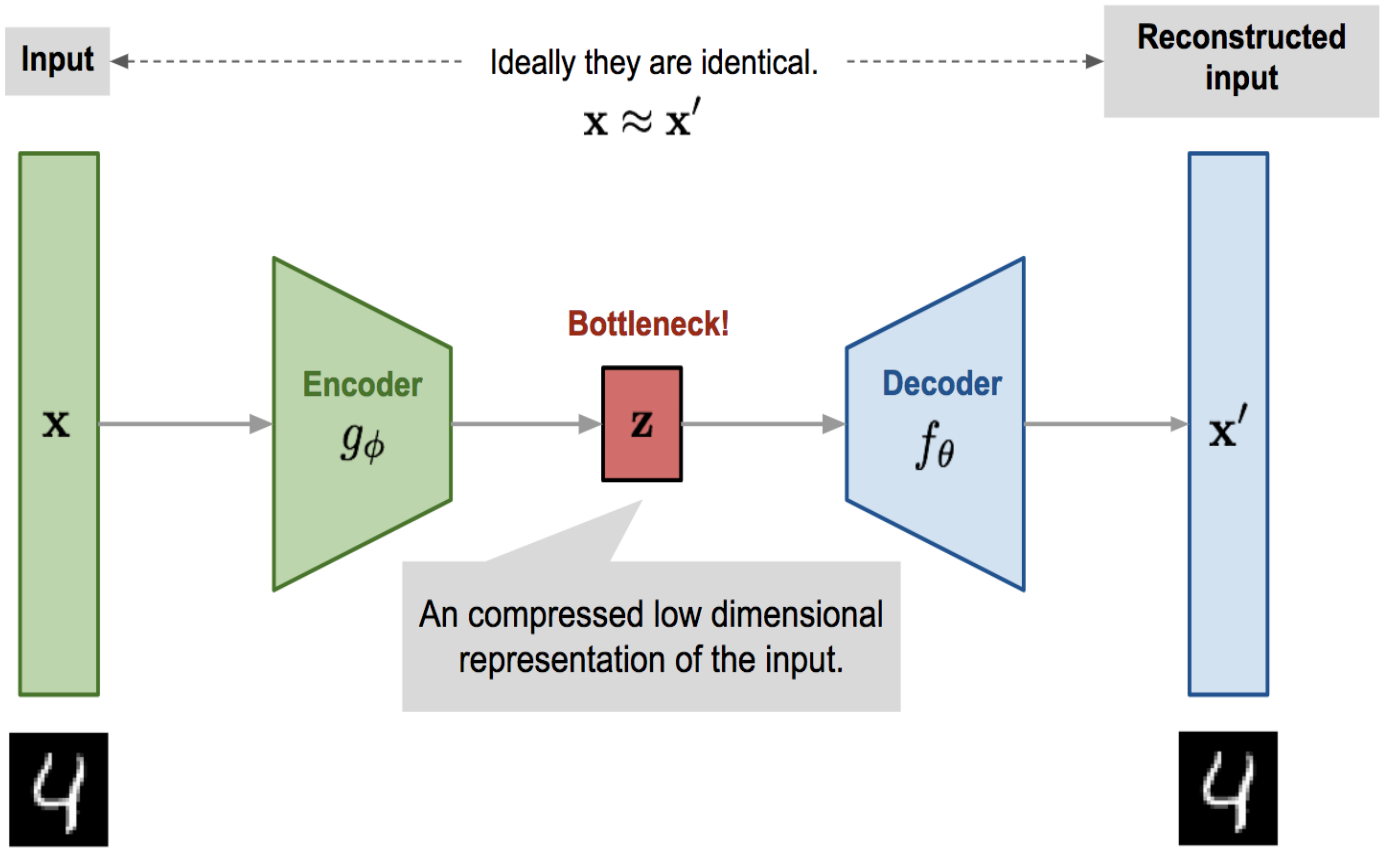
$$\mathcal{L}(\theta, \phi) = n^{-1} \sum_{\mu=1}^{n} l(x_\mu, f_\theta(g_\phi(x_\mu)))$$

Consider a simple model where the encoder and decoder are linear, that is $g_\phi(x) = \Phi x$ where $\Phi \in \mathbb{R}^{r \times p}$ with $r < p$, and the linear decoder $f_\theta(z) = \Theta z$ with $\Theta \in \mathbb{R}^{p \times r}$.
Moreover, consider an entrywise ℓ_2^2 error for $l(x_\mu, f_\theta(g_\phi(x_\mu)))$, then

$$\mathcal{L}(\theta, \phi) = n^{-1} \|X - \Theta \Phi X\|_F$$

where $\Theta\Phi$ is a learned rank r matrix, whose optimal solution is the projector of X to its leading r singular space.

