Advanced Topics in Machine Learning (deep generative modelling)

Lecture 6: Representation learning and evaluation

Nikolay Malkin

24 February 2026

Outline of Lecture 6

Representations and evaluation:

- Review of generative model families
  - Conditional models
- Representation learning
  - Representations from deep networks
  - Probing representations
- Evaluation of generative models
  - Likelihood-based evaluation
  - Sample-based evaluation

- Review of generative model families
  - Conditional models
- Representation learning
  - Representations from deep networks
  - Probing representations
- Evaluation of generative models
  - Likelihood-based evaluation
  - Sample-based evaluation

Summary of models seen so far

We have seen three classes of deep generative models: VAEs, normalising flows, GANs

Summary of models seen so far

We have seen three classes of deep generative models: VAEs, normalising flows, GANs

	VAE	NF	GAN
Likelihood estimation			

Summary of models seen so far

We have seen three classes of deep generative models: VAEs, normalising flows, GANs

	VAE	NF	GAN
Likelihood estimation	approximate	exact	no

Summary of models seen so far

We have seen three classes of deep generative models: VAEs, normalising flows, GANs

	VAE	NF	GAN
Likelihood estimation	approximate	exact	no
Sample quality (fidelity)			

Summary of models seen so far

We have seen three classes of deep generative models: VAEs, normalising flows, GANs

	VAE	NF	GAN
Likelihood estimation	approximate	exact	no
Sample quality (fidelity)	poor	good	best

Summary of models seen so far

We have seen three classes of deep generative models: VAEs, normalising flows, GANs

	VAE	NF	GAN
Likelihood estimation	approximate	exact	no
Sample quality (fidelity)	poor	good	best
Mode coverage (diversity)			

Summary of models seen so far

We have seen three classes of deep generative models: VAEs, normalising flows, GANs

	VAE	NF	GAN
Likelihood estimation	approximate	exact	no
Sample quality (fidelity)	poor	good	best
Mode coverage (diversity)	good	good	poor

Summary of models seen so far

We have seen three classes of deep generative models: VAEs, normalising flows, GANs

	VAE	NF	GAN
Likelihood estimation	approximate	exact	no
Sample quality (fidelity)	poor	good	best
Mode coverage (diversity)	good	good	poor
Trainability at scale			

Summary of models seen so far

We have seen three classes of deep generative models: VAEs, normalising flows, GANs

	VAE	NF	GAN
Likelihood estimation	approximate	exact	no
Sample quality (fidelity)	poor	good	best
Mode coverage (diversity)	good	good	poor
Trainability at scale	best	good	poor

Summary of models seen so far

We have seen three classes of deep generative models: VAEs, normalising flows, GANs

	VAE	NF	GAN
Likelihood estimation	approximate	exact	no
Sample quality (fidelity)	poor	good	best
Mode coverage (diversity)	good	good	poor
Trainability at scale	best	good	poor
Ability to impose inductive biases			

Summary of models seen so far

We have seen three classes of deep generative models: VAEs, normalising flows, GANs

	VAE	NF	GAN
Likelihood estimation	approximate	exact	no
Sample quality (fidelity)	poor	good	best
Mode coverage (diversity)	good	good	poor
Trainability at scale	best	good	poor
Ability to impose inductive biases	good	poor	good

Conditional generative models

Unconditional modelling:

- We have a **data distribution** π_{data} over $\mathbb{R}^d$ (from which we can sample, but we do not know its density function)
- We have a class of **model distributions** $\{\pi_\theta\}$
- We seek θ such that π_θ approximates π_{data} well:

$$\theta^* = \arg\min_\theta D(\pi_\theta, \pi_{\text{data}})$$

Conditional generative models

Unconditional modelling:

- We have a **data distribution** π_{data} over $\mathbb{R}^d$ (from which we can sample, but we do not know its density function)
- We have a class of **model distributions** $\{\pi_\theta\}$
- We seek θ such that π_θ approximates π_{data} well

Conditional modelling:

- We have a joint distribution $\pi_{\text{data}}(x, y)$, where y is a conditioning variable
- We have a family of conditional model distributions $\pi_\theta(\cdot \mid y)$

Conditional generative models

Unconditional modelling:

- We have a **data distribution** π_{data} over $\mathbb{R}^d$ (from which we can sample, but we do not know its density function)
- We have a class of **model distributions** $\{\pi_\theta\}$
- We seek θ such that π_θ approximates π_{data} well

Conditional modelling:

- We have a joint distribution $\pi_{\text{data}}(x, y)$, where y is a conditioning variable
- We have a family of conditional model distributions $\pi_\theta(\cdot \mid y)$
- We seek θ such that $\pi_\theta(\cdot \mid y)$ approximates $\pi_{\text{data}}(\cdot \mid y)$ well for all y:

$$\theta^* = \arg\min_\theta \mathbb{E}_{y \sim \pi_{\text{data}}(y)} D(\pi_\theta(\cdot \mid y), \pi_{\text{data}}(\cdot \mid y))$$

Conditional generative models

Unconditional modelling:

- We have a **data distribution** π_{data} over $\mathbb{R}^d$ (from which we can sample, but we do not know its density function)
- We have a class of **model distributions** $\{\pi_\theta\}$
- We seek θ such that π_θ approximates π_{data} well

Conditional modelling:

- We have a joint distribution $\pi_{\text{data}}(x, y)$, where y is a conditioning variable
- We have a family of conditional model distributions $\pi_\theta(\cdot \mid y)$
- We seek θ such that $\pi_\theta(\cdot \mid y)$ approximates $\pi_{\text{data}}(\cdot \mid y)$ well for all y:

$$\theta^* = \arg\min_\theta \mathbb{E}_{y \sim \pi_{\text{data}}(y)} D(\pi_\theta(\cdot \mid y), \pi_{\text{data}}(\cdot \mid y))$$

Discriminative modelling is a special case of conditional modelling where the conditioning variable is the input and the output is a label (or class).

