Coverage and quality driven training of generative image models

Thomas Lucas, Konstantin Shmelkov, Karteek Alahari,
Cordelia Schmid, Jakob Verbeek

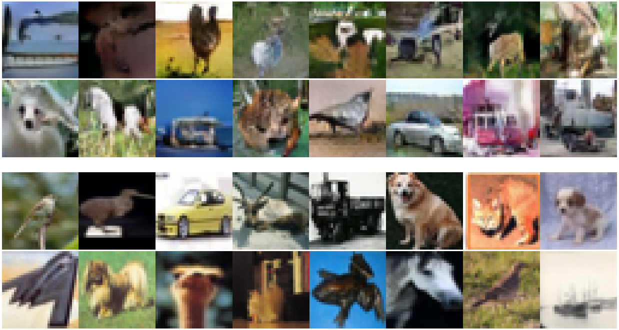
INRIA Grenoble, France

December 2018

What are generative (image) models?

1. Density estimator $p(x)$ from which we can sample

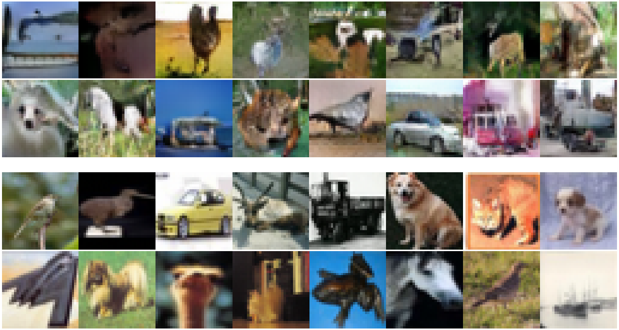
2. Models that generalize outside train data



CIFAR-10: 32×32 samples from model and training set

What are generative (image) models?

1. Density estimator $p(x)$ from which we can sample
2. Models that generalize outside train data
3. Models that allow to assess if (2) actually happened !



CIFAR-10: 32×32 samples from model and training set

Generative image models - motivation

- Sand-box problem to study complex density estimation
  - Images: high-dimensional non-trivial distributions
- Conditional generative models are useful in practice
  - Generate image, speech,. . . conditioned on attributes, text, . . .
  - Conditioning on some input is the "easy" part
- Representation learning from unlabeled data
  - Leveraging latent variables and/or internal feature maps

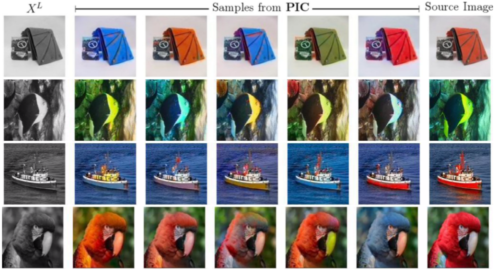


Image colorization results form [Royer et al., 2017]

Current approaches in deep generative modeling

- Autoregressive models, e.g. Pixel-CNN [Oord et al., 2016]
  - Model conditionals in $p(x) = \prod_i p(x_i | x_{<i})$, *e.g.* with CNN, RNN,...
  - Exact likelihoods, slow sequential sampling

Current approaches in deep generative modeling

- Autoregressive models, e.g. Pixel-CNN [Oord et al., 2016]
  - Model conditionals in $p(x) = \prod_i p(x_i | x_{<i})$, *e.g.* with CNN, RNN,...
  - Exact likelihoods, slow sequential sampling

- Flow-based methods, e.g. NVP, IAF
  [Dinh et al., 2017, Kingma et al., 2016a]
  - Exact likelihood through change of variable formula
  - Invertible network layers, with efficient Jacobian determinant

Current approaches in deep generative modeling

- Autoregressive models, e.g. Pixel-CNN [Oord et al., 2016]
  - Model conditionals in $p(x) = \prod_i p(x_i|x_{<i})$, *e.g.* with CNN, RNN,...
  - Exact likelihoods, slow sequential sampling
- Flow-based methods, e.g. NVP, IAF [Dinh et al., 2017, Kingma et al., 2016a]
  - Exact likelihood through change of variable formula
  - Invertible network layers, with efficient Jacobian determinant
- Variational inference [Kingma and Welling, 2014, Rezende et al., 2014]
  - Latent variable model $p(x) = \int_z p(z)p(x|z)$, with $z \in \mathbb{R}^d$
  - VAE: Decoder $p_\theta(x|z)$, encoder $q_\phi(z|x) \approx p(z|x)$

