DD2437 – Artificial Neural Networks and Deep Architectures (annda)

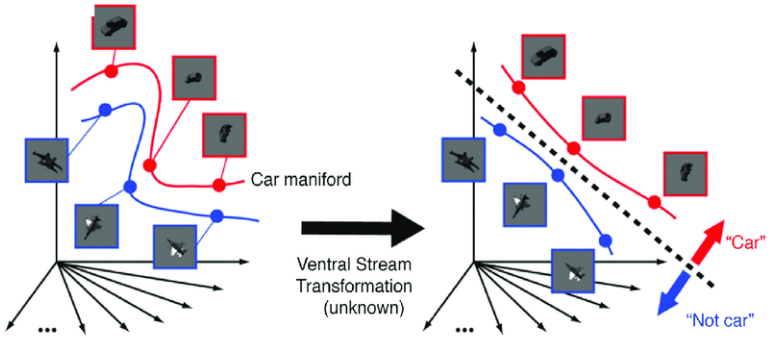
Lecture 11: **Learning deep representations, deep generative models**

Pawel Herman

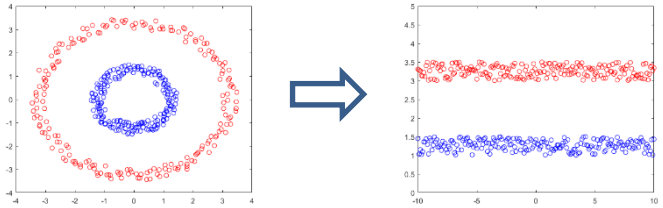
Computational Science and Technology (CST)

KTH Royal Institute of Technology

The importance of representations

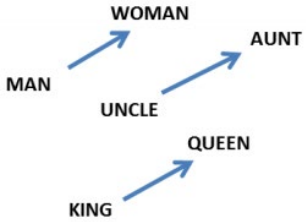


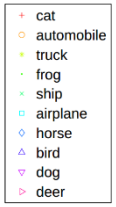
Good representations can make certain computations cheaper and more robust

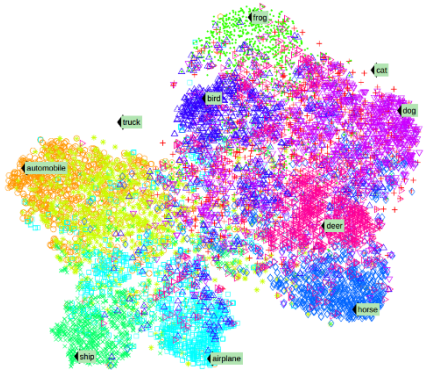


"woman" -> [-0.3, 0.2, 0.5, 0, -0.1, 0…]

"woman" – "man" ≈ "queen" – "king"







Learning representations + complex inference = future AI?

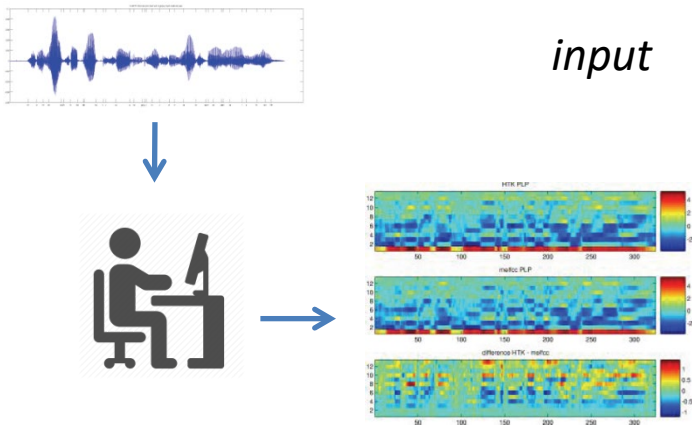
Learning representations as a hallmark of DL

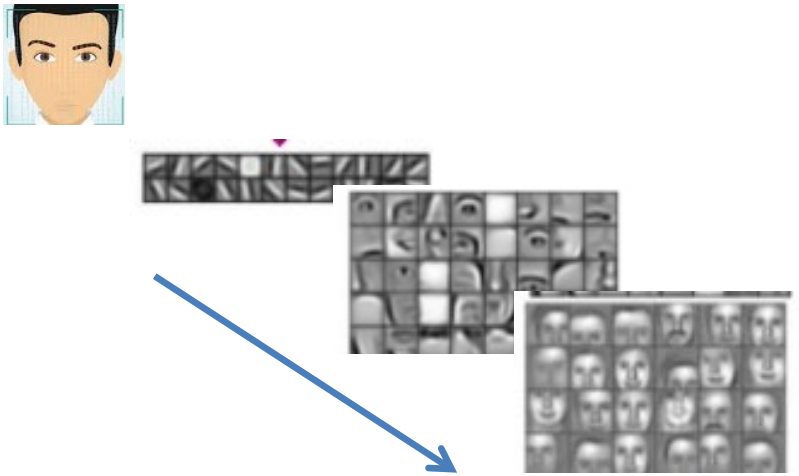
Learning features as part of DL modus operandi

- 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

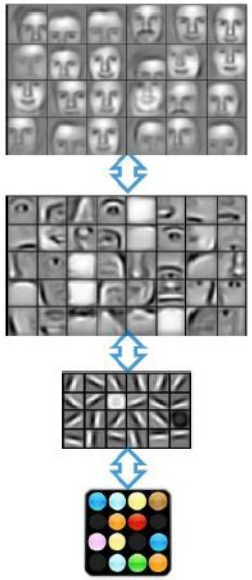




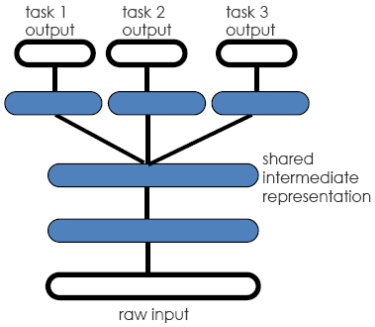
Learning representations as a hallmark of DL