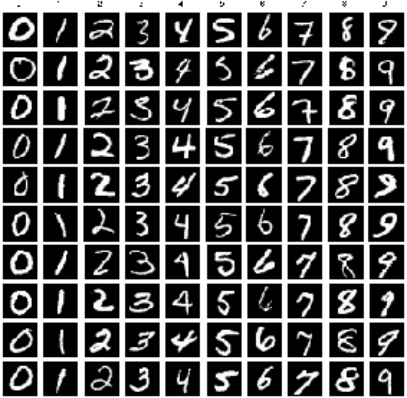
Conditioning generative models

How to make a generative model conditional on y:

Conditioning generative models

How to make a generative model conditional on y:

- VAEs (encoder $q_\phi(z \mid x)$, decoder $p_\theta(x \mid z, y)$):



Conditioning generative models

How to make a generative model conditional on y:

- VAEs (encoder $q_\phi(z \mid x)$, decoder $p_\theta(x \mid z, y)$):
  - Give y as an additional input to the encoder and decoder: $q_\phi(z \mid x, y)$, $p_\theta(x \mid z, y)$

Conditioning generative models

How to make a generative model conditional on y:

- VAEs (encoder $q_\phi(z \mid x)$, decoder $p_\theta(x \mid z, y)$):
  - Give y as an additional input to the encoder and decoder: $q_\phi(z \mid x, y)$, $p_\theta(x \mid z, y)$
  - Alternatively, train as usual, then fit a conditional prior $p_\theta(z \mid y)$ to samples from $q_\phi(z \mid x)$

Conditioning generative models

How to make a generative model conditional on y:

- VAEs (encoder $q_\phi(z \mid x)$, decoder $p_\theta(x \mid z, y)$):
  - Give y as an additional input to the encoder and decoder: $q_\phi(z \mid x, y)$, $p_\theta(x \mid z, y)$
  - Alternatively, train as usual, then fit a conditional prior $p_\theta(z \mid y)$ to samples from $q_\phi(z \mid x)$
- Normalising flows (composition of invertible layers $h_n = f_n(h_{n-1})$):

Conditioning generative models

How to make a generative model conditional on y:

- VAEs (encoder $q_\phi(z \mid x)$, decoder $p_\theta(x \mid z, y)$):
  - Give y as an additional input to the encoder and decoder: $q_\phi(z \mid x, y)$, $p_\theta(x \mid z, y)$
  - Alternatively, train as usual, then fit a conditional prior $p_\theta(z \mid y)$ to samples from $q_\phi(z \mid x)$
- Normalising flows (composition of invertible layers $h_n = f_n(h_{n-1})$):
  - Give y as an additional input: $h_n = f_n(h_{n-1}, y)$

Conditioning generative models

How to make a generative model conditional on y:

- VAEs (encoder $q_\phi(z \mid x)$, decoder $p_\theta(x \mid z, y)$):
  - Give y as an additional input to the encoder and decoder: $q_\phi(z \mid x, y)$, $p_\theta(x \mid z, y)$
  - Alternatively, train as usual, then fit a conditional prior $p_\theta(z \mid y)$ to samples from $q_\phi(z \mid x)$
- Normalising flows (composition of invertible layers $h_n = f_n(h_{n-1})$):
  - Give y as an additional input: $h_n = f_n(h_{n-1}, y)$
  - Alternatively (not common), train as usual, then fit a new conditional base distribution $p_\theta(z \mid y)$
- GANs (generator $G_\theta(z)$, discriminator $D_\phi(\text{real} \mid x)$):

Conditioning generative models

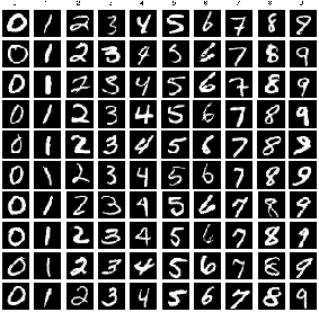
How to make a generative model conditional on y:

- VAEs (encoder $q_\phi(z \mid x)$, decoder $p_\theta(x \mid z, y)$):
  - Give y as an additional input to the encoder and decoder: $q_\phi(z \mid x, y)$, $p_\theta(x \mid z, y)$
  - Alternatively, train as usual, then fit a conditional prior $p_\theta(z \mid y)$ to samples from $q_\phi(z \mid x)$
- Normalising flows (composition of invertible layers $h_n = f_n(h_{n-1})$):
  - Give y as an additional input: $h_n = f_n(h_{n-1}, y)$
  - Alternatively (not common), train as usual, then fit a new conditional base distribution $p_\theta(z \mid y)$
- GANs (generator $G_\theta(z)$, discriminator $D_\phi(\text{real} \mid x)$):
  - Give y as an additional input to the generator and discriminator: $G_\theta(z, y)$, $D_\phi(\text{real} \mid x, y)$

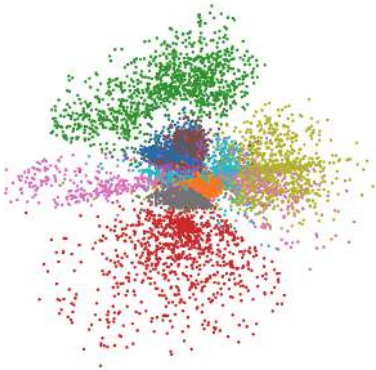
- Review of generative model families
  - Conditional models
- Representation learning
  - Representations from deep networks
  - Probing representations
- Evaluation of generative models
  - Likelihood-based evaluation
  - Sample-based evaluation

