Mathematical
Institute

Autoencoders, GANs, and Diffusion

Theories of Deep Learning: C6.5,
Lecture / Video 13

Prof. Jared Tanner
Mathematical Institute
University of Oxford

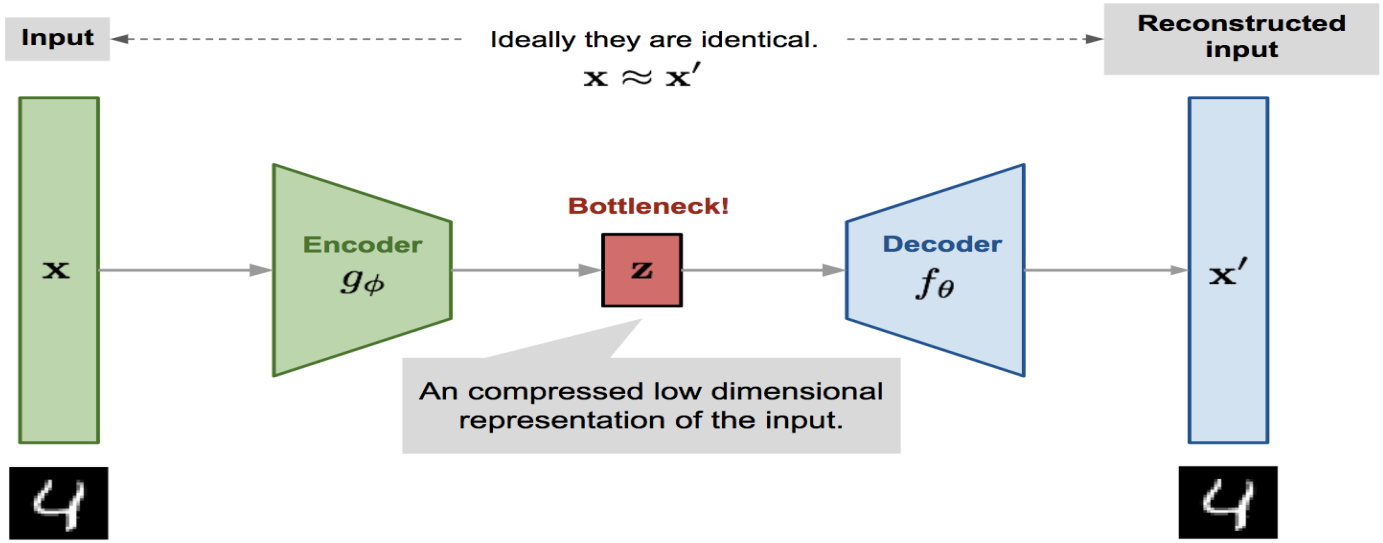
Oxford
Mathematics

(Variational) Auto Encoders; VAEs

Autoencoder (AE) Illustration

Restricting the number of data parameters

University of Oxford
Mathematical Institute



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

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

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

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

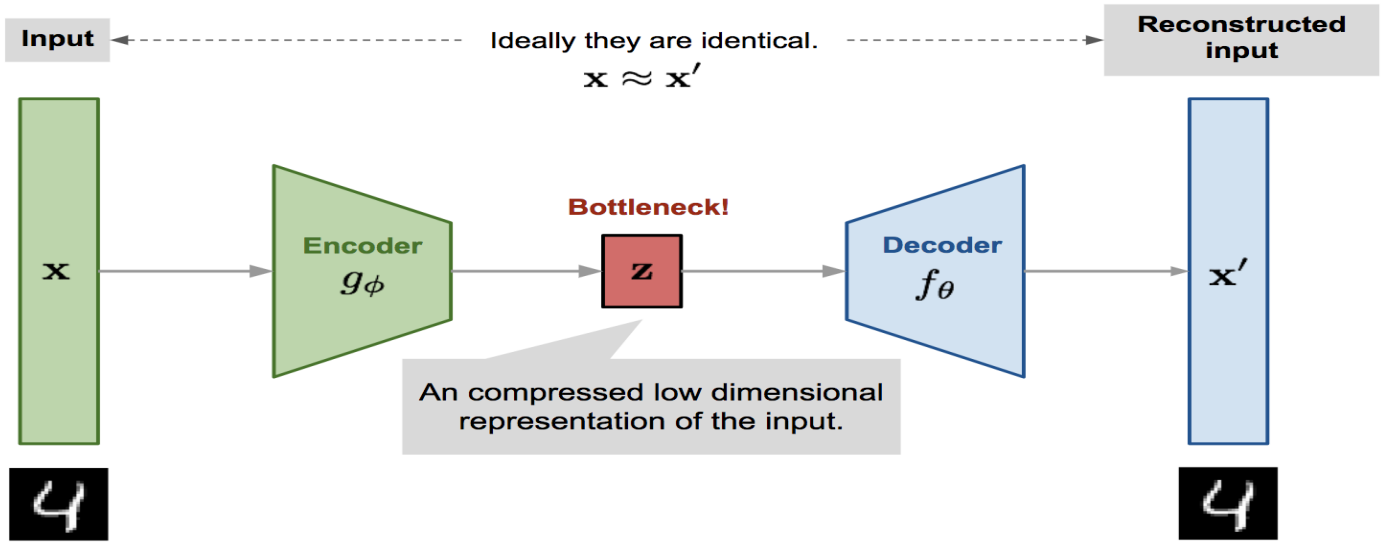
$$\mathcal{L}(\theta, \phi) = n^{-1} \|X - \Theta\Phi X\|_F^2$$

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

Autoencoder extend PCA

More complex maps to low parameter space

Mathematical Institute



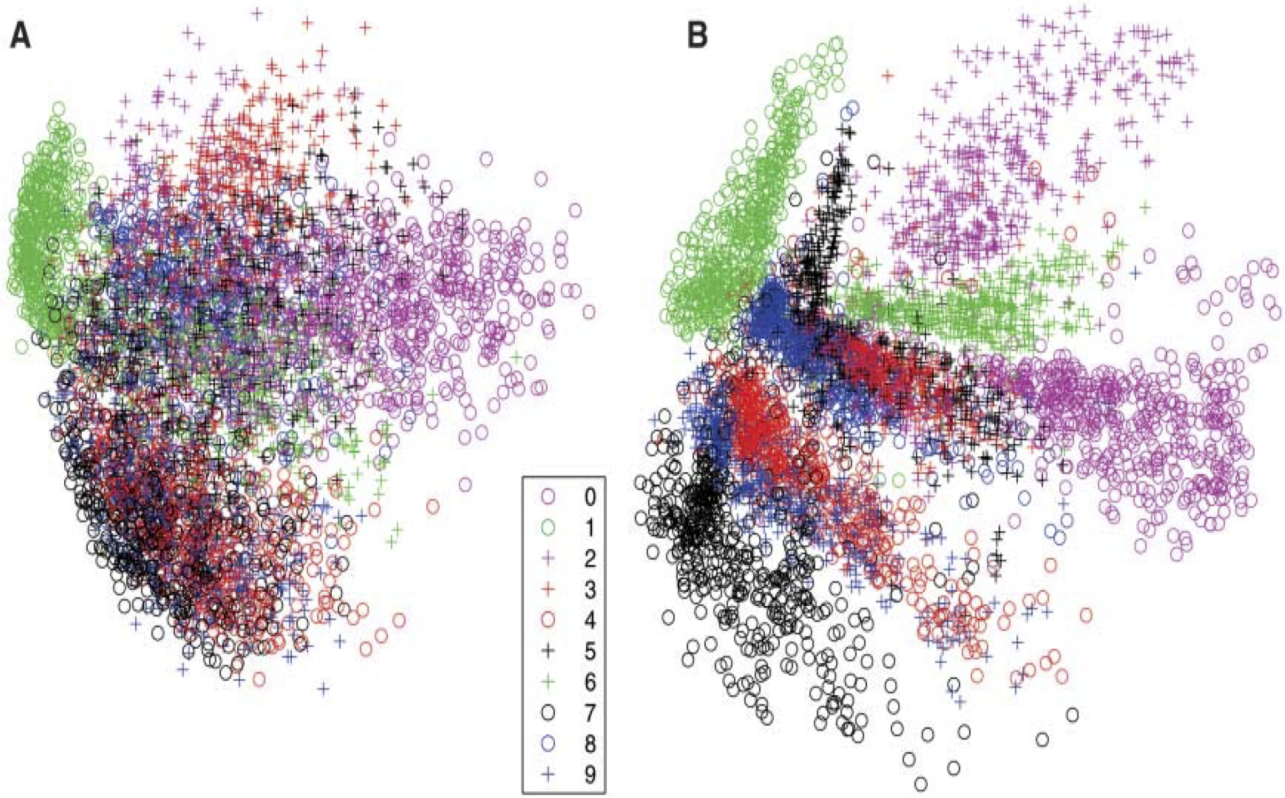
The autoencoder framework allows $g_\phi(\cdot)$ and $f_\theta(\cdot)$ to be more general than linear, and in particular to benefit from the expressively of depth and introduce variation.
`https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html`

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

Improved separation of data classes

Mathematical Institute

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*).



http://science.sciencemag.org/content/313/5786/504

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

Low dimensionality through sparsity

UNIVERSITY OF OXFORD
Mathematical Institute

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.

