DD2437 – Artificial Neural Networks and Deep Architectures (annda)

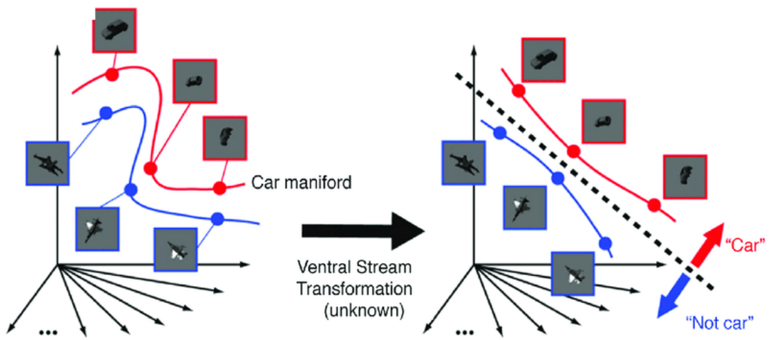
Lecture 11: **Deep representations and deep generative models**

Pawel Herman

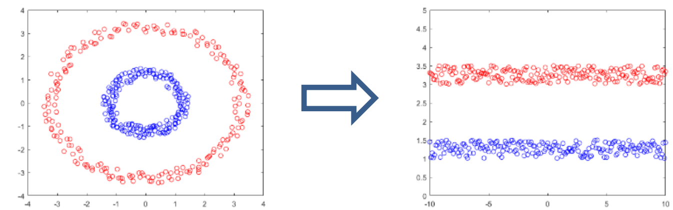
Computational Science and Technology (CST)

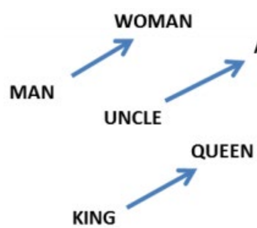
KTH Royal Institute of Technology

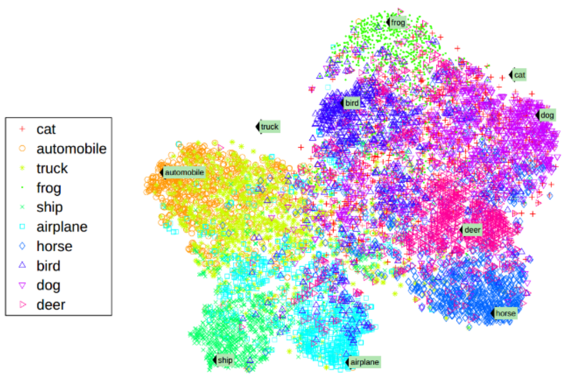
The importance of representations



Good representations can make certain computations cheaper and more robust







"woman" -> [-0.3, 0.2, 0.5, 0, -0.1, 0…]

"woman" – "man" ≈ "queen" – "king"

Learning representations + complex inference = future AI?

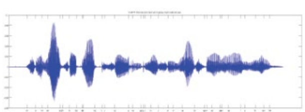
Learning representations as a hallmark of DL

Learning features as part of DL modus operandi

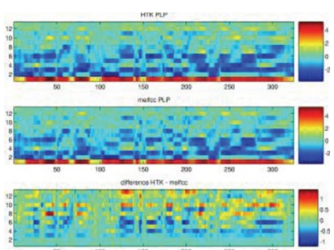
- Traditional pattern recognition *VS* deep neural network approach

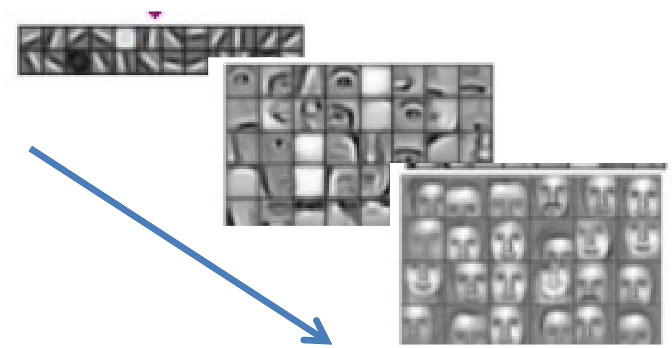
Hand-engineered features in a traditional pattern recognition approach

End-to-end networks with learned features spaces, data representations



input



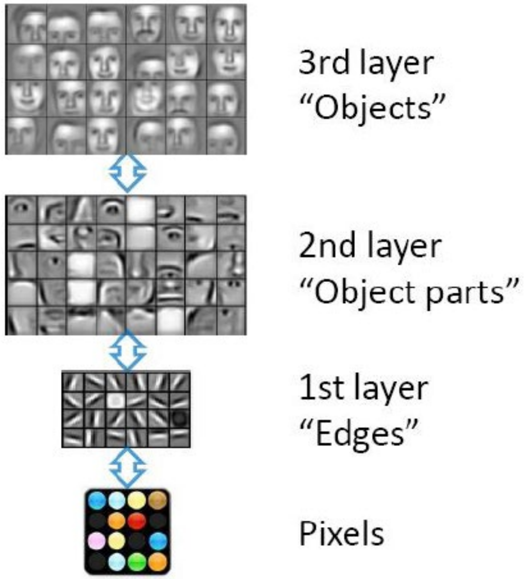


features, representations

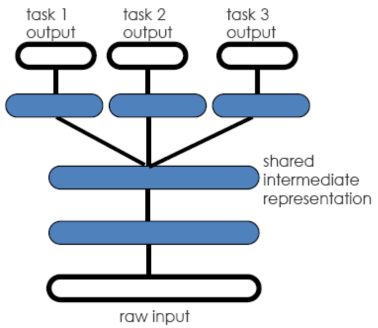
Learning representations as a hallmark of DL

Learning features as part of DL modus operandi with many implications......

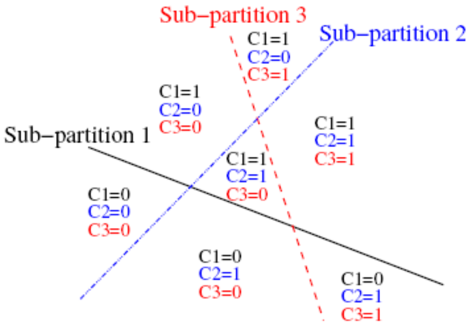
hierarchy of abstraction levels



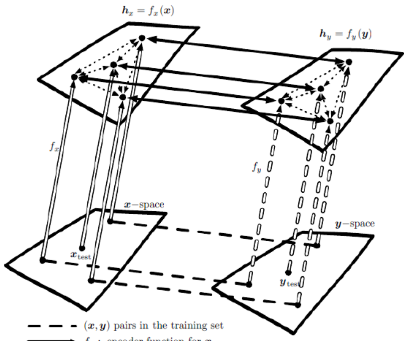
multi-tasking and transfer learning



multi-clustering



zero-shot learning



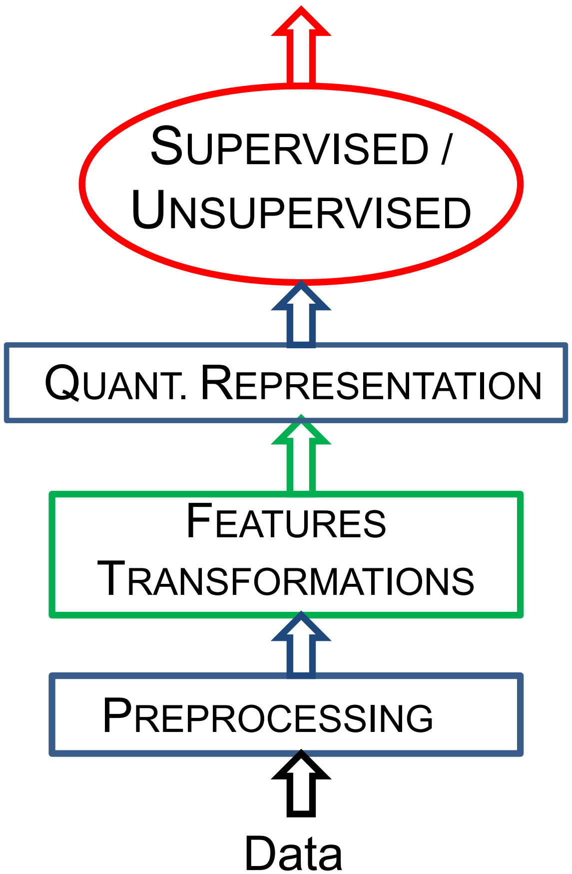
Representation learning problem

What makes representation good? What is desirable/useful information?

factors of variation, causal variables

information about key probabilistic features of data distribution

information needed to solve a given task



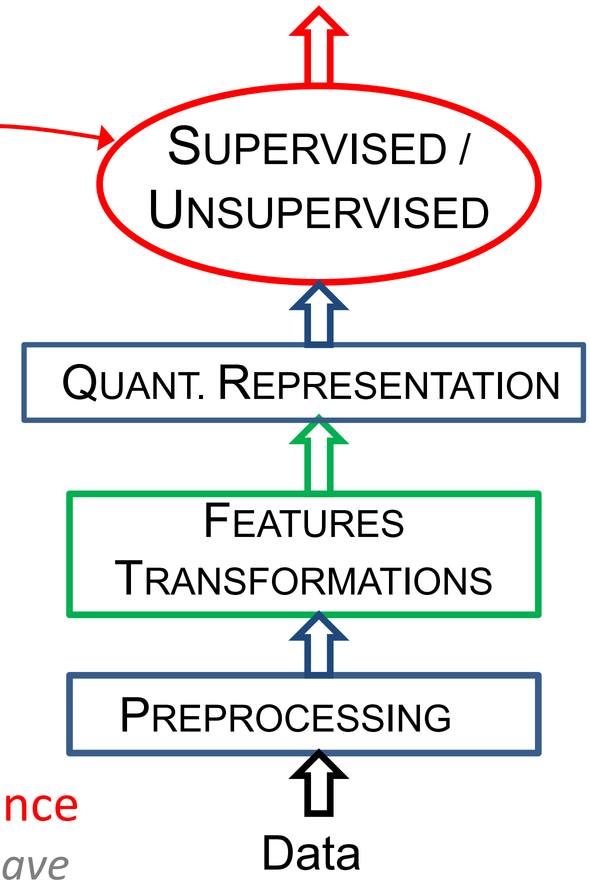
Representation learning problem

What makes representation good? What is desirable/useful information?

factors of variation, causal variables

information about key probabilistic features of data distribution

information needed to solve a given task

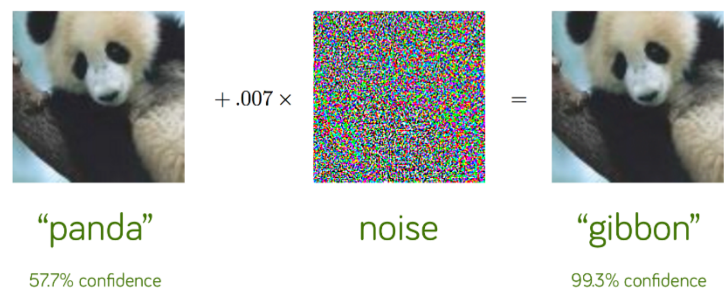


Facilitate the subsequent learning task, maximise performance

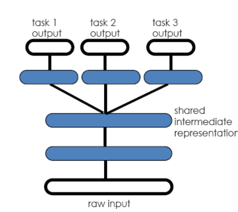
(*easiest to define for supervised learning problems but does not have to lead to "good" representations*)

Supervised vs unsupervised approach

- Supervised – distilling information relevant to a concrete task defined by labels
  - very useful when solving particular tasks
  - strongly relying on the abundance of labelled data and prone to excessive information bottleneck



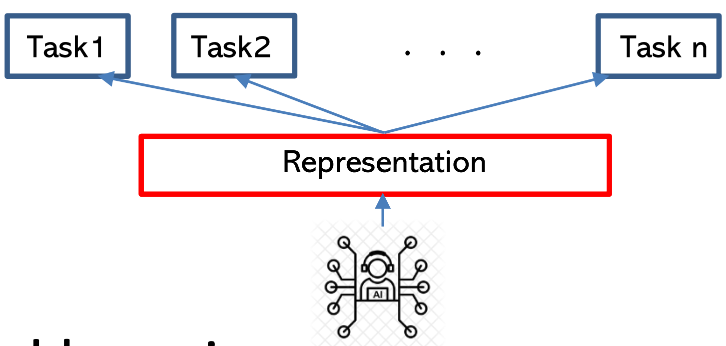
  - lacking “common sense”



  - different ways to “improve” representations

Supervised vs unsupervised approach

- Supervised – distilling information relevant to a concrete task defined by labels
- Unsupervised learning
  - "The idea of learning to represent the world before learning a task – this is what babies do" (*LeCun*)
  - It appears as a more generic approach less susceptible to inf. Loss



- Semi-supervised learning

Representation learning

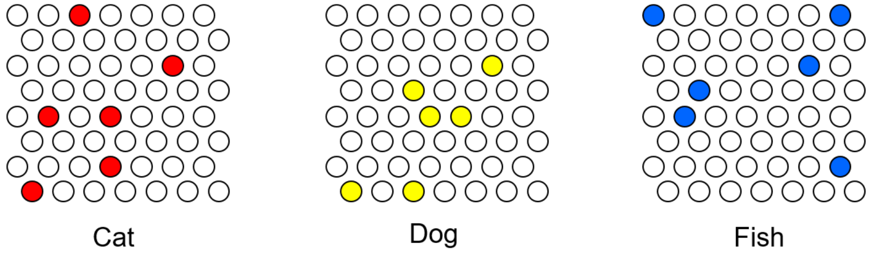
- Computational perspective: disentangling unknown factors causing relevant variation in the data (*factors of variation*)
  - *causes explain* the observed data (discriminative context, both unsupervised and supervised aspects)
  - factors in combination can be used to generate data (generative context)
- Probabilistic perspective
  - Classical approach: *density estimation* – learn probability distribution for data with the use of latent variables (PCA, ICA, GMM etc.) -> explain data
  - P(*data*|<u>*latent var*</u>) for generation and P(*latent var*|*data*) for recognition
  - full probabilistic model advocated by generative models, P(*data*,…)

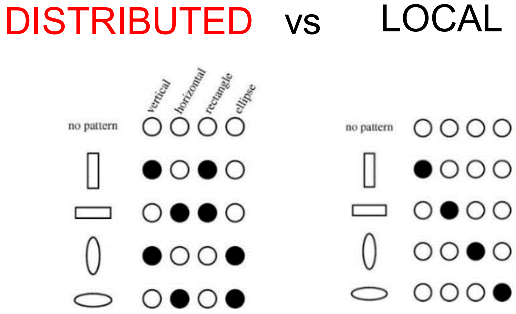
Distributed vs local representations

Information is distributed across many units that account for information about features that are not mutually exclusive.....

... unlike in clustering (cluster centres act as prototypes) with distinct regions where *local generalisation* is observed.

Locality in input space implies different behaviour of the learned function in different regions of data space (local or symbolic representations).





Generalisation due to shared attributes and semantic proximity.