Current approaches in deep generative modeling

- Autoregressive models, e.g. Pixel-CNN [Oord et al., 2016]
  - Model conditionals in $p(x) = \prod_i p(x_i|x_{<i})$, *e.g.* with CNN, RNN,...
  - Exact likelihoods, slow sequential sampling

- Flow-based methods, e.g. NVP, IAF [Dinh et al., 2017, Kingma et al., 2016a]
  - Exact likelihood through change of variable formula
  - Invertible network layers, with efficient Jacobian determinant

- Variational inference [Kingma and Welling, 2014, Rezende et al., 2014]
  - Latent variable model $p(x) = \int_z p(z)p(x|z)$, with $z \in \mathbb{R}^d$
  - VAE: Decoder $p_\theta(x|z)$, encoder $q_\phi(z|x) \approx p(z|x)$

- Generative adversarial networks [Goodfellow et al., 2014]
  - Deterministic $x = G_\theta(z)$, low dim. support, likelihood-free
  - Use discriminator real/synth. samples as "trainable loss"

Adversarial and maximum likelihood training

- Discriminator in GAN trained with binary cross-entropy loss

$$\mathbb{E}_{p_{\text{train}}(x)}[\ln D(x)] + \mathbb{E}_{p_\theta(x)}\left[\ln\left(1 - D(x)\right)\right] \tag{1}$$

Adversarial and maximum likelihood training

- Discriminator in GAN trained with binary cross-entropy loss

$$\mathbb{E}_{p_{\text{train}}(x)}[\ln D(x)] + \mathbb{E}_{p_\theta(x)}\left[\ln\left(1 - D(x)\right)\right] \tag{1}$$

- Train GAN generator with sum both losses proposed by [Goodfellow et al., 2014], see for example [Sønderby et al., 2017]

$$\mathcal{L}_Q(\theta) = -\mathbb{E}_{p(z)}\left[\ln \frac{D(G_\theta(z))}{1 - D(G_\theta(z))}\right] \tag{2}$$

Adversarial and maximum likelihood training

- Discriminator in GAN trained with binary cross-entropy loss

$$\mathbb{E}_{p_{\text{train}}(x)}[\ln D(x)] + \mathbb{E}_{p_\theta(x)}\left[\ln\left(1 - D(x)\right)\right] \tag{1}$$

- Train GAN generator with sum both losses proposed by [Goodfellow et al., 2014], see for example [Sønderby et al., 2017]

$$\mathcal{L}_Q(\theta) = -\mathbb{E}_{p(z)}\left[\ln \frac{D(G_\theta(z))}{1 - D(G_\theta(z))}\right] \tag{2}$$

- For optimal discriminator: $\mathcal{L}_Q(\theta) = D_{\text{KL}}(p_\theta || p_{\text{train}})$

Adversarial and maximum likelihood training

- Discriminator in GAN trained with binary cross-entropy loss

$$\mathbb{E}_{p_{\text{train}}(x)}[\ln D(x)] + \mathbb{E}_{p_\theta(x)}\left[\ln\left(1 - D(x)\right)\right] \tag{1}$$

- Train GAN generator with sum both losses proposed by [Goodfellow et al., 2014], see for example [Sønderby et al., 2017]

$$\mathcal{L}_Q(\theta) = -\mathbb{E}_{p(z)}\left[\ln \frac{D(G_\theta(z))}{1 - D(G_\theta(z))}\right] \tag{2}$$

- For optimal discriminator: $\mathcal{L}_Q(\theta) = D_{\text{KL}}(p_\theta || p_{\text{train}})$

- Compare to maximum likelihood training

$$\mathcal{L}_C(\theta) = -\mathbb{E}_{p_{\text{train}}(x)}[\ln p_\theta(x)] \tag{3}$$

Adversarial and maximum likelihood training

- Discriminator in GAN trained with binary cross-entropy loss

$$\mathbb{E}_{p_{\text{train}}(x)}[\ln D(x)] + \mathbb{E}_{p_\theta(x)}\left[\ln\left(1 - D(x)\right)\right] \tag{1}$$

- Train GAN generator with sum both losses proposed by [Goodfellow et al., 2014], see for example [Sønderby et al., 2017]

$$\mathcal{L}_Q(\theta) = -\mathbb{E}_{p(z)}\left[\ln \frac{D(G_\theta(z))}{1 - D(G_\theta(z))}\right] \tag{2}$$

- For optimal discriminator: $\mathcal{L}_Q(\theta) = D_{\text{KL}}(p_\theta || p_{\text{train}})$