Encodings from generative models

Recall: latent variables 'encode' the cdata



$$\xrightarrow{q_\phi(z|x)}$$



We can treat the inferred latents as feaures extracted from the data

Encodings from generative models

Recall: latent variables ‘encode’ the cdata We can treat the inferred latents as feaures extracted from the data

- VAEs: the latent variables are trained for optimal reconstruction
  - Generalisation of PCA, which is the case of linear encoder and decoder with Gaussian noise

Encodings from generative models

Recall: latent variables 'encode' the cdata We can treat the inferred latents as feaures extracted from the data

- VAEs: the latent variables are trained for optimal reconstruction
  - Generalisation of PCA, which is the case of linear encoder and decoder with Gaussian noise
- Normalising flows: invert the flow to get a latent representation
  - Same dimension as the data, so may not be useful

Encodings from generative models

Recall: latent variables 'encode' the cdata We can treat the inferred latents as feaures extracted from the data

- VAEs: the latent variables are trained for optimal reconstruction
  - Generalisation of PCA, which is the case of linear encoder and decoder with Gaussian noise
- Normalising flows: invert the flow to get a latent representation
  - Same dimension as the data, so may not be useful
- GANs: we cannot reconstruct z from $G_\theta(z)$, and many (or no) z can map to the same x
  - Some work on 'GAN inversion' to reconstruct z from x (not common)

Encodings from generative models

Recall: latent variables 'encode' the cdata We can treat the inferred latents as feaures extracted from the data

- VAEs: the latent variables are trained for optimal reconstruction
  - Generalisation of PCA, which is the case of linear encoder and decoder with Gaussian noise
- Normalising flows: invert the flow to get a latent representation
  - Same dimension as the data, so may not be useful
- GANs: we cannot reconstruct z from $G_\theta(z)$, and many (or no) z can map to the same x
  - Some work on 'GAN inversion' to reconstruct z from x (not common)

These representations are generally not identifiable and emerge only as a side effect of data modelling:

- No way to know which of many possible representations we get
- Compose with invertible transformations to get other representations that are equally good

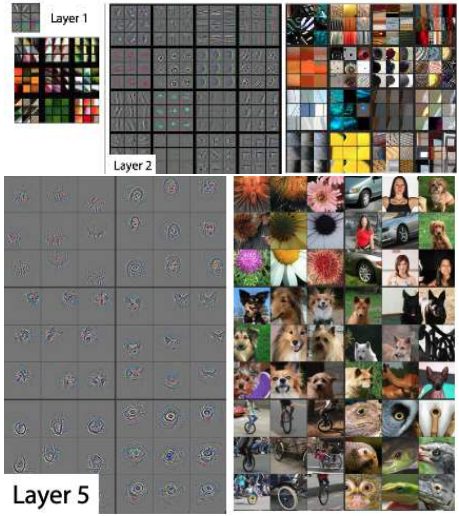
Accidental representation learning without generation

Hidden units of deep networks taking data as input can learn useful representations:

Accidental representation learning without generation

Hidden units of deep networks taking data as input can learn useful representations:

- First observed in CNNs for classification: shallow layers encode local features, deeper layers learn to encode more global features

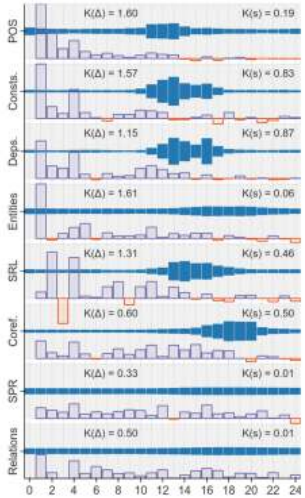


[Zeiler and Fergus, 2014]

Accidental representation learning without generation

Hidden units of deep networks taking data as input can learn useful representations:

- First observed in CNNs for classification: shallow layers encode local features, deeper layers learn to encode more global features
- In 'large' language models, the hidden units correspond to a variety of linguistic features



[Tenney et al., 2019]

Other representation learning methods

Not the topic of this class, but some methods for dimensionality reduction that do not involve training a probabilistic model of the data

Other representation learning methods

Not the topic of this class, but some methods for dimensionality reduction that do not involve training a probabilistic model of the data

- Train a representation $z = f_\theta(x)$ with no reconstruction of x

Other representation learning methods

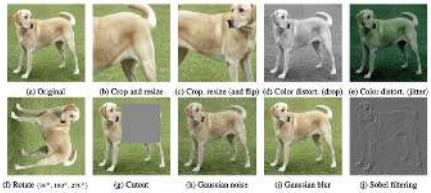
Not the topic of this class, but some methods for dimensionality reduction that do not involve training a probabilistic model of the data

- Train a representation $z = f_\theta(x)$ with no reconstruction of x
- Denoising autoencoders: train a model to reconstruct noisy version of x, use its deep features
  - Masked language models are a special case of this
  - Related to diffusion modelling (in two weeks)

Other representation learning methods

Not the topic of this class, but some methods for dimensionality reduction that do not involve training a probabilistic model of the data

- Train a representation $z = f_\theta(x)$ with no reconstruction of x
- Denoising autoencoders: train a model to reconstruct noisy version of x, use its deep features
- Contrastive learning (*e.g.*, SimCLR, InfoNCE)
  - Take a **batch** of data points $\{x_1, \ldots, x_n\}$
  - Apply a random transformation, to which the representation should be invariant, to get $\{\tilde{x}_1, \ldots, \tilde{x}_n\}$
  - Somehow enforce that $f_\theta(x_i)$ is close to $f_\theta(\tilde{x}_i)$ but far from the $f_\theta(x_j)$ for $j \neq i$