Autoencoders extend PCA



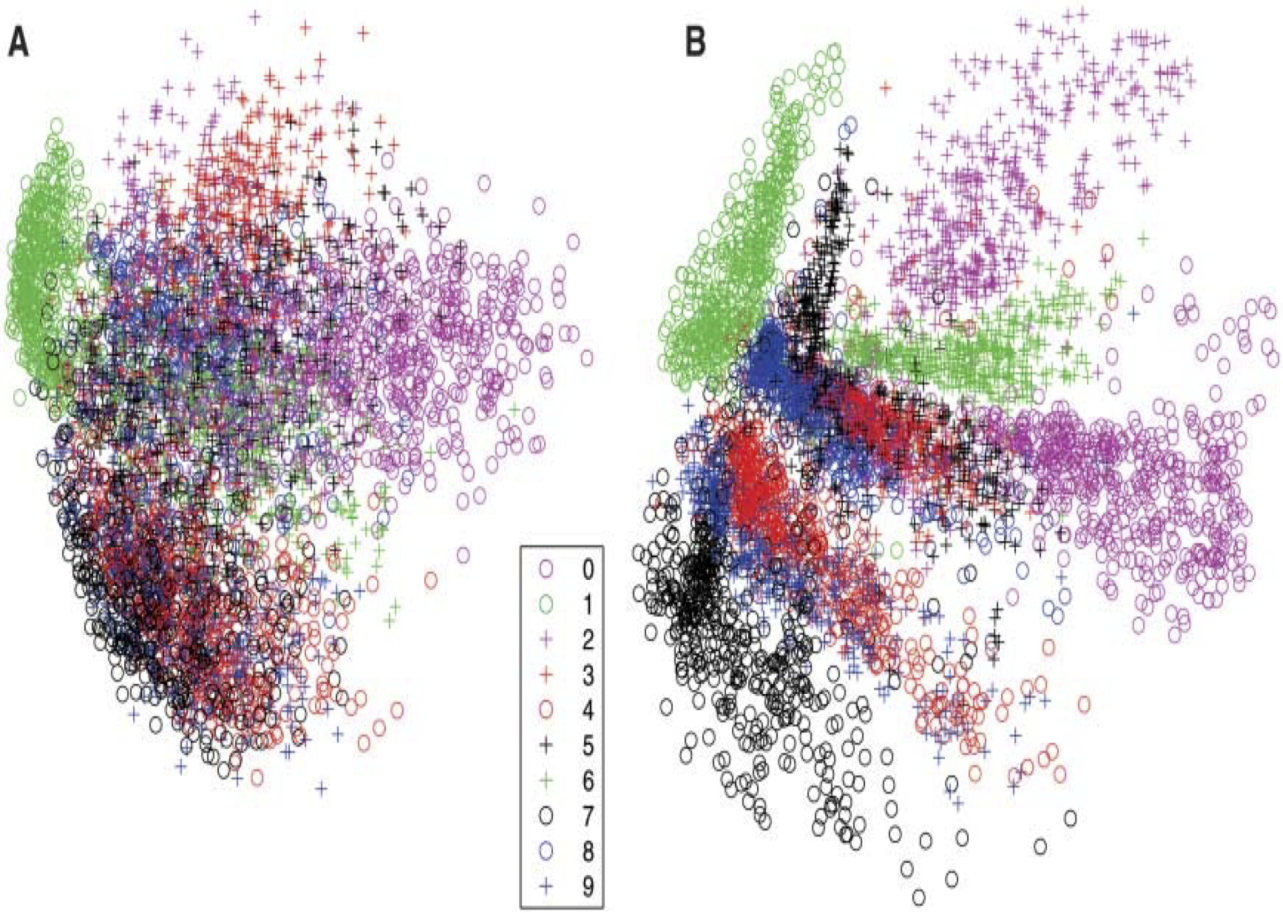
9

The autoencoder framework allows $g_\phi(\cdot)$ and $f_\theta(\cdot)$ to be more general than linear, and in particular to benefit from the expressively of depth and introduce variation.

[9] https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

PCA vs 3 layer Autoencoder: MNIST (Hinton et al. 06'[10])

Fig. 3. (**A**) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (**B**) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (*8*).



There are many variations on how to construct encoder-decoder pairs.

[10] http://science.sciencemag.org/content/313/5786/504

k-sparse autoencoders (Makhzani et al. 13'[11])

k-Sparse Autoencoders:

Training:

1) Perform the feedforward phase and compute

$$\boldsymbol{z} = W^\top \boldsymbol{x} + \boldsymbol{b}$$

2) Find the k largest activations of $\boldsymbol{z}$ and set the rest to zero.

$$\boldsymbol{z}_{(\Gamma)^c} = 0 \quad \text{where} \quad \Gamma = \text{supp}_k(\boldsymbol{z})$$

3) Compute the output and the error using the sparsified $\boldsymbol{z}$.

$$\hat{\boldsymbol{x}} = W\boldsymbol{z} + \boldsymbol{b}'$$
$$E = \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2$$

3) Backpropagate the error through the k largest activations defined by Γ and iterate.