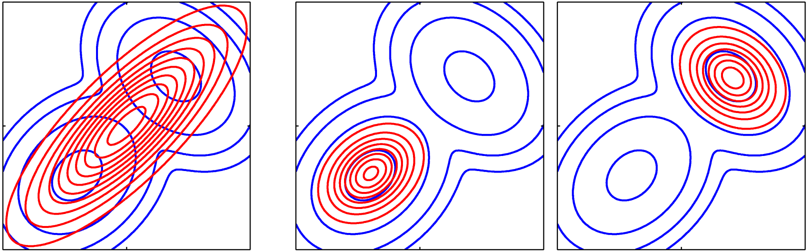
- Compare to maximum likelihood training

$$\mathcal{L}_C(\theta) = -\mathbb{E}_{p_{\text{train}}(x)}[\ln p_\theta(x)] \tag{3}$$

- Adding a constant we obtain $\mathcal{L}_C(\theta) - H(p_{\text{train}}) = D_{\text{KL}}(p_{\text{train}} || p_\theta)$

Mode dropping and over-generalization

- Reversing KL direction yields qualitatively different estimators
  - [Bishop, 2006]: "zero avoiding" or "zero forcing" behavior



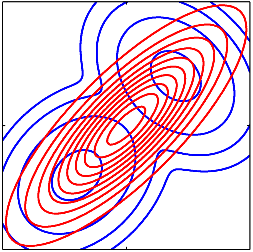
$D_{\mathsf{KL}}(p||q)$ $D_{\mathsf{KL}}(q||p)$

Mode dropping and over-generalization

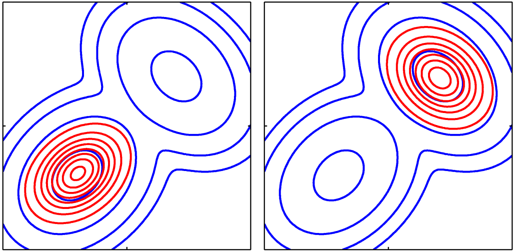
- Reversing KL direction yields qualitatively different estimators
  - [Bishop, 2006]: "zero avoiding" or "zero forcing" behavior

- GANs give nice samples, but mode-drop.

- Likelihood-based models over-generalize to poor samples



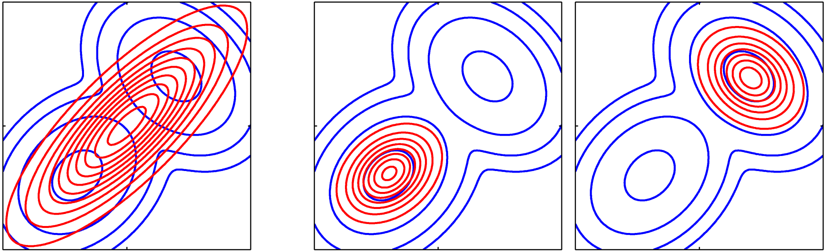
$D_{\mathrm{KL}}(p||q)$



$D_{\mathrm{KL}}(q||p)$

Mode dropping and over-generalization

- Reversing KL direction yields qualitatively different estimators
  - [Bishop, 2006]: "zero avoiding" or "zero forcing" behavior
- GANs give nice samples, but mode-drop.
- Likelihood-based models over-generalize to poor samples



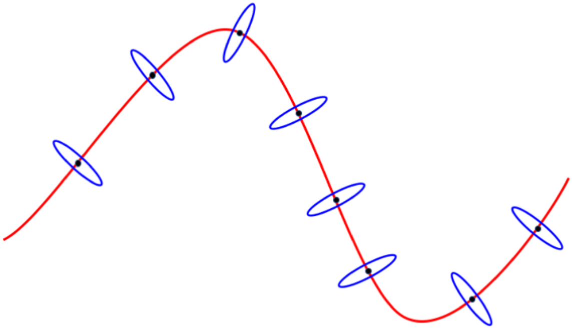
$D_{\mathrm{KL}}(p||q)$
Expectation propagation

$D_{\mathrm{KL}}(q||p)$
Variational inference

Limitation of maximum likelihood estimation

- Only measures the mass on the train data, invariant to where the rest of the mass goes

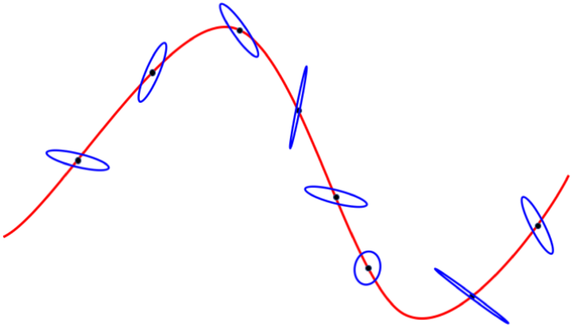
$$\mathcal{L}_C(\theta) = -\mathbb{E}_{p_{\text{train}}(x)}[\ln p_\theta(x)] \approx \frac{1}{n}\sum_{i=1}^{n} \ln p_\theta(x_i) \qquad (4)$$