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

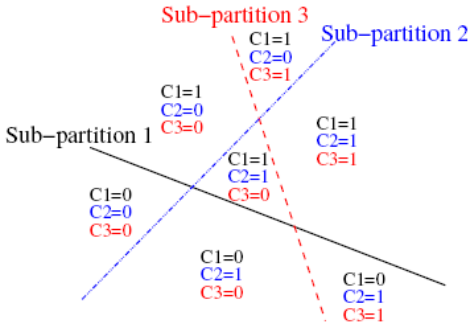
hierarchy of abstraction levels



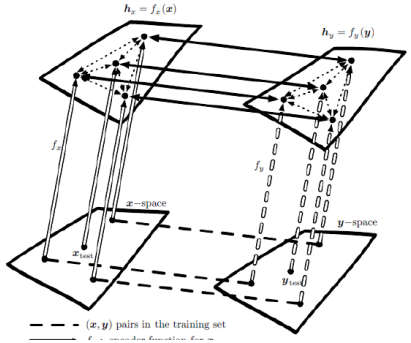
multi-tasking and transfer learning



multi-clustering



zero-shot learning



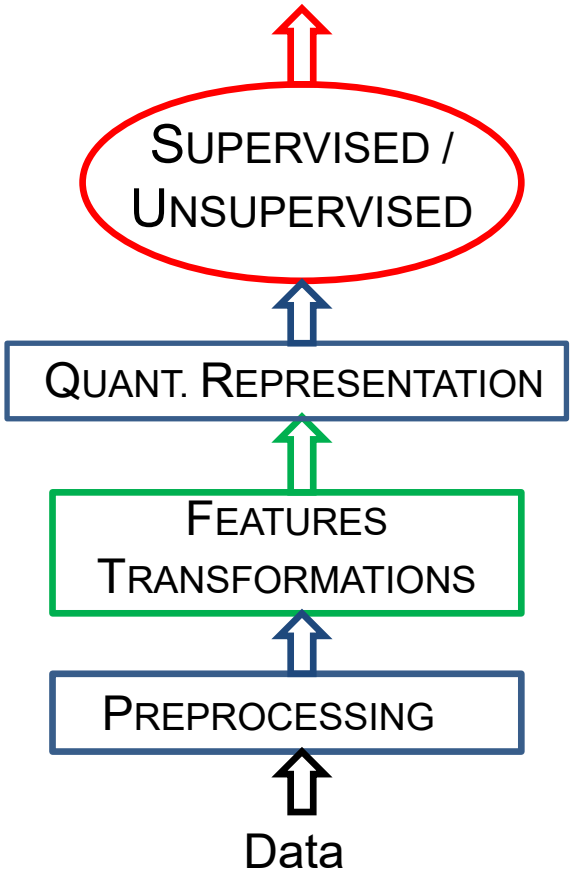
Representation learning problem

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



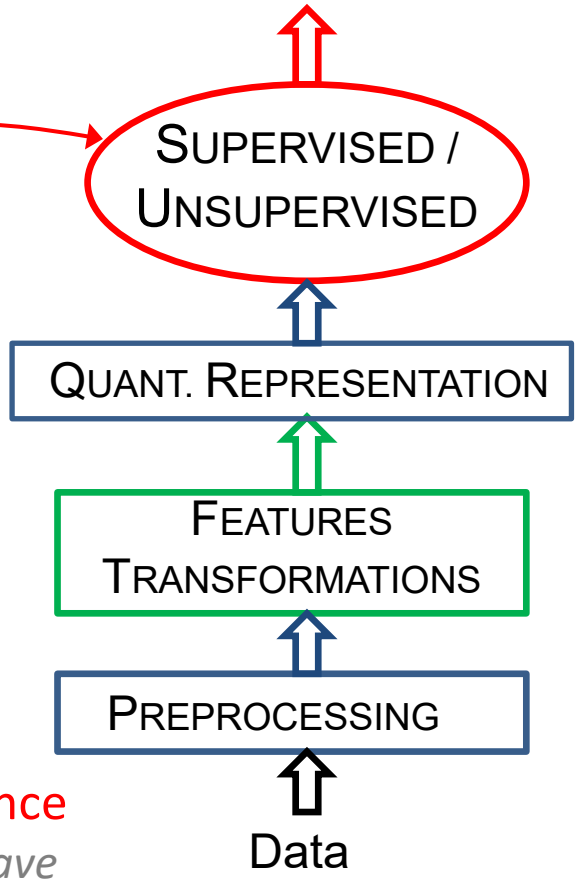
Representation learning problem

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

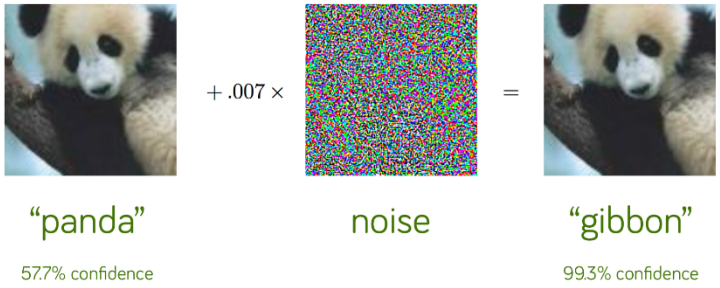


Facilitate the subsequent learning task, maximise performance

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

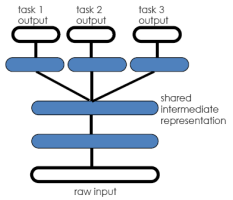
Supervised vs unsupervised approach

- 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



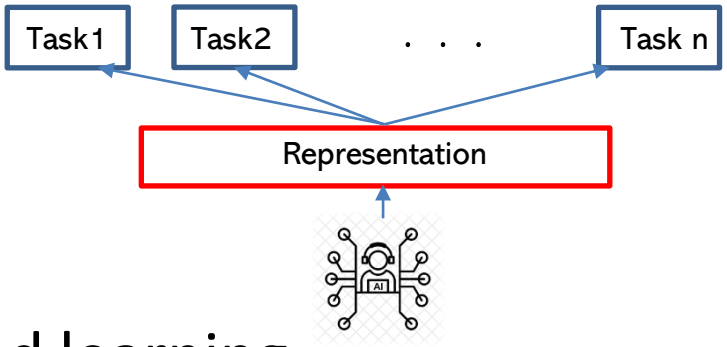
  - lacking “common sense”

  - 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

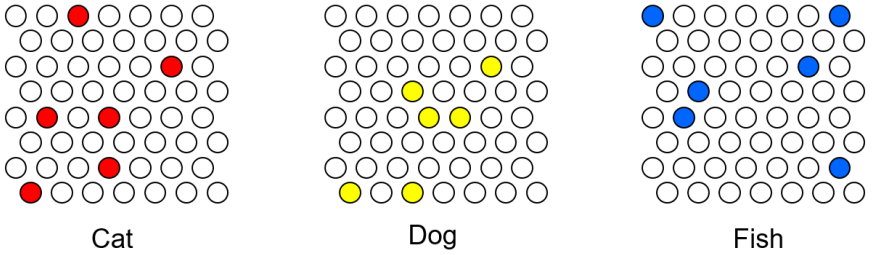
- Computational perspective: disentangling unknown factors causing relevant variation in the data (*factors of variation*)
  - *causes explain* the observed data (discriminative context, both unsupervised and supervised aspects)
  - factors in combination can be used to generate data (generative context)
- 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*|*latent var*) for generation and P(*latent var*|*data*) for recognition
  - full probabilistic model advocated by generative models, P(*data*,…)

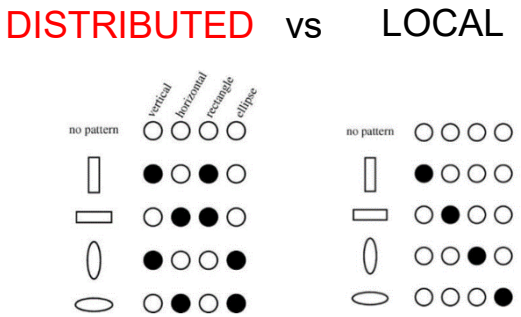
Distributed vs local representations

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).



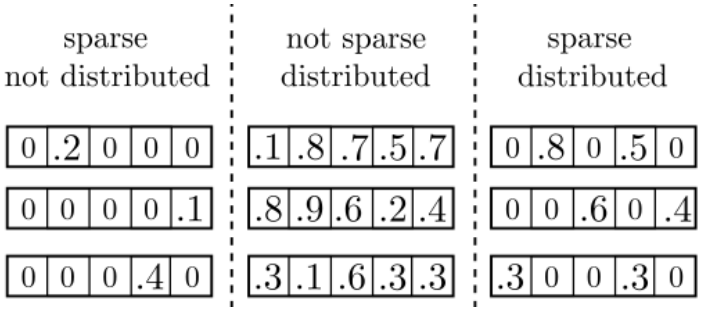


Generalisation due to shared attributes and semantic proximity.

Sparse vs dense representations

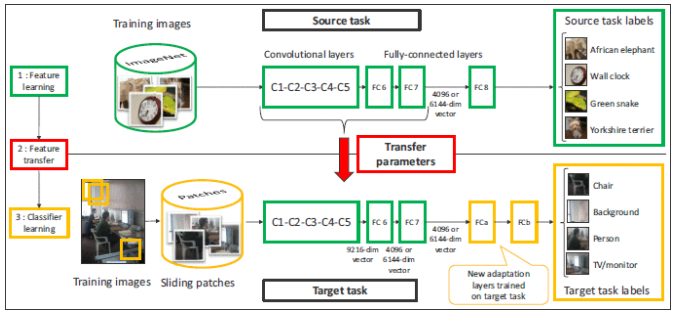
Sparse representations

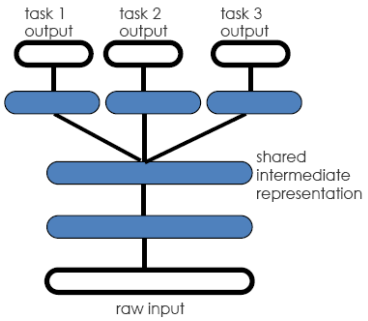
- orthogonalisation/decorrelation – more separable in high-dimensional spaces
- "metabolic" efficiency
- neural selectivity (vs coarse coding with broad tuning)
- balance between sparse local representations that suffer from the *curse of dimensionality* and dense representations that *entangle* factors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

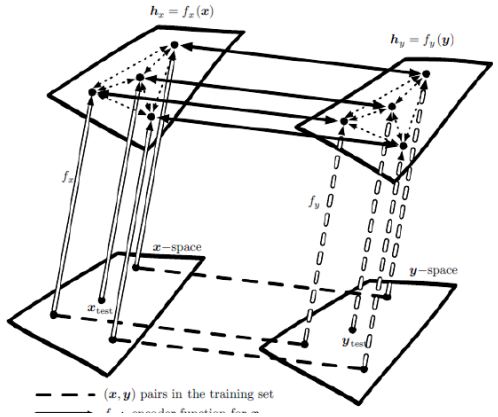


Functional implications of distributed codes

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







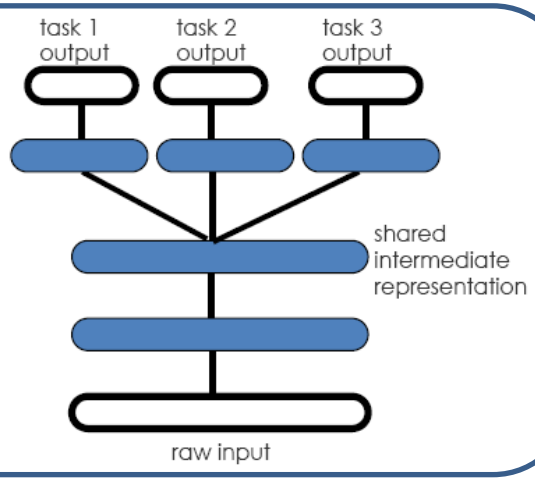
Transfer learning

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



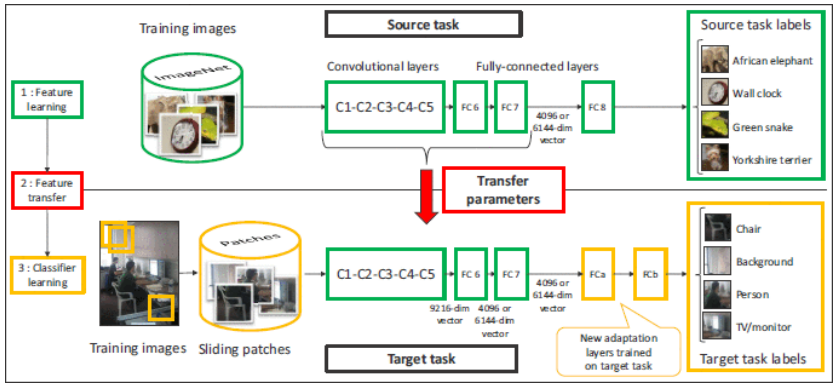
Transfer learning

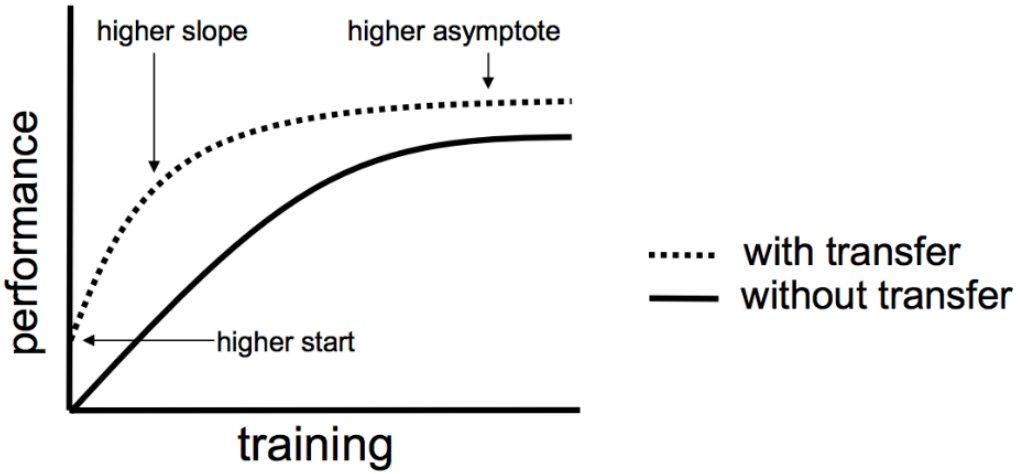
A more general concept in ML: a model developed for one task is reused (re-purposed) as the starting point for a model on another task