Sparse vs dense representations

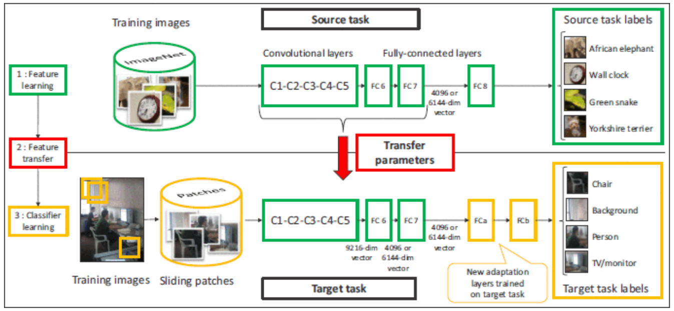
Sparse representations

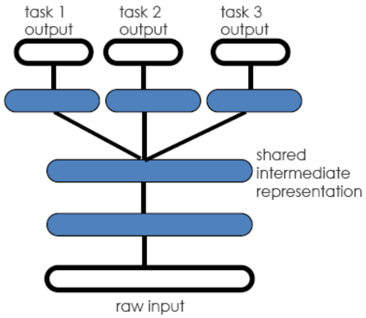
- orthogonalisation/decorrelation – more separable in high-dimensional spaces
- “metabolic” efficiency
- neural selectivity (vs coarse coding with broad tuning)
- balance between sparse local representations that suffer from the *curse of dimensionality* and dense representations that *entangle* factors and can be hard to interpret

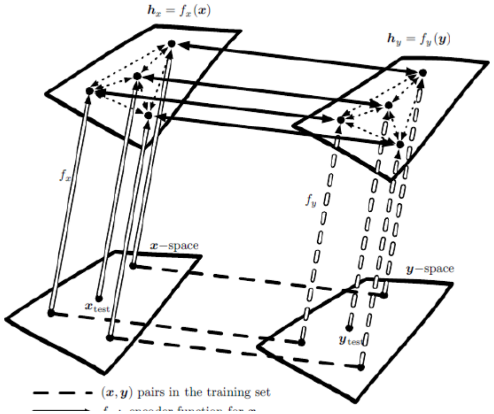
sparse not distributed	not sparse distributed	sparse distributed
0 | .2 | 0 | 0 | 0	.1 | .8 | .7 | .5 | .7	0 | .8 | 0 | .5 | 0
0 | 0 | 0 | 0 | .1	.8 | .9 | .6 | .2 | .4	0 | 0 | .6 | 0 | .4
0 | 0 | 0 | .4 | 0	.3 | .1 | .6 | .3 | .3	.3 | 0 | 0 | .3 | 0

Functional implications of distributed codes

- Transfer learning and domain adaptation
- Multi-task learning
- Zero-shot learning







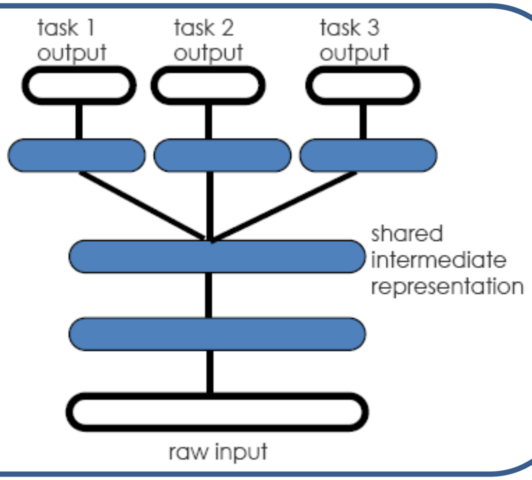
Transfer learning

A more general concept in ML: a model developed for one task is reused (re-purposed) as the starting point for a model on another task

- also referred to as *domain adaptation*
- related to both *multi-tasking* and *concept drift*

Sharing factors across tasks

- assumption that factors explaining the variations in different tasks are shared/common
- especially low-level features are expected to be the same
- supported by hierarchical and distributed representations



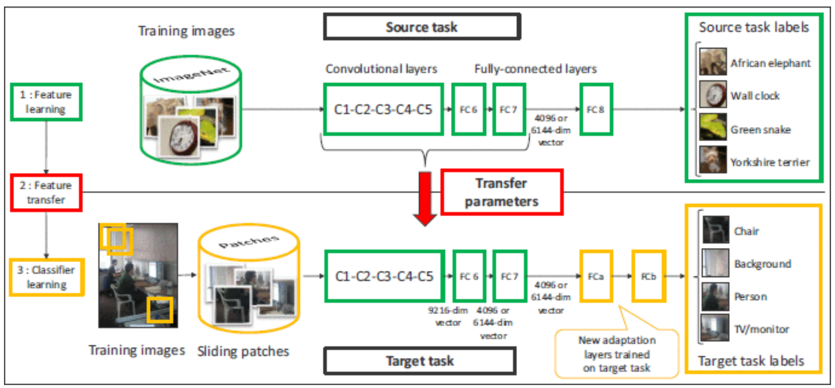
Transfer learning

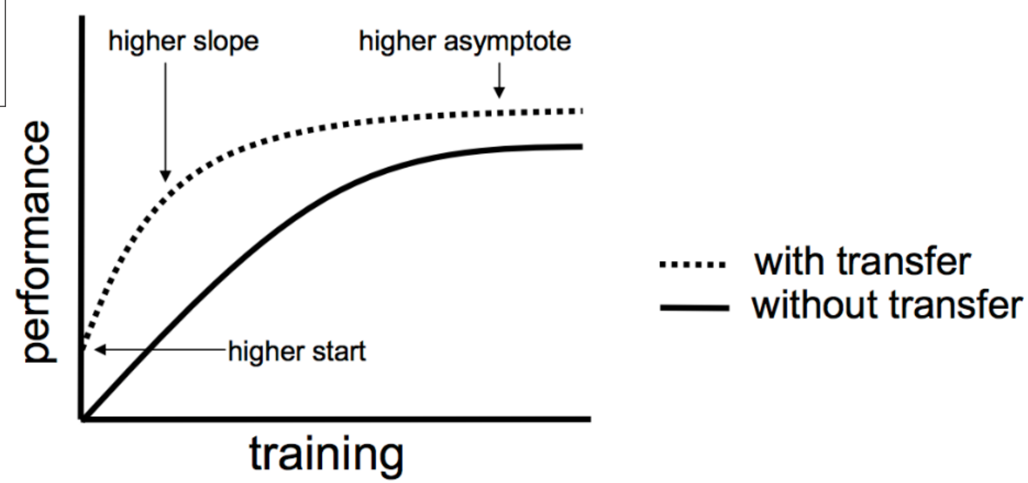
A more general concept in ML: a model developed for one task is reused (re-purposed) as the starting point for a model on another task

- also referred to as *domain adaptation*
- related to both *multi-tasking* and *concept drift*
- popular in DL as it offers an opportunity to save computational costs
- two main approaches: develop model and **pre-trained model**
  - Google Inception model, Oxford VGG model (ImageNet)
  - Stanford's GloVe Model, Google's word2vec Model
  - check out *Caffe Model Zoo*

Transfer learning

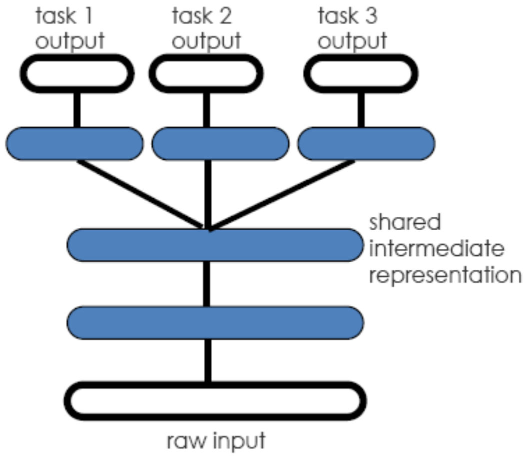
“Transfer learning and domain adaptation refer to the situation where what has been learned in one setting … is exploited to improve generalization in another setting”





Multi-task learning

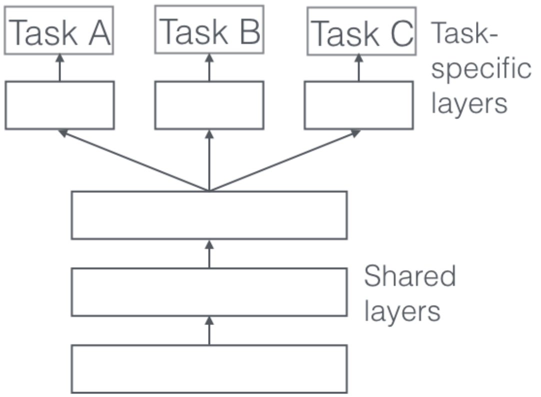
- Excessive focus on a single task may ignore information that can help us perform better on the metric we care about
- Effectively, simultaneously optimizing different cost functions implements multi-task learning
- Form of inductive transfer learning with inductive bias/regularisation mediated by another task
- "Multi-task learning improves generalization by leveraging the domain-specific information contained in the training signals of related tasks" (*Caruana, 1998*)



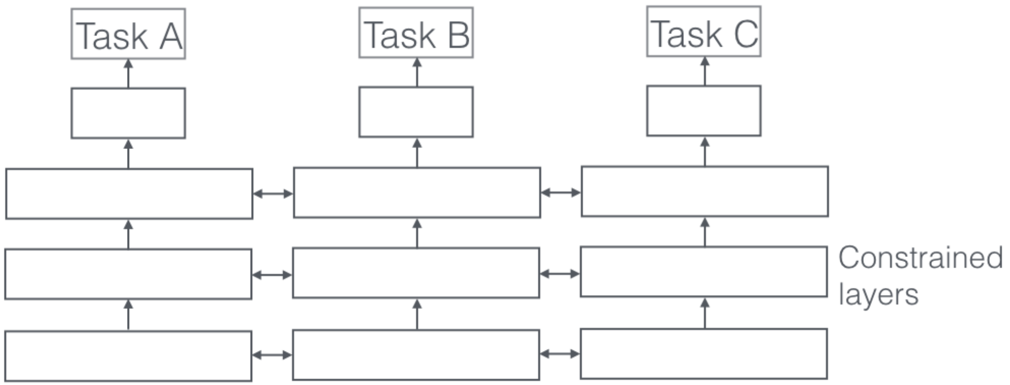
S. Ruder (2017). An Overview of Multi-Task Learning in Deep Neural Networks

Multi-task learning

Hard parameter sharing



Soft parameter sharing



Underlying mechanisms: implicit data augmentation, attention focusing, representation bias, regularisation

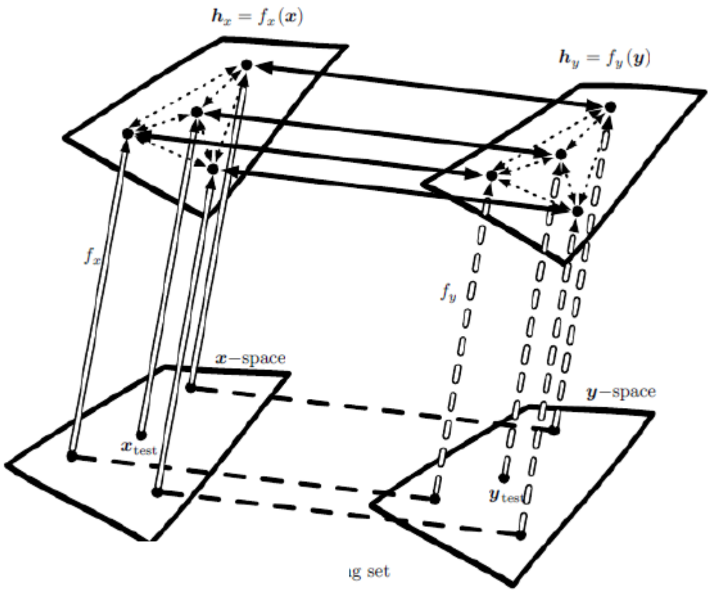
Zero-shot learning

- Zero-shot learning as a specific form of *multi-modal learning* (capturing the relationship between representations in different modalities)

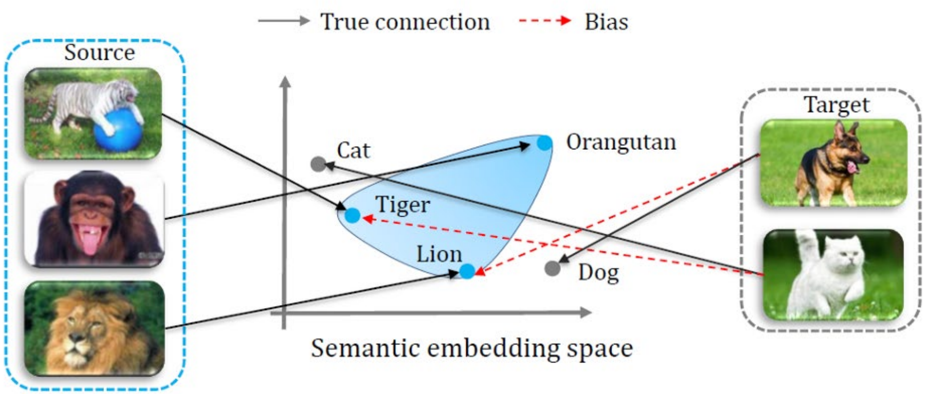
- In zero-shot classification we want to recognize objects from classes that the model has not seen during training.

Available data:

i. **Seen classes** (with labels)
ii. **Unseen classes**: (no labels available during training)
iii. **Auxiliary information**: descriptions/semantic attributes/word embeddings for both seen and unseen classes at train time - a bridge between seen and unseen classes.



Zero-shot learning



Available data:

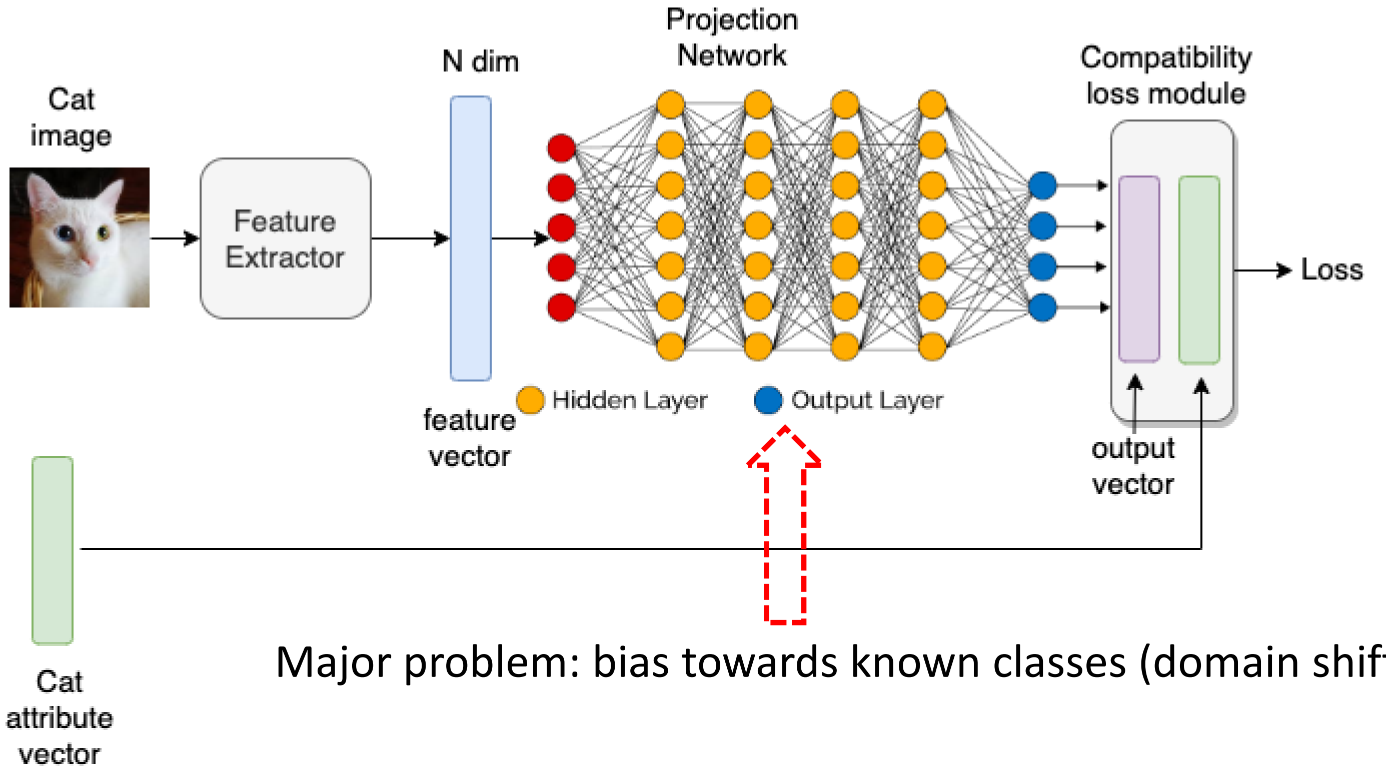
i. **Seen classes** (with labels)

ii. **Unseen classes**: (no labels available during training)

iii. **Auxiliary information**: descriptions/semantic attributes/word embeddings for both seen and unseen classes at train time - a bridge between seen and unseen classes.