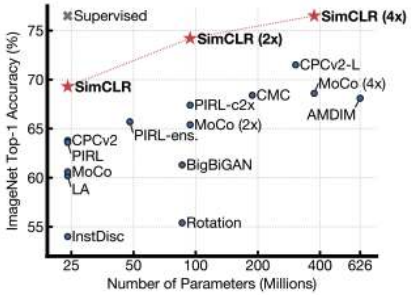


[Chen et al., SimCLR, 2020]

Other representation learning methods

Not the topic of this class, but some methods for dimensionality reduction that do not involve training a probabilistic model of the data

- Train a representation $z = f_\theta(x)$ with no reconstruction of x
- Denoising autoencoders: train a model to reconstruct noisy version of x, use its deep features
- Contrastive learning (*e.g.*, SimCLR, InfoNCE)
  - Take a **batch** of data points $\{x_1, \ldots, x_n\}$
  - Apply a random transformation, to which the representation should be invariant, to get $\{\tilde{x}_1, \ldots, \tilde{x}_n\}$
  - Somehow enforce that $f_\theta(x_i)$ is close to $f_\theta(\tilde{x}_i)$ but far from the $f_\theta(x_j)$ for $j \neq i$



Shallow probing of representations

How do we know if a representation is good?

Shallow probing of representations

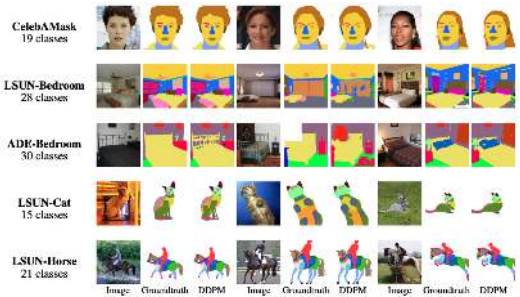
How do we know if a representation is good?

- It should be predictive of something we care about (*e.g.*, a class label)

Shallow probing of representations

How do we know if a representation is good?

- It should be predictive of something we care about (*e.g.*, a class label)
- **Probing:** train a simple model (often a linear/logistic regression) to predict some property of the data from the representation
- The better the probe does, the better the representation is for that property

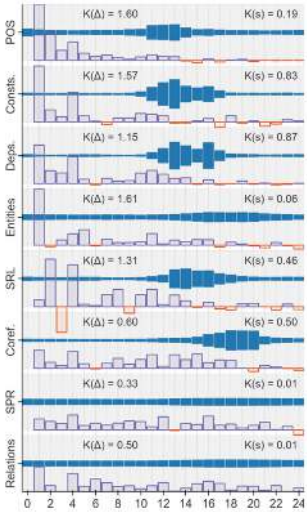


[Baranchuk et al., 2022]

Shallow probing of representations

How do we know if a representation is good?

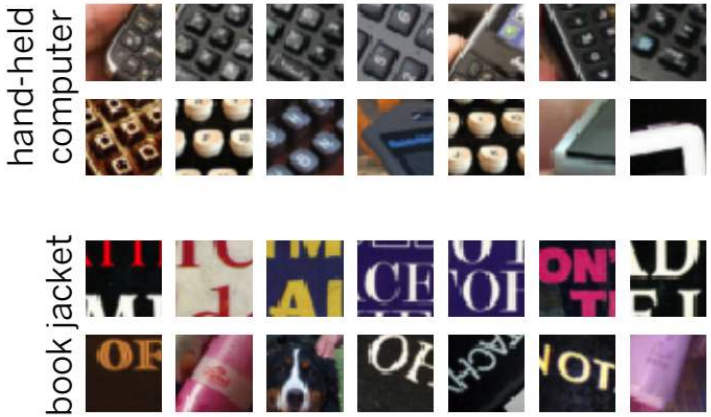
- It should be predictive of something we care about (*e.g.*, a class label)
- **Probing:** train a simple model (often a linear/logistic regression) to predict some property of the data from the representation
- The better the probe does, the better the representation is for that property



[Tenney et al., 2019]

Shallow probing of representations

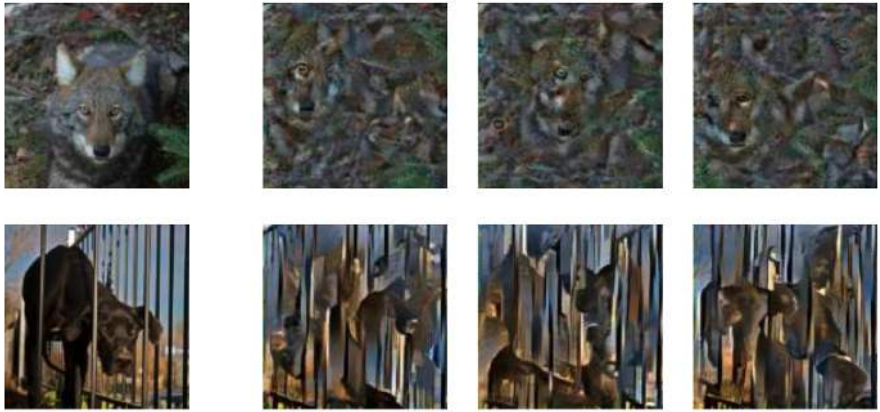
Can also identify the hidden units in a neural net that predict a given property . . and try to maximise their activation



[Brendel and Bethge, BagNet, 2019] ... hacking the convolutional inductive bias

Shallow probing of representations

Can also identify the hidden units in a neural net that predict a given property . . and try to maximise their activation



[Brendel and Bethge, BagNet, 2019] – hacking the convolutional inductive bias

Shallow probing of representations

How do we know if a representation is good?