Limitation of maximum likelihood estimation

- Only measures the mass on the train data, invariant to where the rest of the mass goes

$$\mathcal{L}_C(\theta) = -\mathbb{E}_{p_{\text{train}}(x)}[\ln p_\theta(x)] \approx \frac{1}{n}\sum_{i=1}^{n} \ln p_\theta(x_i) \qquad (4)$$

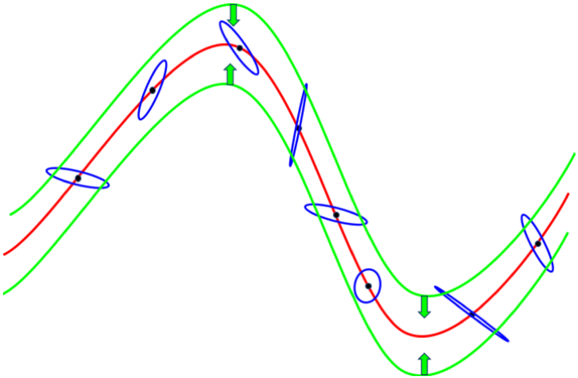


Limitation of maximum likelihood estimation

- Only measures the mass on the train data, invariant to where the rest of the mass goes

$$\mathcal{L}_C(\theta) = -\mathbb{E}_{p_{\text{train}}(x)}[\ln p_\theta(x)] \approx \frac{1}{n}\sum_{i=1}^{n} \ln p_\theta(x_i) \qquad (4)$$

- Idea 1: Use adversarial discriminator to break this invariance
  - Unlike MLE, discriminator is sensitive to model samples



Limitation of basic VAE decoders

- "Vanilla" VAE decoders factorize over data x given latent z

$$p(x|z) = \mathcal{N}\left(x; \mu(z), \mathrm{diag}\left(\sigma(z)\right)\right) \tag{5}$$

Limitation of basic VAE decoders

- "Vanilla" VAE decoders factorize over data x given latent z
  - Assumes pixels are independent (!) given latent code

$$p(x|z) = \mathcal{N}\left(x; \mu(z), \mathrm{diag}\left(\sigma(z)\right)\right) \tag{5}$$

Limitation of basic VAE decoders

- "Vanilla" VAE decoders factorize over data x given latent z
  - Assumes pixels are independent (!) given latent code

$$p(x|z) = \mathcal{N}\left(x; \mu(z), \mathrm{diag}\left(\sigma(z)\right)\right) \tag{5}$$

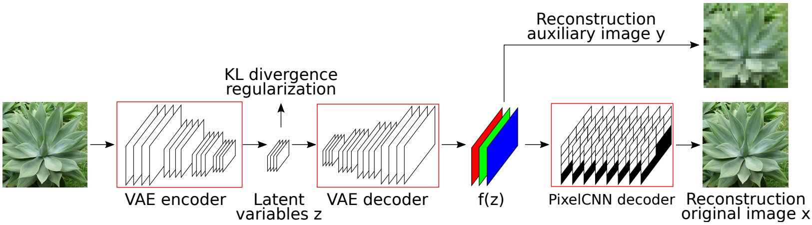
- First idea: use non-factorial Gaussian

Limitation of basic VAE decoders

- "Vanilla" VAE decoders factorize over data x given latent z
  - Assumes pixels are independent (!) given latent code

$$p(x|z) = \mathcal{N}\left(x; \mu(z), \mathrm{diag}\left(\sigma(z)\right)\right) \tag{5}$$

- First idea: use non-factorial Gaussian
- Powerful non-Gaussian decoders, *e.g.* conditional pixel-CNN [Chen et al., 2017, Gulrajani et al., 2017b, Lucas and Verbeek, 2018]

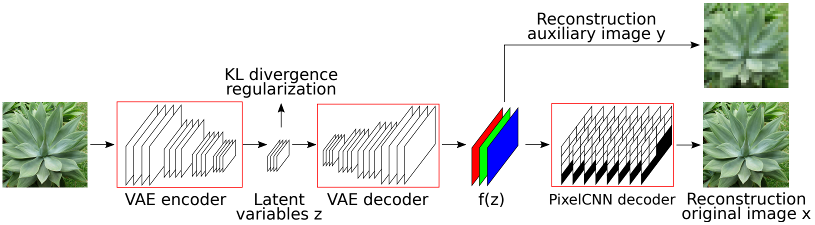


Limitation of basic VAE decoders

- "Vanilla" VAE decoders factorize over data x given latent z
  - Assumes pixels are independent (!) given latent code