Zero-shot learning

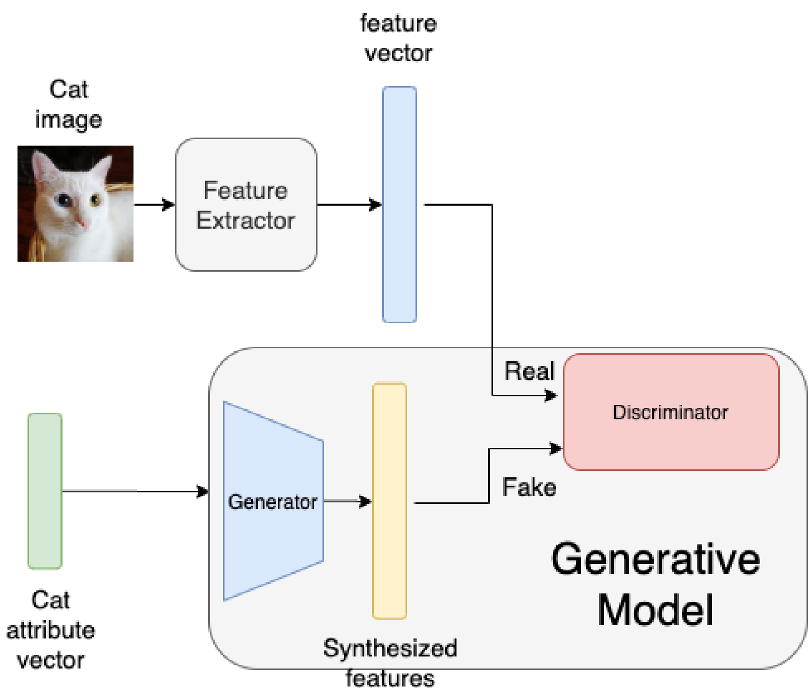
Embedding based methods



Major problem: bias towards known classes (domain shift)

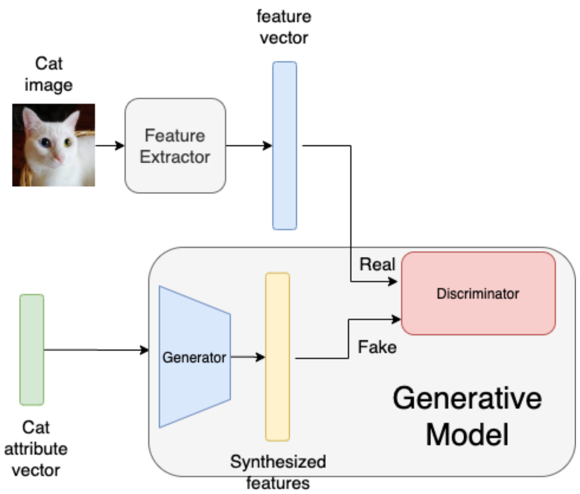
Zero-shot learning

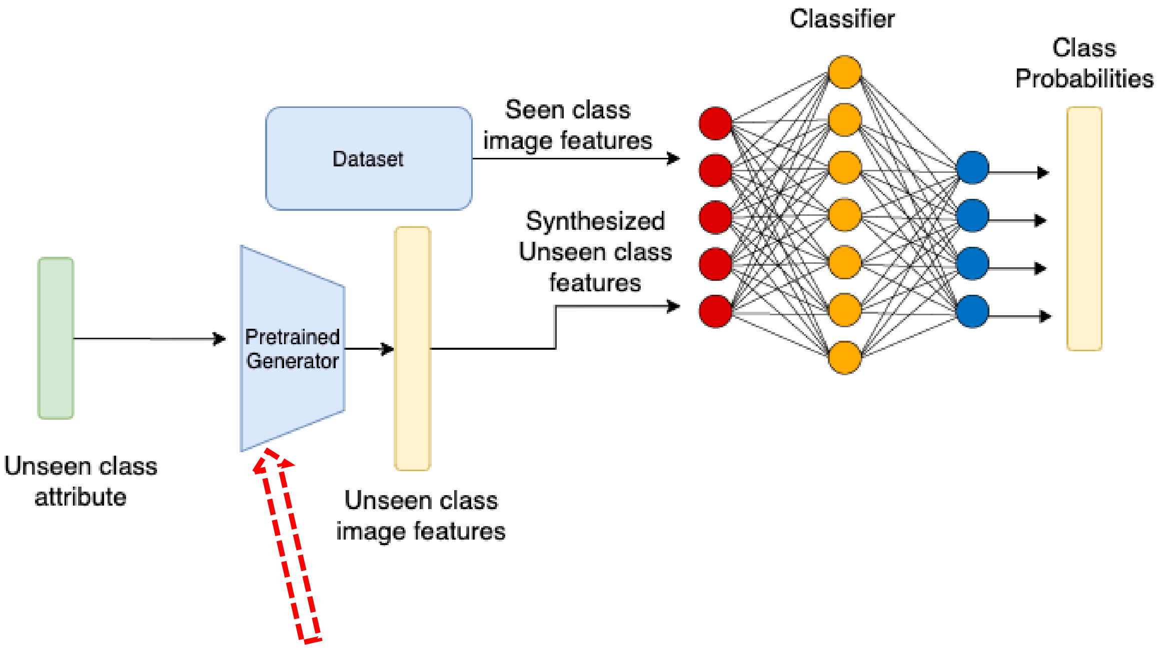
Generative model based methods



Zero-shot learning

Generative model based methods





cross-domain knowledge transfer

+ semantic loss problem

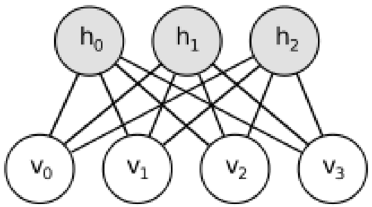
Representation learning in deep models

General philosophy and approach

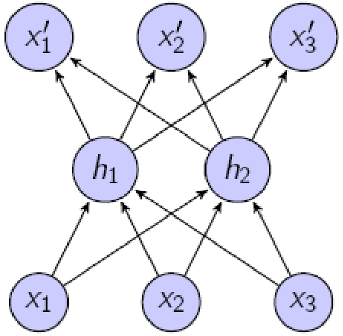
- supervised vs unsupervised
- the concept of unsupervised *pre-training*
- *probabilistic models* (latent variables that describe the distribution) vs *direct encoding* (learning a parametric map from input to representation)
- historically important *manifold learning* and *generative* models
- growing interest in *unsupervised* representation learning with *generative*, *contrastive losses* and *self-supervised learning* approaches

Representation learning in deep models

- Pre-training for fundamental computational blocks
  - regularized Boltzmann machines (RBMs)

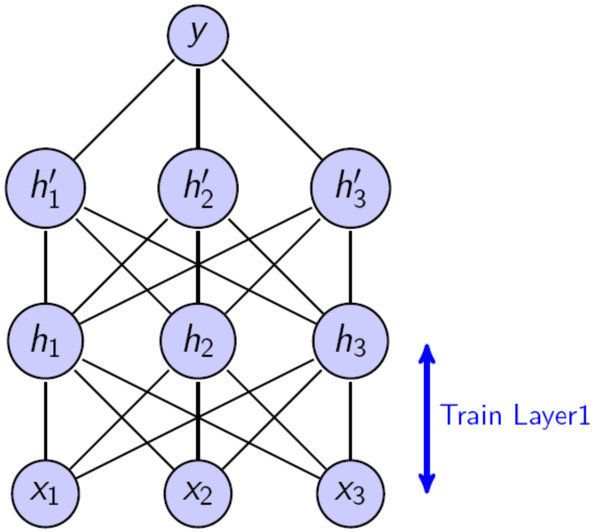


  - autoencoders

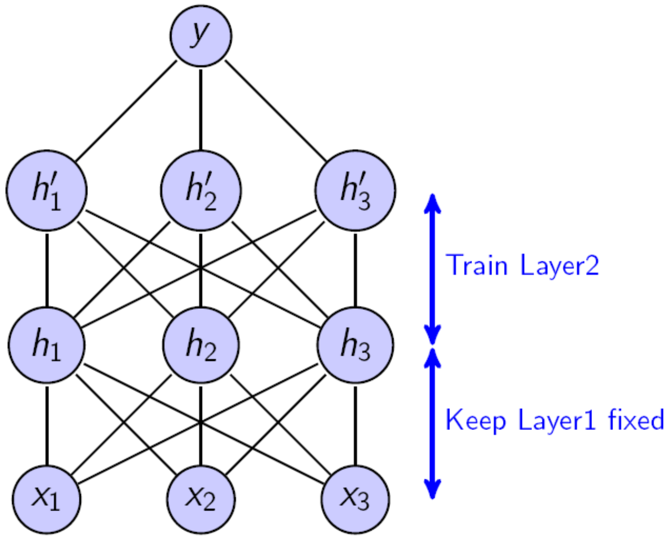


Representation learning in deep models

- The concept of layer-by-layer pretraining
  - greedy layer-wise unsupervised representation learning



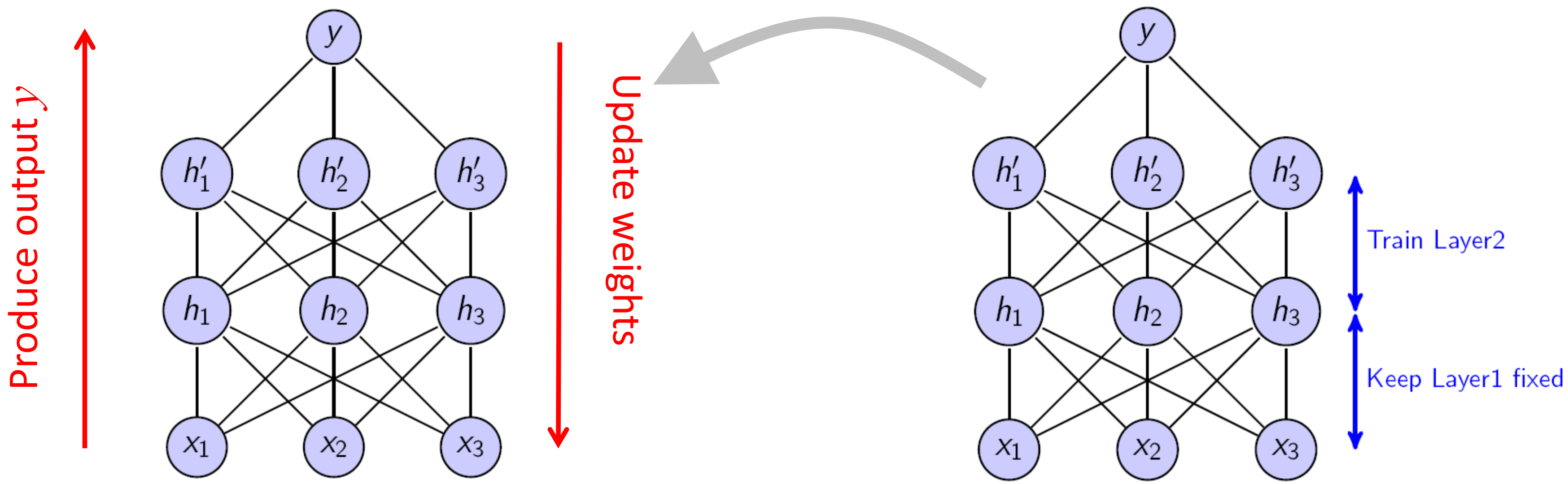
Single layer at a time



Train another layer while keeping the lower layer fixed

Representation learning in deep models

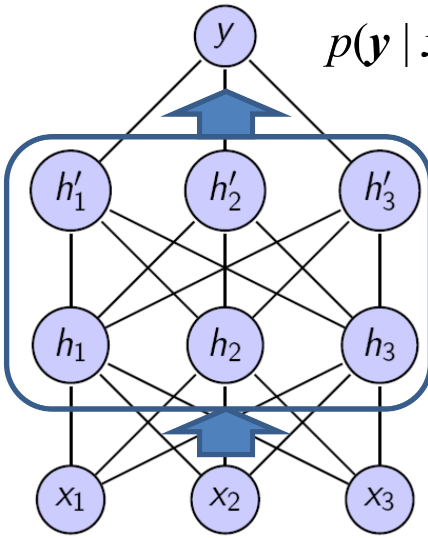
- The concept of layer-by-layer pretraining
  - greedy layer-wise unsupervised representation learning
    - intuitively, learning about the input distribution should help in learning the *mapping* between the input and output space
    - BUT having two separate phases has *disadvantages*



Representation learning in deep models

- The concept of layer-by-layer pretraining
  - greedy layer-wise unsupervised representation learning
    - RBMs, autoencoders
    - leads to lower test classification error
    - pretraining as an initialisation scheme
      - prior to supervised fine-tuning
      - initialisation for other unsupervised algorithms such as DBM, DBN etc.
    - *optimisation* vs *regularisation* hypothesis
      - lower variance in learning, less risk for overfitting
      - as a regulariser, it urges the learning algorithm to discover **features that explain *underlying causes* that generate the data** (also, causal factors often remain *invariant*)

$p(\boldsymbol{y} \mid \boldsymbol{x})$



$p(\boldsymbol{h}|\boldsymbol{x}) = p(\boldsymbol{x}|\boldsymbol{h})\, p(\boldsymbol{h})$

Ideally:

$p(\boldsymbol{h}) = \prod_i p(h_i)$

marginal factor independence

$p(\boldsymbol{x})$

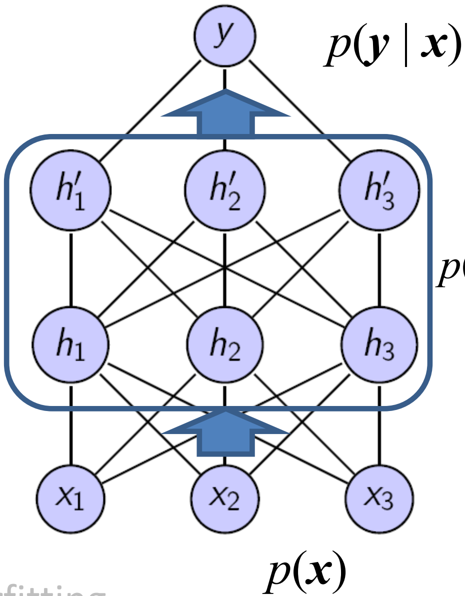
Representation learning in deep models

- The concept of layer-by-layer pretraining
  - greedy layer-wise unsupervised representation learning

Representation learning should strive towards uncovering latent factors, $\boldsymbol{h}$, which capture underlying causes in $\boldsymbol{x}$.

Then, if $\boldsymbol{y}$ is one of them, i.e. $\boldsymbol{y}=h_i$, it should be easy to learn to predict $\boldsymbol{y}$ from this representation.

<u>So, how to make representation encode relevant/salient factors?</u>



$p(\boldsymbol{h}|\boldsymbol{x})=p(\boldsymbol{x}|\boldsymbol{h})\ p(\boldsymbol{h})$

Ideally:

$p(\boldsymbol{h}) = \prod_i p(h_i)$

marginal factor independence

erfitting

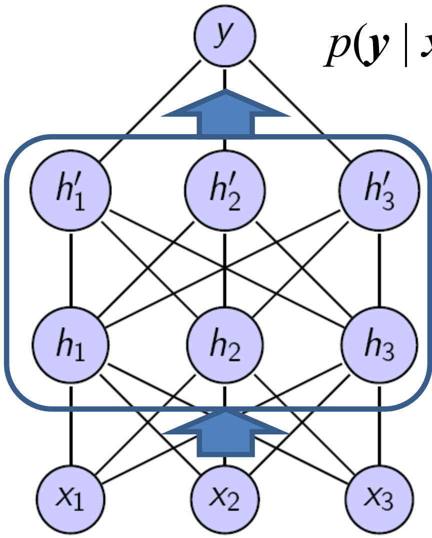
– as a regulariser, it urges the learning algorithm to discover **features that explain *underlying causes* that generate the data** (also, causal factors often remain *invariant*)

Representation learning in deep models

- The concept of layer-by-layer pretraining
  - greedy layer-wise unsupervised representation learning

<u>So, how to make representation encode relevant/salient factors?</u>

1) Guide unsupervised pretraining with a supervised learning signal (e.g. autoencoders).
2) Rely on massive representations with purely unsupervised learning (e.g. RBMs).
3) Redefine the meaning of salience.