- 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

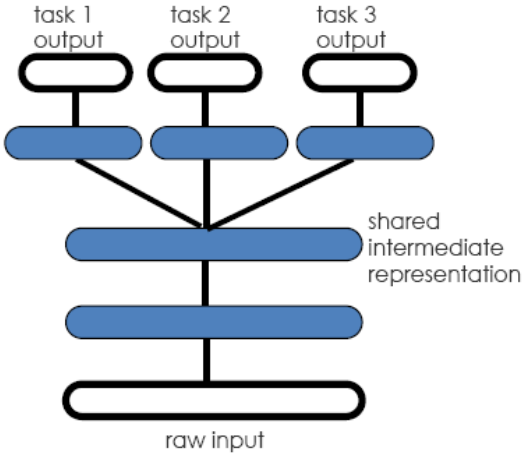
"Transfer learning and domain adaptation refer to the situation where what has been learned in one setting … is exploited to improve generalization in another setting"





Multi-task learning

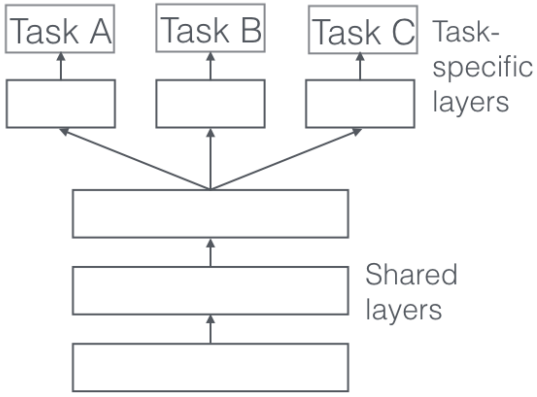
- Excessive focus on a single task may ignore information that can help us perform better on the metric we care about
- Effectively, simultaneously optimizing different cost functions implements multi-task learning
- Form of inductive transfer learning with inductive bias/regularisation mediated by another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



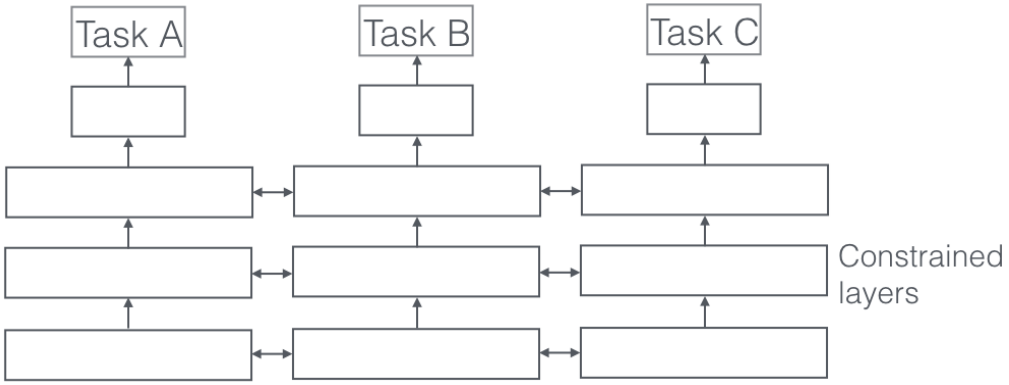
S. Ruder (2017). An Overview of Multi-Task Learning in Deep Neural Networks

Multi-task learning

Hard parameter sharing



Soft parameter sharing



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

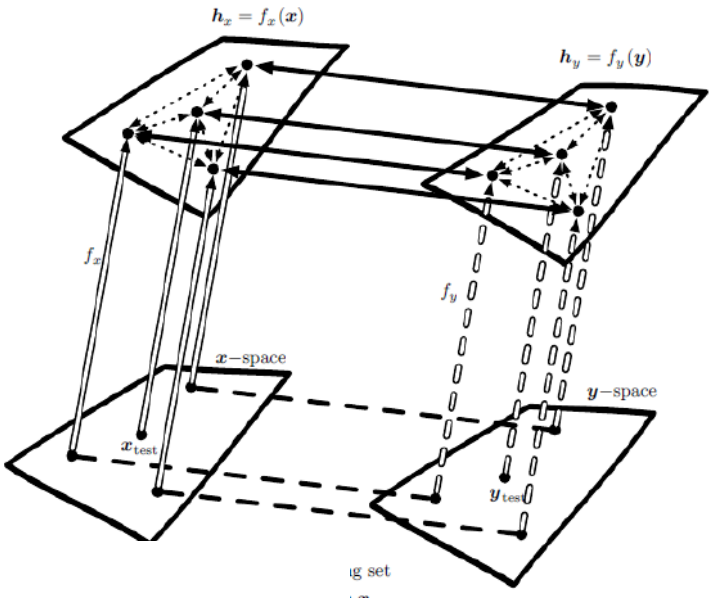
Zero-shot learning

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

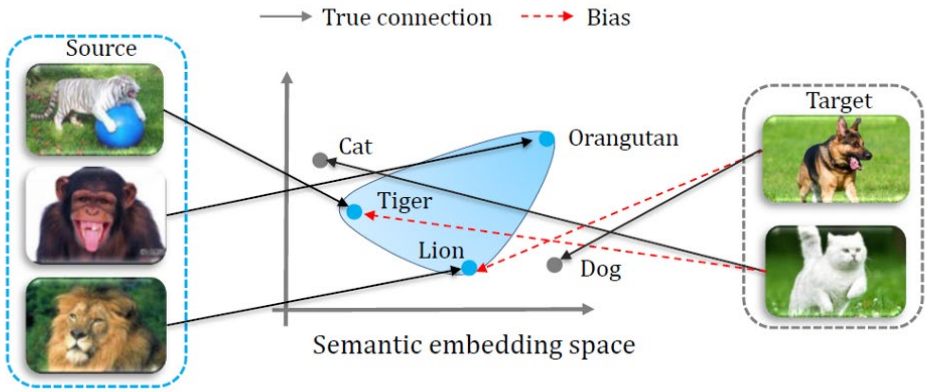
• In zero-shot classification we want to recognize objects from classes that the model has not seen during training.

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:

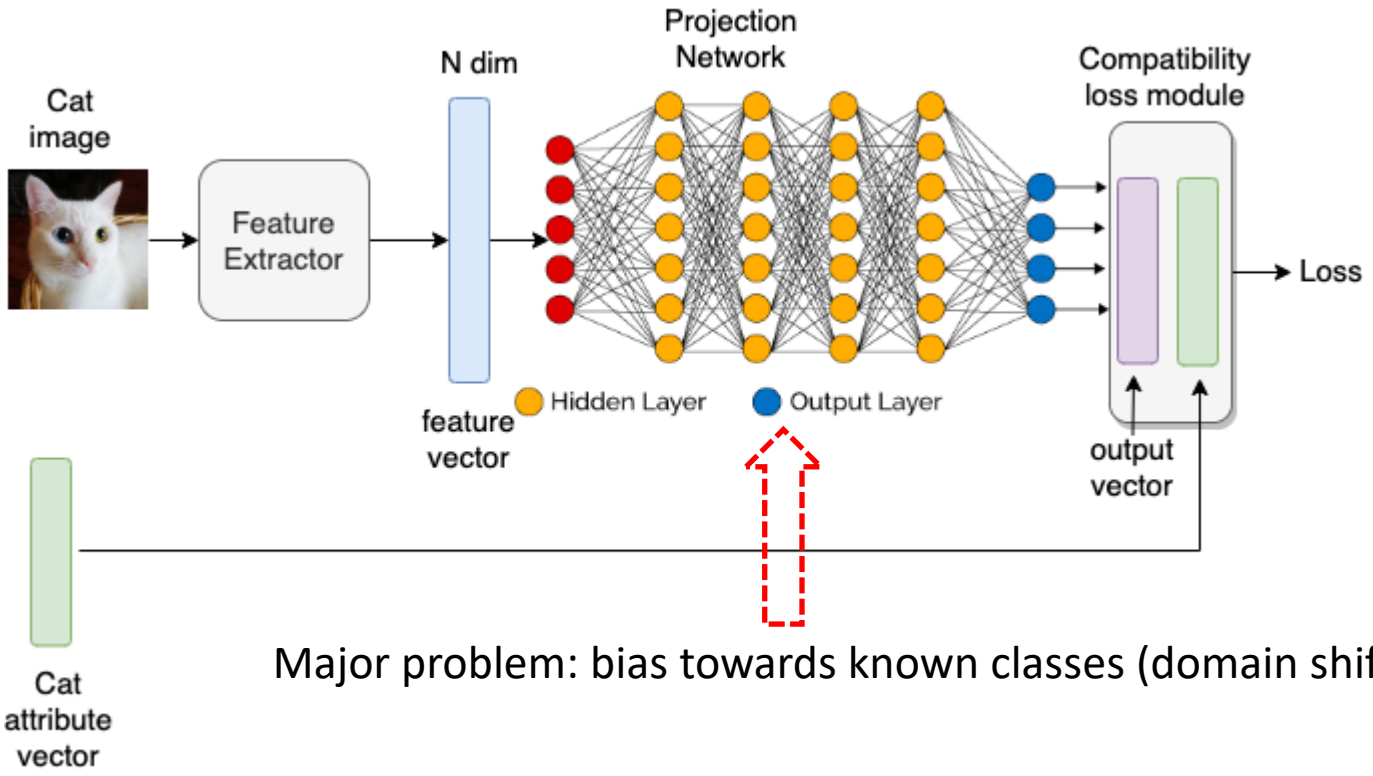
i. **Seen classes** (with labels)