Sparse Encoding:

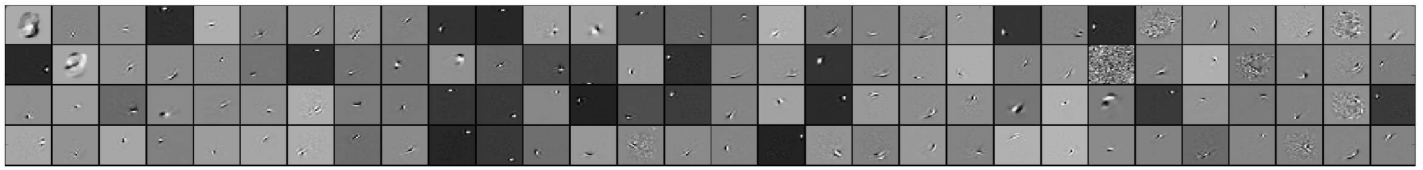
Compute the features $\boldsymbol{h} = W^\top \boldsymbol{x} + \boldsymbol{b}$. Find its αk largest activations and set the rest to zero.

$$\boldsymbol{h}_{(\Gamma)^c} = 0 \quad \text{where} \quad \Gamma = \text{supp}_{\alpha k}(\boldsymbol{h})$$

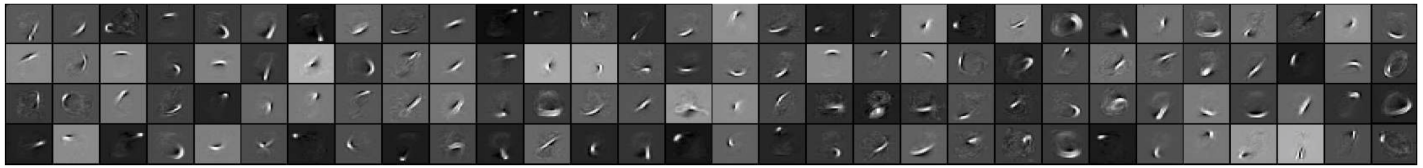
This framework includes nonlinearity and can be rigorously analysed using the material such as in earlier sparse approximation lectures, it lacks depth.

[11] https://arxiv.org/pdf/1312.5663.pdf

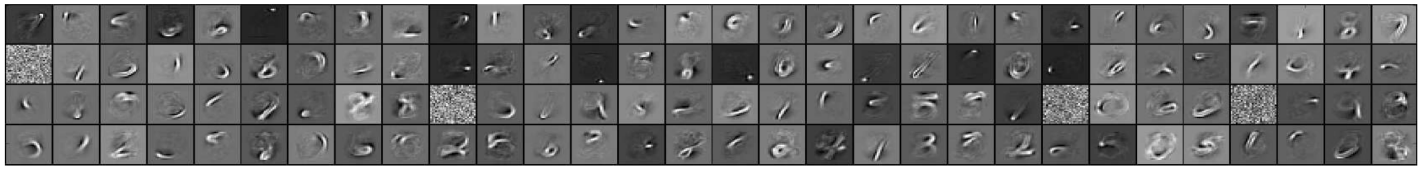
k-sparse autoencoders (Makhzani et al. 13'[12])



(a) $k = 70$



(b) $k = 40$



(c) $k = 25$

(d) $k = 10$

Figure 1. Filters of the k-sparse autoencoder for different sparsity levels k, learnt from MNIST with 1000 hidden units.

This framework includes nonlinearity and can be rigorously analysed using the material such as in earlier sparse approximation lectures, it lacks depth.

[12] https://arxiv.org/pdf/1312.5663.pdf

k-sparse autoencoders (Makhzani et al. 13'[13])

	Error Rate
Raw Pixels	7.20%
RBM	1.81%
Dropout Autoencoder (50% hidden)	1.80%
Denoising Autoencoder (20% input dropout)	1.95%
Dropout + Denoising Autoencoder (20% input and 50% hidden)	1.60%
k-Sparse Autoencoder, $k = 40$	1.54%
k-Sparse Autoencoder, $k = 25$	1.35%
k-Sparse Autoencoder, $k = 10$	2.10%