- It should be predictive of something we care about (*e.g.*, a class label)
- **Probing:** train a simple model (often a linear/logistic regression) to predict some property of the data from the representation
- The better the probe does, the better the representation is for that property

Can also identify the hidden units in a neural net that predict a given property

- Very large models may contain 'circuits' responsible for approximate execution of simple symbolic routines (copying, memory, . . .)
  - Controversial: you can find a circuit for anything if you look hard enough. . .

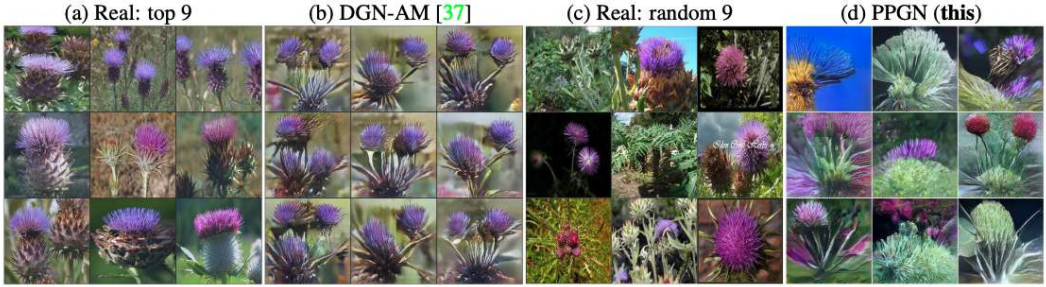
Interventions and controllability

Representations can be useful if they can be modified to control the output:

Interventions and controllability

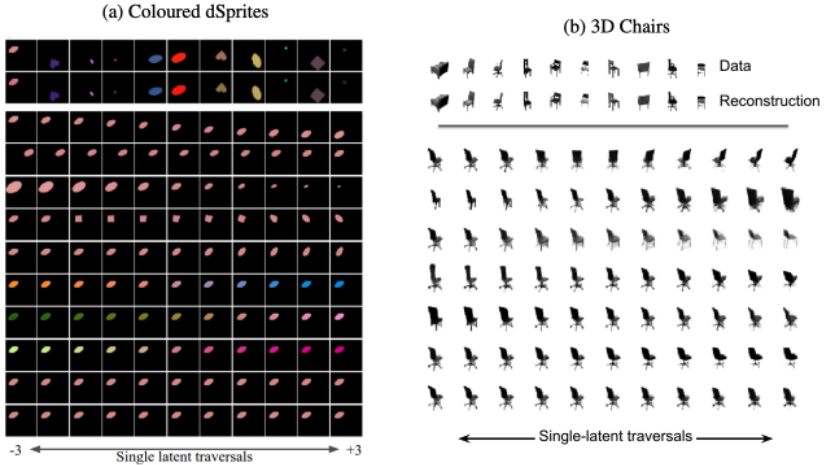
Representations can be useful if they can be modified to control the output:

- It may be desirable to **intervene** on a hidden unit / latent representation to change the output in a desired way:
  - Style/content transfer for images; conditional samples from unconditional models
  - Changing the sentiment of a text while keeping the content



[Nguyen et al., 'Plug & Play Generative Networks', 2017]

Interventions and controllability



[Burgess et al., Understanding disentangling in β-VAE, 2018]

Interventions and controllability

Representations can be useful if they can be modified to control the output:

- It may be desirable to **intervene** on a hidden unit / latent representation to change the output in a desired way:
  - Style/content transfer for images; conditional samples from unconditional models
  - Changing the sentiment of a text while keeping the content
- **Disentanglement:** different dimensions of the representation correspond to different properties of the data, can be independently controlled
  - In principle, should allow **compositionality** of representations $\rightsquigarrow$ **systematic generalisation**
  - Active area of research (but somewhat lacking in clear definitions and metrics)

- Review of generative model families
  - Conditional models
- Representation learning
  - Representations from deep networks
  - Probing representations
- Evaluation of generative models
  - Likelihood-based evaluation
  - Sample-based evaluation

Log-likelihood as a metric

- Recall: Minimising $\mathrm{KL}(\pi_{\text{data}} \| \pi_\theta) \equiv$ maximising sample log-likelihood $\mathbb{E}_{X \sim \pi_{\text{data}}}[\log p_\theta(X)]$

Log-likelihood as a metric

- Recall: Minimising $\mathrm{KL}(\pi_{\text{data}} \| \pi_\theta) \equiv$ maximising sample log-likelihood $\mathbb{E}_{X \sim \pi_{\text{data}}}[\log p_\theta(X)]$
- Use the negative log-likelihood (NLL) of a held-out test set (an independent sample from π_{data}) as an evaluation metric:

$$\mathrm{NLL} = -\mathbb{E}_{X \sim \pi_{\text{data}}}[\log p_\theta(X)]$$

Log-likelihood as a metric

- Recall: Minimising $\mathrm{KL}(\pi_{\text{data}} \| \pi_\theta) \equiv$ maximising sample log-likelihood $\mathbb{E}_{X \sim \pi_{\text{data}}}[\log p_\theta(X)]$
- Use the negative log-likelihood (NLL) of a held-out test set (an independent sample from π_{data}) as an evaluation metric:

$$\begin{aligned} \text{NLL} &= -\mathbb{E}_{X \sim \pi_{\text{data}}}[\log p_\theta(X)] \\ &\approx -\frac{1}{n} \sum_{i=1}^{n} \log p_\theta(x_i), \qquad x_i \sim \pi_{\text{data}} \end{aligned}$$