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

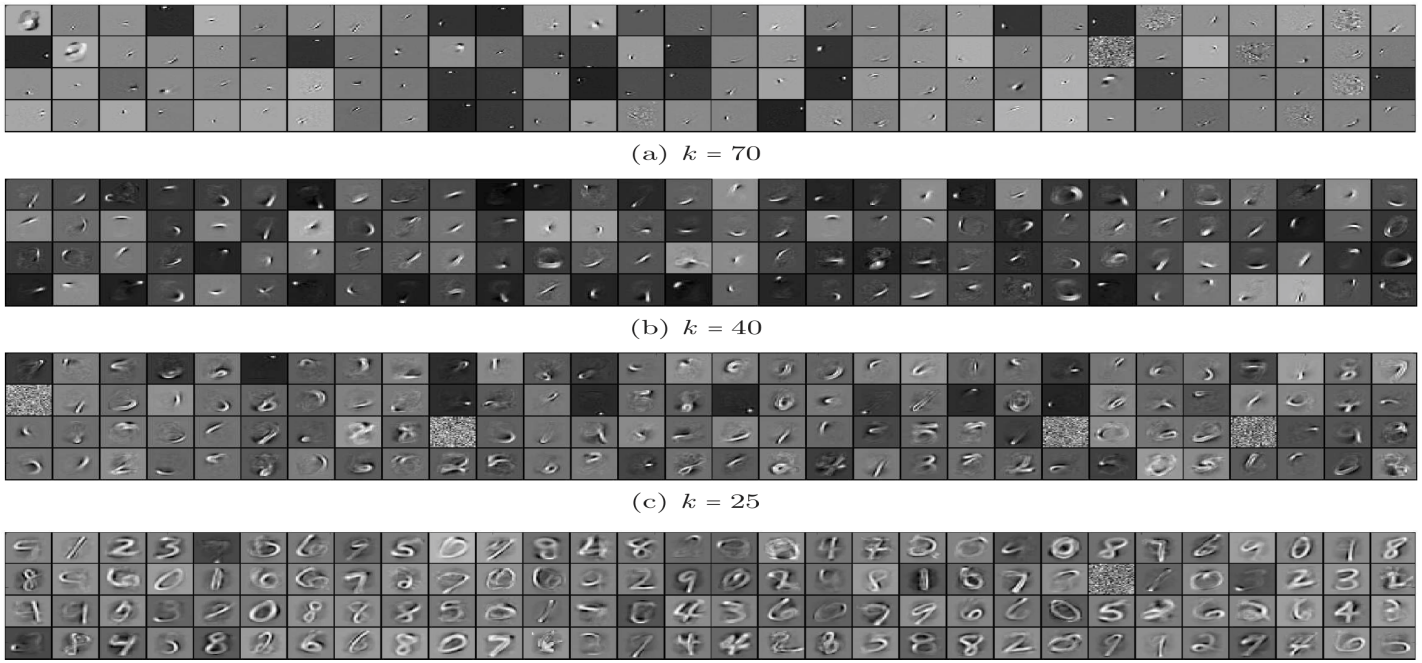
This framework includes nonlinearity and can be rigorously analysed using techniques from sparse approximation, but it lacks depth.

`https://arxiv.org/pdf/1312.5663.pdf`

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

Learned elements: MNIST

UNIVERSITY OF OXFORD
Mathematical Institute



(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.

Elements learned depend on number of components, sparsity, allowed; k small are class elements, k large are basis elements.

`https://arxiv.org/pdf/1312.5663.pdf`

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

Performance vs other autoencoders

	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**.

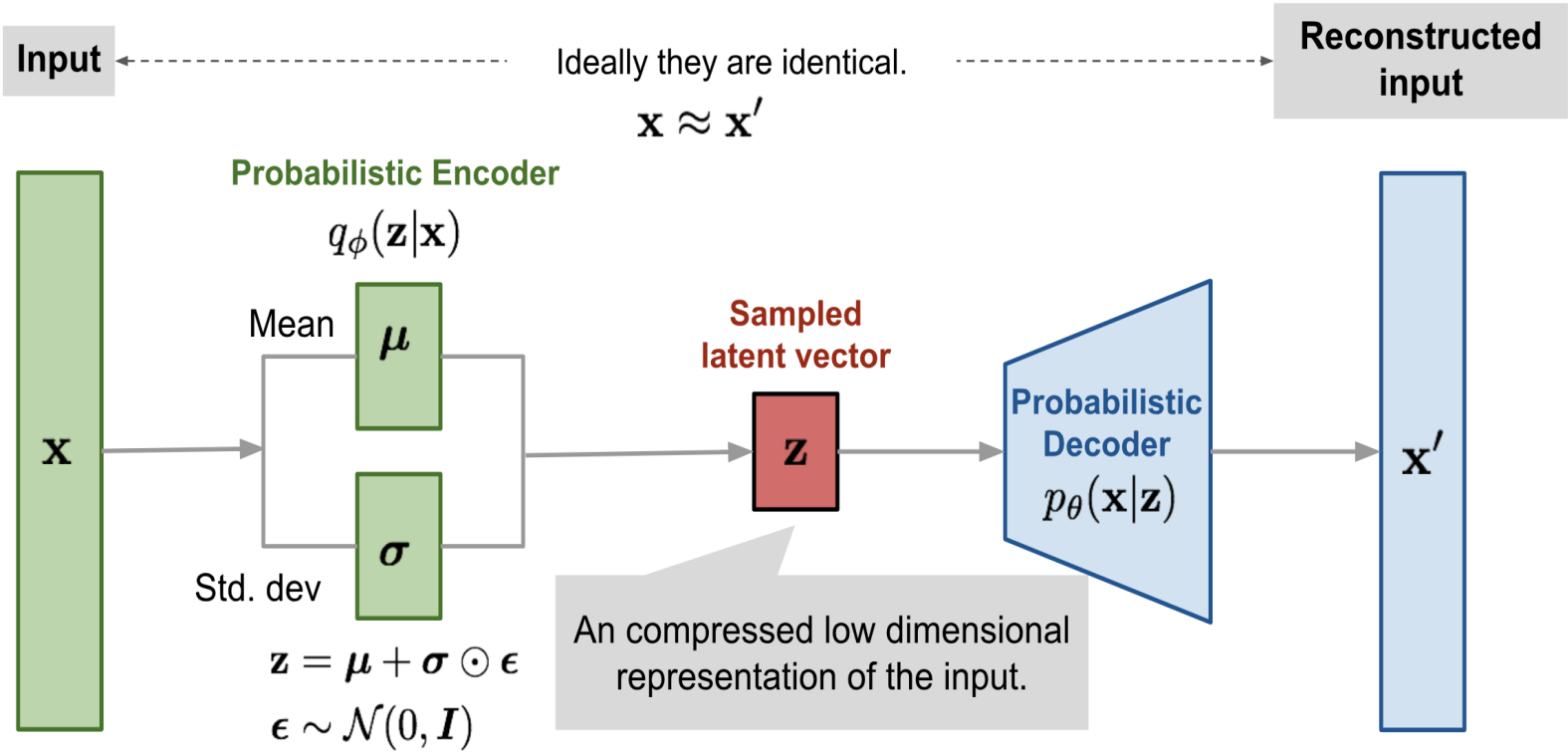
	Error
Without Pre-Training	1.60%
RBM + F.T.	1.24%
Shallow Dropout AE + F.T. (%50 hidden)	1.05%
Denoising AE + F.T. (%20 input dropout)	1.20%
Deep Dropout AE + F.T. (Layer-wise pre-training, %50 hidden)	0.85%
k-Sparse AE + F.T. (k=25)	1.08%
Deep k-Sparse AE + F.T. (Layer-wise pre-training)	0.97%

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.