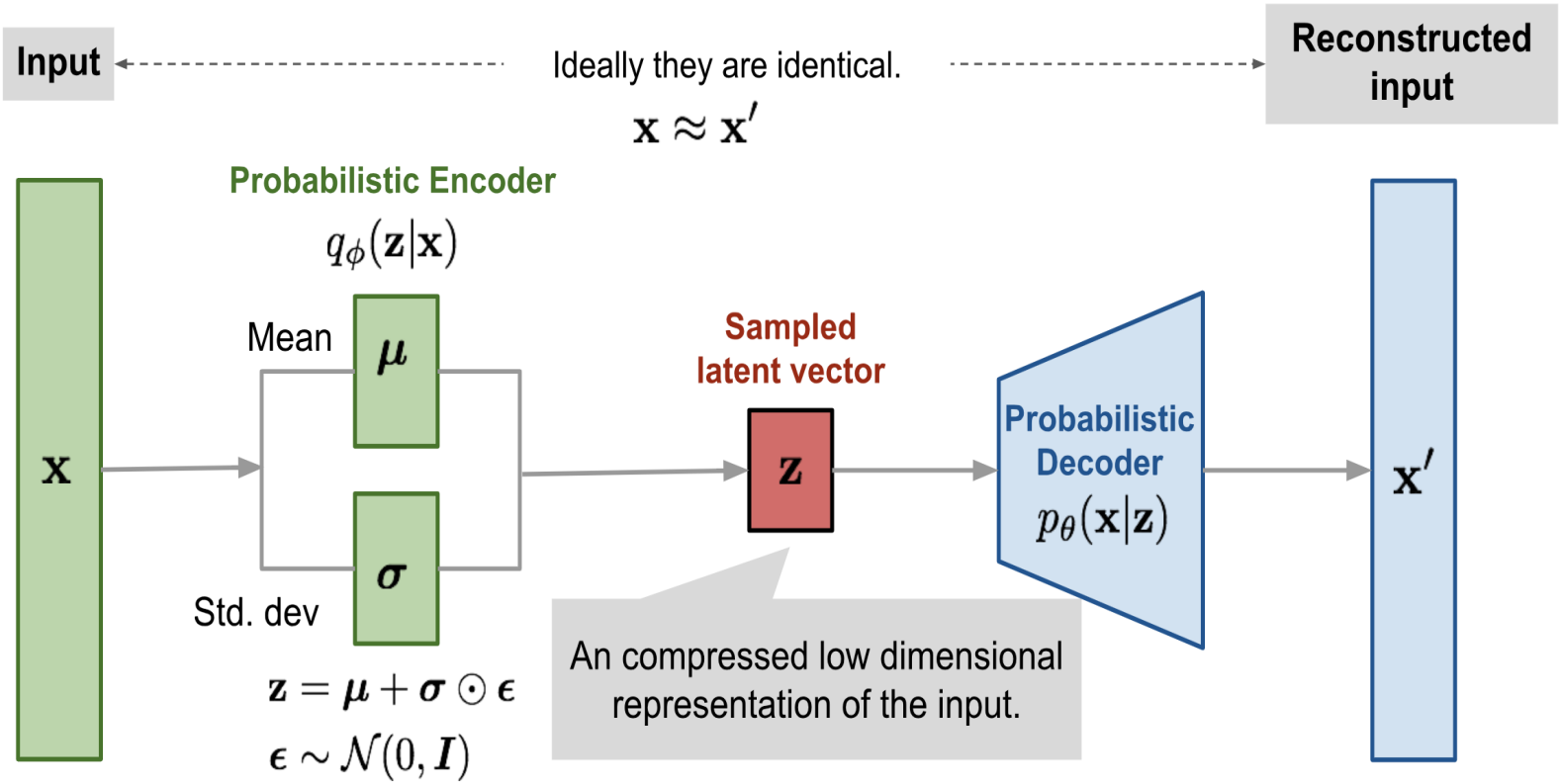
Table 1. Performance of unsupervised learning methods (without fine-tuning) with 1000 hidden units on **MNIST**.

	Error
Without Pre-Training	1.60%
RBM + F.T.	1.24%
Shallow Dropout AE + F.T. (%50 hidden)	1.05%
Denoising AE + F.T. (%20 input dropout)	1.20%
Deep Dropout AE + F.T. (Layer-wise pre-training, %50 hidden)	0.85%
k-Sparse AE + F.T. (k=25)	1.08%
Deep k-Sparse AE + F.T. (Layer-wise pre-training)	0.97%

Table 3. Performance of supervised learning methods on **MNIST**. Pre-training was performed using the corresponding unsupervised learning algorithm with 1000 hidden units, and then the model was fine-tuned.

[13] https://arxiv.org/pdf/1312.5663.pdf

Variational Autoencoders (Kingma et al. 13'[15])

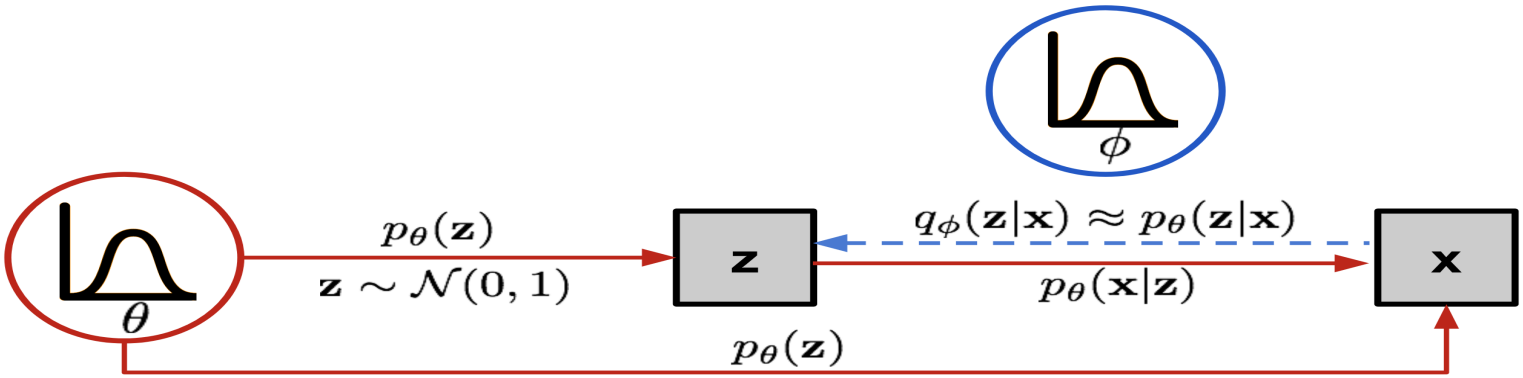


14

[14] https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

[15] https://arxiv.org/pdf/1312.6114.pdf

Variational Autoencoders (Kingma et al. 13'[17])



16

$p_\theta(x|z)$ acts as the generators, analogous to the decoder $f_\theta(x|z)$, and is called a probabilistic decoder
$q_\phi(z|x)$ acts as the encoder, analogous to $g_\phi(z|x)$, and is used to approximate $p_\theta(z|x)$.
The parameters ϕ, θ for a model are then learned so minimize a distance, or divergence, between $q_\phi(z|x)$ and $p_\theta(z|x)$; Kingma proposed minimising the Kullback-Leibler divergence.

