Deep Learning

Lecture 11 – Autoencoders

Kumar Bipin

BE, MS, PhD (MMMTU, IISc, IIIT-Hyderabad)

Robotics, Computer Vision, Deep Learning, Machine Learning, System Software

Diederik P. Kingma

FOLLOW

Research Scientist, Google Brain
Verified email at google.com - Homepage

Machine Learning Deep Learning Neural Networks Variational Inference Identifiability

TITLE	CITED BY	YEAR
Auto-Encoding Variational Bayes DP Kingma, M Welling Proceedings of the 2nd International Conference on Learning Representations ... arXiv preprint arXiv:1606.03498	12897	2013

Ian Goodfellow

FOLLOW

Unknown affiliation
Verified email at cs.stanford.edu - Homepage

Deep Learning

TITLE	CITED BY	YEAR
Generative adversarial networks IJ Goodfellow, J Pouget-Abadie, M Mirza, B Xu, D Warde-Farley, S Ozair, ... arXiv preprint arXiv:1406.2661	27462	2014

Agenda

11.1
Latent Variable Models

Learning Problems

Supervised Learning:

- Learn model using dataset with **data-label pairs** $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$
- Examples: Classification, regression, structured prediction

Unsupervised Learning:

- Learn model using dataset **without labels** $\{\mathbf{x}_i\}_{i=1}^N$
- Examples: Clustering, dimensionality reduction, generative models
- Some of them use latent variables to capture structure $\rightarrow$ topic for today

Latent Variable Models

LVMs map between **observation space** $\mathbf{x} \in \mathbb{R}^D$ and **latent space** $\mathbf{z} \in \mathbb{R}^Q$:

$$(f_{\mathbf{w}} : \mathbf{x} \mapsto \mathbf{z}) \qquad g_{\mathbf{w}} : \mathbf{z} \mapsto \hat{\mathbf{x}}$$

- One **latent variable** gets associated with each data point in the training set
- The latent vectors are smaller than the observations $(Q < D) \Rightarrow$ **compression**
- Models are linear or non-linear, deterministic or stochastic, with/without encoder

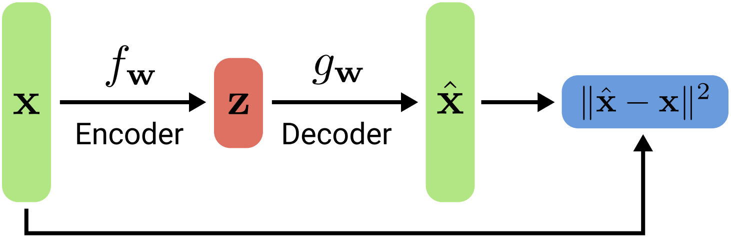
A little taxonomy:

	Deterministic	Probabilistic
Linear	Principle Component Analysis	Probabilistic PCA
Non-Linear w/ Encoder	Autoencoder	Variational Autoencoder
Non-Linear w/o Encoder		Gen. Adv. Networks

Autoencoders

Autoencoders comprise an **encoder** $f_{\mathbf{w}}$ as well as a **decoder** $g_{\mathbf{w}}$:

$$f_{\mathbf{w}} : \mathbf{x} \mapsto \mathbf{z} \qquad g_{\mathbf{w}} : \mathbf{z} \mapsto \hat{\mathbf{x}}$$

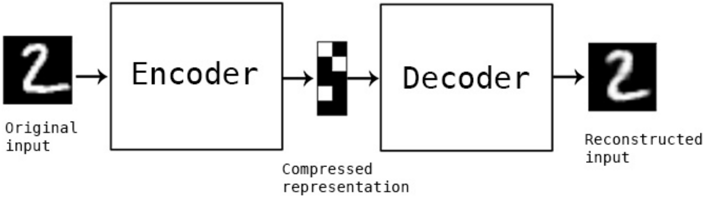


- Models of this type are called **autoencoders** as they predict their input as output
- In contrast, Generative adversarial networks (next lecture) only have a decoder $g_{\mathbf{w}}$