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

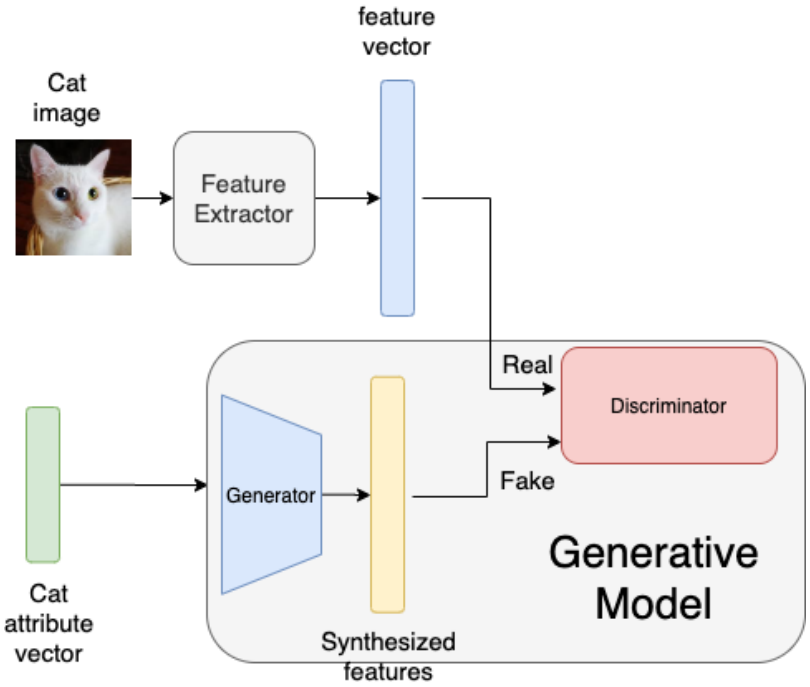
Embedding based methods



Major problem: bias towards known classes (domain shift)

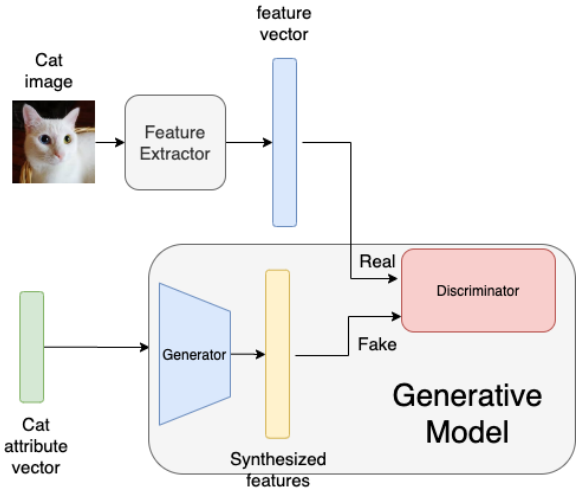
Zero-shot learning

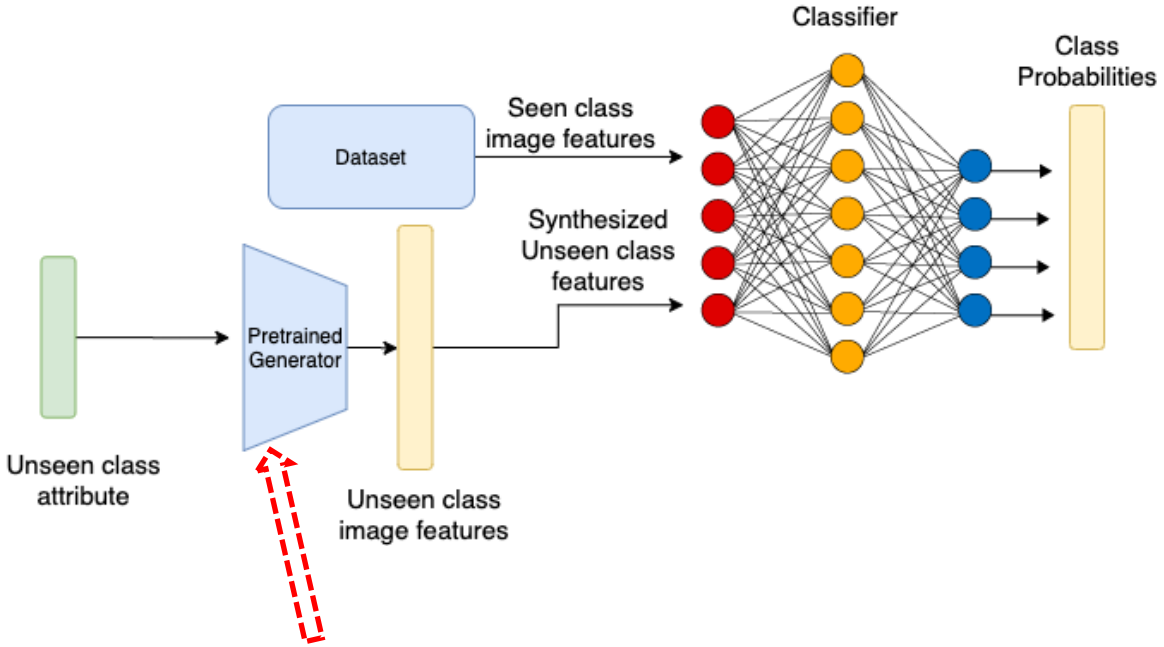
Generative model based methods



Zero-shot learning

Generative model based methods





cross-domain knowledge transfer

+ semantic loss problem

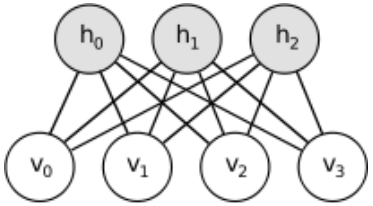
Representation learning in deep models

General philosophy and approach

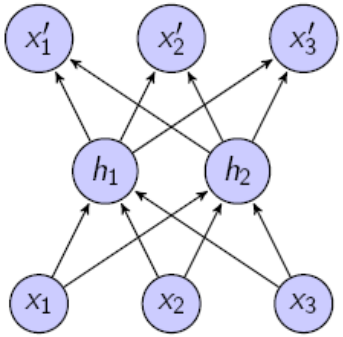
- supervised vs unsupervised
- the concept of unsupervised *pre-training*
- *probabilistic models* (latent variables that describe the distribution) vs *direct encoding* (learning a parametric map from input to representation)
- historically important *manifold learning* and *generative* models
- growing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

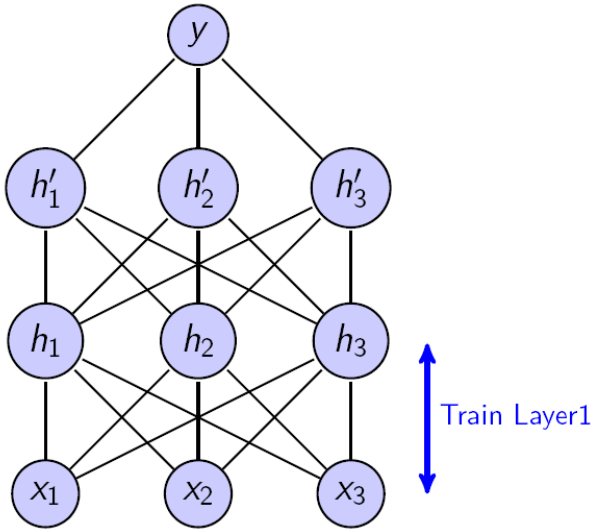


  - autoencoders

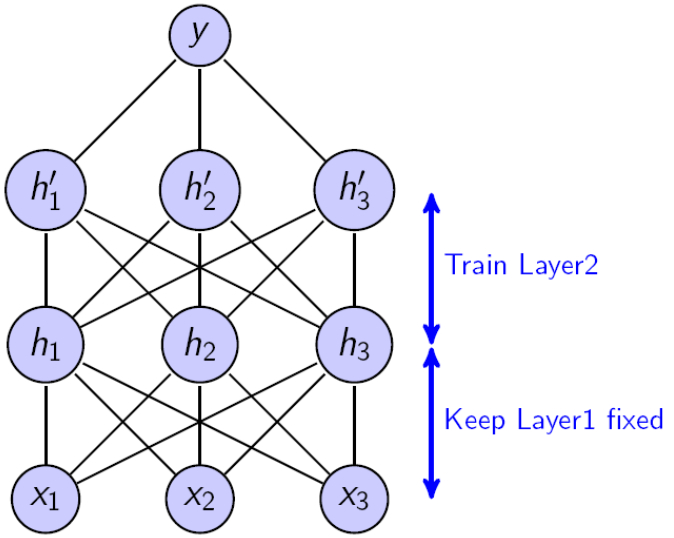


Representation learning in deep models

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



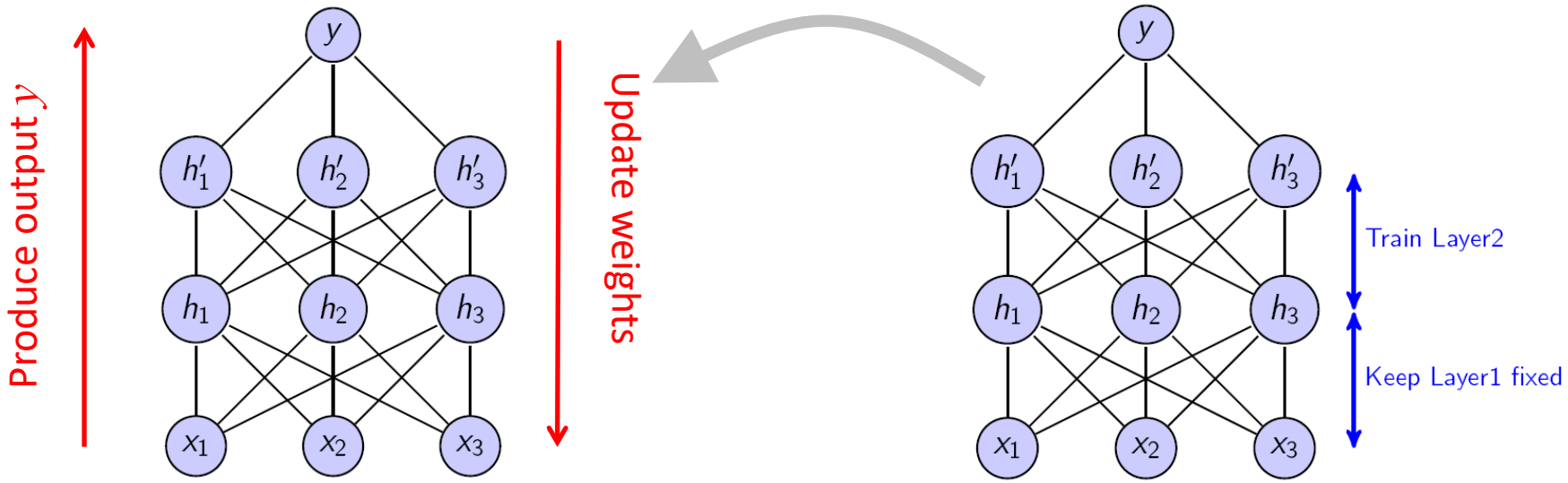
Single layer at a time



Train another layer while keeping the lower layer fixed

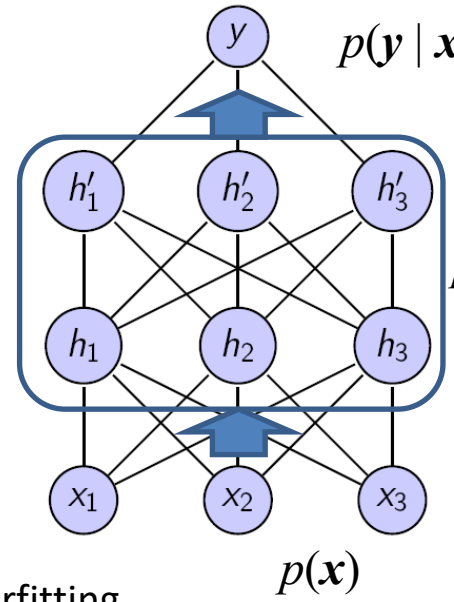
Representation learning in deep models

- 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*)