`https://arxiv.org/pdf/1312.5663.pdf`

Variational Autoencoders (VAE) (Kingma et al. 13')

Introduction of noise as a generative model



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

https://arxiv.org/pdf/1312.6114.pdf

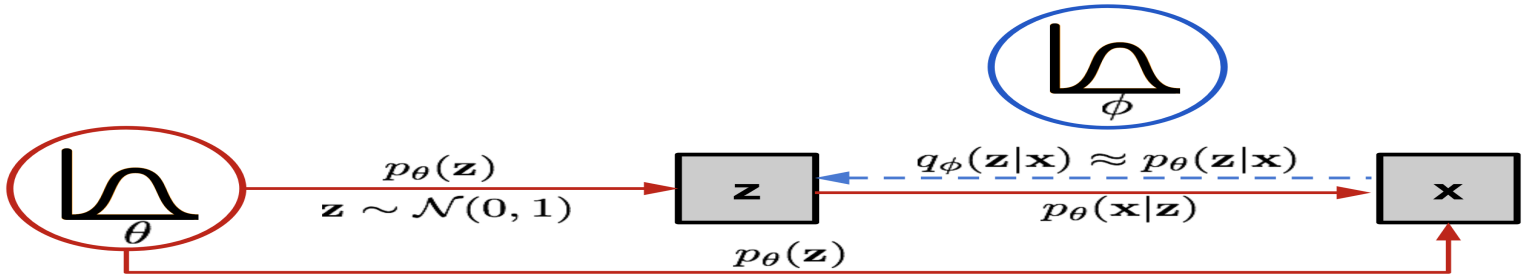
Examples of encoder or decoder structure is a multivariate Gaussian with a diagonal covariance structure; e.g. for the decoder:

$$\begin{aligned} h &= \tanh(W_3 z + b_3) \\ \text{with } \mu &= W_4 h + b_4 \\ \log(\sigma^2) &= W_5 h + b_5 \\ \log p(x|z) &= \log \mathcal{N}(x; \mu, \sigma^2 I) \end{aligned}$$

`https://arxiv.org/pdf/1312.6114.pdf`

Variational Autoencoders (VAE) (Kingma et al. 13')

Distribution for generative model



$p_\theta(x|z)$ 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.

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

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

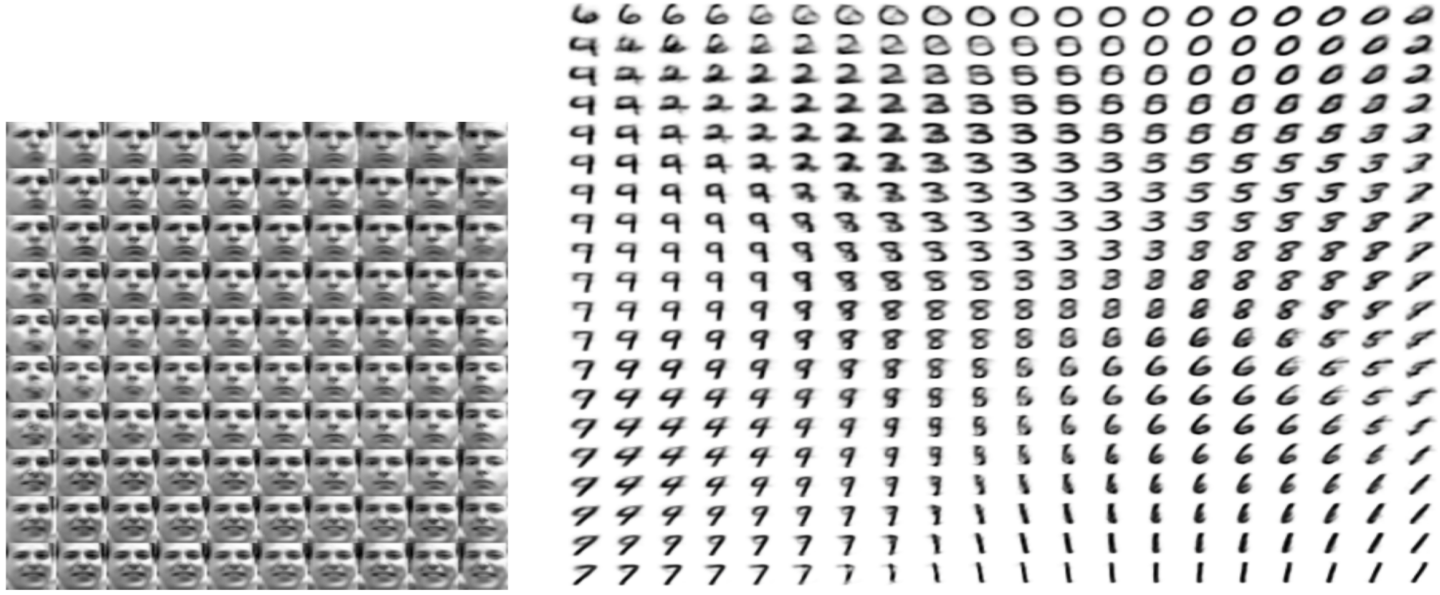
$$\begin{aligned}
\log p_\theta(x) &= \log\left(\int p_\theta(x|z)p_\theta(z)dz\right) \\
&= \log\left(\int p_\theta(x|z)\frac{p_\theta(z)}{q_\phi(z)}q_\phi(z)dz\right) \\
&\geq \int \log\left(p_\theta(x|z)\frac{p_\theta(z)}{q_\phi(z)}\right) q_\phi(z)dz \\
&= \mathbb{E}_{q_\phi(z|x)} \log(p_\theta(x|z))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

https://arxiv.org/pdf/1312.6114.pdf

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

Learned two dimensional space: faces and MNIST

UNIVERSITY OF OXFORD
Mathematical Institute



(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 $\boldsymbol{\theta}$.

https://arxiv.org/pdf/1312.6114.pdf

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

Dependence on manifold dimension: MNIST

(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.

`https://arxiv.org/pdf/1312.6114.pdf`

Inference Variational Autoencoders (Zhao et al. 17')

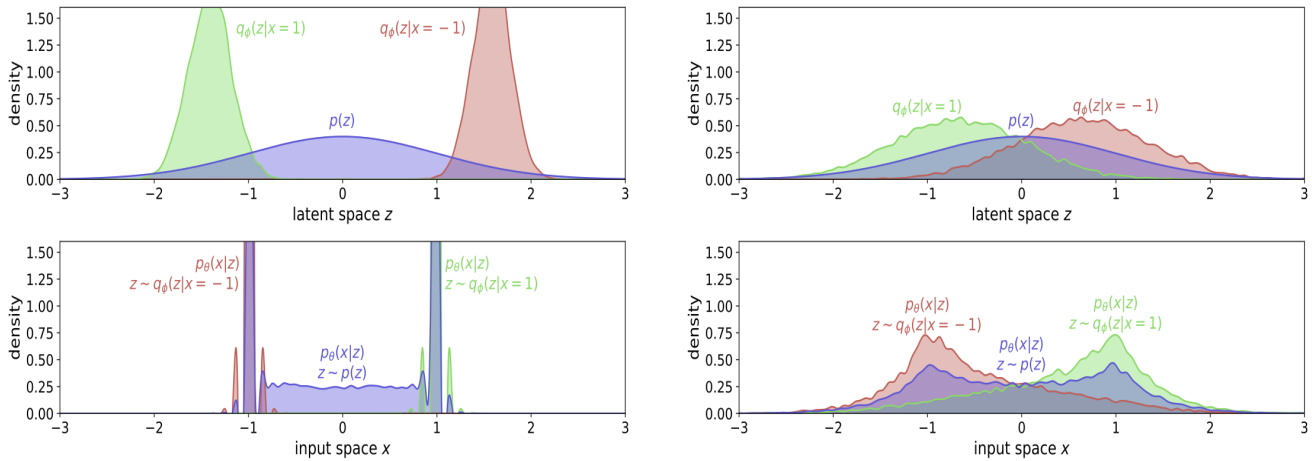
$p_\theta(x|z)$ acts as the generators, analogous to the decoder $f_\theta(x|z)$, and is called a probabilistic decoder; $q_\phi(z|x)$ acts as the encoder, analogous to $g_\phi(z|x)$, and is used to approximate $p_\theta(z|x)$.
The parameters ϕ, θ for a model are then learned to minimize a distance, 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.
`https://arxiv.org/pdf/1706.02262.pdf`

Inference Variational Autoencoders (Zhao et al. 17')