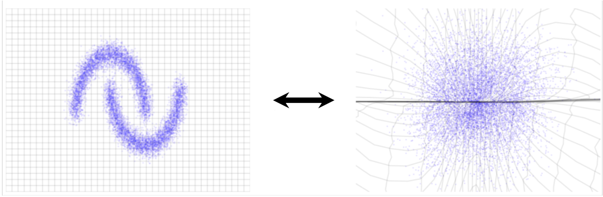
$$p(x|z) = \mathcal{N}\left(x; \mu(z), \mathrm{diag}\left(\sigma(z)\right)\right) \tag{5}$$

- First idea: use non-factorial Gaussian
- Powerful non-Gaussian decoders, *e.g.* conditional pixel-CNN [Chen et al., 2017, Gulrajani et al., 2017b, Lucas and Verbeek, 2018]
  - Too slow to sample sequential pixelCNN during training



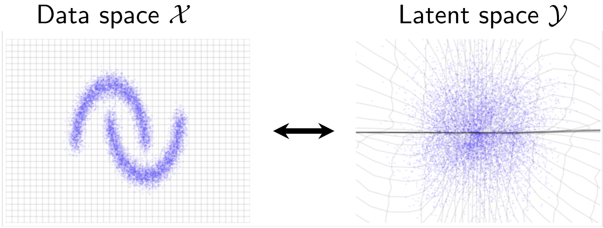
Idea 2: Use NVP in a VAE decoder

Data space $\mathcal{X}$ Latent space $\mathcal{Y}$



- Invertible transformation between image x and feature y

Idea 2: Use NVP in a VAE decoder



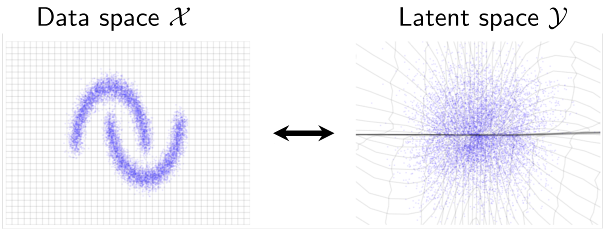
- Invertible transformation between image x and feature y
- Factorized Gaussian decoder over feature y given z

$$p_y(y|z) = \mathcal{N}\left(y; \mu(z), \mathrm{diag}\left(\sigma(z)\right)\right) \tag{6}$$

$$x = f^{-1}(y) \tag{7}$$

$$p_x(x|z) = p_y(f(x)|z) \times \left|\det\left(\frac{\partial f(x)}{\partial x^\top}\right)\right| \tag{8}$$

Idea 2: Use NVP in a VAE decoder

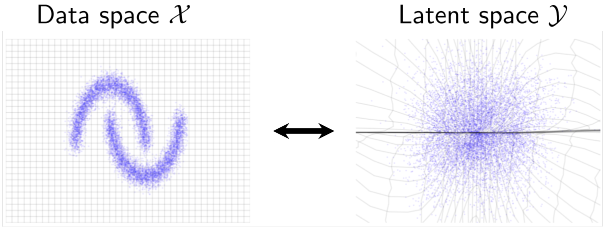


- Invertible transformation between image x and feature y
- Factorized Gaussian decoder over feature y given z

$$p_y(y|z) = \mathcal{N}\left(y; \mu(z), \mathrm{diag}\left(\sigma(z)\right)\right) \tag{6}$$
$$x = f^{-1}(y) \tag{7}$$
$$p_x(x|z) = p_y(f(x)|z) \times \left|\det\left(\frac{\partial f(x)}{\partial x^\top}\right)\right| \tag{8}$$

- Non-factorial non-Gaussian conditional distribution $p(x|z)$

Idea 2: Use NVP in a VAE decoder



- Invertible transformation between image x and feature y
- Factorized Gaussian decoder over feature y given z

$$p_y(y|z) = \mathcal{N}\left(y; \mu(z), \mathrm{diag}\left(\sigma(z)\right)\right) \quad (6)$$

$$x = f^{-1}(y) \quad (7)$$

$$p_x(x|z) = p_y(f(x)|z) \times \left|\det\left(\frac{\partial f(x)}{\partial x^\top}\right)\right| \quad (8)$$

- Non-factorial non-Gaussian conditional distribution $p(x|z)$
  - Better samples & likelihoods