Ideally:

$p(\boldsymbol{h}) = \prod_i p(h_i)$

marginal factor independence

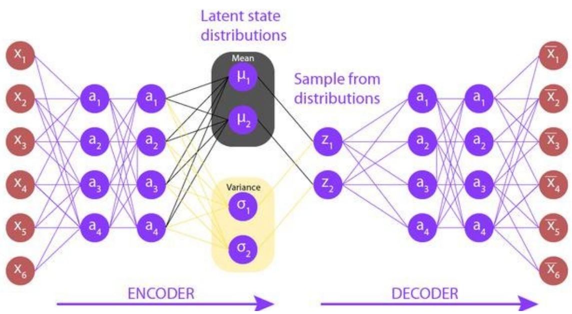
erfitting

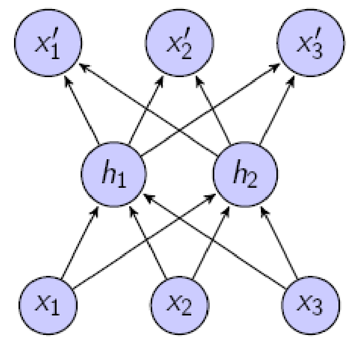
– as a regulariser, it urges the learning algorithm to discover **features that explain *underlying causes* that generate the data** (also, causal factors often remain *invariant*)

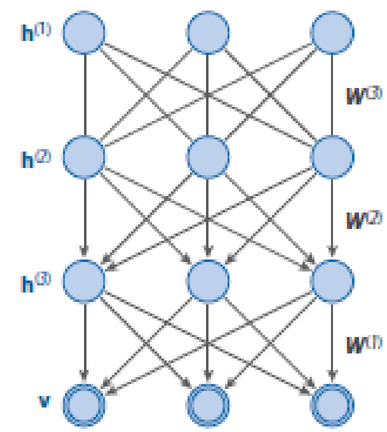
Representation learning in deep models

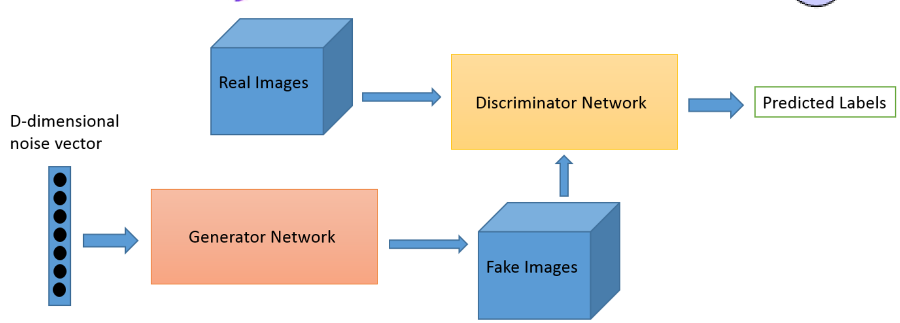
DNN architectures of special interest

- stacked autoencoders (manifold learning)
- generative models: DBNs, DBMs or Generative Adversarial Networks (GANs), Variational Autoencoders (VAE)



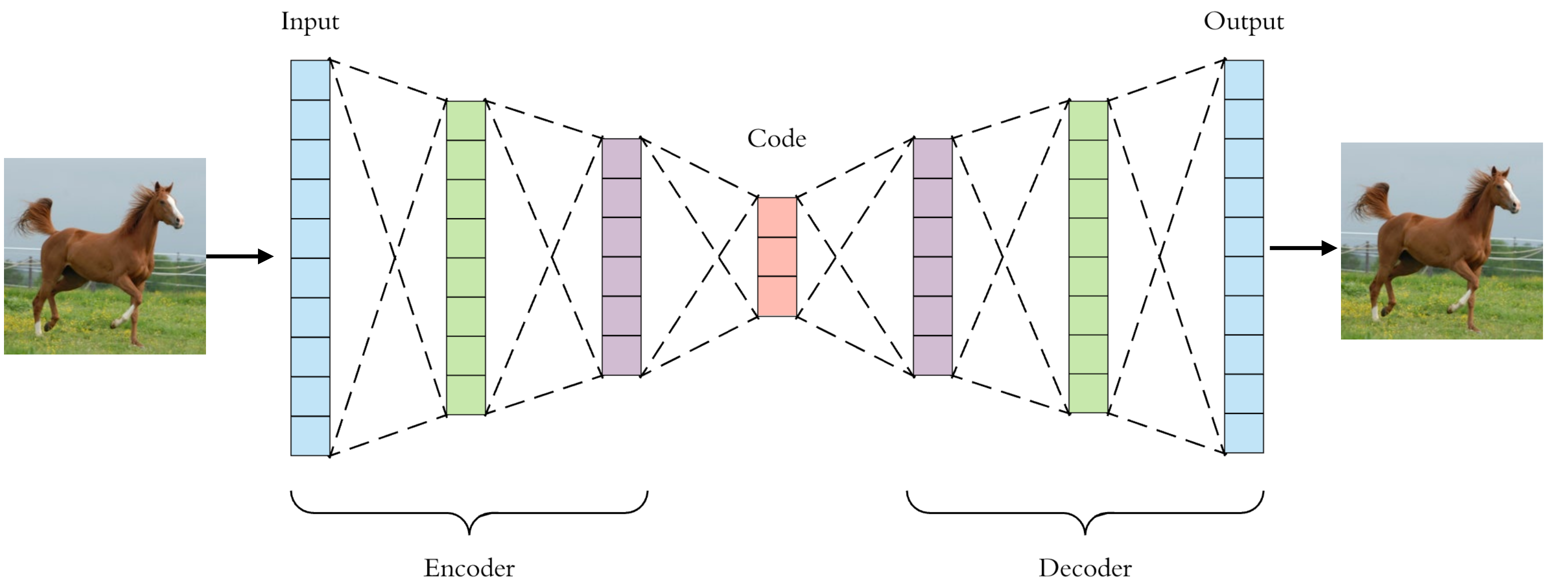






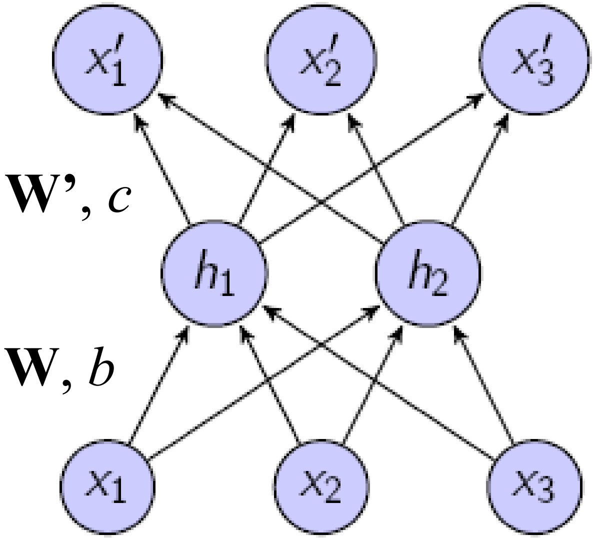
Introduction to autoencoders

- The objective is to extract/learn representation (latent code) of data without any labels



Traditionally, these latent representations are expected to be **low-dimensional** in the spirit of *dimensionality reduction* (compressed knowledge – *inf. bottleneck*).

Introduction to autoencoders



Decoder: $\boldsymbol{x}' = d(\boldsymbol{h})$, e.g. $\boldsymbol{x}' = \sigma(\mathbf{W}'\boldsymbol{h} + c)$

Encoder: $\boldsymbol{h} = e(\boldsymbol{x})$, e.g. $\boldsymbol{h} = \sigma(\mathbf{W}\boldsymbol{x} + b)$

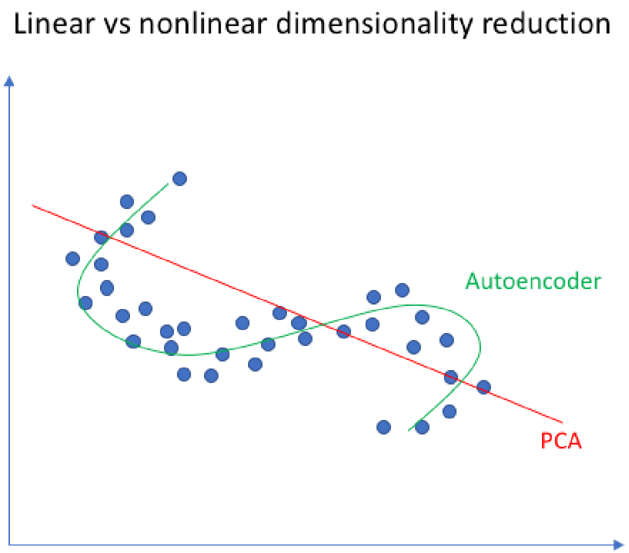
Encourage $\boldsymbol{h}$ to produce low reconstruction error (loss L) – training with backprop

Reconstruction $\boldsymbol{x}' = d(e(\boldsymbol{x}))$, e.g. $\boldsymbol{x}' = \sigma(\mathbf{W}'\sigma(\mathbf{W}\boldsymbol{x} + b) + c)$

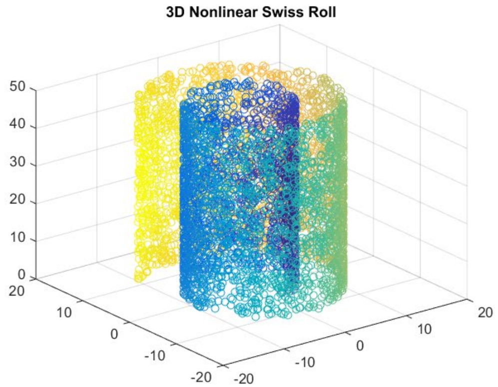
Loss $L(\boldsymbol{x}, d(e(\boldsymbol{x}))$, e.g. $L = \frac{1}{M}\sum_m \left\| \boldsymbol{x}^{(m)} - d(e(\boldsymbol{x}^{(m)}) \right\|_2$

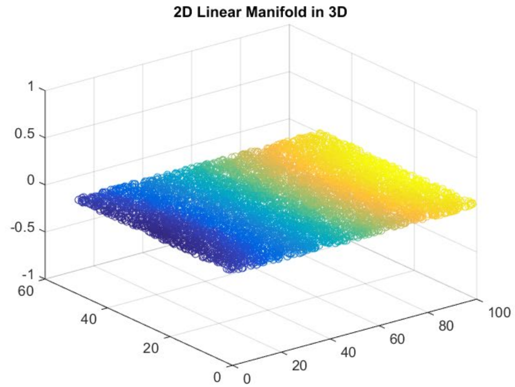
Dimensionality reduction with autoencoders

Learning nonlinear manifolds, dimensionality reduction



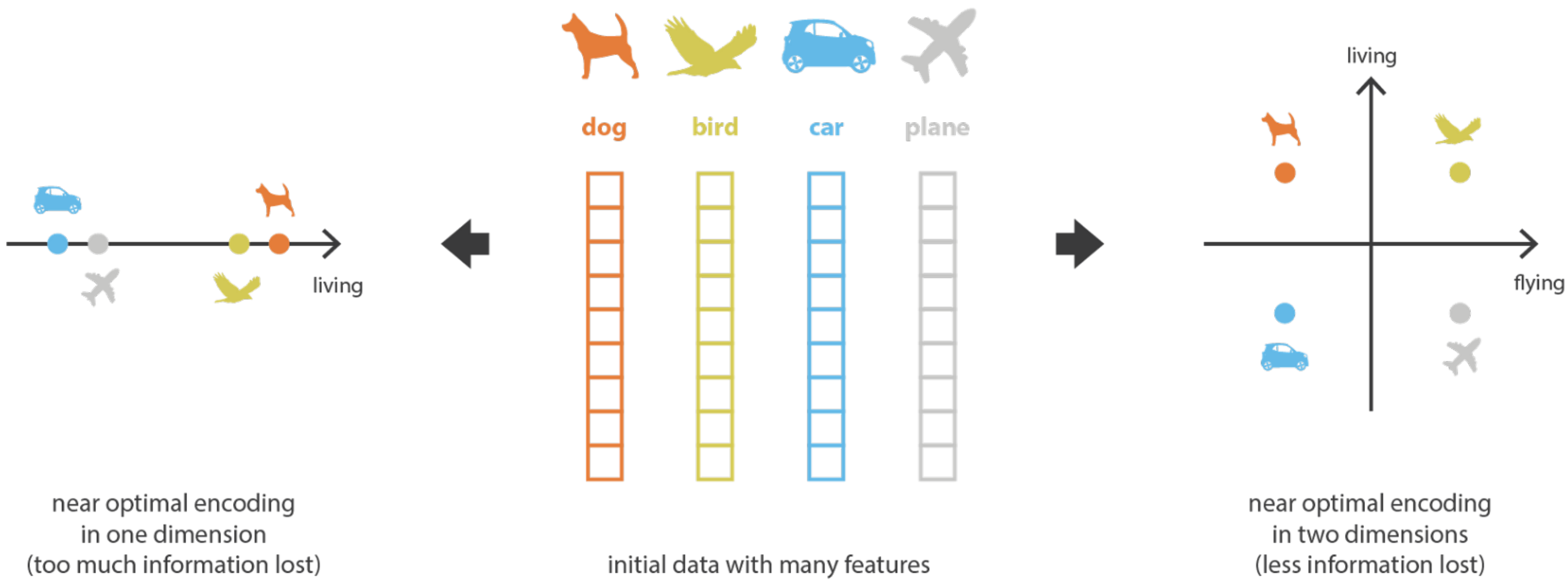
- Key assumption: there is low-dimensional structure of the data
- Balance in sensitivity to data between building accurate reconstructions and preventing from memorizing (overfitting)





Dimensionality reduction with autoencoders

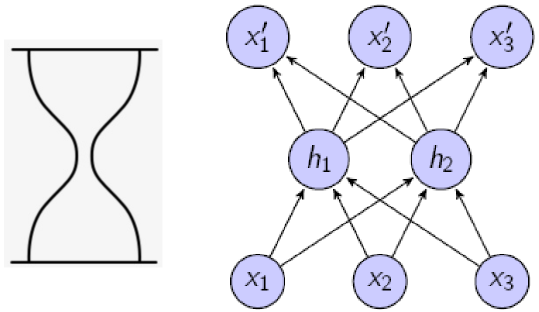
What is a suitable dimensionality?



adapted from J. Jordan

Undercomplete autoencoders

- Information bottleneck is realised by the lower dimensionality of the hidden layer than that of input & output (*undercomplete autoencoders*)
- So, no need for explicit regularization term
- For deep autoencoders, we must be aware of the capacity of the encoder/decoder to avoid highly-nonlinear mapping and memorizing/overfitting



hour-glass architecture

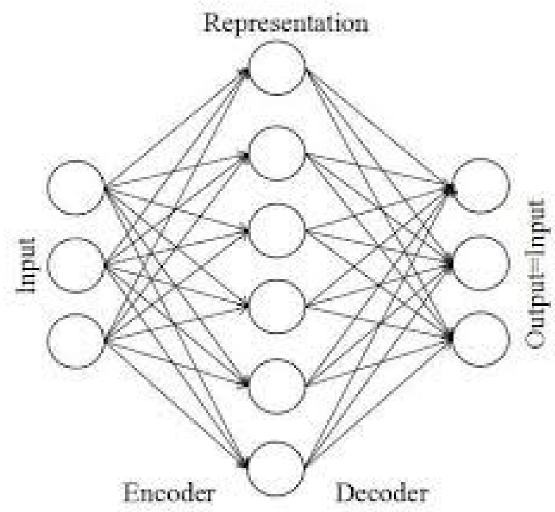
- For complex (highly nonlinear) manifolds we might have to go to higher dimensionalities -> *overcomplete* autoencoders