Autoencoders

Autoencoders comprise an **encoder** $f_{\mathbf{w}}$ as well as a **decoder** $g_{\mathbf{w}}$:

$$f_{\mathbf{w}} : \mathbf{x} \mapsto \mathbf{z} \qquad\qquad g_{\mathbf{w}} : \mathbf{z} \mapsto \hat{\mathbf{x}}$$



- Models of this type are called **autoencoders** as they predict their input as output
- In contrast, Generative adversarial networks (next lecture) only have a decoder $g_{\mathbf{w}}$

Generative Models

- Generative modeling is a broad area of machine learning which deals with models of **probability distributions** $p(\mathbf{x})$ over data points $\mathbf{x}$ (e.g., images)
- The generative model's task is to capture dependencies / structural regularities in the data (e.g., between pixels in images)
- **Generative latent variable models** capture the structure in **latent variables**
- Intuitively, we are trying to establish a **theory** for what we observe
- Some generative models (e.g., normalizing flows) allow for computing $p(\mathbf{x})$
- Others (e.g., VAEs) only approximate $p(\mathbf{x})$, but allow to draw samples from $p(\mathbf{x})$

Generative Latent Variable Models

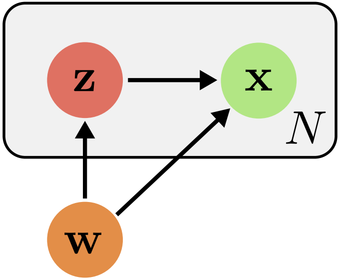
Generative latent variable models often consider a simple **Bayesian model:**

$$p(\mathbf{x}) = \int_{\mathbf{z}} \underbrace{p(\mathbf{z})p(\mathbf{x}|\mathbf{z})}_{\text{Generative Process}} \, d\mathbf{z} = \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})} p(\mathbf{x}|\mathbf{z})$$

- $p(\mathbf{z})$ is the **prior** over the **latent variable** $\mathbf{z} \in \mathbb{R}^Q$
- $p(\mathbf{x}|\mathbf{z})$ is the **likelihood** (= decoder that transforms $\mathbf{z}$ into a distribution over $\mathbf{x}$)
- $p(\mathbf{x})$ is the **marginal** of the joint distribution $p(\mathbf{x}, \mathbf{z})$

The goal is to maximize $p(\mathbf{x})$ for a given dataset $\mathcal{X}$ by learning the two models $p(\mathbf{z})$ and $p(\mathbf{x}|\mathbf{z})$ such that the latent variables $\mathbf{z}$ best capture the latent structure of the data.

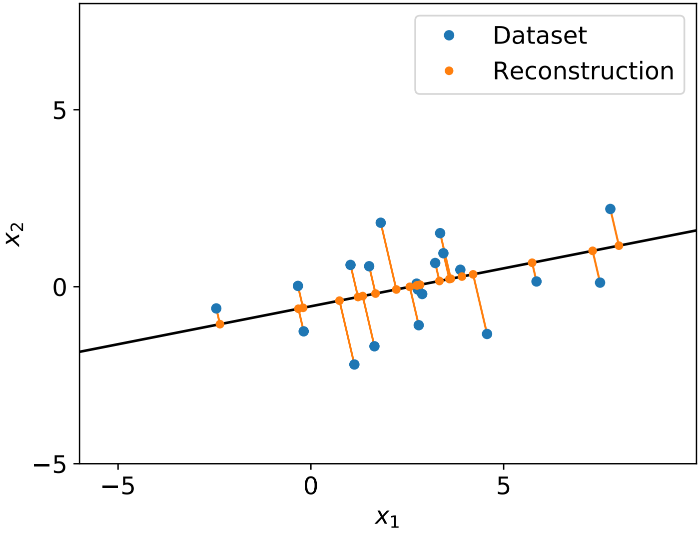
Generative Latent Variable Models



Representation as Graphical Model in Plate Notation:

- Variables inside plates are replicated (we have N data points to explain)
- Each data point $\mathbf{x}$ is associated with a latent variable $\mathbf{z}$
- In contrast, parameters are global (exist only once)
- Remark: We use a single $\mathbf{w}$ to refer to all model parameters