[16] `https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html`

[17] `https://arxiv.org/pdf/1312.6114.pdf`

Evidence lower bound loss (Kingma et al. 13'[18])

We claimed before the VAE parameters ϕ, θ are given by

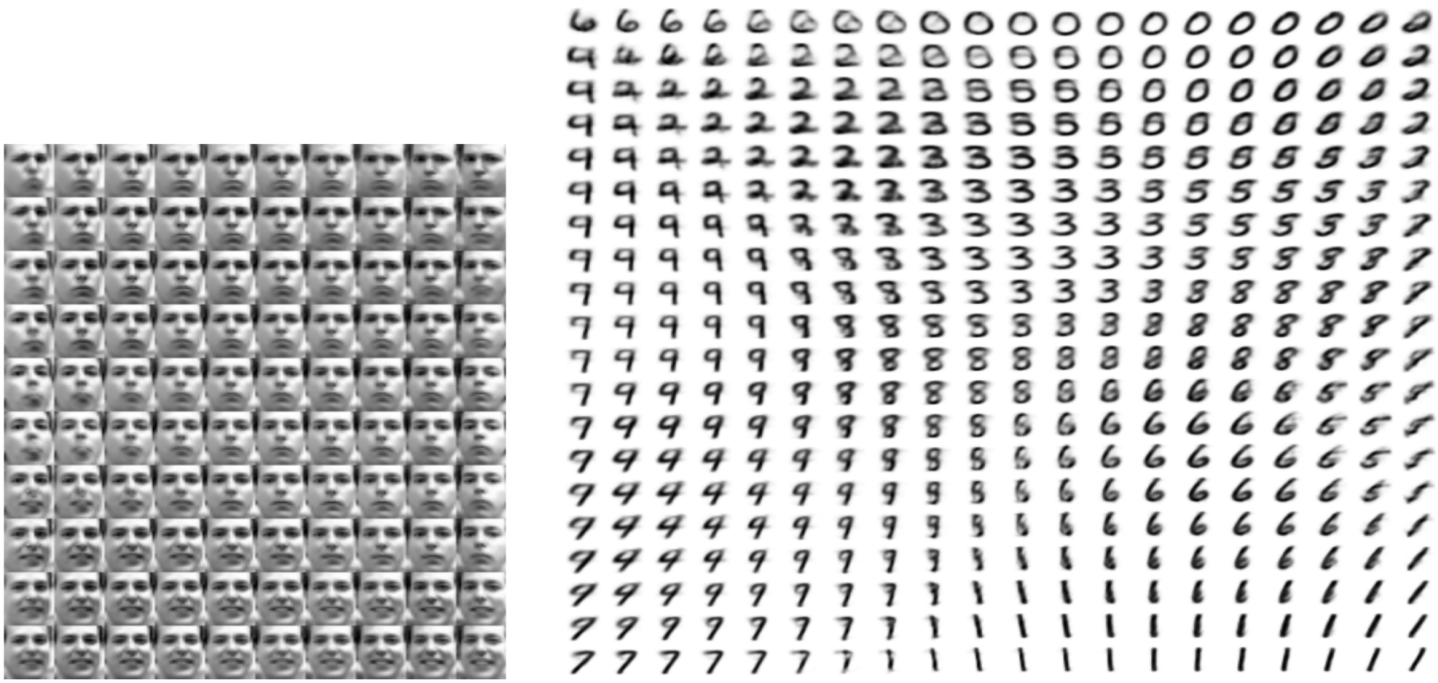
$$(\phi^*, \theta^*) = \text{argmin}_{(\phi,\theta)} \mathcal{L}_{ELBO}(\phi, \theta; X)$$

The formulae for $\mathcal{L}_{ELBO}(\phi, \theta; X)$, the evidence lower bound (ELBO), follows from minimising a lower bound of $\sum_{\mu=1}^{n} \log p_\theta(x_\mu)$:

$$\begin{aligned}
\log p_\theta(x) &= \log \left(\int p_\theta(x|z) p_\theta(z) dz \right) \\
&= \log \left(\int p_\theta(x|z) \frac{p_\theta(z)}{q_\phi(z)} q_\phi(z) dz \right) \\
&\geq \int \log \left(p_\theta(x|z) \frac{p_\theta(z)}{q_\phi(z)} \right) q_\phi(z) dz \\
&= \mathbb{E}_{q_\phi(z|x)} \log(p_\theta(x|z)) - D_{KL}(q_\phi(z|x)|p_\theta(z|x)) \\
&=: \mathcal{L}_{ELBO}(\phi, \theta; x)
\end{aligned}$$

[18] https://arxiv.org/pdf/1312.6114.pdf

Variational Autoencoder: manifold (Kingma et al. 13'[19])



(a) Learned Frey Face manifold

(b) Learned MNIST manifold

Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables $\mathbf{z}$. For each of these values $\mathbf{z}$, we plotted the corresponding generative $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with the learned parameters $\boldsymbol{\theta}$.