Impact of VAE objective ($\lambda = \beta$, in $\beta-$VAE



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.

https://arxiv.org/pdf/1706.02262.pdf

Inference Variational Autoencoders (Zhao et al. 17')

Impact of VAE objective

UNIVERSITY OF OXFORD
Mathematical Institute

reconstruction generation

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.

https://arxiv.org/pdf/1706.02262.pdf

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

Explainable latent space

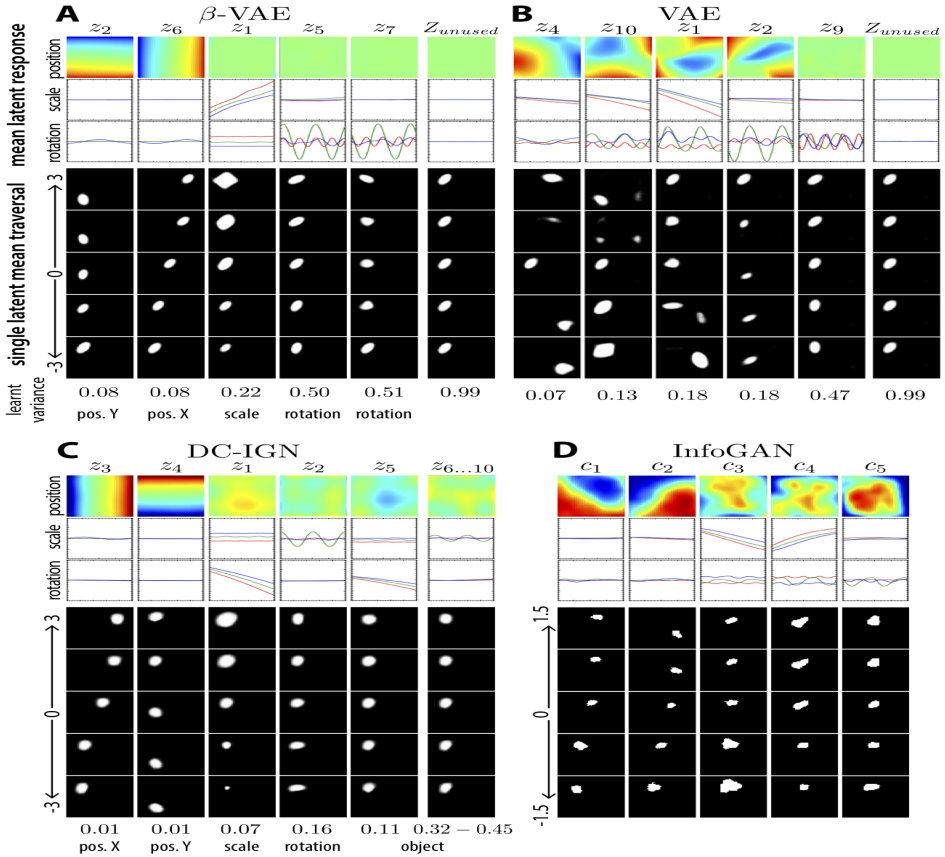


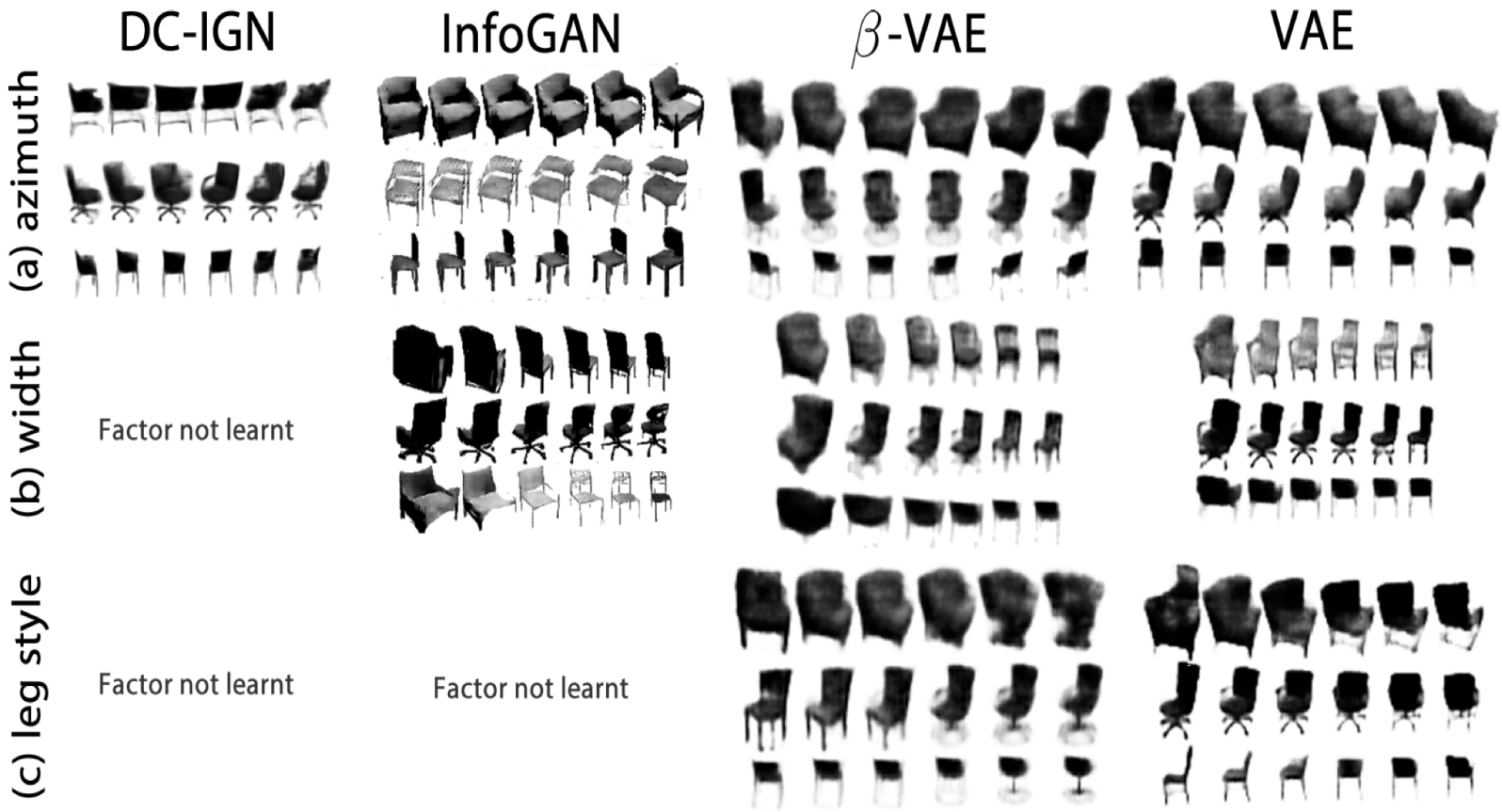
Figure 7: **A:** Representations learnt by a β-VAE ($\beta = 4$). Each column represents a latent z_i, ordered according to the learnt Gaussian variance (last row). Row 1 (position) shows the mean activation (red represents high values) of each latent z_i as a function of all 32x32 locations averaged 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).

https://arxiv.org/pdf/1706.02262.pdf

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

Explainable latent space: chairs

University of Oxford
Mathematical Institute



https://arxiv.org/pdf/1706.02262.pdf

β-VAEs disentangling features pt. 3 (Higgins et al. 17')

Architectures for encoder-decoders

UNIVERSITY OF OXFORD
Mathematical Institute

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.

https://arxiv.org/pdf/1706.02262.pdf

Autoencoders summary

NNs used as nonlinear maps to lower dimensional latent spaces

UNIVERSITY OF OXFORD
Mathematical Institute

- Principal component analysis (PCA) reveals the low dimensional latent space within a data matrix by projecting to the space of low-rank matrices.
- Autoencoders (AE) extend this notion allowing more general maps to and from a low dimensional parameter space.
- Variational AEs (VAEs) give a probabilistic notion that gives a natural generative model.
- Inference VAEs and $\beta-$VAEs are further extensions to improve VAEs and for interpretability respectively.
- (V)AEs are a flexible structure allowing general maps; an area open for great further analysis.