Overcomplete autoencoders

Overcomplete regularised autoencoders

- larger hidden layer size than that of the input and output
- the need for explicit regularisation (to avoid overfitting - copying input to the output):
  Reconstruction loss + *Regularisation* penalty

$$L\left(x,d\left(h\right)\right)+\Omega(h),\quad h=e(x)$$

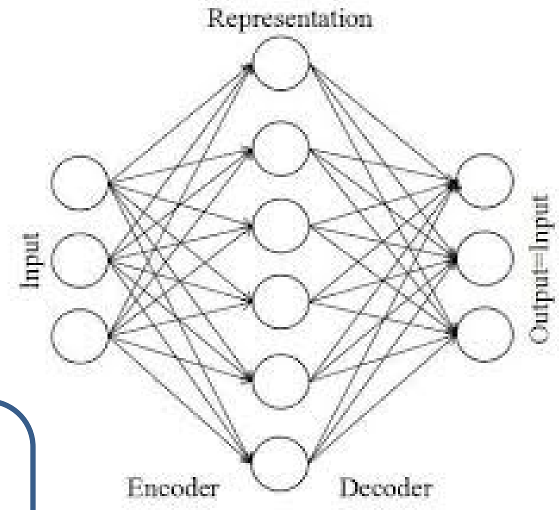


Overcomplete autoencoders

Overcomplete regularised autoencoders

- larger hidden layer size than that of the input and output
- the need for explicit regularisation (to avoid overfitting - copying input to the output):
  Reconstruction loss + *Regularisation* penalty

$$L\left(x, d\left(h\right)\right) + \Omega(h), \quad h = e(x)$$



When training autoencoders there is a **compromise**

I. Need to approximately recover $\boldsymbol{x}$ – *reconstruction* force

II. Need to satisfy the regularization term – *regularisation* force.

Sparse autoencoders

- Penalizing non-sparse solutions can be seen as maximum likelihood training of a model with latent variables (and the model's prior over latent variables $p_{\text{model}}(\boldsymbol{h})$ – different from a typical prior over the model's parameters, e.g. weights)

$$\log p_{\text{model}}(\boldsymbol{h},\boldsymbol{x}) = \log p_{\text{model}}(\boldsymbol{h}) + \log p_{\text{model}}(\boldsymbol{x} \mid \boldsymbol{h})$$

for example: $p_{\text{model}}(\boldsymbol{h}) = \prod_i \frac{\lambda}{2} e^{-\lambda |h_i|} \Rightarrow \boxed{\Omega(\boldsymbol{h}) = \lambda \sum |h_i|}$ *L1-regularisation*

Sparse autoencoders

- Penalizing non-sparse solutions can be seen as maximum likelihood training of a model with latent variables (and the model's prior over latent variables $p_{\text{model}}(\boldsymbol{h})$ – different from a typical prior over the model's parameters, e.g. weights)

$$\log p_{\text{model}}(\boldsymbol{h},\boldsymbol{x}) = \log p_{\text{model}}(\boldsymbol{h}) + \log p_{\text{model}}(\boldsymbol{x} \mid \boldsymbol{h})$$

Kullback-Leibler divergence regularisation

or:

$$\boxed{\Omega(\boldsymbol{h}) = \sum_i^{|\boldsymbol{h}|} \rho \log \frac{\rho}{\hat{\rho}_i} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_i} = \sum_i^{|\boldsymbol{h}|} \mathrm{KL}\left(\rho \,\|\, \hat{\rho}_i\right)}$$

$$\hat{\rho}_i = \frac{1}{m} \sum_j^m h_i(\boldsymbol{x}^{(j)})$$

with a sparsity parameter ρ, e.g. $\rho = 0.05$

Sparse autoencoders

- Penalizing non-sparse solutions can be seen as maximum likelihood training of a model with latent variables (and the model's prior over latent variables $p_{\text{model}}(\boldsymbol{h})$ – different from a typical prior over the model's parameters, e.g. weights)

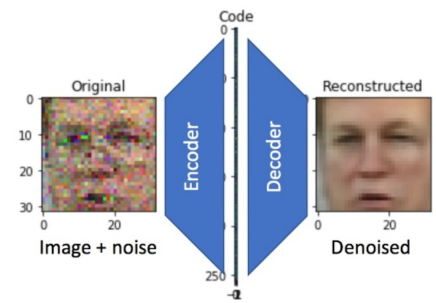
$$\log p_{\text{model}}(\boldsymbol{h},\boldsymbol{x}) = \log p_{\text{model}}(\boldsymbol{h}) + \log p_{\text{model}}(\boldsymbol{x} \mid \boldsymbol{h})$$

Sparse autoencoder selectively activates parts of the network depending on the input samples (thus avoiding the memorisation) unlike undercomplete autoencoder that relies on the entire network for every input samples.

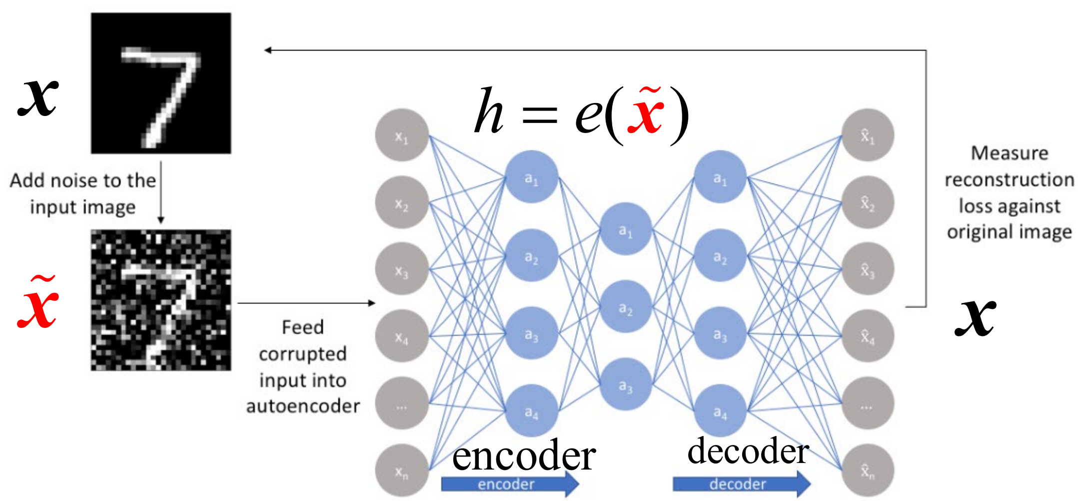
Denoising autoencoders

$$L\left(\boldsymbol{x}, d\left(e(\color{red}{\tilde{\boldsymbol{x}}})\right)\right), \quad h = e(\color{red}{\tilde{\boldsymbol{x}}})$$

Corrupted copy of $\boldsymbol{x}$

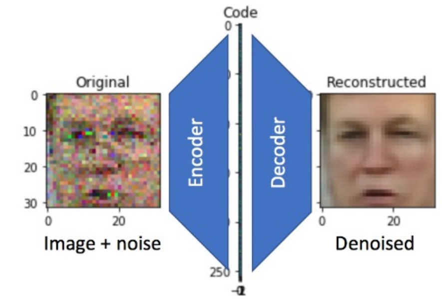


Autoencoders have to undo this corruption beyond simply coping the input.



Denoising autoencoders

$$L\left(\boldsymbol{x}, d\left(e(\tilde{\boldsymbol{x}})\right)\right), \quad h = e(\tilde{\boldsymbol{x}})$$



Corrupted copy of $\boldsymbol{x}$

Autoencoders have to undo this corruption beyond simply coping the input.

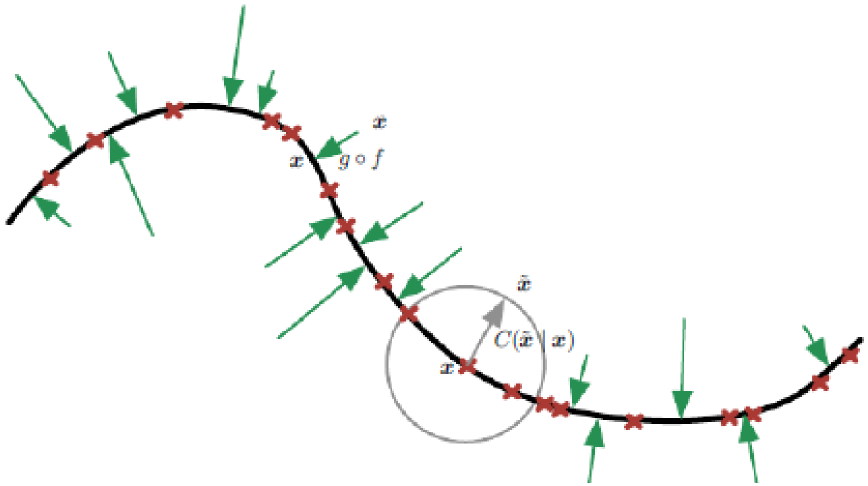
1. A training sample is sampled from the training data.
2. A corrupted version of the sample $\boldsymbol{x}$ is drawn from some corruption process

$$C(\tilde{\boldsymbol{x}} \mid \boldsymbol{x} = \boldsymbol{s})$$

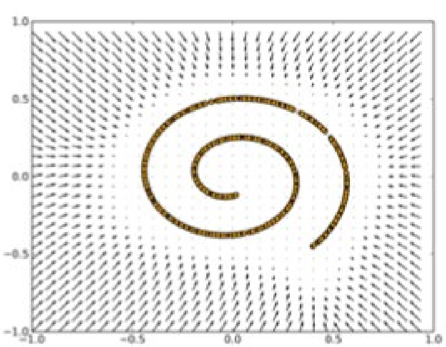
3. $(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ is used as a training sample to estimate the autoencoder's reconstruction distribution $\quad p_{reconstruction}(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) = p_{decoder}(\boldsymbol{x} \mid \boldsymbol{h}), \quad \boldsymbol{h} = e(\tilde{\boldsymbol{x}})$

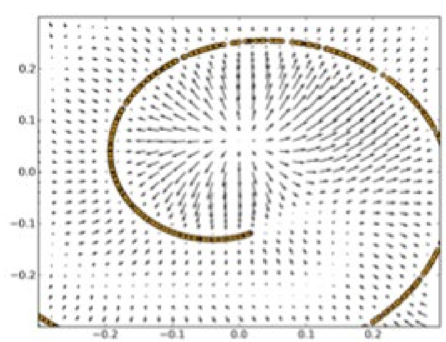
Denoising autoencoders

Learning a *vector field* around a *low-dimensional manifold . . .*



. . . with the principle that only the variations tangent to the manifold around $\boldsymbol{x}$ should be accounted for by changes in $\boldsymbol{h}$



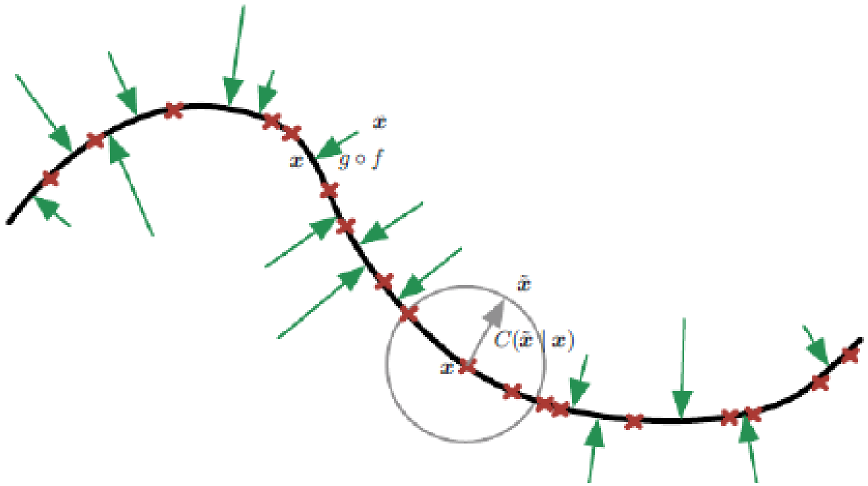


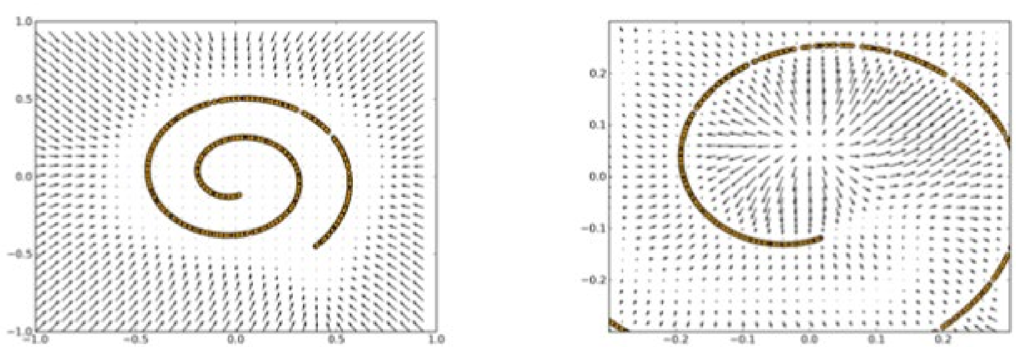
vector field pushing towards the data distribution on the manifold

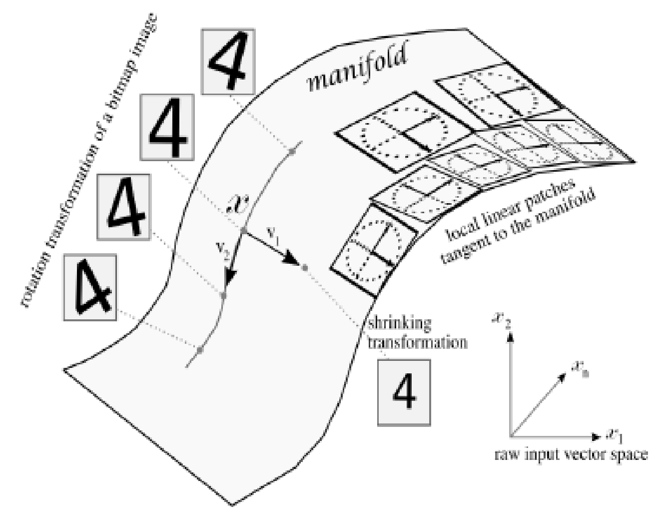
Goodfellow et al.

Denoising autoencoders

Learning a *vector field* around a *low-dimensional manifold . . .*







Goodfellow et al.