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

Ideally:

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

marginal factor independence

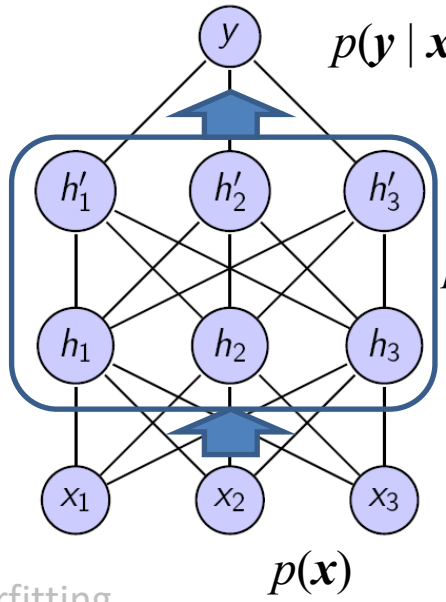
Representation learning in deep models

- 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.

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



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

Ideally:

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

marginal factor independence

verfitting

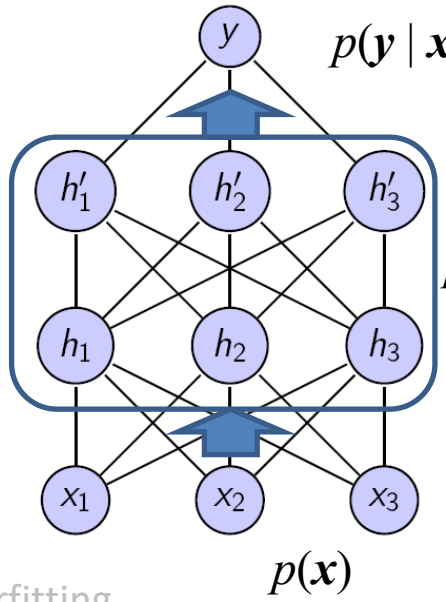
- as a regulariser, it urges the learning algorithm to discover **features that explain *underlying causes* that generate the data** (also, causal factors often remain *invariant*)

Representation learning in deep models

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

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

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.



$p(h|x)=p(x|h)\ p(h)$

Ideally:

$p(h) = \prod_i p(h_i)$

marginal factor independence

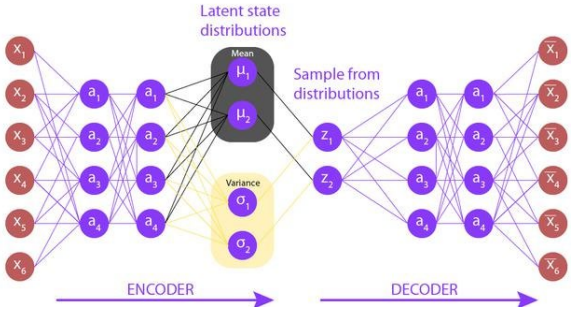
verfitting

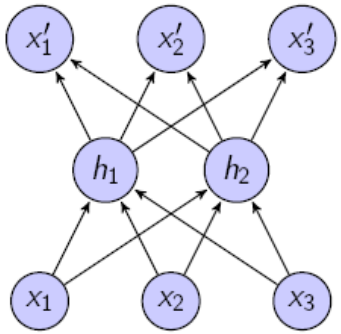
– as a regulariser, it urges the learning algorithm to discover **features that explain *underlying causes* that generate the data** (also, causal factors often remain *invariant*)

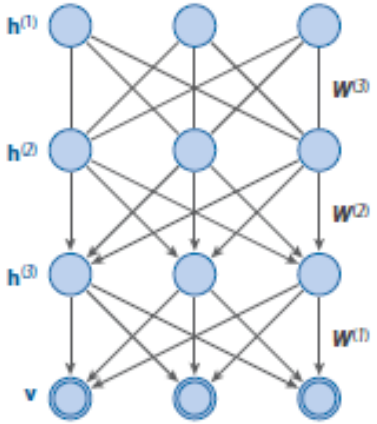
Representation learning in deep models

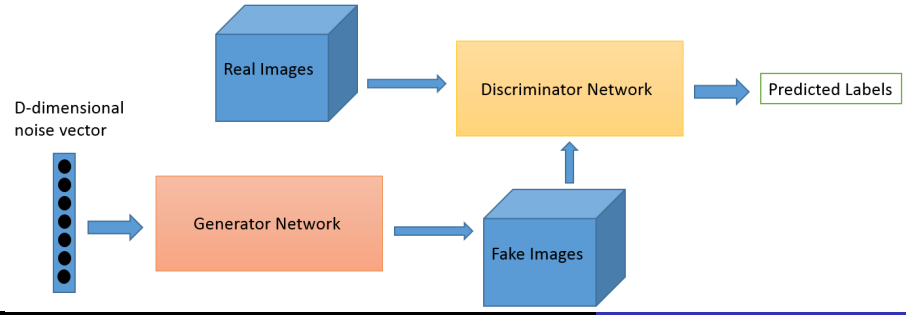
DNN architectures of special interest

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

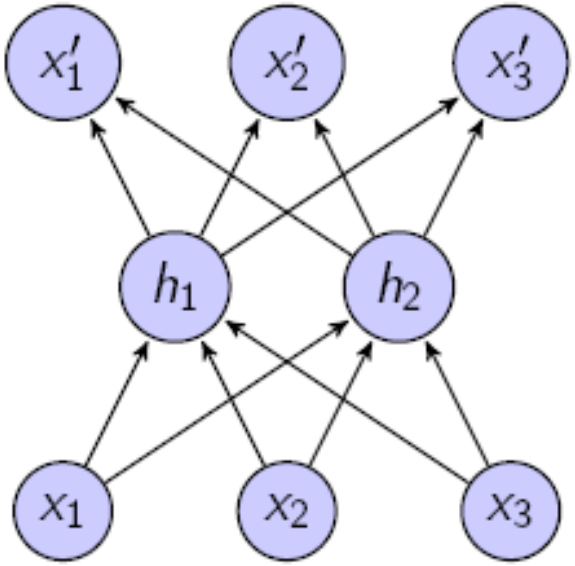








Autoencoders



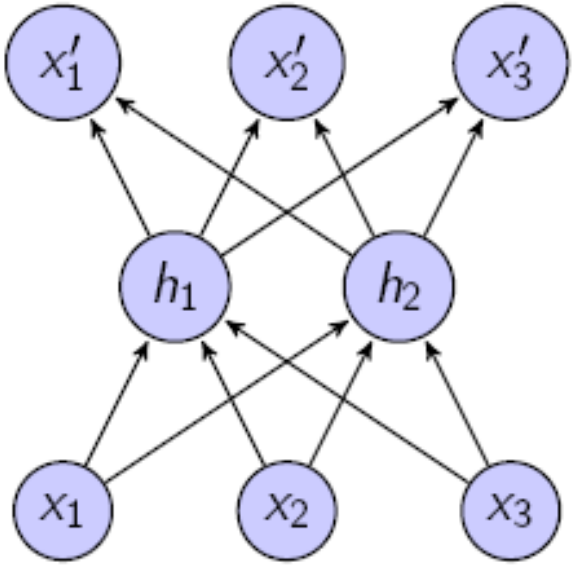
Decoder: $x' = f(x)$

Encoder: $h = g(f(x))$

The idea is to minimise the loss function, L:

$$L(x, g(f(x)))$$

Autoencoders



Decoder: $x' = \sigma(W'h + d)$

Encoder: $h = \sigma(Wx + b)$

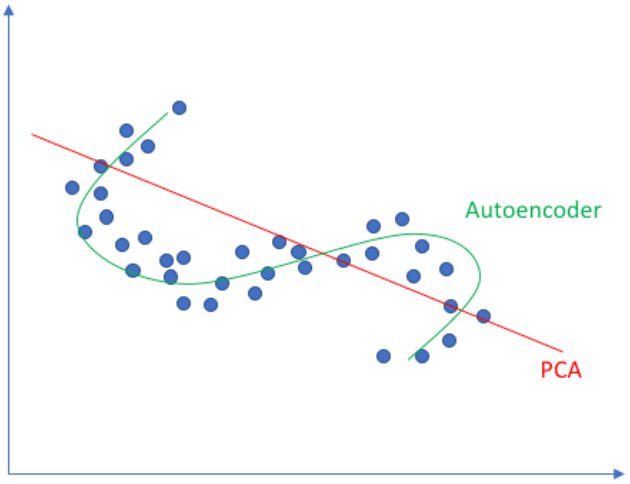
Encourage h to give small reconstruction error:

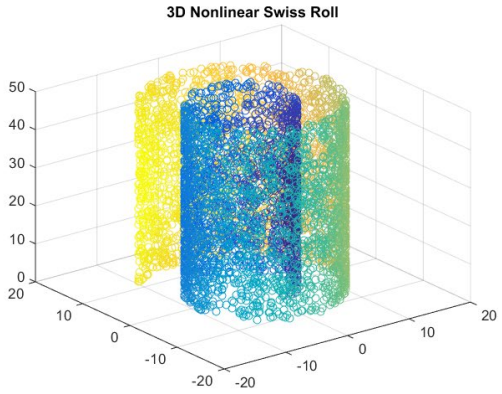
- e.g. $Loss = \sum_m ||x^{(m)} - DECODER(ENCODER(x^{(m)}))||^2$
- Reconstruction: $x' = \sigma(W'\sigma(Wx + b) + d)$