Log-likelihood as a metric

- Recall: Minimising $\mathrm{KL}(\pi_{\text{data}} \| \pi_\theta) \equiv$ maximising sample log-likelihood $\mathbb{E}_{X \sim \pi_{\text{data}}}[\log p_\theta(X)]$
- Use the negative log-likelihood (NLL) of a held-out test set (an independent sample from π_{data}) as an evaluation metric:

$$\begin{aligned} \mathrm{NLL} &= -\mathbb{E}_{X \sim \pi_{\text{data}}}[\log p_\theta(X)] \\ &\approx -\frac{1}{n} \sum_{i=1}^{n} \log p_\theta(x_i), \qquad x_i \sim \pi_{\text{data}} \end{aligned}$$

- Often reported in 'bits per dimension': $\mathrm{BPD} = \frac{\mathrm{NLL}}{d \log 2}$ for data in $\mathbb{R}^d$
  - Best BPDs for MNIST are ~ 1 (note $\log_2 256 = 8$ for a uniform distribution over 8-bit pixel values),
  - Best bits per character for character-level language modelling: ~ 1 bit per character (note $\log_2 27 \approx 4.8$ bpc for a uniform distribution)

Problems with likelihood-based evaluation

What is wrong with using NLL as an evaluation metric?

Problems with likelihood-based evaluation

What is wrong with using NLL as an evaluation metric?

- Cannot evaluate it when we do not have the denisty (such as for GANs)

Problems with likelihood-based evaluation

What is wrong with using NLL as an evaluation metric?

- Cannot evaluate it when we do not have the denisty (such as for GANs)
- Not always correlated with sample quality as judged by humans (for images, NLL is often dominated by low-level details/textures)

Problems with likelihood-based evaluation

What is wrong with using NLL as an evaluation metric?

- Cannot evaluate it when we do not have the denisty (such as for GANs)
- Not always correlated with sample quality as judged by humans (for images, NLL is often dominated by low-level details/textures)
- In some domains, considered to be 'saturated' (no longer meaningfully distinguishes between models)

Sample-based evaluation

Compare samples from the model to *samples* from the data distribution

Sample-based evaluation

Compare samples from the model to *samples* from the data distribution

- Draw a large sample $\{x_1, \ldots, x_n\}$ from the data distribution π_{data} (a held-out test set)
- Draw a large sample $\{\tilde{x}_1, \ldots, \tilde{x}_m\}$ from the model distribution π_θ

Sample-based evaluation

Compare samples from the model to *samples* from the data distribution

- Draw a large sample $\{x_1, \ldots, x_n\}$ from the data distribution π_{data} (a held-out test set)
- Draw a large sample $\{\tilde{x}_1, \ldots, \tilde{x}_m\}$ from the model distribution π_θ
- Compute a divergence between the **empirical** distributions $\frac{1}{n}\sum_{i=1}^{n} \delta_{x_i}$ and $\frac{1}{m}\sum_{j=1}^{m} \delta_{\tilde{x}_j}$

Sample-based evaluation

Compare samples from the model to *samples* from the data distribution

- Draw a large sample $\{x_1, \ldots, x_n\}$ from the data distribution π_{data} (a held-out test set)
- Draw a large sample $\{\tilde{x}_1, \ldots, \tilde{x}_m\}$ from the model distribution π_θ
- Compute a divergence between the **empirical** distributions $\frac{1}{n}\sum_{i=1}^{n} \delta_{x_i}$ and $\frac{1}{m}\sum_{j=1}^{m} \delta_{\tilde{x}_j}$

Often, first map to a representation space, that is, compare $\{f(x_1), \ldots, f(x_n)\}$ to $\{f(\tilde{x}_1), \ldots, f(\tilde{x}_m)\}$ for some pretrained representation f

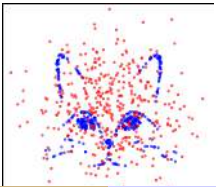
Sample-based evaluation

Compare samples from the model to *samples* from the data distribution

- Draw a large sample $\{x_1, \ldots, x_n\}$ from the data distribution π_{data} (a held-out test set)
- Draw a large sample $\{\tilde{x}_1, \ldots, \tilde{x}_m\}$ from the model distribution π_θ
- Compute a divergence between the **empirical** distributions $\frac{1}{n}\sum_{i=1}^{n} \delta_{x_i}$ and $\frac{1}{m}\sum_{j=1}^{m} \delta_{\tilde{x}_j}$

Often, first map to a representation space, that is, compare $\{f(x_1), \ldots, f(x_n)\}$ to $\{f(\tilde{x}_1), \ldots, f(\tilde{x}_m)\}$ for some pretrained representation f

- What divergence should be used?



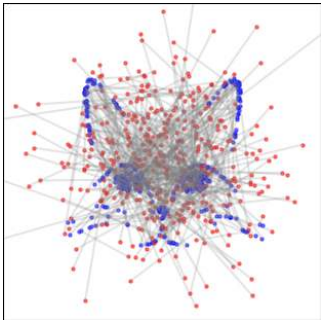
Optimal transport

The field of **optimal transport** studies distances between probability distributions defined by moving samples from one distribution to another at minimal cost...

Optimal transport

The field of **optimal transport** studies distances between probability distributions defined by moving samples from one distribution to another at minimal cost...

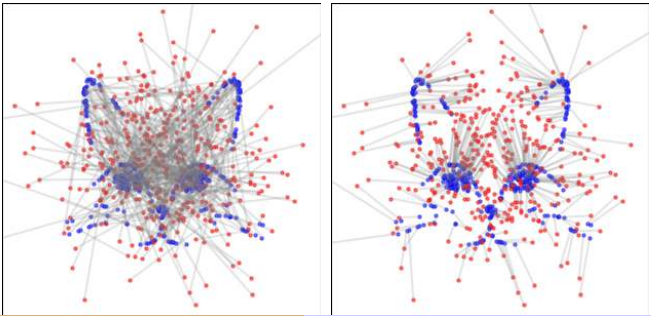
- For two samples of size n in $\mathbb{R}^d$, we can pair the two samples in $n!$ says



Optimal transport

The field of **optimal transport** studies distances between probability distributions defined by moving samples from one distribution to another at minimal cost...