Contractive autoencoders

- Contractive autoencoders (CAE) are regularized to resist small (local) perturbations of the input

$$\Omega(\boldsymbol{h}) = \lambda \left\| \frac{\partial e(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_F^2$$

F – Frobenius norm: sum of squared components

  - encouraged to map a neighbourhood of input samples to a small neighbourhood in the output space (warping space)
  - learning the local manifold structure of the data
  - Learning is heavy for deep CAE so it may be better to greedily stack shallow CAEs on top of each other (in the same way as we develop stacked autoencoders)

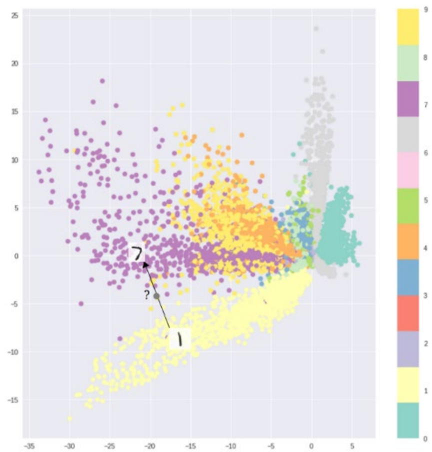
Generative modelling

- Unsupervised learning
  - no labels
  - the objective is to capture the hidden structure in the data
  - density estimation, clustering, feature learning, dimensionality reduction
- Generative approach allows even for generating samples
  - probabilistic in nature: learning **the joint $P(x, y)$** -> ambitious
  - training data $\sim P_{data}(x)$ -> generated samples $\sim P_{model}(x)$
  - capable for uncovering underlying latent variables
  - could be used for many purposes, e.g. debiasing or outlier detection
  - some flagship examples of deep latent variable models: DBN, GAN, VAE

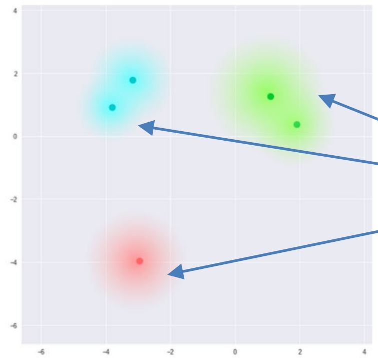
Variational autoencoders (VAE) - motivation

- The expectation from the generative model
  - capability to generate new samples from the learned distribution

MNIST example



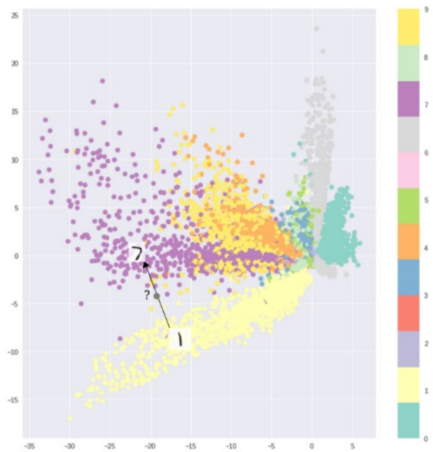
Classical autoencoders (that are not generative models) offer latent space that is often *discontinuous* and does *not* allow *easy interpolation* (it is not well organised).



distinct and isolated islands in the latent space

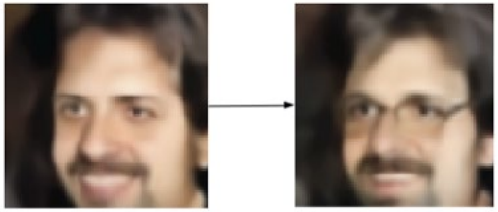
Variational autoencoders (VAE) - motivation

- The expectation from the generative model
  - capability to generate new samples from the learned distribution



Classical autoencoders (that are not generative models) offer latent space that is often *discontinuous* and does *not* allow *easy interpolation* (it is not well organised).

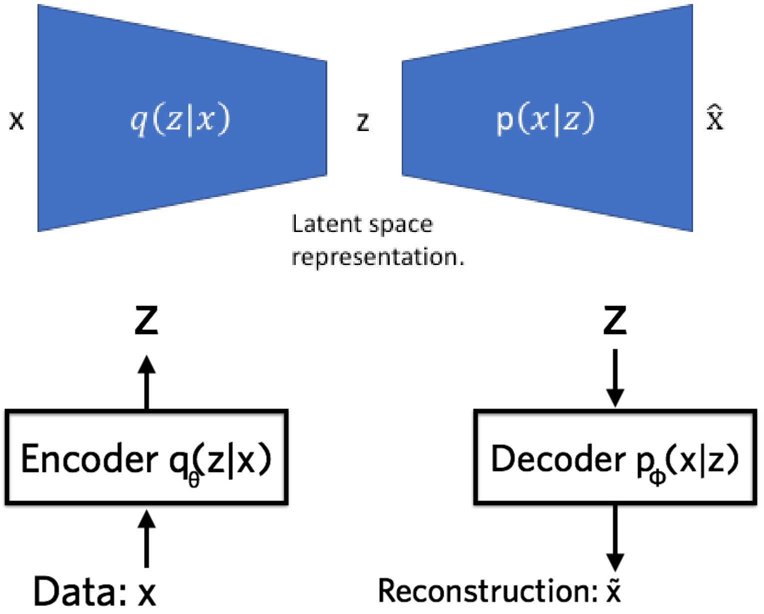
  - alter and explore variations on the existing data

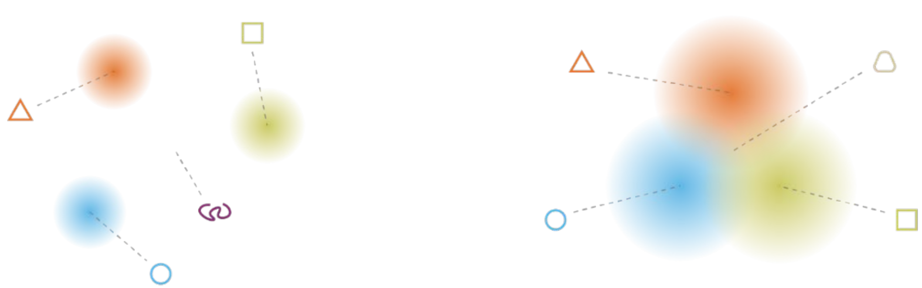


VAE mechanics

VAE is a probabilistic twist on an autoencoder with *regularised* training:

- ➢ to minimize the risk of overfitting
- ➢ to ensure "nice" properties of the latent space: regularity, continuity, completeness.



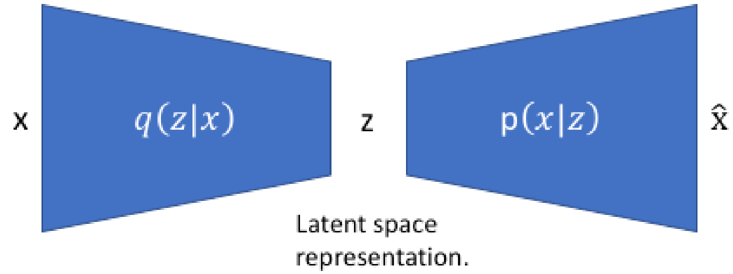


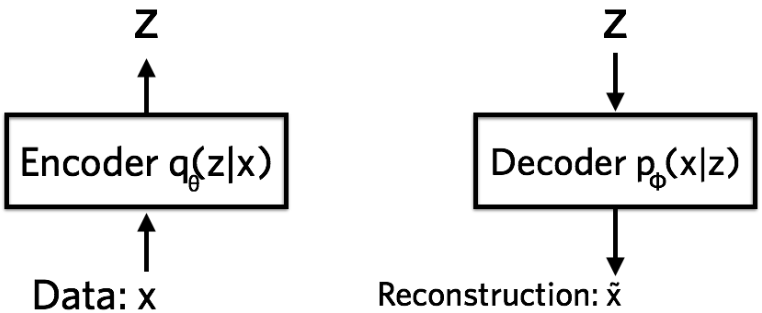
"bad" latent space "good" properties of the latent space

VAE mechanics

VAE is a probabilistic twist on an autoencoder with *regularised* training:

- to minimize the risk of overfitting
- to ensure "nice" properties of the latent space: regularity, continuity, completeness.





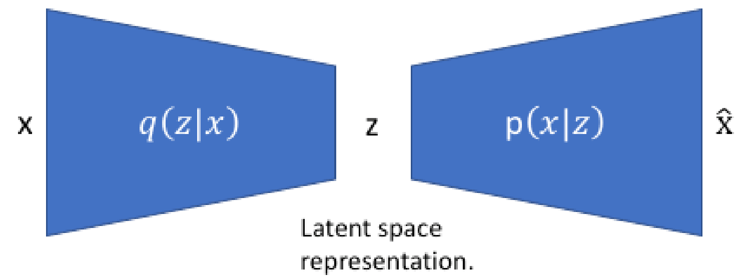
VAE replaces *deterministic* hidden layer (latent) z with *stochastic* sampling:

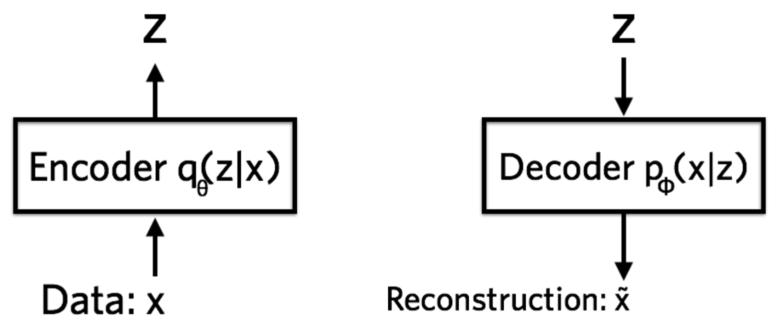
$$\text{input } x \;\rightarrow\; \text{latent space } p(z\,|\,\boldsymbol{x}) \;\rightarrow\; \text{sampling } z \sim p(z\,|\,\boldsymbol{x}) \;\rightarrow\; \text{input reconstr. } \tilde{\boldsymbol{x}} = d(z)$$

VAE mechanics

VAE is a probabilistic twist on an autoencoder with *regularised* training:

- to minimize the risk of overfitting
- to ensure "nice" properties of the latent space: regularity, continuity, completeness.

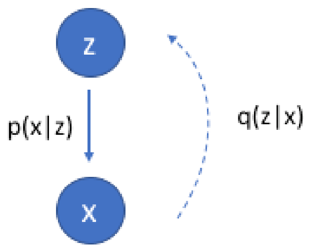


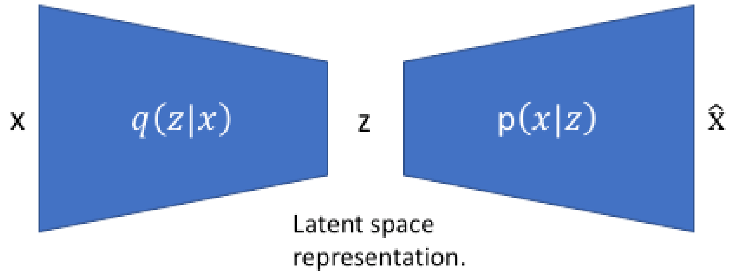


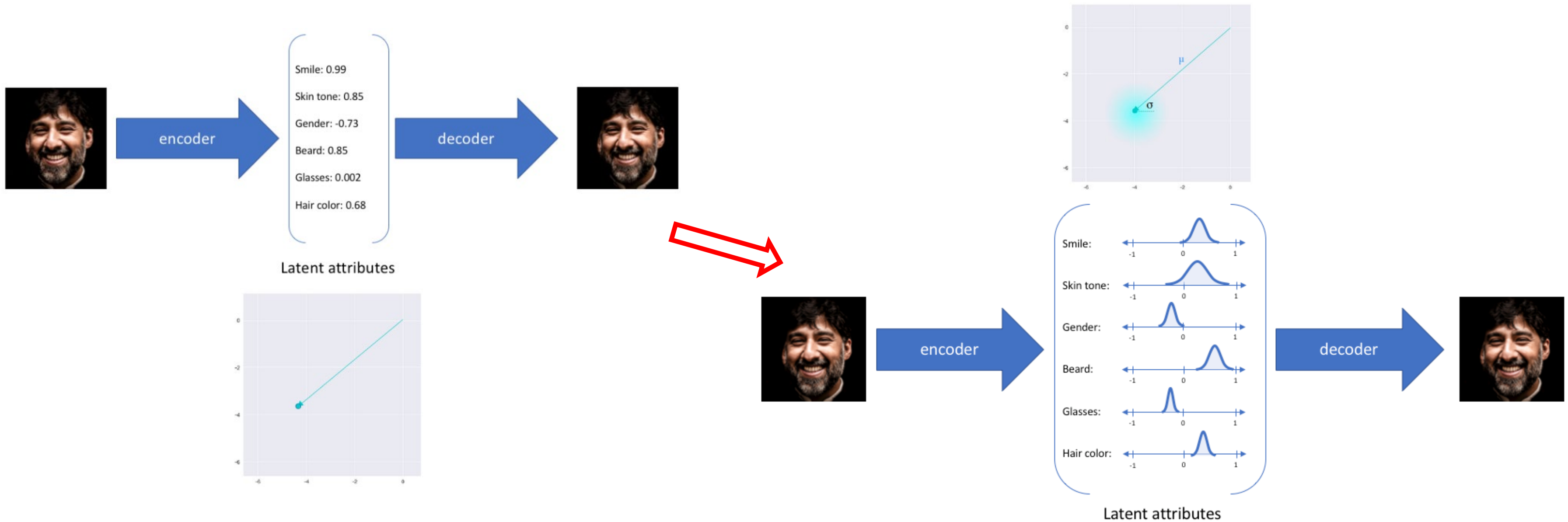
VAE replaces *deterministic* hidden layer (latent) z with *stochastic* sampling:

$$\text{input } x \;\rightarrow\; \underbrace{\text{latent space } p(\boldsymbol{z}\,|\,\boldsymbol{x}) \;\rightarrow\; \text{sampling } z \sim p(\boldsymbol{z}\,|\,\boldsymbol{x})}_{\text{in a } \textit{classical} \text{ autoencoder: } \boldsymbol{z} = e(\boldsymbol{x})} \;\rightarrow\; \text{input reconstr. } \tilde{\boldsymbol{x}} = d(\boldsymbol{z})$$

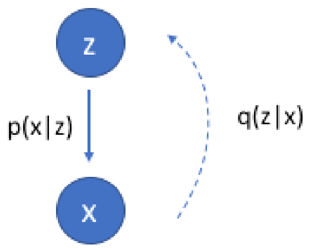
Intuition behind a probabilistic “twist” on AE

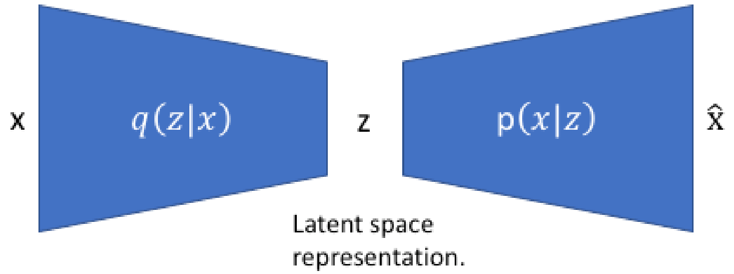






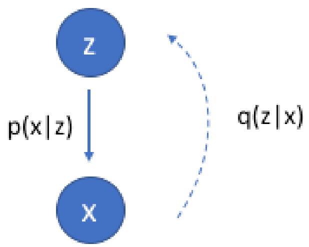
VAE mechanics

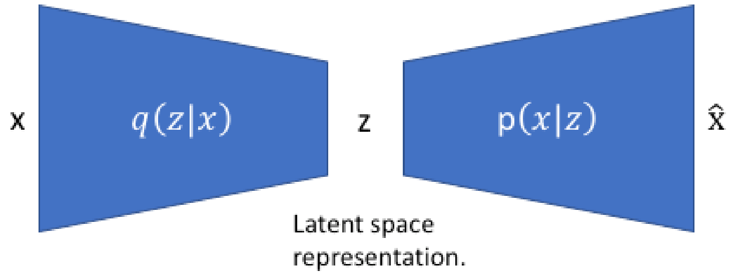




The aim is to estimate a probabilistic decoder: $p(z|x) = \frac{p(x|z)p(z)}{p(x)} = \frac{p(x|z)p(z)}{\int p(x|u)p(u)du}$

VAE mechanics





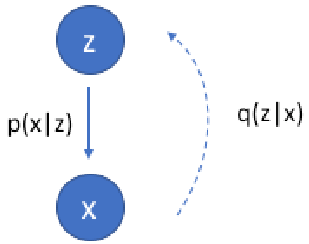
The aim is to estimate a probabilistic decoder: $p(z|x) = \frac{p(x|z)p(z)}{p(x)} = \frac{p(x|z)p(z)}{\int p(x|u)p(u)du}$