Generative Adversarial Networks (GANs)

Example of a deep convolutional generator:

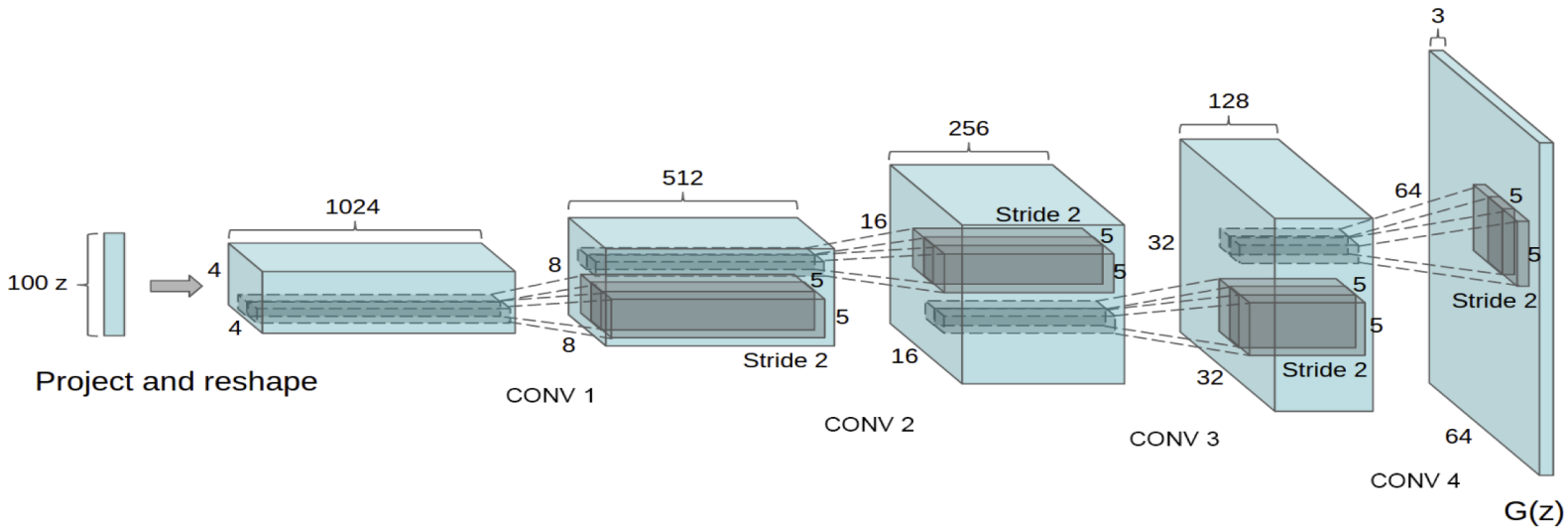


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps.

https://arxiv.org/pdf/1511.06434.pdf

https://arxiv.org/pdf/1406.2661.pdf

Generative deep nets (Goodfellow et al. 14')

Generative model from 100 latent variables

UNIVERSITY OF OXFORD
Mathematical Institute

Train the two network parameters using the objective

$$\min_G \max_D n^{-1} \sum_{\mu=1}^{n} \log(D(x_\mu, y_\mu)) + p^{-1} \sum_p \log\left(1 - D(G(z_p), y_p)\right)$$

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

https://arxiv.org/pdf/1406.2661.pdf

Generative deep nets (Radford et al. 16')

Early training examples

UNIVERSITY OF OXFORD
Mathematical Institute

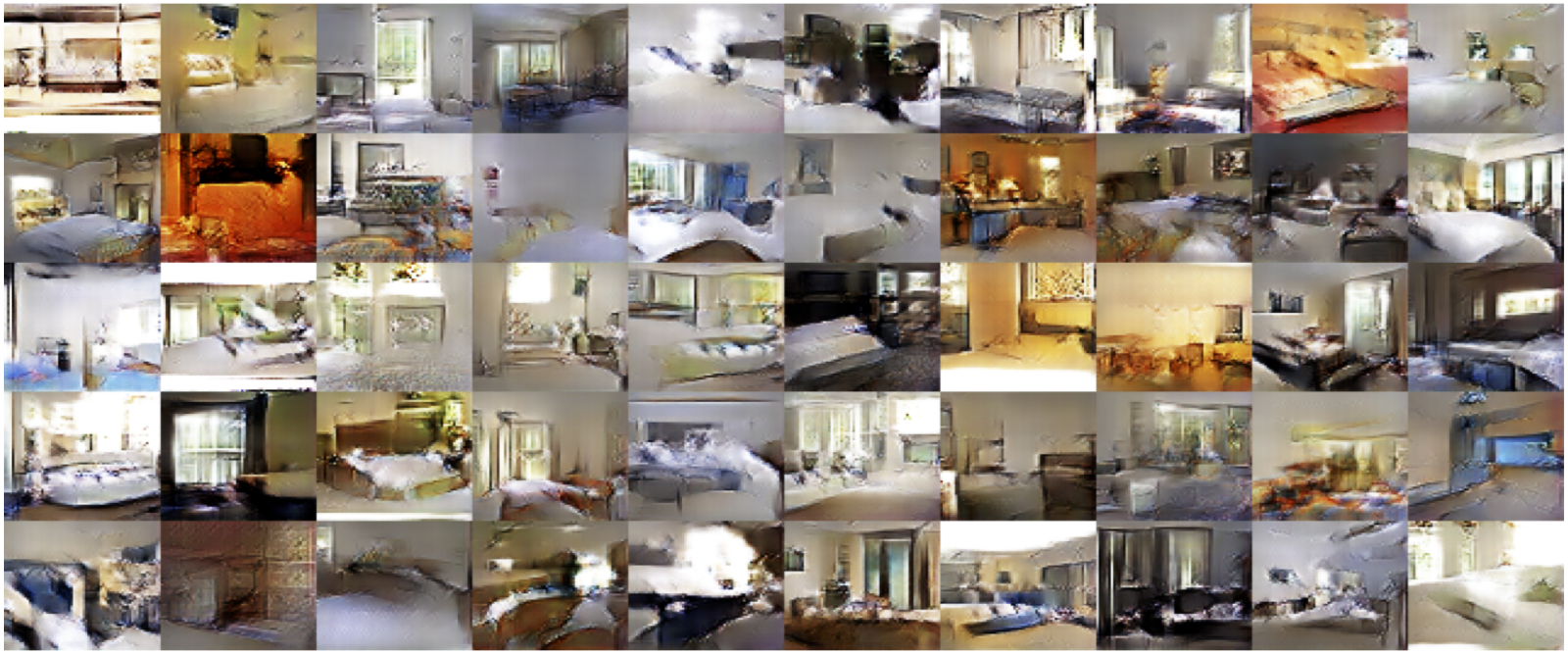


Figure 2: Generated bedrooms after one training pass through the dataset. Theoretically, the model could learn to memorize training examples, but this is experimentally unlikely as we train with a small learning rate and minibatch SGD. We are aware of no prior empirical evidence demonstrating memorization with SGD and a small learning rate.

`https://arxiv.org/pdf/1511.06434.pdf`

Generative deep nets (Radford et al. 16')

Later training examples

Mathematical Institute

Figure 3: Generated bedrooms after five epochs of training. There appears to be evidence of visual under-fitting via repeated noise textures across multiple samples such as the base boards of some of the beds.

`https://arxiv.org/pdf/1511.06434.pdf`

Wasserstein GAN (Arjovsky et al. 17')

Optimal transport decoder

University of Oxford
Mathematical Institute

One of the central challenges with GANs is the ability to train the parameters. Improvements have been made through choice of generative architecture (DC-GAN of Radford) and through different training objective functions (W-GAN)

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

while θ has not converged **do**
  for $t = 1, \ldots, n_{\text{critic}}$ **do**
    for $i = 1, \ldots, m$ **do**
      Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
      $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
      $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
      $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
    end for
    $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2)$
  end for
  Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^m \sim p(\boldsymbol{z})$.
  $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^m -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while

https://arxiv.org/pdf/1704.00028.pdf
https://arxiv.org/pdf/1701.07875.pdf

Wasserstein GAN (Arjovsky et al. 17')