[19] https://arxiv.org/pdf/1312.6114.pdf

Variational Autoencoder: MNIST (Kingma et al. 13'[20])

(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Figure 5: Random samples from learned generative models of MNIST for different dimensionalities of latent space.

[20] https://arxiv.org/pdf/1312.6114.pdf

Inference Variational Autoencoders (Zhao et al. 17'[21])

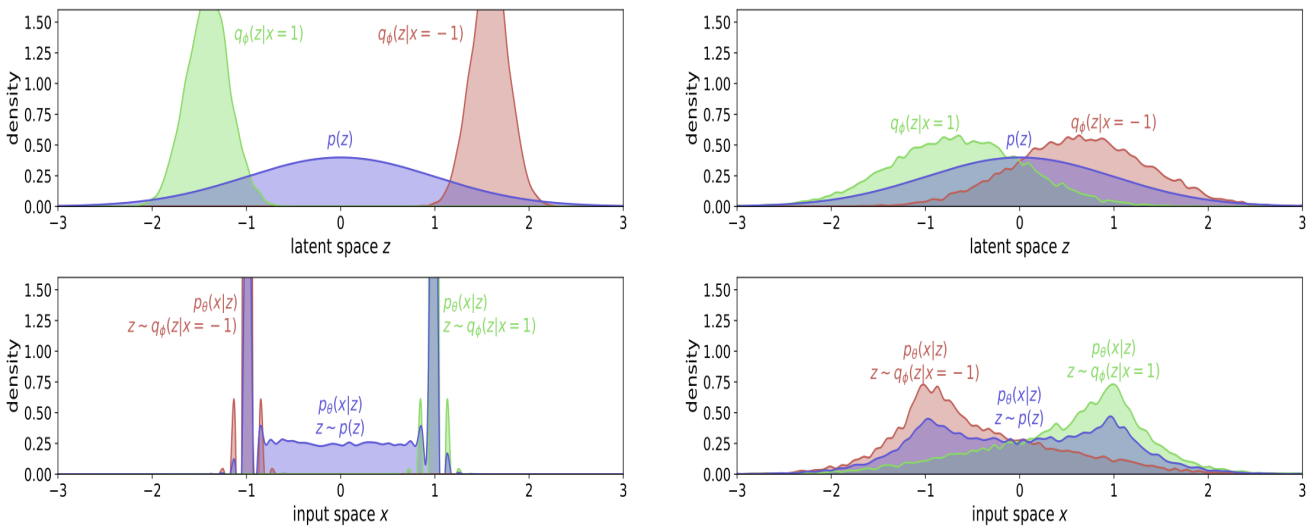
$p_\theta(x|z)$ acts as the generators, analogous to the decoder $f_\theta(x|z)$, and is called a probabilistic decoder
$q_\phi(z|x)$ acts as the encoder, analogous to $g_\phi(z|x)$, and is used to approximate $p_\theta(z|x)$.
The parameters ϕ, θ for a model are then learned so minimize a distance, or divergence, between $q_\phi(z|x)$ and $p_\theta(z|x)$; Kingma proposed minimising the Kullback-Leibler divergence, giving the evidence lower bound (ELBO)

$$\mathcal{L}_{ELBO} := \mathbb{E}_{q_{\phi(z|x)}} \log(p_\theta(x|z)) - \beta D_{KL}(q_\phi(x,z)||p_\theta(x,z))$$

VAEs originally use $\beta = 1$, with larger $\beta > 1$ called β-VAEs.
Zhao et al. propose including a mutual information term to avoid mode separation and collapse.

[21] https://arxiv.org/pdf/1706.02262.pdf

Inference Variational Autoencoders (Zhao et al. 17'[22])



ELBO InfoVAE ($\lambda = 500$)

Figure 1: Verification of Proposition 1 where the dataset only contains two examples $\{-1, 1\}$. **Top:** density of the distributions $q_\phi(z|x)$ when $x = 1$ (red) and $x = -1$ (green) compared with the true prior $p(z)$ (purple). **Bottom:** The "reconstruction" $p_\theta(x|z)$ when z is sampled from $q_\phi(z|x = 1)$ (green) and $q_\phi(z|x = -1)$ (red). Also plotted is $p_\theta(x|z)$ when z is sampled from the true prior $p(z)$ (purple). When the dataset consists of only two data points, ELBO (**left**) will push the density in latent space Z away from 0, while InfoVAE (**right**) does not suffer from this problem.