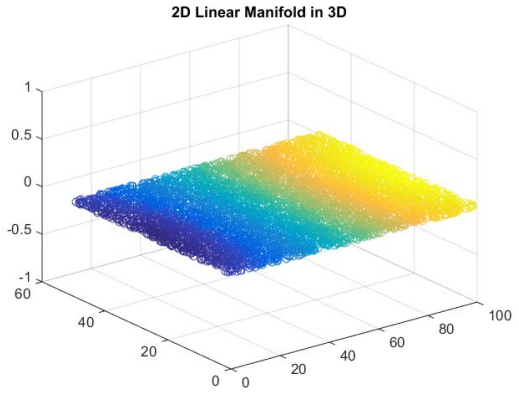
Autoencoders

Learning nonlinear manifolds

Linear vs nonlinear dimensionality reduction

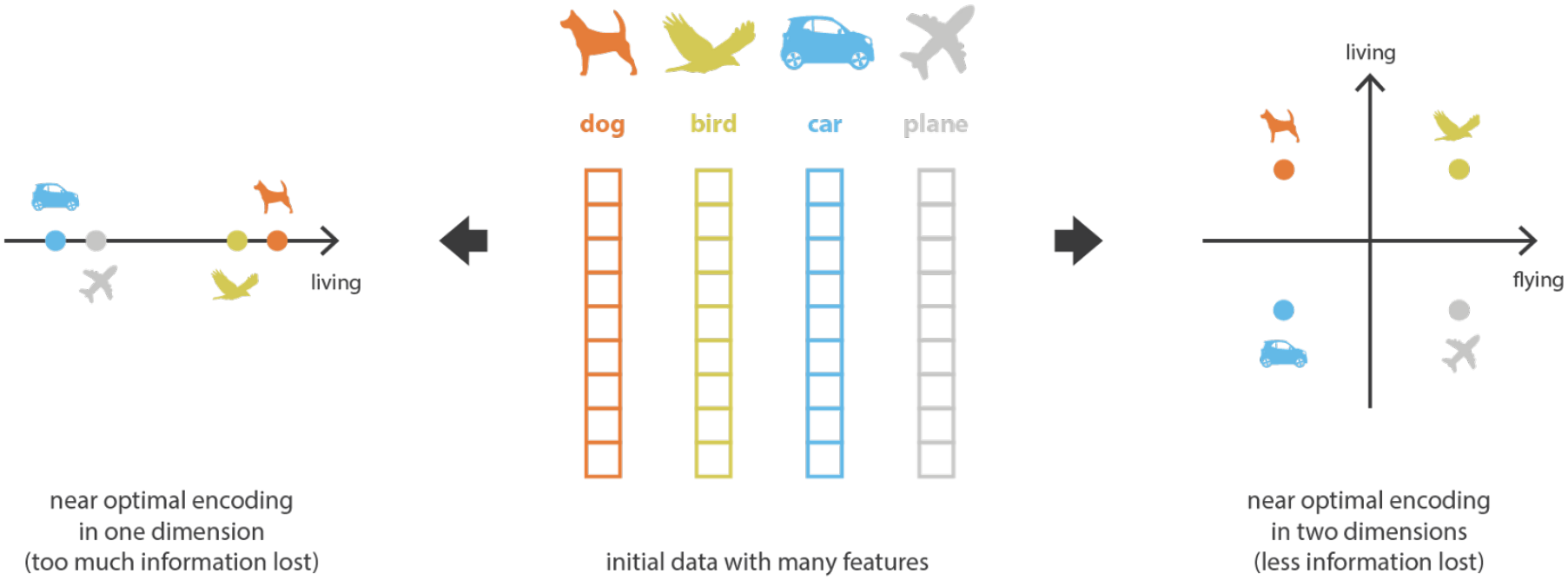






Autoencoders

Learning nonlinear manifolds



Different types of autoencoders

- Undercomplete autoencoders
  - hidden layer is smaller than the input dimensionality
- Overcomplete regularised autoencoders
  - Larger hidden layer size with the regularisation (to avoid overfitting and copying input to the output)

$$L\big(x, g\big(f(x)\big)\big) + \Omega(h), \quad h = f(x)$$

  - Sparse autoencoders, denosing autoencoders
  - Deep autoencoders

Sparse autoencoders

- Penalizing non-sparse solutions can be seen as adding latent variables with a prior and maximising likelihood

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

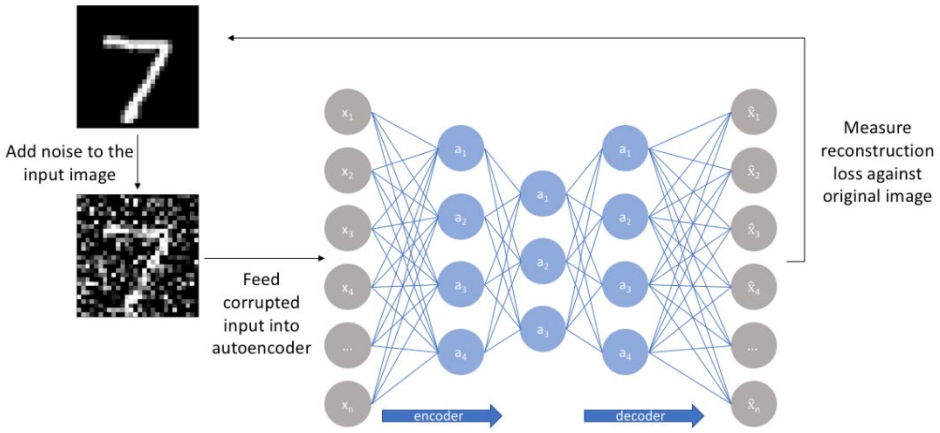
for example: $p_{\text{model}}(\boldsymbol{h}) = \prod_i \frac{\lambda}{2} e^{-\lambda |h_i|} \Rightarrow \boxed{\Omega(\boldsymbol{h}) = \lambda \sum |h_i|}$

Denoising autoencoders

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

Corrupted copy of x

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



Denoising autoencoders

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

Corrupted copy of 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 ***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 $p_{reconstruction}(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) = p_{decoder}(\boldsymbol{x} \mid \boldsymbol{h}), \quad \boldsymbol{h} = f(\tilde{\boldsymbol{x}})$

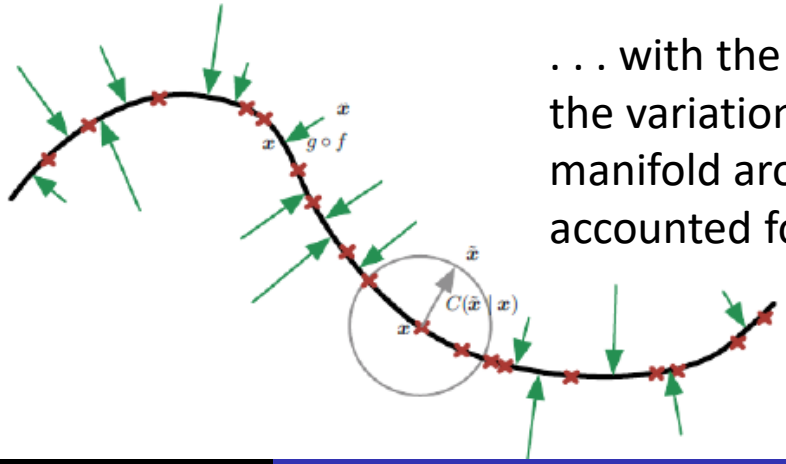
Denoising autoencoders

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

Corrupted copy of x

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

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



. . . with the principle that only the variations tangent to the manifold around ***x*** should be accounted for by changes in ***h***

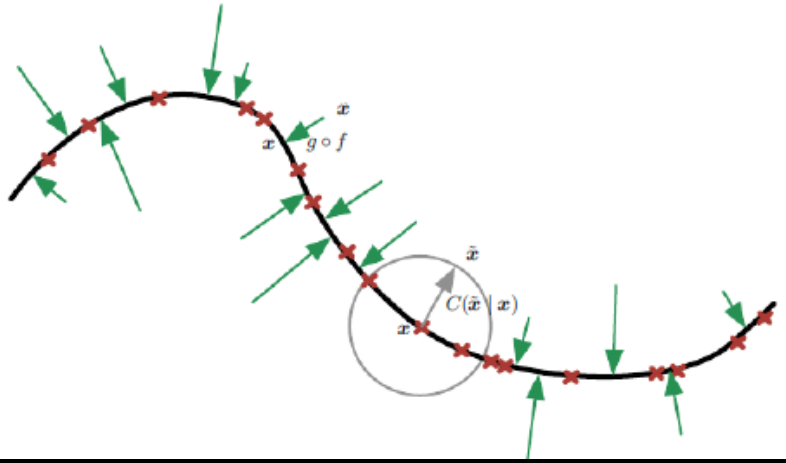
Goodfellow et al.