Examples of output from GAN architectures

University of Oxford
Mathematical Institute

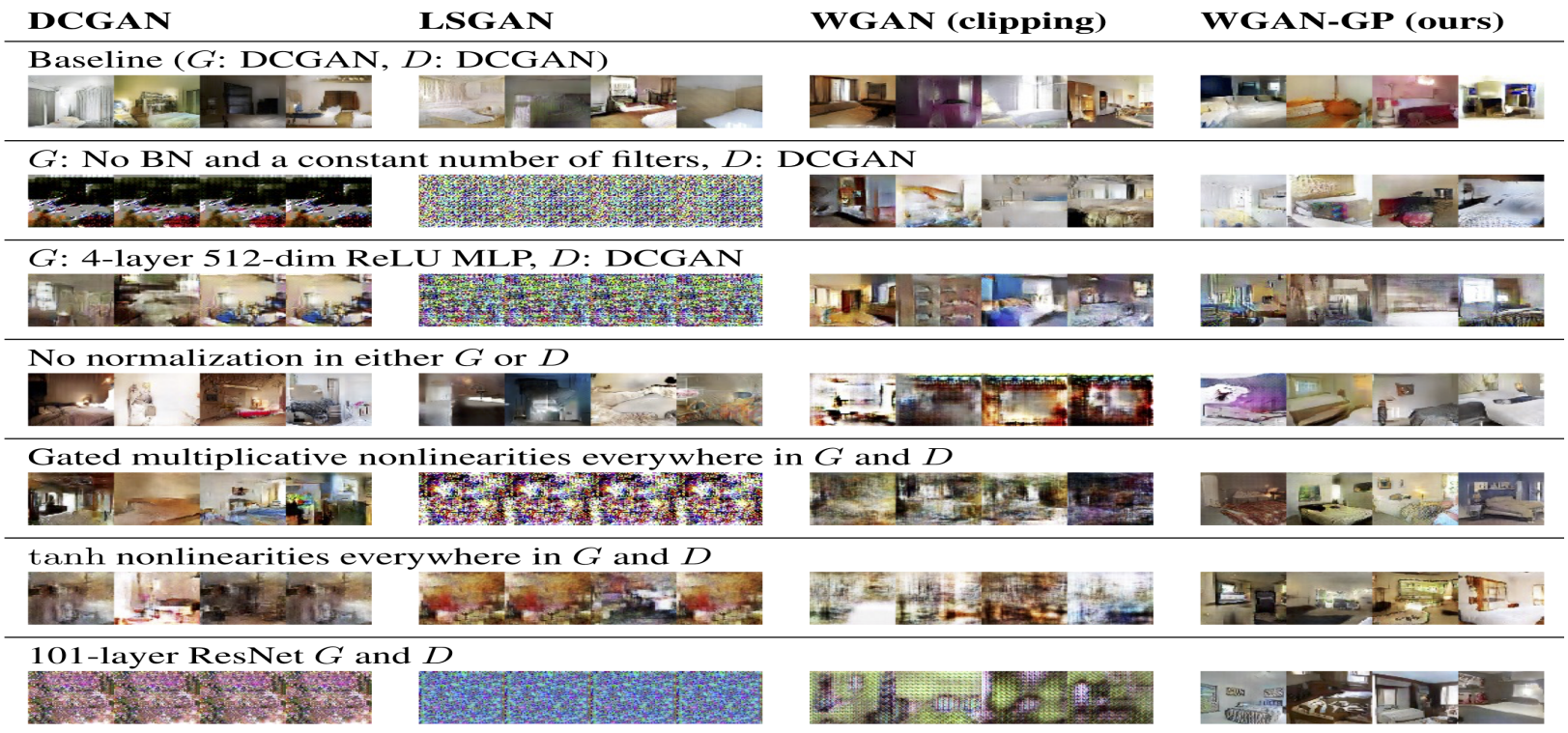


Figure 2: Different GAN architectures trained with different methods. We only succeeded in training every architecture with a shared set of hyperparameters using WGAN-GP.

https://arxiv.org/pdf/1704.00028.pdf

Wasserstein GAN (Arjovsky et al. 17')

Training rate

Mathematical Institute

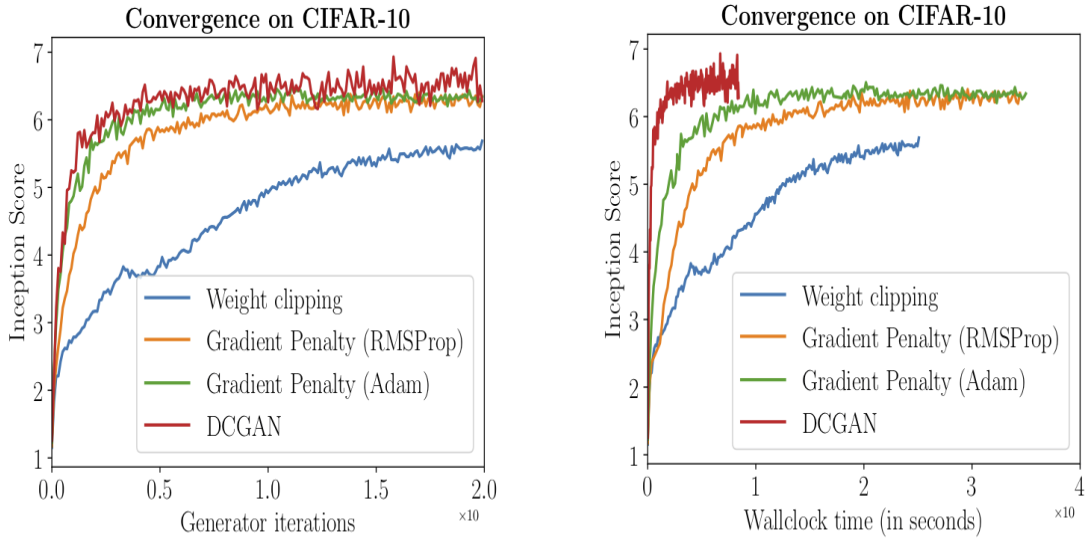


Figure 3: CIFAR-10 Inception score over generator iterations (left) or wall-clock time (right) for four models: WGAN with weight clipping, WGAN-GP with RMSProp and Adam (to control for the optimizer), and DCGAN. WGAN-GP significantly outperforms weight clipping and performs comparably to DCGAN.

`https://arxiv.org/pdf/1704.00028.pdf`

Large scale WGAN (Karras et al. 18')

Growing the encoder/decoder complexity

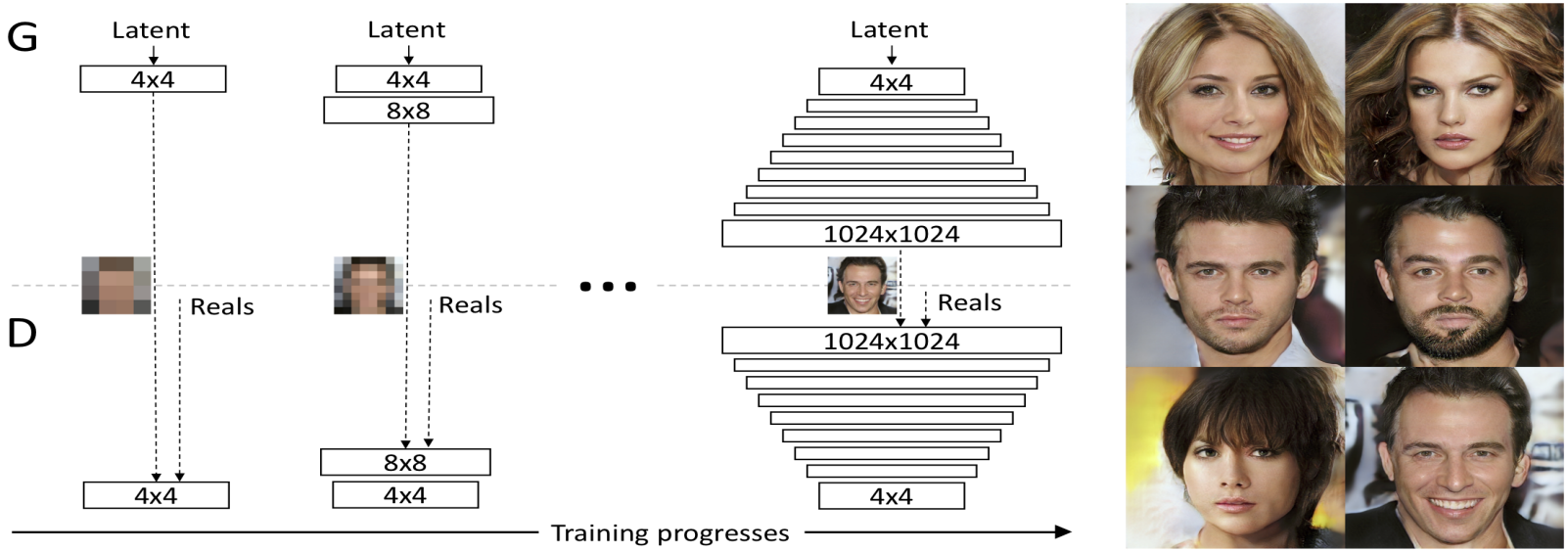


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Here $N \times N$ refers to convolutional layers operating on $N \times N$ spatial resolution. This allows stable synthesis in high resolutions and also speeds up training considerably. One the right we show six example images generated using progressive growing at 1024×1024.