[22] https://arxiv.org/pdf/1706.02262.pdf

Inference Variational Autoencoders (Zhao et al. 17'[23])

Figure 5: Samples generated by ELBO vs. MMD InfoVAE ($\lambda = 1000$) after training on 500 samples (plotting mean of $p_\theta(x|z)$). **Top:** Samples generated by ELBO. Even though ELBO generates very sharp reconstruction for samples on the training set, model samples $p(z)p_\theta(x|z)$ is very poor, and differ significantly from the reconstruction samples, indicating over-fitting, and mismatch between $q_\phi(z)$ and $p(z)$. **Bottom:** Samples generated by InfoVAE. The reconstructed samples and model samples look similar in quality and appearance, suggesting better generalization in the latent space.

[23] https://arxiv.org/pdf/1706.02262.pdf

β-VAEs disentangling features pt. 1 (Higgins et al. 17'[24])

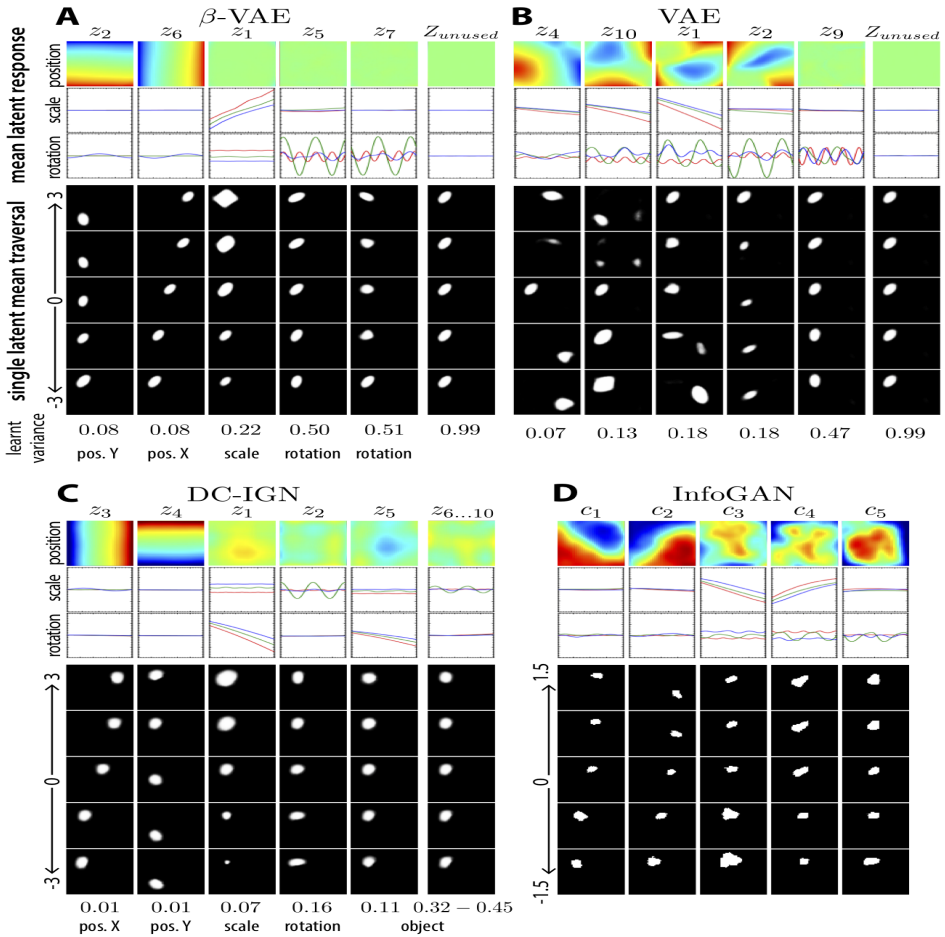
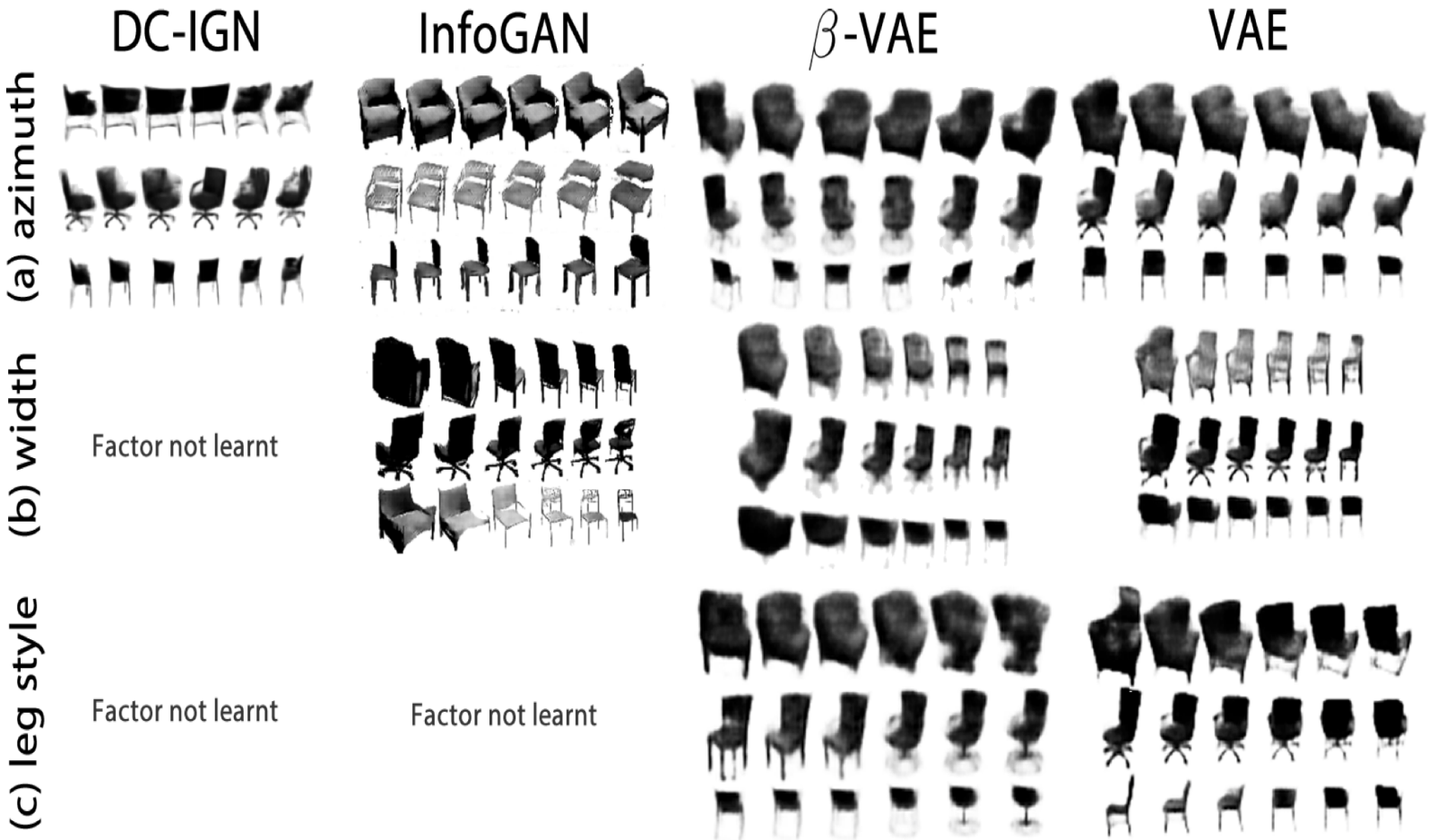