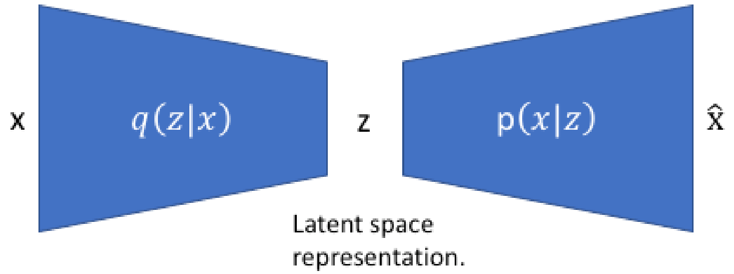
hard to estimate

We need to resort to a variational approach with some $q(z|x)$ and minimise KL divergence:

$$\min \mathrm{KL}\big(q(z\,|\,x)\,\|\,p(z\,|\,x)\big)$$

VAE mechanics





The aim is to estimate a probabilistic encoder: $p(z|x) = \frac{p(x|z)p(z)}{p(x)} = \frac{p(x|z)p(z)}{\int p(x|u)p(u)du}$

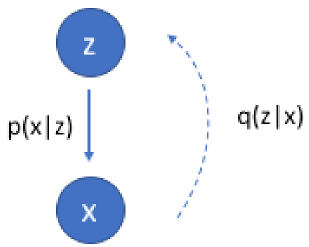
We need to resort to a variational approach with some $q(z|x)$ and minimise KL divergence:

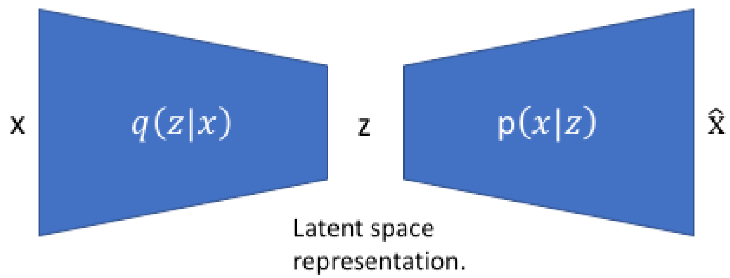
$$\min \mathrm{KL}\left(q(z|x)\,\|\,p(z|x)\right)$$

$$\Downarrow$$

$$\max E_{q(z|x)} \log p(x|z) - \mathrm{KL}\left(q(z|x)\,\|\,p(z)\right)$$

VAE mechanics





The aim is to estimate a probabilistic encoder: $p(z|x) = \frac{p(x|z)p(z)}{p(x)} = \frac{p(x|z)p(z)}{\int p(x|u)p(u)du}$

We need to resort to a variational approach with some $q(z|x)$ and minimise KL divergence:

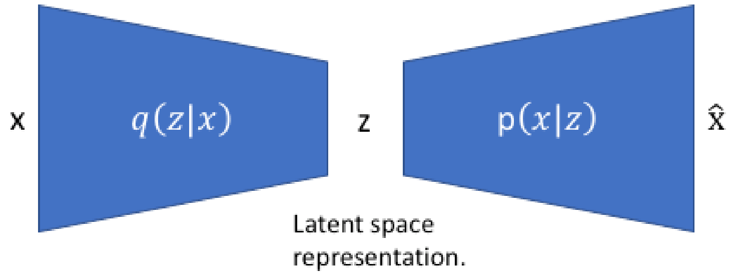
$$\min \mathrm{KL}\left(q(\boldsymbol{z}\,|\,\boldsymbol{x}) \,\|\, p(\boldsymbol{z}\,|\,\boldsymbol{x})\right)$$

$$\Downarrow$$

$$\max E_{q(z|x)} \log p(\boldsymbol{x}\,|\,\boldsymbol{z}) - \mathrm{KL}\left(q(\boldsymbol{z}\,|\,\boldsymbol{x}) \,\|\, p(\boldsymbol{z})\right) \leq \log p(\boldsymbol{x})$$

Lower bound for the likelihood

VAE training – network implementation



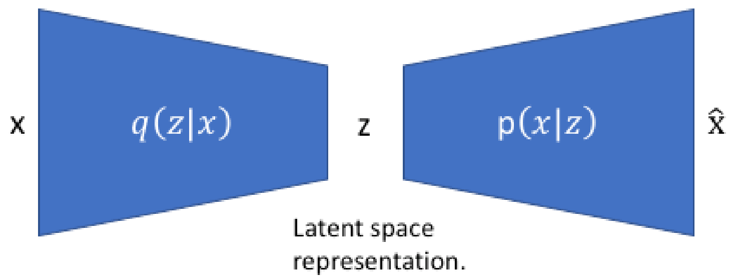
$$\max E_{q(z|x)} \log p(\boldsymbol{x} \mid \boldsymbol{z}) - \mathrm{KL}\left(q(\boldsymbol{z} \mid \boldsymbol{x}) \,\|\, p(\boldsymbol{z})\right)$$

$$\Downarrow$$

Loss function:

$$\min \underbrace{L(\boldsymbol{x}, \tilde{\boldsymbol{x}})}_{\textit{reconstruction}\text{ error}} + \underbrace{\sum_{j}^{|z|} \mathrm{KL}\left(q_j\left(z_j \mid \boldsymbol{x}\right) \,\|\, p\left(z_j\right)\right)}_{\textit{regularisation}}$$

VAE training – network implementation



$$\max E_{q(\boldsymbol{z}|\boldsymbol{x})} \log p(\boldsymbol{x}\,|\,\boldsymbol{z}) - \mathrm{KL}\left(q(\boldsymbol{z}\,|\,\boldsymbol{x}) \,\|\, p(\boldsymbol{z})\right)$$

$$\Downarrow$$

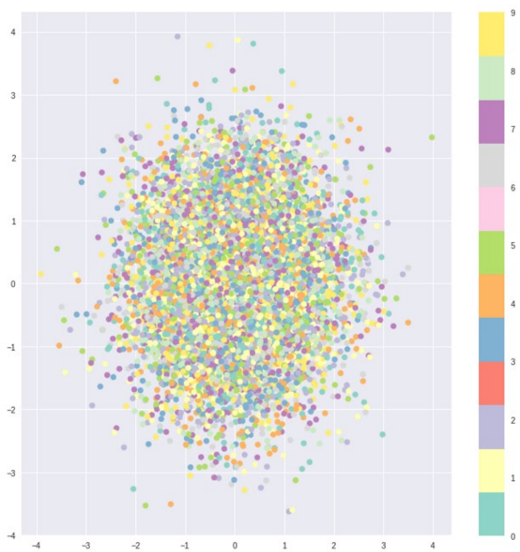
Loss function:

$$\min L(\boldsymbol{x},\tilde{\boldsymbol{x}}) + \sum_{j}^{|z|} \mathrm{KL}\left(q_j(z_j\,|\,\boldsymbol{x}) \,\|\, p(z_j)\right)$$

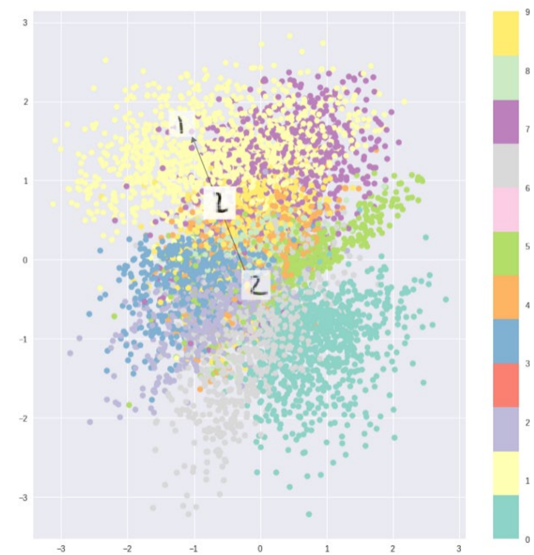
$$z_j \sim \mathrm{N}(\mu_j, \sigma_j) \Longrightarrow \sum_{i=1}^{n} \sigma_i^2 + \mu_i^2 - \log(\sigma_i) - 1$$

The importance of regularization – MNIST example

MNIST example



Only the KL-divergence loss



Reconstruction error + KL-divergence loss

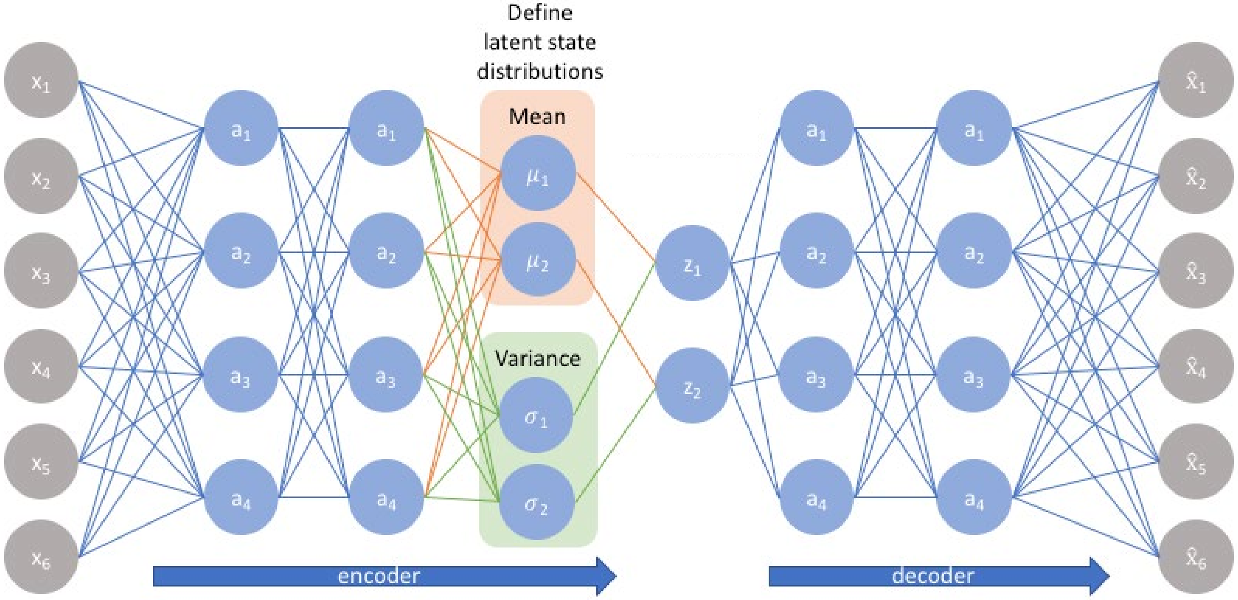
adapted from Irum Shafkat (towardsdatascience)

VAE as a network implementation

- How do we go about this in practice?

Encoder outputs parameters of the distributions

(we assume here that the covariance of $p(z)$ is diagonal)



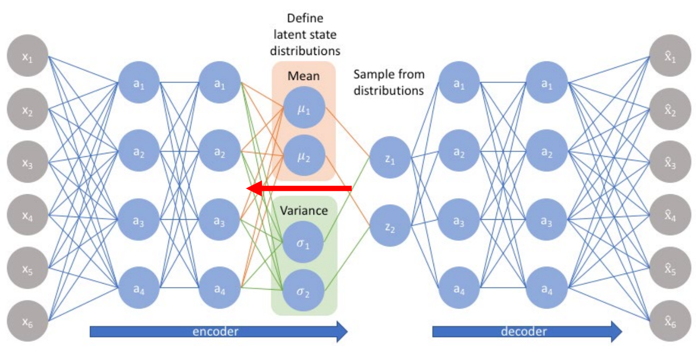
adapted from Jeremy Jordan

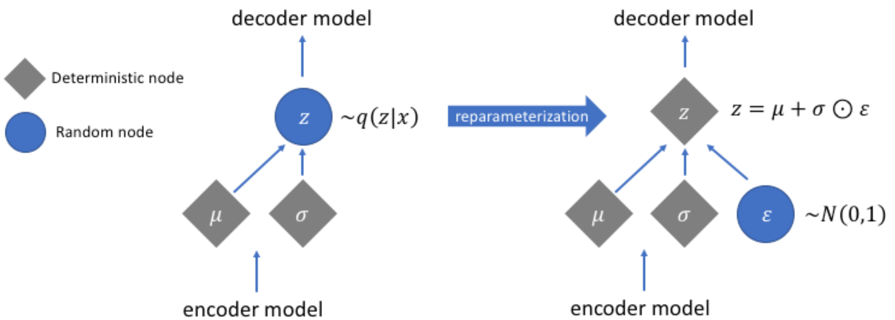
VAE as a network implementation

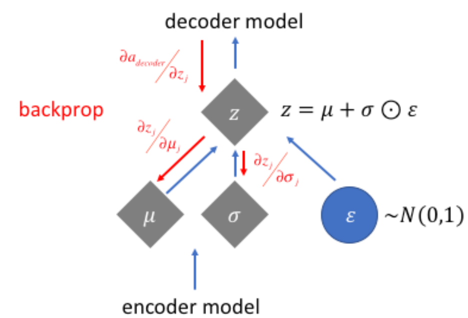
- How do we go about this in practice?

Reparameterisation ***trick*** to propagate gradients wrt. deterministc variables

backpropagation of gradients

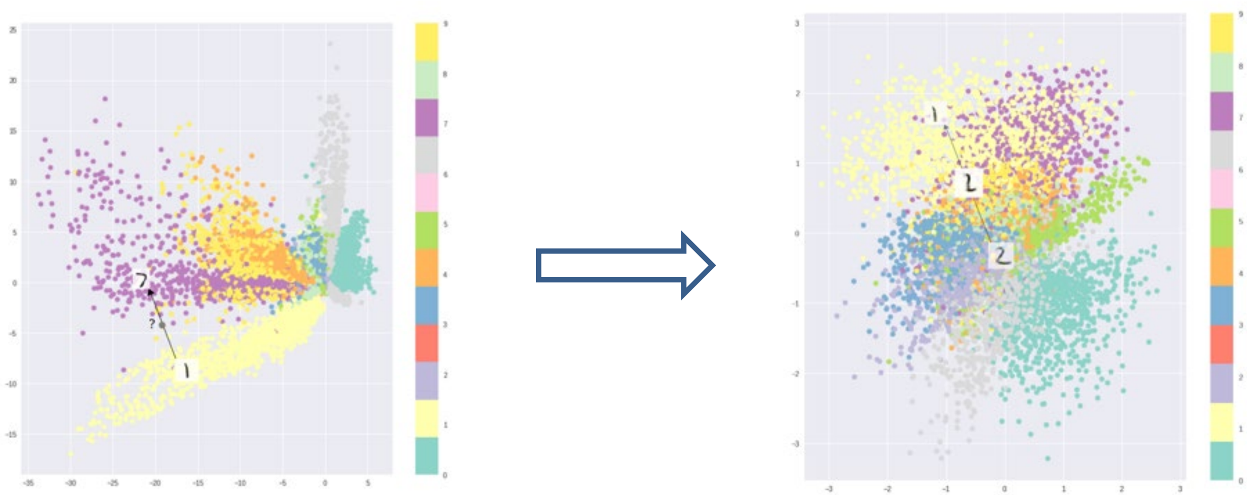






adapted from Jeremy Jordan

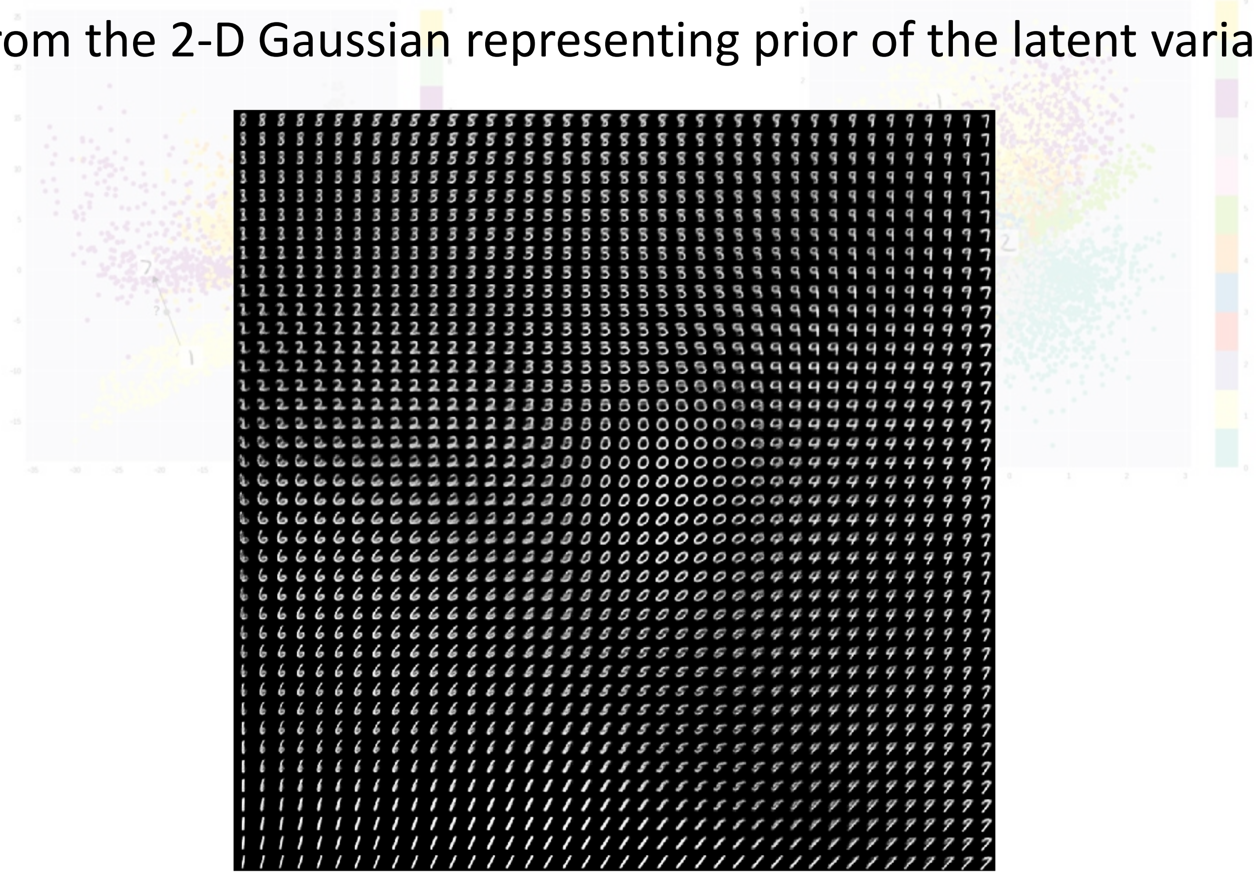
VAE implications



adapted from J. Jordan (towards data science)