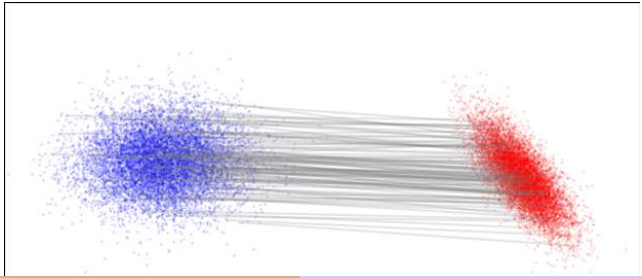
- For two samples of size n in $\mathbb{R}^d$, we can pair the two samples in $n!$ says
- Find the one that minimises the average squared distance between paired points; this is the (squared) **2-Wasserstein distance**



Optimal transport

The field of **optimal transport** studies distances between probability distributions defined by moving samples from one distribution to another at minimal cost...

- For two samples of size n in $\mathbb{R}^d$, we can pair the two samples in $n!$ says
- Find the one that minimises the average squared distance between paired points; this is the (squared) **2-Wasserstein distance**
- Can also be defined for densities or different numbers of samples



Optimal transport

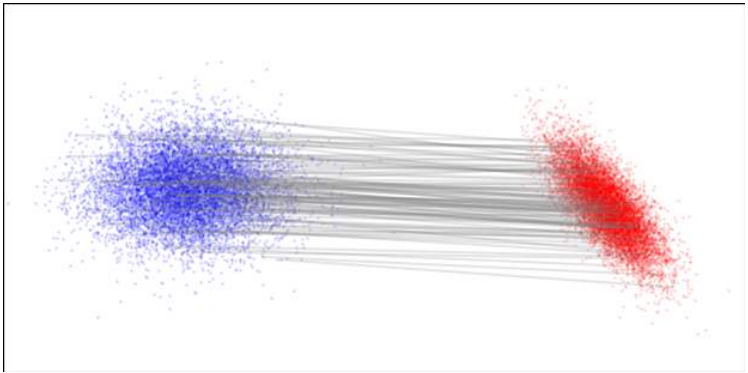
The field of **optimal transport** studies distances between probability distributions defined by moving samples from one distribution to another at minimal cost...

- For two samples of size n in $\mathbb{R}^d$, we can pair the two samples in $n!$ says
- Find the one that minimises the average squared distance between paired points; this is the (squared) **2-Wasserstein distance**
- Can also be defined for densities or different numbers of samples
- Many generalisations and rich mathematical theory, connections throughout ML
  - Replace the squared distance with some other cost function
  - Add entropy regularisation ($\rightsquigarrow$ **Sinkhorn distance**)
  - Connections with diffusion modelling, GANs, optimisation (see [Peyré, 'Optimal and Diffusion Transports in Machine Learning'] for interesting examples)

Fréchet Inception Distance

The Wasserstein distance between two Gaussian distributions has a closed-form solution:

$$\mathcal{W}_2^2(\mathcal{N}(\mu_1, \Sigma_1), \mathcal{N}(\mu_2, \Sigma_2)) = \|\mu_1 - \mu_2\|^2 + \mathrm{Tr}\left(\Sigma_1 + \Sigma_2 - 2\left(\Sigma_1^{1/2}\Sigma_2\Sigma_1^{1/2}\right)^{1/2}\right)$$



Fréchet Inception Distance

The Wasserstein distance between two Gaussian distributions has a closed-form solution:

$$\mathcal{W}_2^2(\mathcal{N}(\mu_1, \Sigma_1), \mathcal{N}(\mu_2, \Sigma_2)) = \|\mu_1 - \mu_2\|^2 + \mathrm{Tr}\left(\Sigma_1 + \Sigma_2 - 2\left(\Sigma_1^{1/2}\Sigma_2\Sigma_1^{1/2}\right)^{1/2}\right)$$

To compute the **Fréchet Inception Distance (FID)**:

- Embed samples from the data and model into a representation space (using Inception model) to obtain $f(x_i)$ and $f(\tilde{x}_j)$

Fréchet Inception Distance

The Wasserstein distance between two Gaussian distributions has a closed-form solution:

$$\mathcal{W}_2^2(\mathcal{N}(\mu_1, \Sigma_1), \mathcal{N}(\mu_2, \Sigma_2)) = \|\mu_1 - \mu_2\|^2 + \mathrm{Tr}\left(\Sigma_1 + \Sigma_2 - 2\left(\Sigma_1^{1/2}\Sigma_2\Sigma_1^{1/2}\right)^{1/2}\right)$$

To compute the **Fréchet Inception Distance (FID)**:

- Embed samples from the data and model into a representation space (using Inception model) to obtain $f(x_i)$ and $f(\tilde{x}_j)$
- Fit a Gaussian to each set of embedded samples

Fréchet Inception Distance

The Wasserstein distance between two Gaussian distributions has a closed-form solution:

$$\mathcal{W}_2^2(\mathcal{N}(\mu_1, \Sigma_1), \mathcal{N}(\mu_2, \Sigma_2)) = \|\mu_1 - \mu_2\|^2 + \mathrm{Tr}\left(\Sigma_1 + \Sigma_2 - 2\left(\Sigma_1^{1/2}\Sigma_2\Sigma_1^{1/2}\right)^{1/2}\right)$$

To compute the **Fréchet Inception Distance (FID)**:

- Embed samples from the data and model into a representation space (using Inception model) to obtain $f(x_i)$ and $f(\tilde{x}_j)$
- Fit a Gaussian to each set of embedded samples
- FID is the Wasserstein distance between these two Gaussians

Fréchet Inception Distance

CIFAR-10

Model	NFE↓	FID↓
BigGAN [3]	1	14.73
TransGAN [87]	1	9.26
ViTGAN [42]	1	6.66
DDGAN [94]	4	3.75
Diffusion StyleGAN2 [90]	1	3.19
StyleGAN2 + ADA [30]	1	2.42
StyleGAN3-R + ADA [32, 25]	1	10.83
DDPM [20]	1000	3.21
DDIM [76]	50	4.67
VE [78, 33]	35	3.11
VP [78, 33]	35	2.48
Ours—Config E	1	1.96

ImageNet (64×64)

Model	NFE↓	FID↓
BigGAN-deep [3]	1	4.06
DDPM [20]	250	11.0
DDIM [76]	50	13.7
ADM [7]	§250	2.91
EDM [33]	79	2.23
CT [79]	2	11.1
CD [79]	3	4.32
iCT-deep [77]	2	2.77
DMD [96]	1	2.62
Ours—Config E	1	2.09

from [Huang et al., 'The GAN is Dead; Long Live the GAN!', 2025]

Other sample-based metrics

Other options exist, some domain-specific:

Other sample-based metrics

Other options exist, some domain-specific:

- Inception score (involves fitting a classifier to the data, comparing class distributions of real and generated samples)

Other sample-based metrics

Other options exist, some domain-specific:

- Inception score (involves fitting a classifier to the data, comparing class distributions of real and generated samples)
- Maximum mean discrepancy (MMD)

Other sample-based metrics

Other options exist, some domain-specific:

- Inception score (involves fitting a classifier to the data, comparing class distributions of real and generated samples)
- Maximum mean discrepancy (MMD)
- Various metrics intended to better capture quality, diversity, and novelty, *e.g.*, feature likelihood divergence (FLD)

Other sample-based metrics

Other options exist, some domain-specific:

- Inception score (involves fitting a classifier to the data, comparing class distributions of real and generated samples)
- Maximum mean discrepancy (MMD)
- Various metrics intended to better capture quality, diversity, and novelty, *e.g.*, feature likelihood divergence (FLD)
- For images: LPIPS (not a distributional distance, but a distance between individual samples, correlated with human judgement)

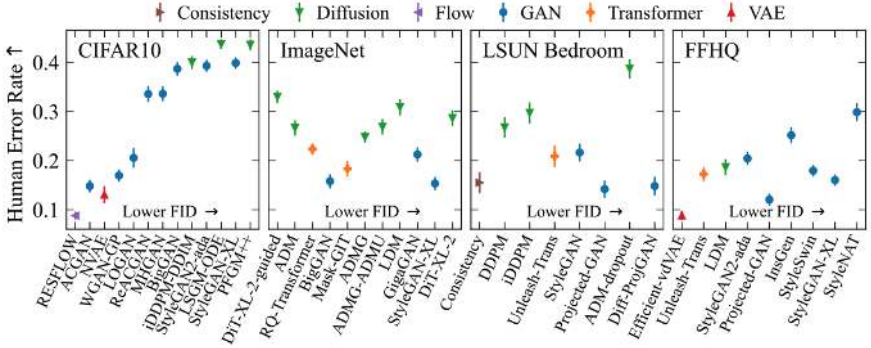
Problems with sample-based evaluation

What is wrong with sample-based evaluation?

Problems with sample-based evaluation

What is wrong with sample-based evaluation?

- Not always correlated with human judgement of sample quality, ignores information not contained in the representations (for FID, in their second-moment statistics)



[Stein et al., 'Exposing flaws of generative model evaluation metrics. . . ', 2023]

Problems with sample-based evaluation

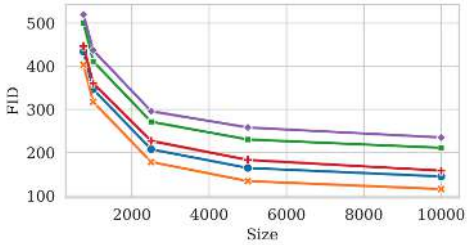
What is wrong with sample-based evaluation?

- Not always correlated with human judgement of sample quality, ignores information not contained in the representations (for FID, in their second-moment statistics)
- Not suitable for conditional settings where only one sample of x is available for each y (*e.g.*, text-to-image generation)

Problems with sample-based evaluation

What is wrong with sample-based evaluation?

- Not always correlated with human judgement of sample quality, ignores information not contained in the representations (for FID, in their second-moment statistics)
- Not suitable for conditional settings where only one sample of x is available for each y (*e.g.*, text-to-image generation)
- Often noisy (requires a large number of samples to get a reliable estimate)



[Jiralerspong et al., 'Feature Likelihood Divergence...', 2023]

Conclusion and looking ahead

- Representation learning is a side effect of generative model training
- Evaluation of generative models requires selecting a metric that is appropriate for the model and the domain and reflection on the features whose distribution is being compared
- Likelihood-based metrics not always appropriate, especially for very high-dimensional data where deep features are more important than low-level details
- Sample-based metrics can be more correlated with human judgement, but are often noisy and rely on a good choice of representation

Conclusion and looking ahead

- Representation learning is a side effect of generative model training
- Evaluation of generative models requires selecting a metric that is appropriate for the model and the domain and reflection on the features whose distribution is being compared
- Likelihood-based metrics not always appropriate, especially for very high-dimensional data where deep features are more important than low-level details
- Sample-based metrics can be more correlated with human judgement, but are often noisy and rely on a good choice of representation
- Next time: a different view on evaluating 'generative AI'