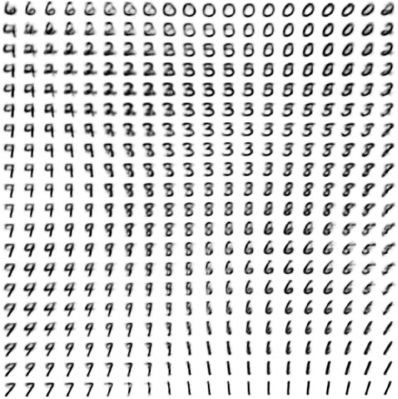
Example: 1D Manifold in 2D Space



Diederik P. Kingma, Max Welling: Auto-Encoding Variational Bayes. ICLR, 2014.

Example: Natural Image Manifolds

(a) Learned Frey Face manifold

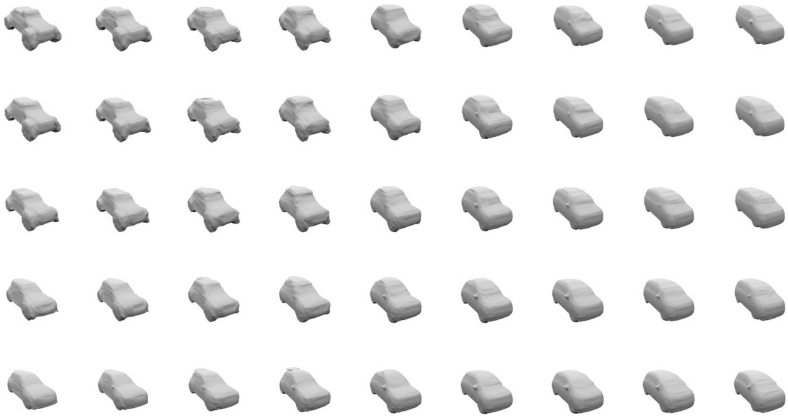


(b) Learned MNIST manifold

- Visualizing the latent space gives insights into the learned semantics

Diederik P. Kingma, Max Welling: Auto-Encoding Variational Bayes. ICLR, 2014.

Example: 3D Shape Manifolds



- We can also learn a latent space for 3D shapes and interpolate between them

Mescheder, Oechsle, Niemeyer, Nowozin and Geiger: Occupancy Networks: Learning 3D Reconstruction in Function Space. CVPR, 2019.

Example: Sentence Manifolds

“ i want to talk to you . ”
“i want to be with you . ”
“i do n’t want to be with you . ”
i do n’t want to be with you .
she did n’t want to be with him .
he was silent for a long moment .
he was silent for a moment .
it was quiet for a moment .
it was dark and cold .
there was a pause .
it was my turn .

- It is also possible to learn a latent space for sequences of words

Bowman, Vilnis, Vinyals, Dai, Józefowicz and Bengio: Generating Sentences from a Continuous Space. CoNLL, 2016.

11.2
Principal Component Analysis

Preliminaries

- Let $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)^\top \in \mathbb{R}^{N \times D}$ denote a **dataset** of observations $\mathbf{x}_i \in \mathbb{R}^D$
- Let $\mathbf{Z} = (\mathbf{z}_1, \dots, \mathbf{z}_N)^\top \in \mathbb{R}^{N \times Q}$ be the corresponding **latent variables** $\mathbf{z}_i \in \mathbb{R}^Q$
- While $\mathbf{X}$ is observed, $\mathbf{Z}$ is unobserved and needs to be inferred
- Typically, we assume $Q < D$, i.e., we want to obtain a **compressed** representation
- In PCA, our goal is to learn a **linear bidirectional mapping** $\mathcal{Z} \leftrightarrow \mathcal{X}$ such that as much information of $\mathcal{X}$ as possible is retained in $\mathcal{Z}$
- In other words, we want to encode $\mathbf{x} \to \mathbf{z}$ such that if we decode $\mathbf{z} \to \hat{\mathbf{x}}$, then $\hat{\mathbf{x}}$ is a good approximation to the original $\mathbf{x}$ (in most cases $\hat{\mathbf{x}} \neq \mathbf{x}$)