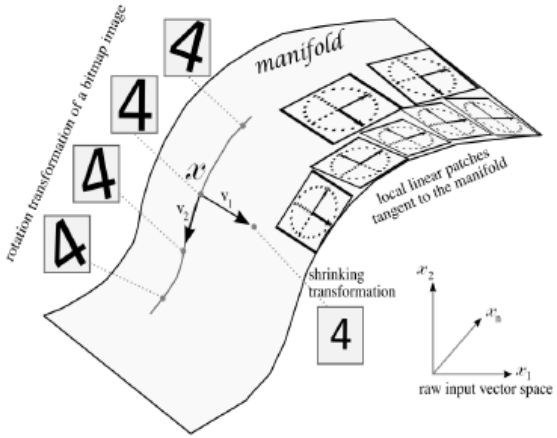
Denoising autoencoders

When training autoencoders there is a **compromise**

I. Need to approximately recover ***x*** – *reconstruction* force

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



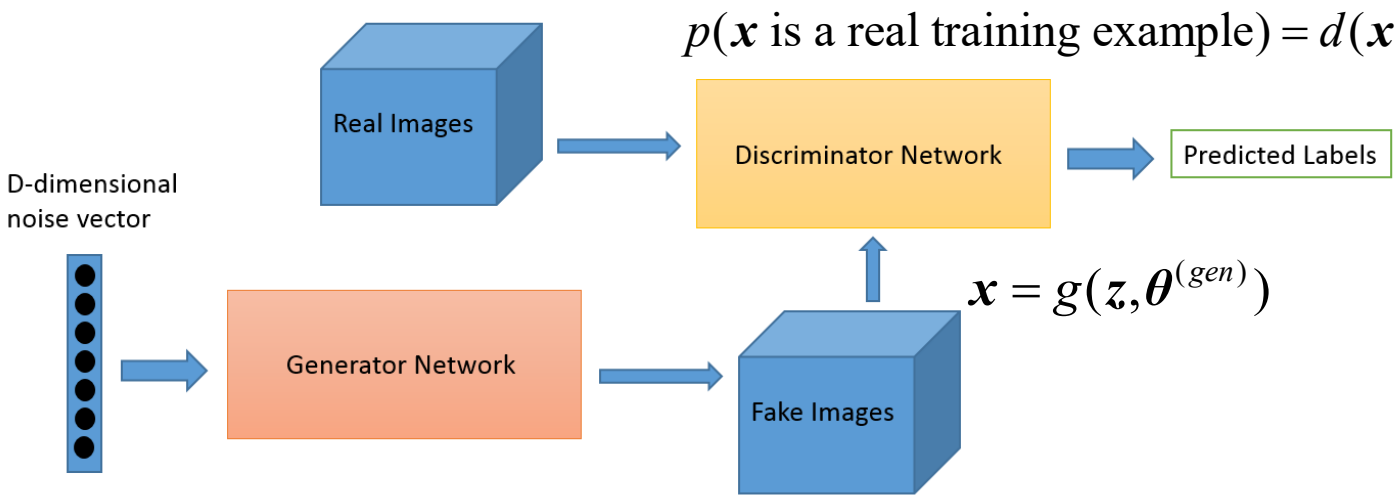


Goodfellow et al.

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

Generative adversarial networks (GANs)



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:

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

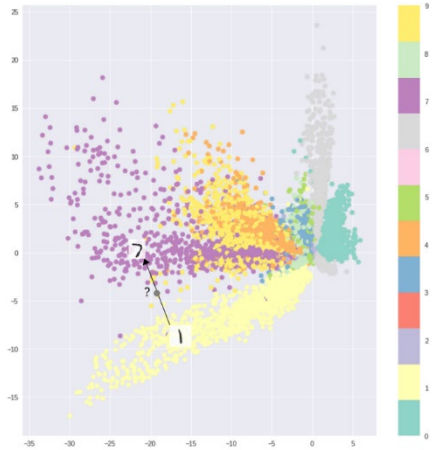
$$v(\boldsymbol{\theta}^{(g)}, \boldsymbol{\theta}^{(d)}) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log d(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{x} \sim p_{\text{model}}} \log (1 - d(\boldsymbol{x}))$$

Generative adversarial networks (GANs)

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

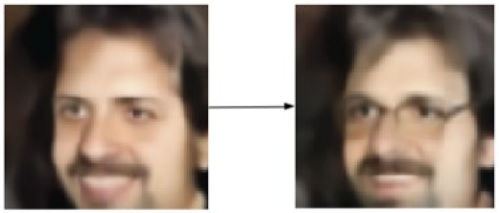
Variational autoencoders (VAE)

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

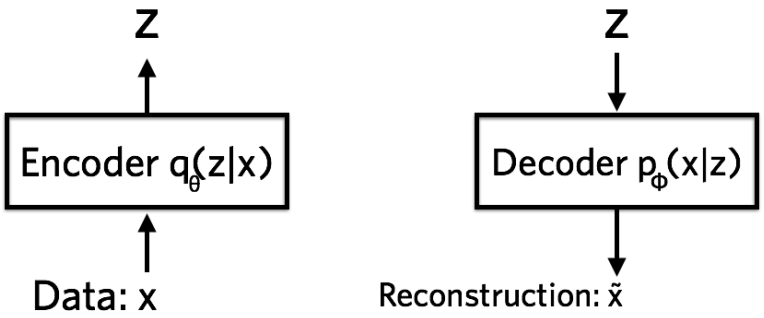


Classical autoencoders (that are not generative models) offer latent space that is often discontinuous and does not allow easy interpolation.

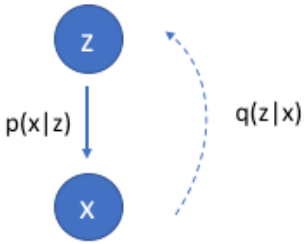
  - alter and explore variations on the existing data

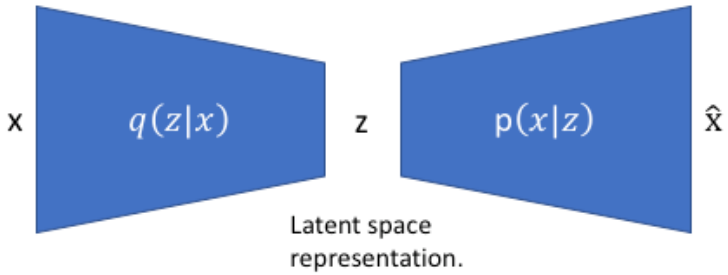


VAE mechanics



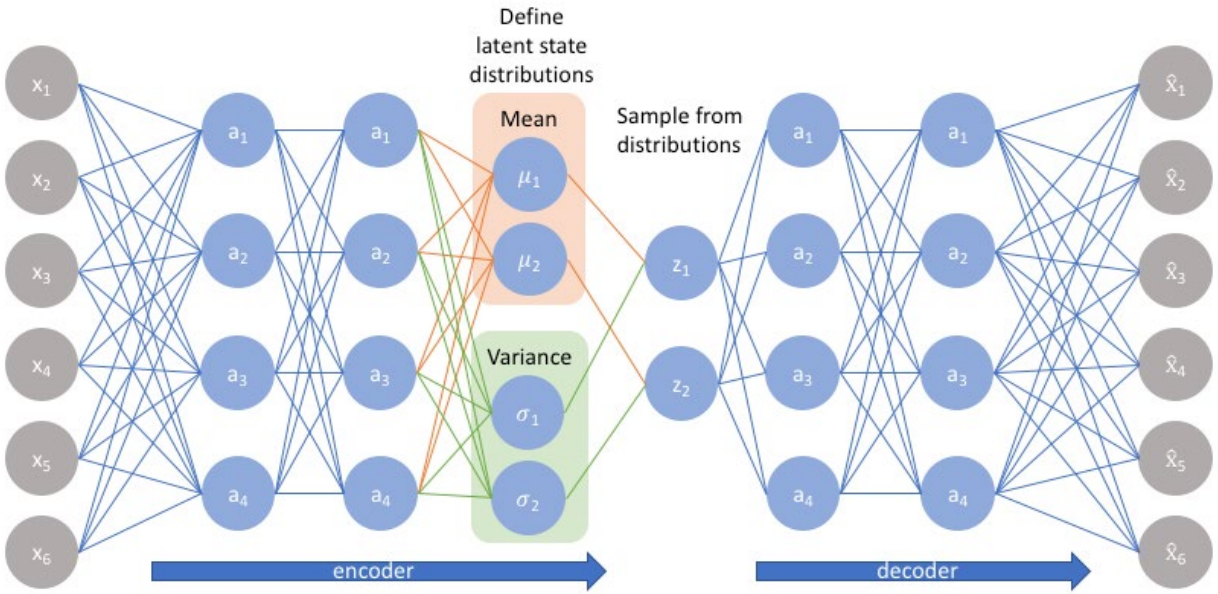
$$L(x,\hat{x}) + \sum_j KL\left(q_j(z|x) \| p(z)\right)$$





VAE as a network implementation

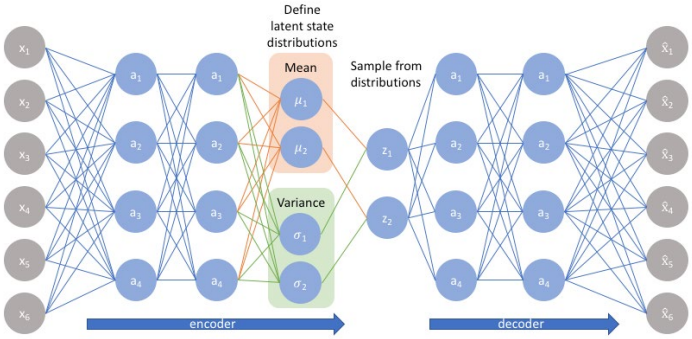
- How do we go about this in practice?

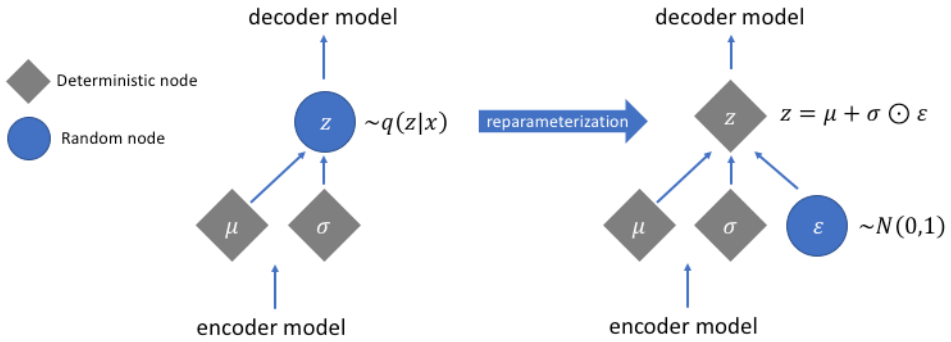


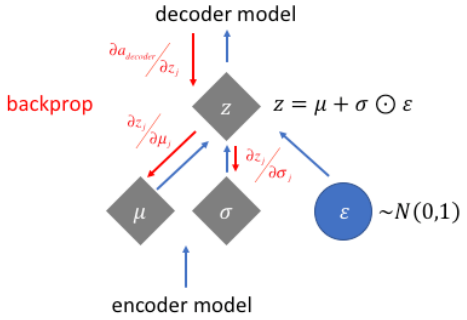
adapted from Jeremy Jordan

VAE as a network implementation

- How do we go about this in practice?