Figure 7: **A:** Representations learnt by a β-VAE ($\beta = 4$). Each column represents a latent z_i, ordered according to the learnt Gaussian variance (last row). Row 1 (position) shows the mean activation (red represents high values) of each latent z_i as a function of all 32x32 locations averaged across objects, rotations and scales. Row 2 and 3 show the mean activation of each unit z_i as a function of scale (respectively rotation), averaged across rotations and positions (respectively scales and positions). *Square* is red, *oval* is green and *heart* is blue. Rows 4-8 (second group) show reconstructions resulting from the traversal of each latent z_i over three standard deviations around the unit Gaussian prior mean while keeping the remaining 9/10 latent units fixed to the values obtained by running inference on an image from the dataset. **B:** Similar analysis for VAE ($\beta = 1$). **C:** Similar analysis for DC-IGN, clamping a single latent each for scale, positions, orientation and 5 for shape. **D:** Similar analysis for InfoGAN, using 5 continuous latents regularized using the mutual information cost, and 5 additional unconstrained noise latents (not shown).

[24] https://arxiv.org/pdf/1706.02262.pdf

β-VAEs disentangling features pt. 2 (Higgins et al. 17'[25])



[25] https://arxiv.org/pdf/1706.02262.pdf

β-VAEs architectures (Higgins et al. 17'[26])

Dataset	Optimiser		Architecture
2D shapes (VAE)	Adagrad 1e-2	Input	4096 (flattened 64x64x1).
		Encoder	FC 1200, 1200. ReLU activation.
		Latents	10
		Decoder	FC 1200, 1200, 1200, 4096. Tanh activation. Bernoulli.
2D shapes (DC-IGN)	rmsprop (as in Kulkarni et al., 2015)	Input	64x64x1.
		Encoder	Conv 96x3x3, 48x3x3, 48x3x3 (padding 1). ReLU activation and Max pooling 2x2.
		Latents	10
		Decoder	Unpooling, Conv 48x3x3, 96x3x3, 1x3x3. ReLU activation, Sigmoid.
2D shapes (InfoGAN)	Adam 1e-3 (gen) 2e-4 (dis)	Generator	FC 256, 256, Deconv 128x4x4, 64x4x4 (stride 2). Tanh.
		Discriminator	Conv and FC reverse of generator. Leaky ReLU activation. FC 1. Sigmoid activation.
		Recognition	Conv and FC shared with discriminator. FC 128, 5. Gaussian
		Latents	10: $z_{1\ldots5} \sim Unif(-1,1)$, $c_{1\ldots5} \sim Unif(-1,1)$
Chairs (VAE)	Adam 1e-4	Input	64x64x1.
		Encoder	Conv 32x4x4 (stride 2), 32x4x4 (stride 2), 64x4x4 (stride 2), 64x4x4 (stride 2), FC 256. ReLU activation.
		Latents	32
		Decoder	Deconv reverse of encoder. ReLU activation. Bernoulli.

[26] https://arxiv.org/pdf/1706.02262.pdf