Principal Component Analysis

Let us assume the following **linear mapping** from latent space to observation space

$$\hat{\mathbf{x}}_i = \bar{\mathbf{x}} + \sum_{j=1}^{Q} z_{ij} \mathbf{v}_j$$

where $\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$ is the **data mean** and $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_Q)$ an **orthonormal basis.**

Our goal is to minimize the L_2 **reconstruction loss** wrt. $\mathbf{Z}$ and $\mathbf{V}$:

$$\mathcal{L}(\mathbf{Z}, \mathbf{V}) = \sum_{i=1}^{N} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|^2 = \sum_{i=1}^{N} \| \underbrace{\bar{\mathbf{x}} + \sum_{j=1}^{Q} z_{ij} \mathbf{v}_j}_{= \hat{\mathbf{x}}_i} - \mathbf{x}_i \|^2$$

Principal Component Analysis

Considering that $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_Q)$ is an **orthonormal basis,** we expand $\mathcal{L}$ as follows:

$$
\begin{aligned}
\mathcal{L}(\mathbf{Z}, \mathbf{V}) &= \sum_{i=1}^{N} \|\bar{\mathbf{x}} + \sum_{j=1}^{Q} z_{ij}\mathbf{v}_j - \mathbf{x}_i\|^2 \\
&= \sum_{i=1}^{N} \| \sum_{j=1}^{Q} z_{ij}\mathbf{v}_j + \bar{\mathbf{x}} - \mathbf{x}_i\|^2 \\
&= \sum_{i=1}^{N} \left[\sum_{j=1}^{Q} z_{ij}^2 + 2\sum_{j=1}^{Q} z_{ij}\mathbf{v}_j^\top (\bar{\mathbf{x}} - \mathbf{x}_i) + \|\bar{\mathbf{x}} - \mathbf{x}_i\|^2 \right]
\end{aligned}
$$

Principal Component Analysis

The **reconstruction loss** can therefore be minimized in closed form wrt. $\mathbf{Z}$:

$$\mathcal{L}(\mathbf{Z},\mathbf{V}) = \sum_{i=1}^{N}\left[\sum_{j=1}^{Q}\left[z_{ij}^2 + 2z_{ij}\mathbf{v}_j^\top(\bar{\mathbf{x}} - \mathbf{x}_i)\right] + \|\bar{\mathbf{x}} - \mathbf{x}_i\|^2\right]$$

$$\frac{\partial \mathcal{L}(\mathbf{Z},\mathbf{V})}{\partial z_{ij}} = 2z_{ij} + 2\mathbf{v}_j^\top(\bar{\mathbf{x}} - \mathbf{x}_i) \overset{!}{=} 0$$

$$\Rightarrow \quad z_{ij}^* = -\mathbf{v}_j^\top(\bar{\mathbf{x}} - \mathbf{x}_i)$$

For $\mathbf{Z} = \mathbf{Z}^*$, the reconstruction loss simplifies to:

$$\mathcal{L}(\mathbf{Z}^*,\mathbf{V}) = \sum_{i=1}^{N}\left[-\sum_{j=1}^{Q} {z_{ij}^*}^2 + \|\bar{\mathbf{x}} - \mathbf{x}_i\|^2\right]$$

Principal Component Analysis

The **reconstruction loss** at $z_{ij}^* = -\mathbf{v}_j^\top(\bar{\mathbf{x}} - \mathbf{x}_i)$ can be rewritten as

$$\mathcal{L}(\mathbf{Z}^*, \mathbf{V}) = \sum_{i=1}^{N} \left[-\sum_{j=1}^{Q} {z_{ij}^*}^2 + \|\bar{\mathbf{x}} - \mathbf{x}_i\|^2 \right]$$
$$= -\sum_{j=1}^{Q} \mathbf{v}_j^\top \mathbf{S} \mathbf{v}_j + \sum_{i=1}^{N} \|\bar{\mathbf{x}} - \mathbf{x}_i\|^2$$

with $\mathbf{S}$ the **scatter matrix** (unnormalized sample covariance matrix) of $\mathbf{x}$:

$$\mathbf{S} = \sum_{i=1}^{N} (\bar{\mathbf{x}} - \mathbf{x}_i)(\bar{\mathbf{x}} - \mathbf{x}_i)^\top$$