https://arxiv.org/abs/1710.10196

Large scale WGAN (Karras et al. 18')

Examples of synthetic faces

UNIVERSITY OF OXFORD
Mathematical Institute

Figure 10: Top: Our CELEBA-HQ results. Next five rows: Nearest neighbors found from the training data, based on feature-space distance. We used activations from five VGG layers, as suggested by Chen & Koltun (2017). Only the crop highlighted in bottom right image was used for comparison in order to exclude image background and focus the search on matching facial features.

https://arxiv.org/abs/1710.10196

Understanding individual units in a DNN (Bau et al. 20')

Single units which reliably detect object classes

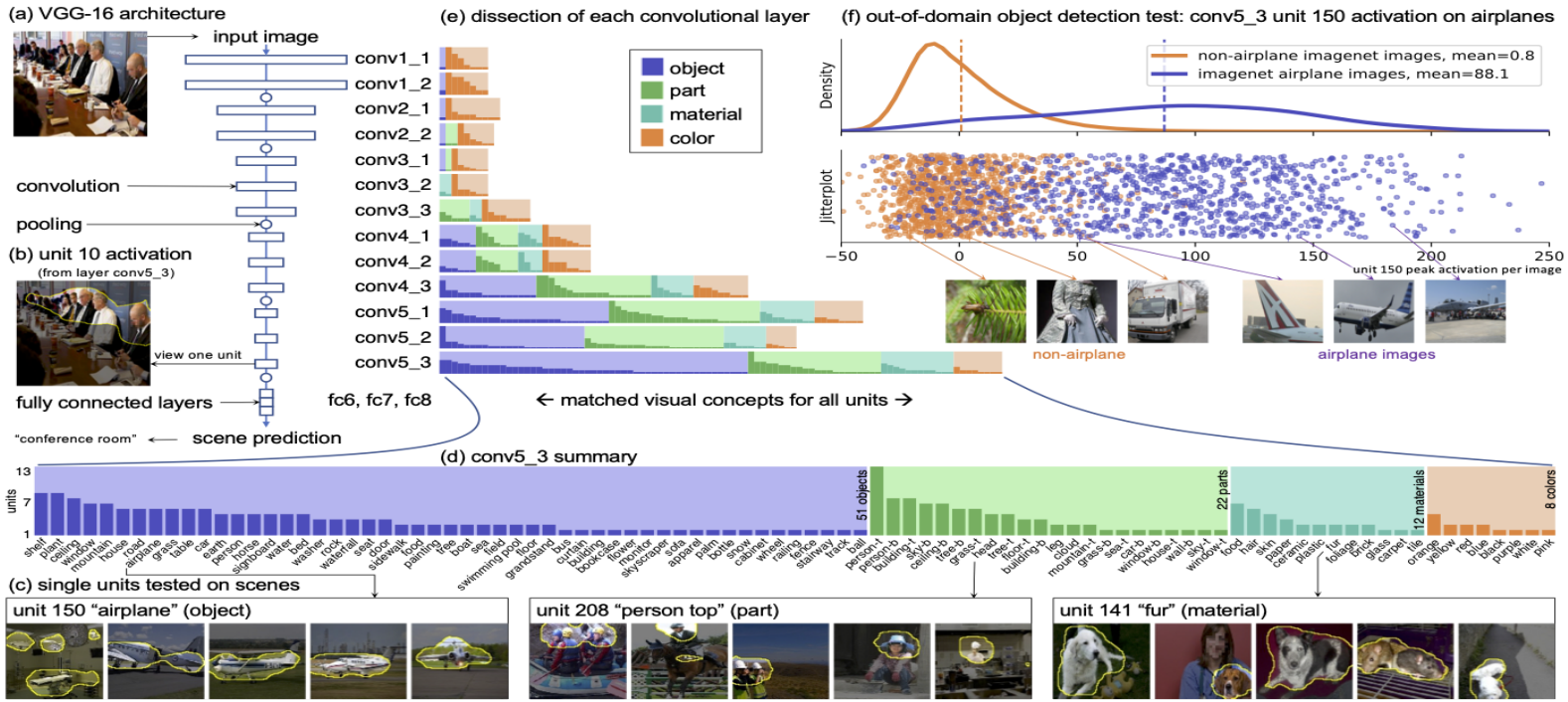


Fig. 1. The emergence of single-unit object detectors within a VGG-16 scene classifier. (a) VGG-16 consists of 13 convolutional layers, `conv1_1` through `conv5_3`, followed by three fully connected layers, `fc6,7,8`. (b) The activation of a single filter on an input image can be visualized as the region where the filter activates beyond its top 1% quantile level. (c) Single units are scored by matching high-activating regions against a set of human-interpretable visual concepts; each unit is labeled with its best-matching concept and visualized with maximally-activating images. (d) Concepts that match units in the final convolutional layer are summarized, showing a broad diversity of detectors for objects, object parts, materials, and colors. Many concepts are associated with multiple units. (e) Comparing all the layers of the network reveals that most object detectors emerge at the last convolutional layers. (f) Although the training set contains no object labels, unit 150 emerges as an 'airplane' object detector that activates much more strongly on airplane objects than non-airplane objects, as tested against a dataset of labeled object images not previously seen by the network. The jitter plot shows peak activations for the unit on randomly sampled 1,000 airplane and 1,000 non-airplane Imagenet images, and the curves show the kernel density estimates of these activations.

https://arxiv.org/abs/2009.05041

Understanding individual units in a DNN (Bau et al. 20')