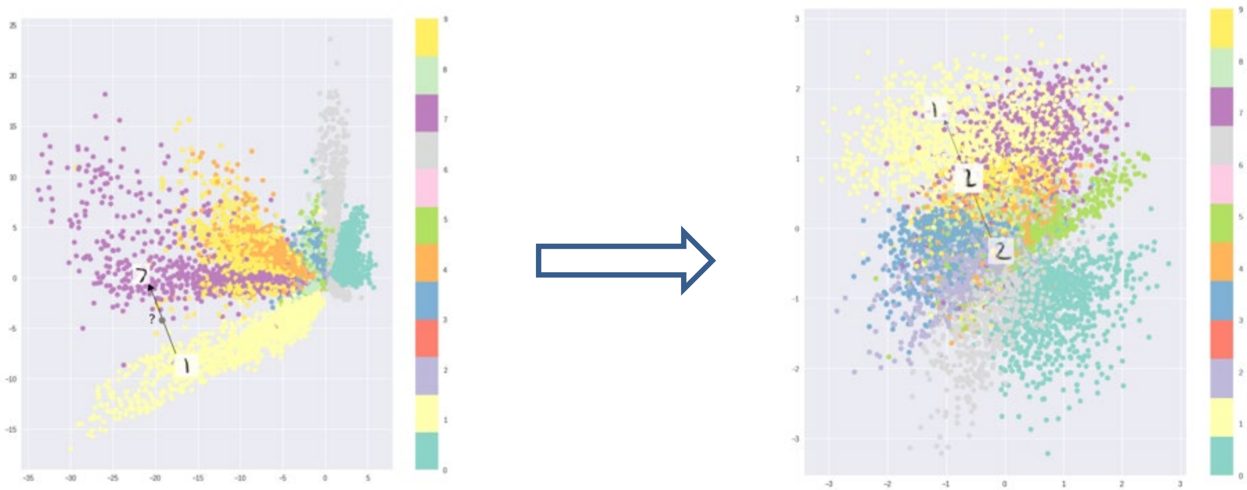
VAE implications

Sampling from the 2-D Gaussian representing prior of the latent variables

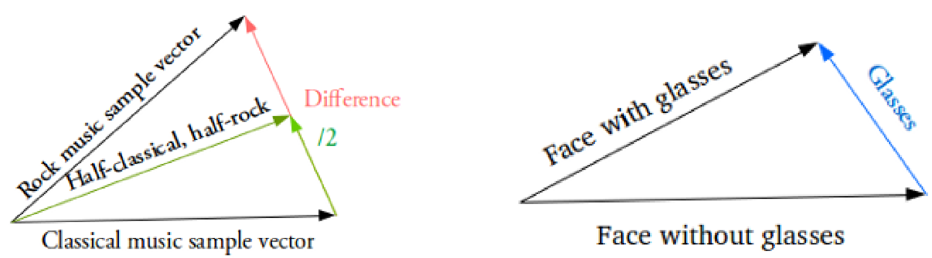


adapted from J. Jordan (towards data science)

VAE implications



Vector arithmetic in the latent space



adapted from Irum Shafkat's (towards data science)

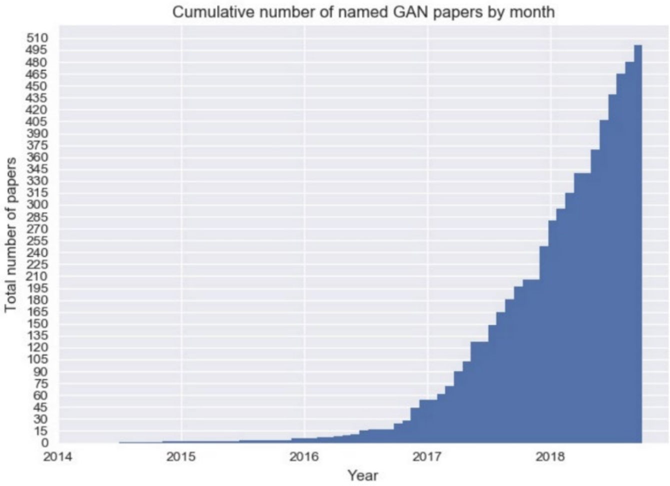
Impressive VAE applications

Deep Feature Consistent Variational Autoencoder

Google Brain's Magenta's MusicVAE
(Roberts et al., 2017)

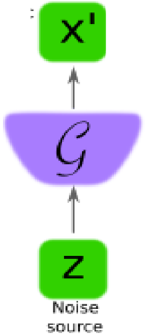
Generative adversarial networks (GANs)

- *"Generative Adversarial Networks is the most interesting idea in the last 10 years in Machine Learning"*, Yann LeCun



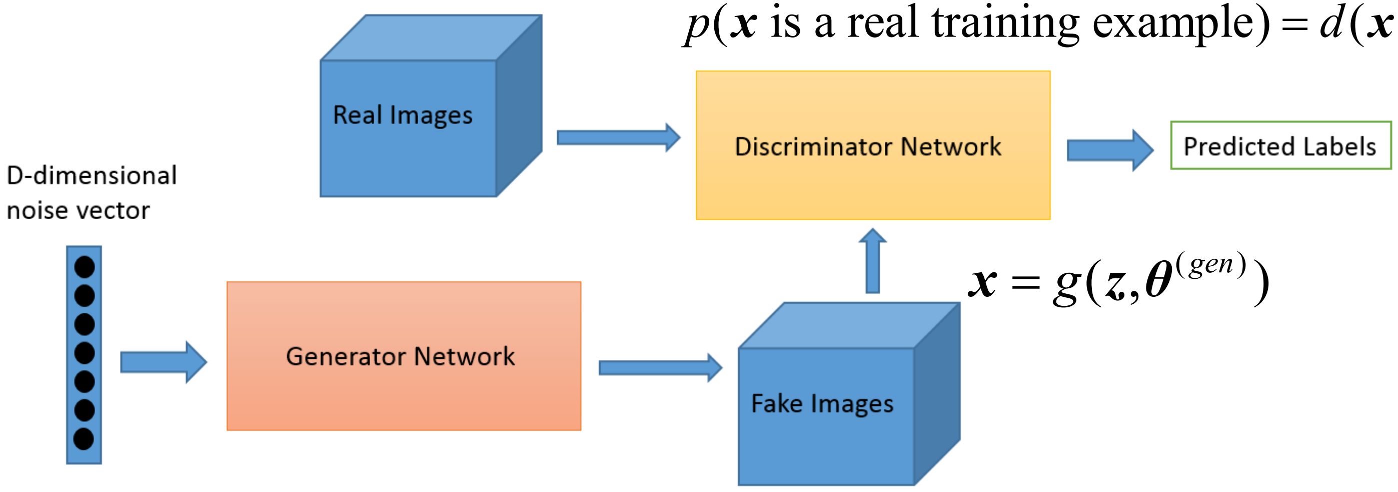
Generative adversarial networks (GANs)

- *"Generative Adversarial Networks is the most interesting idea in the last 10 years in Machine Learning"*, Yann LeCun
- Unlike in VAEs, the probability is modelled implicitly
- Instead, the idea is to sample from a complex distribution
- However, it is challenging to sample from a complex distribution, it cannot be done directly
- So, one solution is to sample from a simpler distribution (e.g. noise) and learn the transformation (generator) to the data distribution



GANs – operational mechanisms

$$p(\boldsymbol{x} \text{ is a real training example}) = d(\boldsymbol{x}, \boldsymbol{\theta}^{(discr)})$$



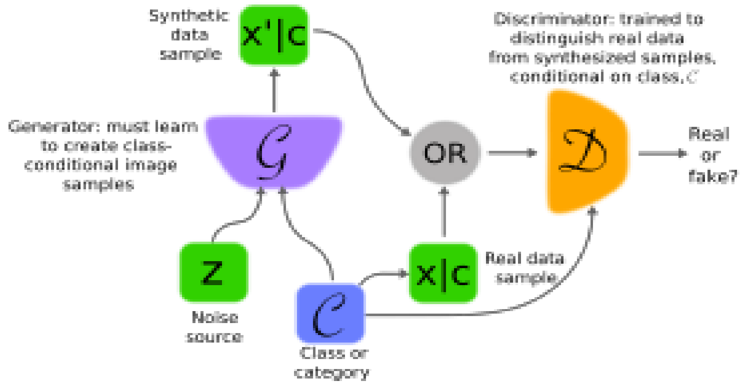
$$\boldsymbol{x} = g(\boldsymbol{z}, \boldsymbol{\theta}^{(gen)})$$

Discriminator network received some payoff v and the generator receives $-v$, so it is a zero-sum game. Both attempt to maximise their own payoff, so at the convergence:

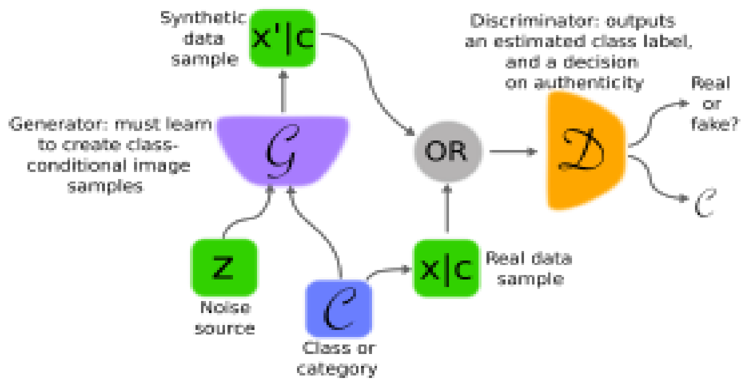
$$g^* = \arg\min_g \max_d v(g, d)$$

$$v(\boldsymbol{\theta}^{(g)}, \boldsymbol{\theta}^{(d)}) = \mathbb{E}_{\mathrm{x} \sim p_{\text{data}}} \log d(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{x} \sim p_{\text{model}}} \log(1 - d(\boldsymbol{x}))$$

A wide range of GAN models



conditional GAN

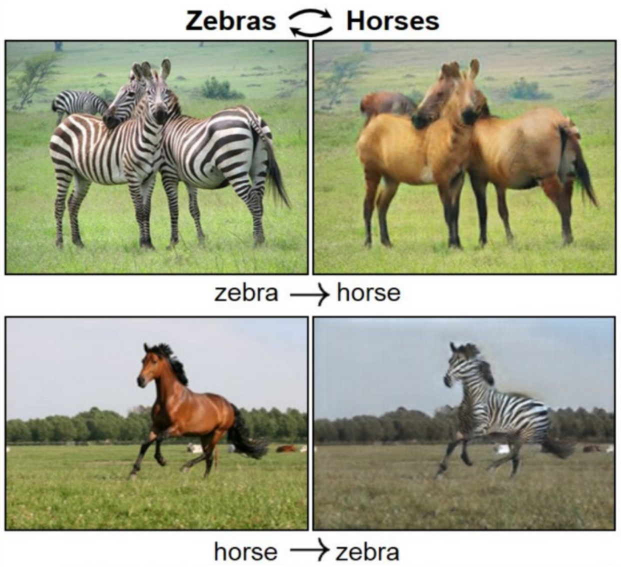


Creswell et al., 2017

info GAN

Applications of GANs

- What can GANs be used for?
  - data augmentation
  - creating art
  - image-to-image translation
  - creating images with higher resolution than the original



Discussion

1. What is the essence of deep generative modelling – what are shared characteristics (the common denominator)?
2. DBN, VAE, GAN are stochastic models – but what are key differences?
3. What could be the reason fro comparatively more blurry reconstruction of images with VAEs than with GANs? What is the underlying difference in their generative philosophy (how they model data distribution)?
4. What problem do overcomplete AEs address when compared to undercomplete AEs? How would you explore the latent manifold in AEs?
5. What role can CNNs play in deep generative modelling? Think of some concrete examples.

Summary of deep generative models

- Generative modelling aims at modelling data distribution (probabilistic approach)
  - learning **the joint $P(x, y)$** -> ambitious (if labels exist)
  - rapidly growing interest in unsupervised learning (no labels)
  - scope for generating new data samples: training data $\sim P_{data}(x)$ -> generated samples $\sim P_{model}(x)$
  - capable for uncovering underlying latent variables (learning represent.)
  - some flagship examples of deep latent variable models: DBN, GAN, VAE
- Traditionally, DBNs have been used to model joint distributions
  - greedy layer-wise pretraining of the latent (hidden) representations
  - often acting as a hybrid model – discriminative and generative

Summary of deep generative models

- Autoencoders (AE) offer flexibility in extracting latent representations (identifying a low-dimensional manifold) acting as inf. bottleneck
  - Undercomplete AE facilitate learning compressed representations where dimensionality reduction can be seen as an encoding process, but they require control of the encoding/decoding network capacity
  - More complex manifolds can be handled by overcomplete AEs (distributed representations) but require some explicit regularisation
  - The regularisation can help obtain some "nice" properties of the latent code, e.g. sparseness (sparse AE), smallness of the derivative of the representations (contractive AE) and robustness to noise and missing inputs (denosing AE)
  - Still, classical AE (that are not generative models) do not allow for controlling the completeness and continuity in the latent space (manifolds)

Summary of deep generative models

- Generative Variational Autoencoders (VAE) balance the reconstruction accuracy and “desirable/good” properties in the latent space
  - the latent space irregularity is tackled by
    - the encoder returning a distribution over the latent space instead of a single point (a probabilistic “twist” on AE -> stochastic behaviour)
    - the loss function incorporating a regularisation term over the returned latent distribution to ensure a good organisation of the latent space
  - VAE allows for defining a meaningful generative process using well-organised latent space, which enables sampling from the distribution as well as altering and exploring variations on the existing data
  - VAEs have a probabilistic interpretation and a network implementation, which for gradient descent based optimization exploits a reparametrisation trick