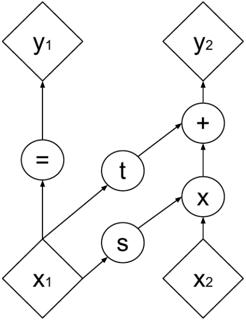
NVP in a nutshell

$$p_X(x) = p_Y(f(x)) \times \left|\det\left(\frac{\partial f(x)}{\partial x^\top}\right)\right|$$

NVP in a nutshell

$$p_X(x) = p_Y(f(x)) \times \left| \det \left(\frac{\partial f(x)}{\partial x^\top} \right) \right|$$

1. Partition variables in two groups

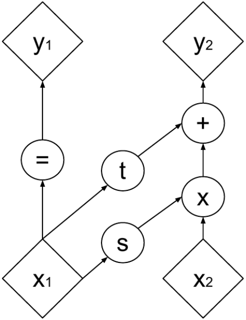


NVP in a nutshell

$$p_X(x) = p_Y(f(x)) \times \left|\det\left(\frac{\partial f(x)}{\partial x^\top}\right)\right|$$

1. Partition variables in two groups

2. Keep one group unchanged

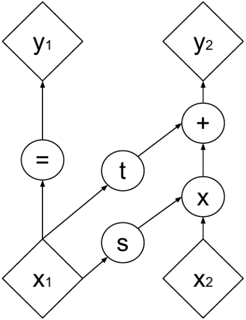


NVP in a nutshell

$$p_X(x) = p_Y(f(x)) \times \left| \det \left(\frac{\partial f(x)}{\partial x^\top} \right) \right|$$

1. Partition variables in two groups
2. Keep one group unchanged
3. Let one group transform the other via translation ans scaling

$$\begin{aligned} y_1 &= x_1 \\ y_2 &= t\left(x_1\right) + \mathrm{diag}\left(s(x_1)\right) \cdot x_2 \end{aligned}$$

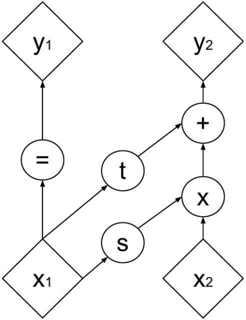


NVP in a nutshell

$$p_X(x) = p_Y(f(x)) \times \left| \det \left(\frac{\partial f(x)}{\partial x^\top} \right) \right|$$

1. Partition variables in two groups
2. Keep one group unchanged
3. Let one group transform the other via translation ans scaling

$$\begin{aligned} y_1 &= x_1 \\ y_2 &= t\left(x_1\right) + \mathrm{diag}\left(s(x_1)\right) \cdot x_2 \end{aligned}$$



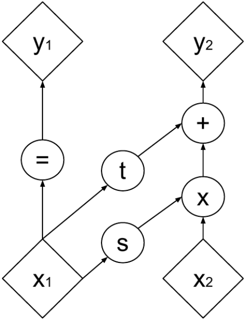
- Triangular Jacobian, determinant is prod$(s(x_1))$

NVP in a nutshell

$$p_X(x) = p_Y(f(x)) \times \left| \det \left(\frac{\partial f(x)}{\partial x^\top} \right) \right|$$

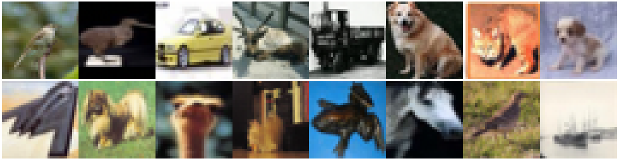
1. Partition variables in two groups
2. Keep one group unchanged
3. Let one group transform the other via translation ans scaling

$$\begin{aligned} y_1 &= x_1 \\ y_2 &= t\left(x_1\right) + \operatorname{diag}\left(s(x_1)\right) \cdot x_2 \end{aligned}$$



- Triangular Jacobian, determinant is $\operatorname{prod}(s(x_1))$
- Functions $s(\cdot)$ and $t(\cdot)$ may be non-invertible, *e.g.* CNN

Experimental setup



- Focus here on experiments on CIFAR-10 32×32
  - Also quant. + qual. evaluation on STL-10, CelebA, ImageNet, LSUN-bedrooms
- Evaluation metrics
  - Bits per dimension (i.e. negative log-likelihood)
  - Inception score [Salimans et al., 2016]: images should have low label-entropy, and high marginal label entropy
  - Fréchet inception distance [Heusel et al., 2017]: distance real and sampled images in 1st and 2nd moments CNN features

Impact of training objectives and NVP decoder

VAE

CIFAR-10

	$\mathcal{L}_Q$	$\mathcal{L}_C$	NVP	BPD ↓	IS ↑	FID ↓
VAE		✓		4.4	2.0	171.0
GAN	✓			7.0 *	6.8	31.4