Single units which reliably generate object classes

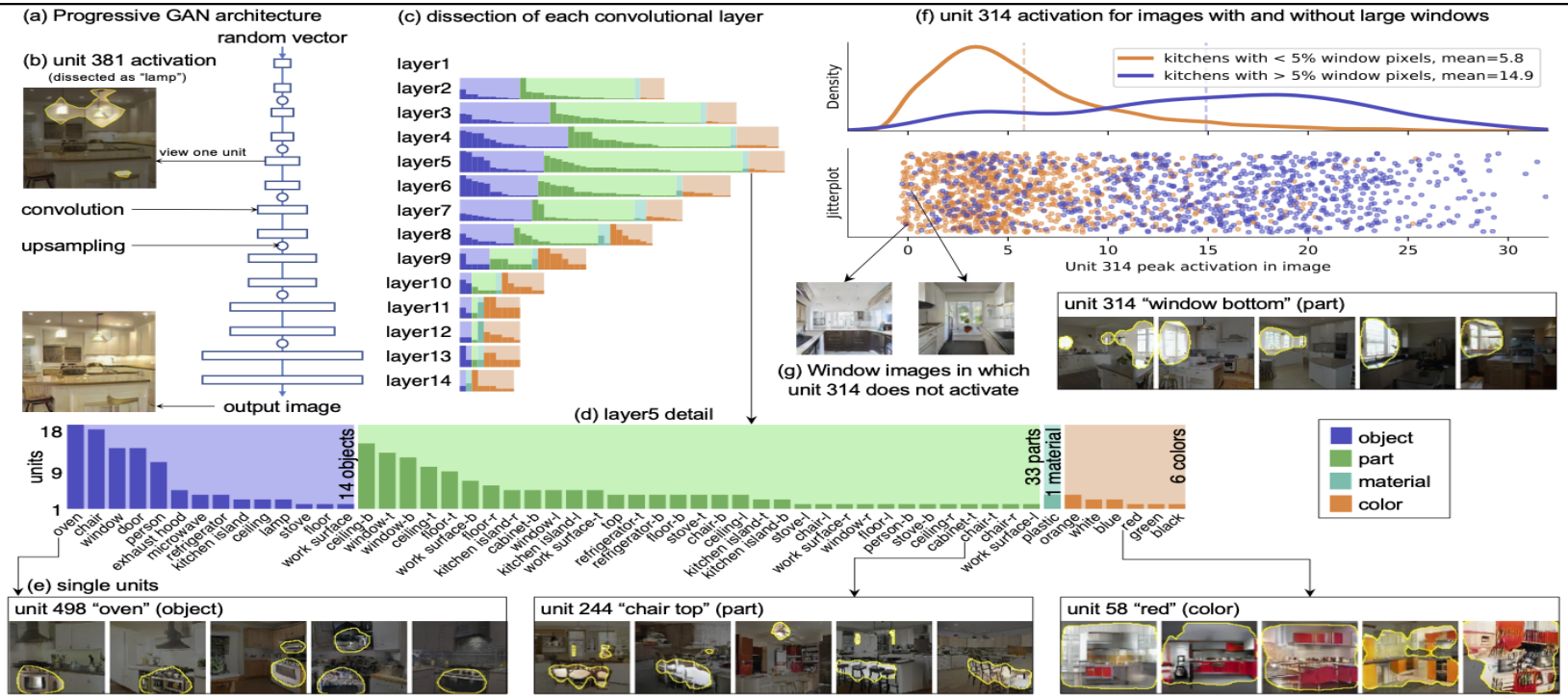


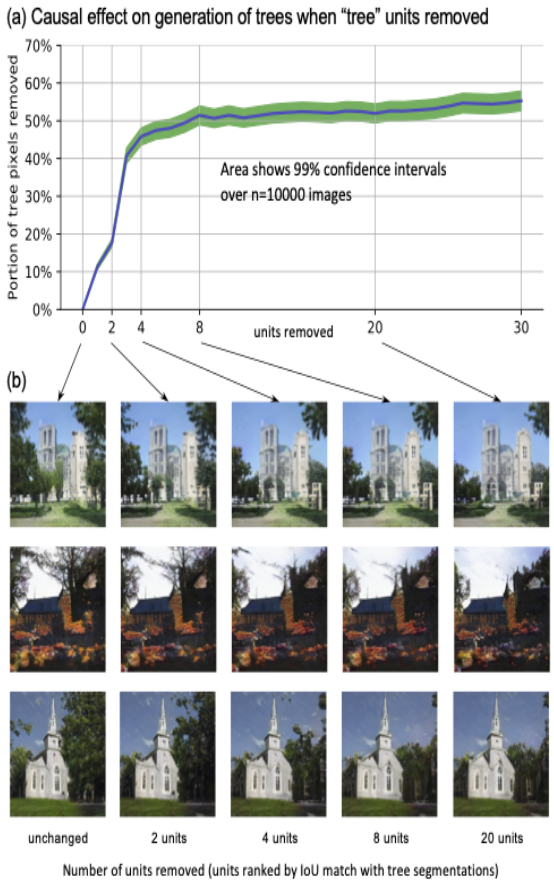
Fig. 3. The emergence of object- and part-specific units within a Progressive GAN generator'(19). (a) The analyzed Progressive GAN consists of 15 convolutional layers that transform a random input vector into a synthesized image of a kitchen. (b) A single filter is visualized as the region of the output image where the filter activates beyond its top 1% quantile level; note that the filters are all precursors to the output. (c) Dissecting all the layers of the network shows a peak in object-specific units at `layer5` of the network. (d) A detailed examination of `layer5` shows more part-specific units than objects, and many visual concepts corresponding to multiple units. (e) Units do not correspond to exact pixel patterns: a wide range of visual appearances for ovens and chairs are generated when an oven or chair part unit are activated. (f) When a unit specific to window parts is tested as a classifier, on average the unit activates more strongly on generated images that contain large windows than images that do not. The jitter plot shows the peak activation of unit 314 on 800 generated images that have windows larger than 5% of the image area as estimated by a segmentation algorithm, and 800 generated images that do not. (g) Some counterexamples: images for which unit 314 does not activate but where windows are synthesized nevertheless.

`https://arxiv.org/abs/2009.05041`

Editing individual units in a DNN (Bau et al. 20')

Removing units of classes decreases their generation

Mathematical Institute



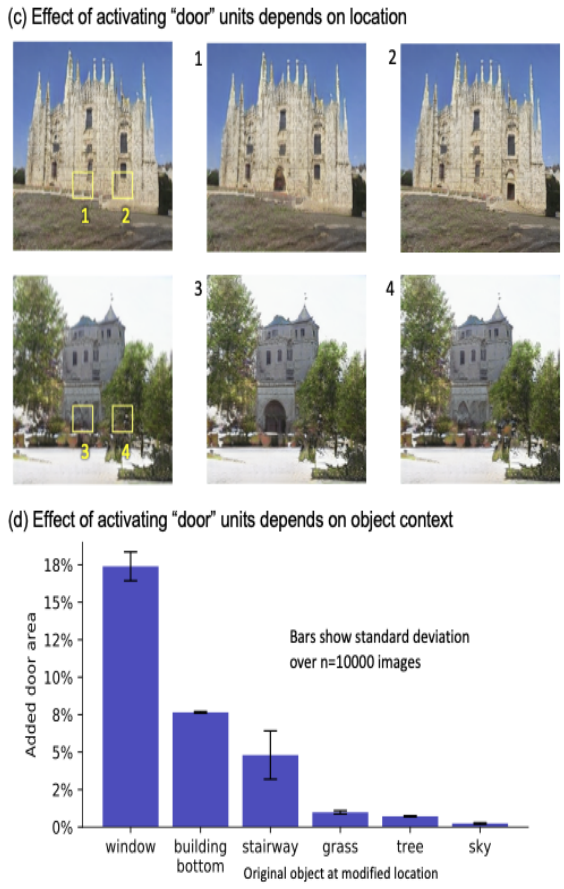


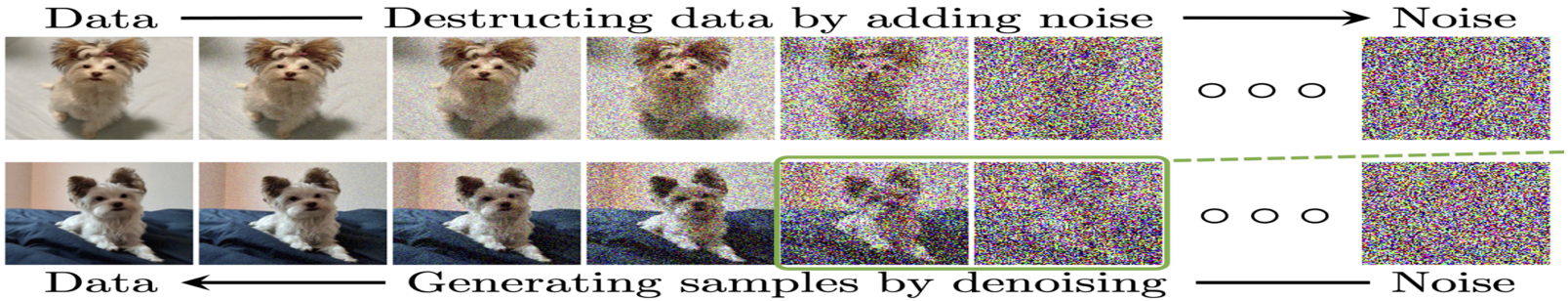
Fig. 4. The causal effect of altering units within a GAN generator. (a) When successively larger sets of units are removed from a GAN trained to generate outdoor church scenes, the tree area of the generated images is reduced. Removing 20 tree units removes more than half the generated tree pixels from the output. (b) Qualitative results: removing tree units affects trees while leaving other objects intact. Building parts that were previously occluded by trees are rendered as if revealing the objects that were behind the trees. (c) Doors can be added to buildings by activating 20 door units. The location, shape, size, and style of the rendered door depends on the location of the activated units. The same activation levels produce different doors, or no door at all (case 4) depending on locations. (d) Similar context dependence can be seen quantitatively: doors can be added in reasonable locations such as at the location of a window, but not in abnormal locations such as on a tree or in the sky.

https://arxiv.org/abs/2009.05041

Diffusion models

Diffusion processes

Deterministic encoder for information preserving GAN sample



Given an input x_0, repeatedly scale and add additive noise

$$x_t = (1-\beta_t)^{1/2} x_{t-1} + \mathcal{N}(0, \beta_t)$$

which is repeated from initial data x_0 to $x_t = \bar{\alpha}_t^{1/2} x_0 + (1-\bar{\alpha}_t)^{1/2}\epsilon$ where $\epsilon \sim \mathcal{N}(0, (1-\bar{\alpha}_t)I)$ with $\alpha_t = 1-\beta_t$ and $\bar{\alpha}_t := \Pi_{s=0}^{t} \alpha_s$.
The deep network is trained to denoise x_t to recover x_0, unlike in a normal GAN x_t is not purely noise. New data generated by drawing from $\mathcal{N}(0, (1-\bar{\alpha}_t)I)$ as if the x_0 is lost in the noise.

`https://arxiv.org/abs/2209.00796`