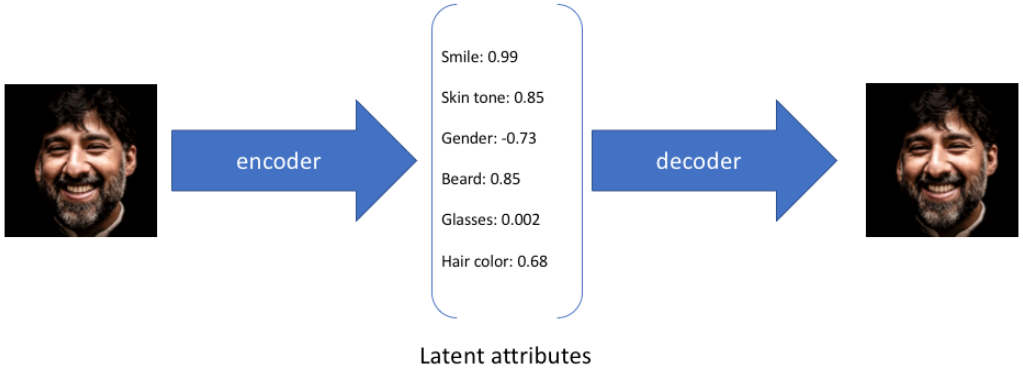


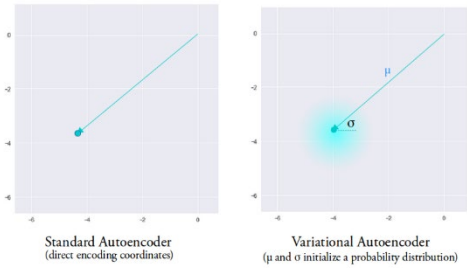


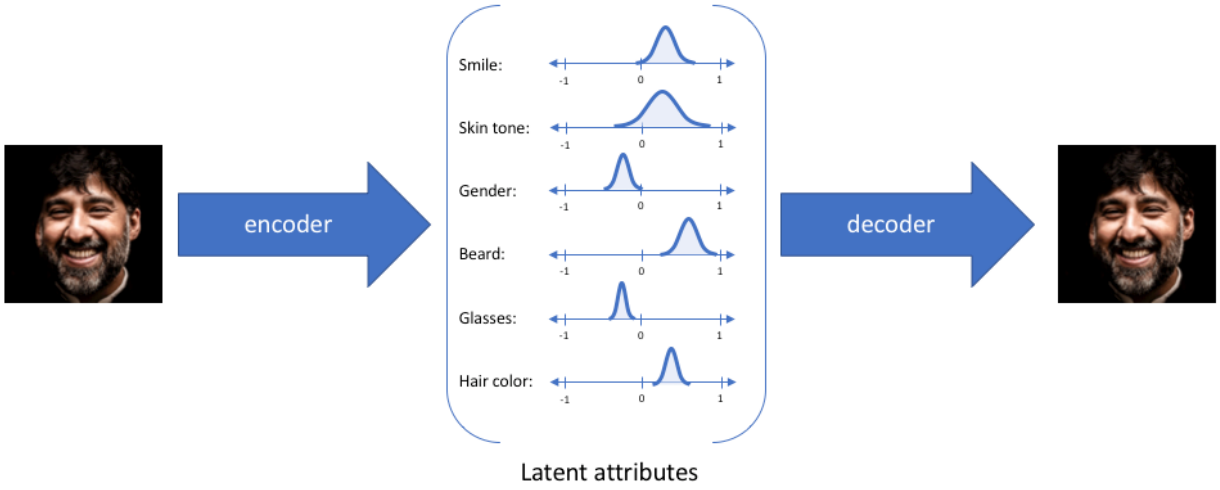


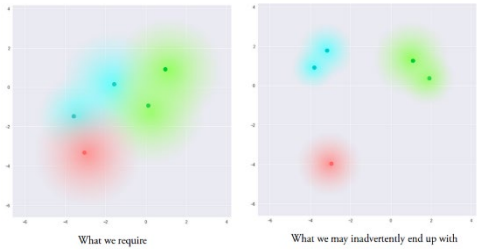
adapted from Jeremy Jordan

The intuition behind VAEs



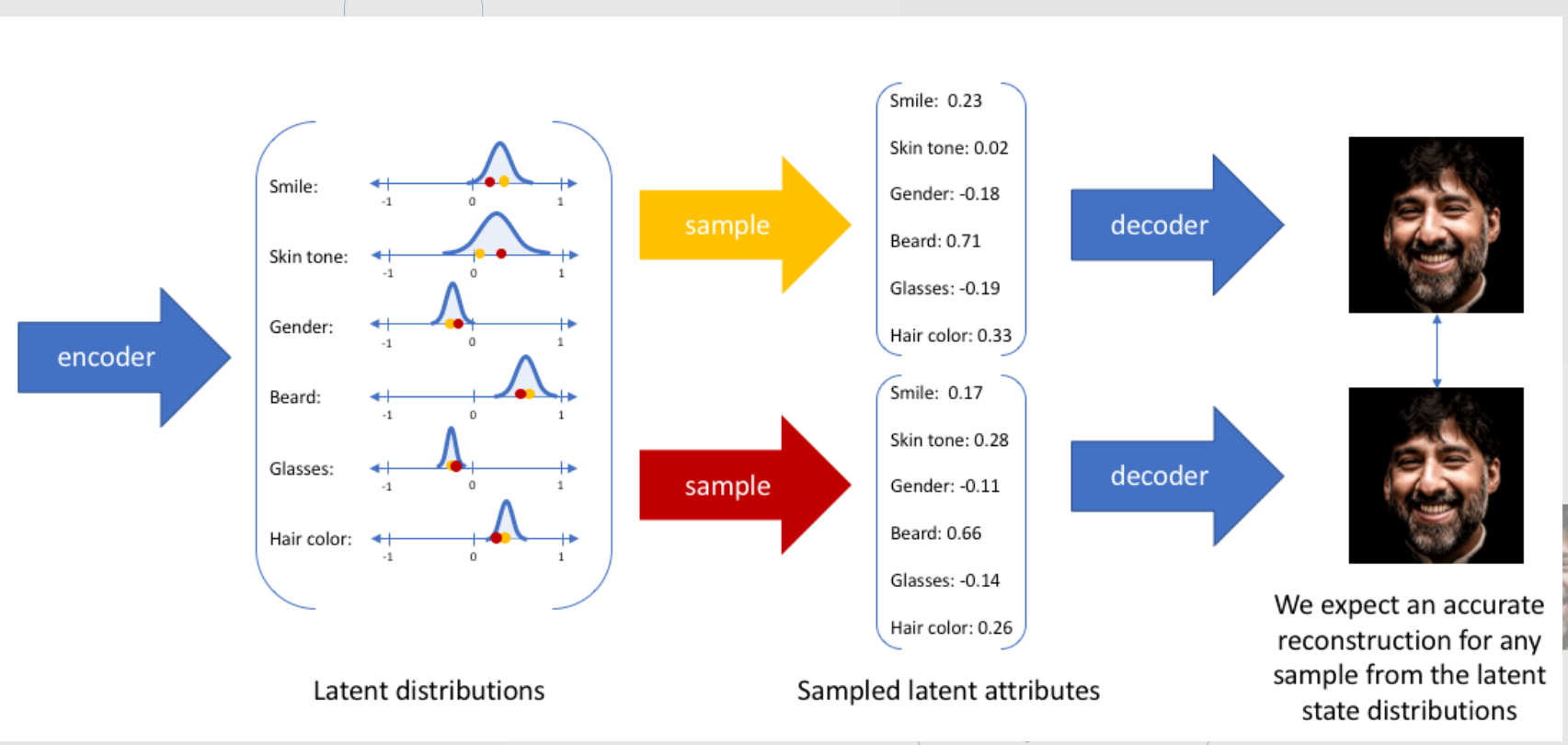






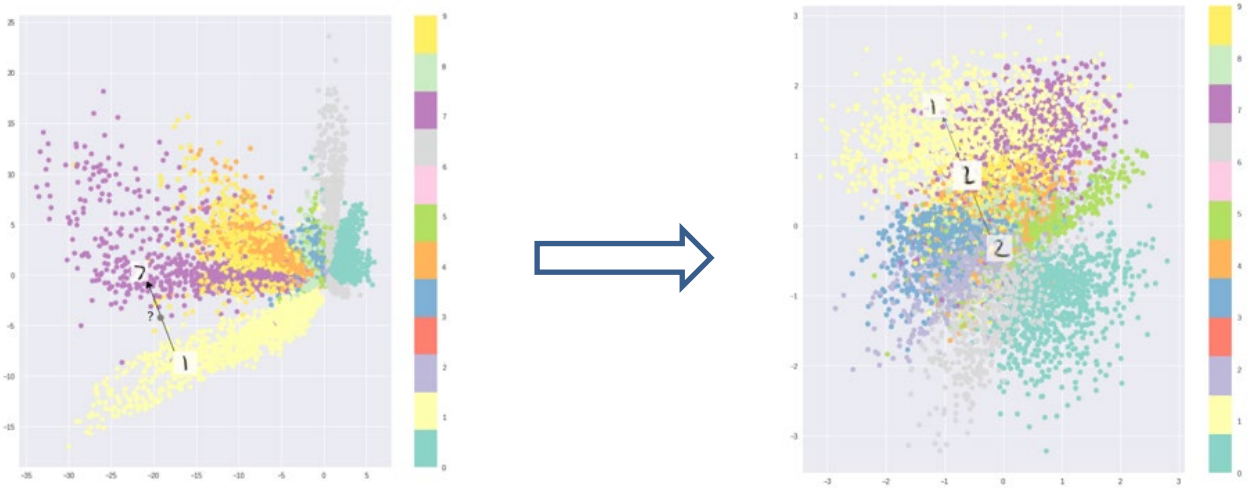
adapted from Irum Shafkat's (towards data science)

The intuition behind VAEs

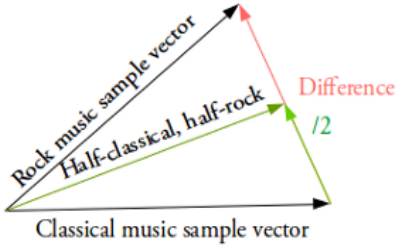


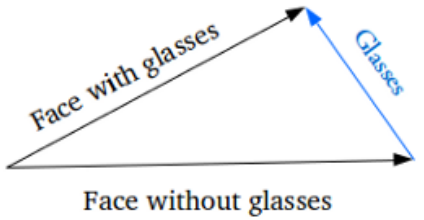
adapted from Irum Shafkat's (towards data science)

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)