*: Obtained using VAE with frozen GAN decoder

GAN

Impact of training objectives and NVP decoder

CIFAR-10

	$\mathcal{L}_Q$	$\mathcal{L}_C$	NVP	BPD $\downarrow$	IS $\uparrow$	FID $\downarrow$
VAE		✓		4.4	2.0	171.0
VAE-F		✓	✓	**3.5**	3.0	112.0
GAN	✓			7.0 *	6.8	31.4

*: Obtained using VAE with frozen GAN decoder

VAE

VAE-F

GAN

Impact of training objectives and NVP decoder

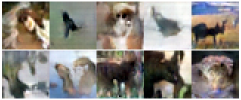
CIFAR-10

	$\mathcal{L}_Q$	$\mathcal{L}_C$	NVP	BPD ↓	IS ↑	FID ↓
VAE		✓		4.4	2.0	171.0
VAE-F		✓	✓	**3.5**	3.0	112.0
CQ	✓	✓		4.4	5.1	58.6
GAN	✓			7.0 *	6.8	31.4

*: Obtained using VAE with frozen GAN decoder

VAE

VAE-F



CQ

GAN

Impact of training objectives and NVP decoder

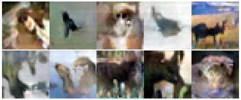
CIFAR-10

	$\mathcal{L}_Q$	$\mathcal{L}_C$	NVP	BPD ↓	IS ↑	FID ↓
VAE		✓		4.4	2.0	171.0
VAE-F		✓	✓	**3.5**	3.0	112.0
CQ	✓	✓		4.4	5.1	58.6
CQ-F	✓	✓	✓	3.9	**7.1**	**28.0**
GAN	✓			7.0 *	6.8	31.4

*: Obtained using VAE with frozen GAN decoder


VAE


VAE-F


CQ


CQ-F


GAN

Evaluation of more advanced architectures

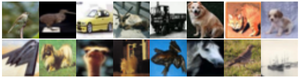
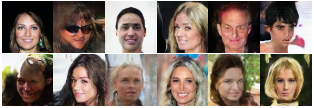
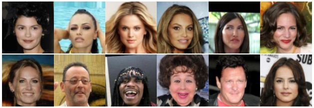
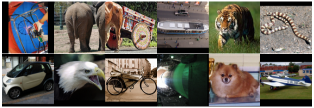
CIFAR-10

	IAF	Residual	BPD $\downarrow$	IS $\uparrow$	FID $\downarrow$
GAN			7.0 (*)	6.8	31.4
GAN		✓	—	7.4	24.0
CQF			3.90	7.1	28.0
CQF		✓	3.84	7.5	26.0
CQF	✓	✓	3.77	7.9	20.1
CQF (large Discr.)	✓	✓	**3.74**	**8.1**	**18.6**

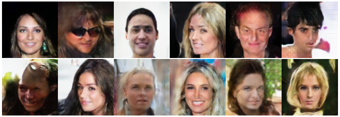
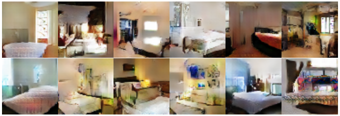
Comparison to the state of the art

	CIFAR-10			STL-10		
	BPD ↓	IS ↑	FID ↓	BPD ↓	IS ↑	FID ↓
DCGAN [Radford et al., 2016]		6.6				
SNGAN [Miyato et al., 2018]		7.4	29.3		8.3	53.1
SNGAN-Hinge [Miyato et al., 2018]					8.7	47.5
BatchGAN [Lucas et al., 2018]		7.5	23.7		8.7	51
WGAN-GP [Gulrajani et al., 2017a]		7.9				
Improved Training GAN [Salimans et al., 2016]		8.1				
SNGAN-ResNet-Hinge [Miyato et al., 2018]		8.2	21.7		**9.1**	**40.1**
Prog-GAN [Karras et al., 2018]		**8.8**				
NVP [Dinh et al., 2017]	3.49					
VAE-IAF [Kingma et al., 2016b]	3.11					
PixelRNN [van den Oord et al., 2016]	3.00					
PixelCNN++ [Salimans et al., 2017]	**2.92**					
CQF [+Residual, +flow, +large D] (Ours)	3.74	8.1	**18.6**	4.00	8.6	52.7
CQF [+Residual, +flow, +2 scales] (Ours)	3.48	6.9	28.9	**3.82**	8.6	52.1

Unconditional CQF samples

	Generated	Training
CIFAR-10		
Celeb-A		
LSUN-bedrooms		
STL-10		
ImageNet-64		

Comparison to NVP samples

	Our samples	NVP samples
CIFAR-10		
Celeb-A		
LSUN-bedrooms		
ImageNet-64		

Take-away

- MLE training cares how much mass goes on the train data, not where the rest goes

Take-away

- MLE training cares how much mass goes on the train data, not where the rest goes
- Adversarial loss is sensitive to mass off the train data, a "trainable" inductive bias beyond architecture design

Take-away

- MLE training cares how much mass goes on the train data, not where the rest goes
- Adversarial loss is sensitive to mass off the train data, a "trainable" inductive bias beyond architecture design
- Using NVP allows for non-factorial decoders in VAEs, improving likelihoods and sample quality

Take-away

- MLE training cares how much mass goes on the train data, not where the rest goes
- Adversarial loss is sensitive to mass off the train data, a “trainable” inductive bias beyond architecture design
- Using NVP allows for non-factorial decoders in VAEs, improving likelihoods and sample quality
- First systematic joint BPD and (IS, FID) evaluation on 7 datasets, results competitive with the state of the art

Take-away

- MLE training cares how much mass goes on the train data, not where the rest goes
- Adversarial loss is sensitive to mass off the train data, a “trainable” inductive bias beyond architecture design
- Using NVP allows for non-factorial decoders in VAEs, improving likelihoods and sample quality
- First systematic joint BPD and (IS, FID) evaluation on 7 datasets, results competitive with the state of the art
- Main message: We can have generative models with full support, *and* high quality samples

References I

[Bishop, 2006] Bishop, C. (2006).
Pattern recognition and machine learning.
Spinger-Verlag.

[Chen et al., 2017] Chen, X., Kingma, D., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. (2017).
Variational lossy autoencoder.
In *ICLR*.

[Dinh et al., 2017] Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017).
Density estimation using real NVP.
In *ICLR*.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014).
Generative adversarial nets.
In *NIPS*.

[Gulrajani et al., 2017a] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017a).
Improved training of Wasserstein GANs.
In *NIPS*.

[Gulrajani et al., 2017b] Gulrajani, I., Kumar, K., Ahmed, F., Taiga, A. A., Visin, F., Vazquez, D., and Courville, A. (2017b).
PixelVAE: A latent variable model for natural images.
In *ICLR*.

[Heusel et al., 2017] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017).
GANs trained by a two time-scale update rule converge to a local Nash equilibrium.
In *NIPS*.

[Karras et al., 2018] Karras, T., Aila, T., and abd J. Lehtinen, S. L. (2018).
Progressive growing of GANSs for improved quality, stability, and variation.
In *ICLR*.

References II

[Kingma et al., 2016a] Kingma, D., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016a).
Improved variational inference with inverse autoregressive flow.
In *NIPS*.

[Kingma and Welling, 2014] Kingma, D. and Welling, M. (2014).
Auto-encoding variational Bayes.
In *ICLR*.

[Kingma et al., 2016b] Kingma, D. P., Salimans, T., Józefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016b).
Improving variational autoencoders with inverse autoregressive flow.
In *NIPS*.

[Lucas et al., 2018] Lucas, T., Tallec, C., Ollivier, Y., and Verbeek, J. (2018).
Mixed batches and symmetric discriminators for GAN training.
In *ICML*.

[Lucas and Verbeek, 2018] Lucas, T. and Verbeek, J. (2018).
Auxiliary guided autoregressive variational autoencoders.
In *ECML*.

[Miyato et al., 2018] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018).
Spectral normalization for generative adversarial networks.
In *ICLR*.

[Oord et al., 2016] Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. (2016).
Pixel recurrent neural networks.
In *ICML*.

[Radford et al., 2016] Radford, A., Metz, L., and Chintala, S. (2016).
Unsupervised representation learning with deep convolutional generative adversarial networks.
In *ICLR*.

References III

[Rezende et al., 2014] Rezende, D., Mohamed, S., and Wierstra, D. (2014).
Stochastic backpropagation and approximate inference in deep generative models.
In *ICML*.

[Royer et al., 2017] Royer, A., Kolesnikov, A., and Lampert, C. (2017).
Probabilistic image colorization.
In *BMVC*.

[Salimans et al., 2016] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016).
Improved techniques for training GANs.
In *NIPS*.

[Salimans et al., 2017] Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. (2017).
PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications.
In *ICLR*.

[Sønderby et al., 2017] Sønderby, C., Caballero, J., Theis, L., Shi, W., and Huszár, F. (2017).
Amortised MAP inference for image super-resolution.
In *ICLR*.

[van den Oord et al., 2016] van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016).
Pixel recurrent neural networks.
In *ICML*.