Principal Component Analysis

To enforce $\|\mathbf{v}_j\| = 1$, we introduce **Lagrange multipliers** λ_j into the loss:

$$\mathcal{L}(\mathbf{Z}^*, \mathbf{V}, \boldsymbol{\lambda}) = -\sum_{j=1}^{Q} \mathbf{v}_j^\top \mathbf{S} \mathbf{v}_j + \sum_{i=1}^{N} \|\bar{\mathbf{x}} - \mathbf{x}_i\|^2 + \sum_{j=1}^{Q} \lambda_j (\mathbf{v}_j^\top \mathbf{v}_j - 1)$$

$$\frac{\partial \mathcal{L}(\mathbf{Z}^*, \mathbf{V}, \boldsymbol{\lambda})}{\partial \mathbf{v}_j} = -2\mathbf{S}\mathbf{v}_j + 2\lambda_j \mathbf{v}_j \stackrel{!}{=} 0$$

$$\Rightarrow \mathbf{S}\mathbf{v}_j = \lambda_j \mathbf{v}_j$$

We see that $\{\boldsymbol{\lambda}, \mathbf{V}\}$ is the solution to an **eigenvalue problem.**

We also observe that as we have ${\mathbf{v}_j}^\top \mathbf{S}\mathbf{v}_j = \lambda_j {\mathbf{v}_j}^\top \mathbf{v}_j = \lambda_j$, the loss $\mathcal{L}$ is minimized by choosing (for $\mathbf{V}$) the eigenvectors $\mathbf{v}_j$ of $\mathbf{S}$ corresponding to the top Q eigenvalues.

Principal Component Analysis

Consider again the **linear model** that we started with:

$$\hat{\mathbf{x}}_i = \bar{\mathbf{x}} + \sum_{j=1}^{Q} z_{ij}\mathbf{v}_j$$

Both the PCA decoder and encoder are simple **linear mappings:**

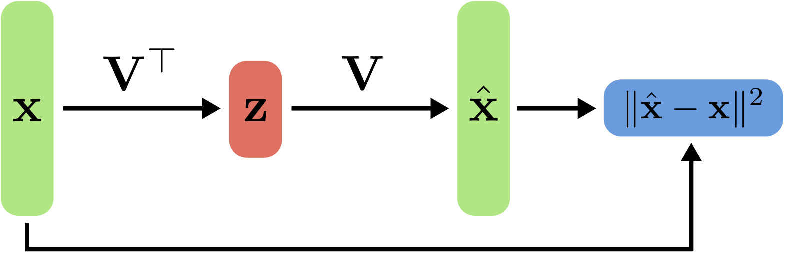
Decoder:

$$\mathbf{x} = \mathbf{V}\mathbf{z} + \bar{\mathbf{x}}$$

Encoder:

$$\mathbf{z} = \mathbf{V}^\top(\mathbf{x} - \bar{\mathbf{x}})$$

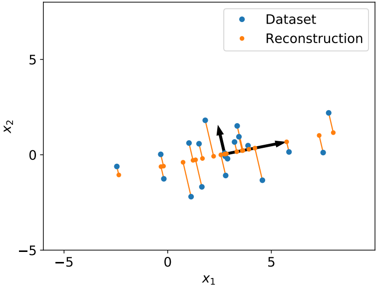
Principal Component Analysis

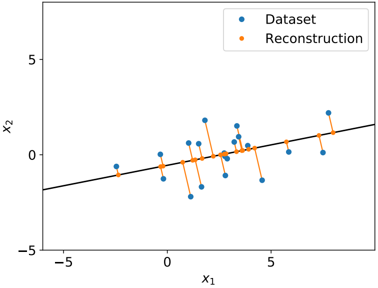


PCA Recipe:

- Given a dataset $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ of observations $\mathbf{x}_i \in \mathbb{R}^D$
- Compute the **data mean** $\bar{\mathbf{x}}$ and **scatter matrix** $\mathbf{S} = \sum_{i=1}^N (\bar{\mathbf{x}} - \mathbf{x}_i)(\bar{\mathbf{x}} - \mathbf{x}_i)^\top$
- Compute the **eigen decomposition** of $\mathbf{S}$
- Select the Q eigenvectors corresponding to the Q largest eigenvalues for $\mathbf{V}$

Principal Component Analysis





There are 2 perspectives on PCA:

- We saw that PCA can be motivated by **minimizing the L_2 reconstruction error**
- However, we can also motivate PCA by **maximizing the variance** of latent points
- In other words, we like to find an embedding that captures most of the variation in the original dataset while using a smaller dimensionality $Q \ll D$

Variance Maximization Perspective

Consider the following (one-dimensional) encoding of vector $\mathbf{x}$:

$$\mathbf{z} = \mathbf{v}^\top(\mathbf{x} - \bar{\mathbf{x}})$$

Our goal is to **maximize variance** in latent space:

$$\begin{aligned}
\mathrm{Var}(\mathbf{z}) &= \mathbb{E}\left[\left(\mathbf{v}^\top(\mathbf{x} - \bar{\mathbf{x}}) - \mathbb{E}\left[\mathbf{v}^\top(\mathbf{x} - \bar{\mathbf{x}})\right]\right)^2\right] \\
&= \mathbb{E}\left[\left(\mathbf{v}^\top(\mathbf{x} - \bar{\mathbf{x}})\right)^2\right] \qquad \text{(as } \mathbb{E}[\mathbf{x}] = \bar{\mathbf{x}}\text{)} \\
&= \mathbb{E}\left[\mathbf{v}^\top(\bar{\mathbf{x}} - \mathbf{x})(\bar{\mathbf{x}} - \mathbf{x})^\top\mathbf{v}\right] \\
&\propto \mathbf{v}^\top\mathbf{S}\mathbf{v} \qquad \text{(as } \mathbf{S} \text{ is not normalized)}
\end{aligned}$$

Variance Maximization Perspective

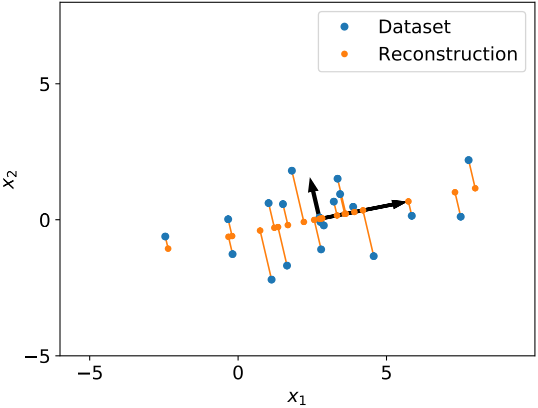
Let us now again assume an orthonormal basis $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_Q)$ of dimension Q. **Maximizing the sum of variances** along each dimension subject to normalization constraints leads to the optimization problem we are already familiar with:

$$\boldsymbol{\lambda}^*, \mathbf{V}^* = \underset{\boldsymbol{\lambda}, \mathbf{V}}{\operatorname{argmax}} \sum_{j=1}^{Q} \mathbf{v}_j^\top \mathbf{S} \mathbf{v}_j + \sum_{j=1}^{Q} \lambda_j (\mathbf{v}_j^\top \mathbf{v}_j - 1)$$

A solution is given by the Q largest eigenvalues and corresponding eigenvectors of $\mathbf{S}$. Remark: ${\mathbf{v}_j}^\top \mathbf{S} \mathbf{v}_j = \lambda_j {\mathbf{v}_j}^\top \mathbf{v}_j = \lambda_j$ is the **variance** along the j'th principal component if the scatter matrix $\mathbf{S}$ is normalized by the number of data points (=covariance matrix).

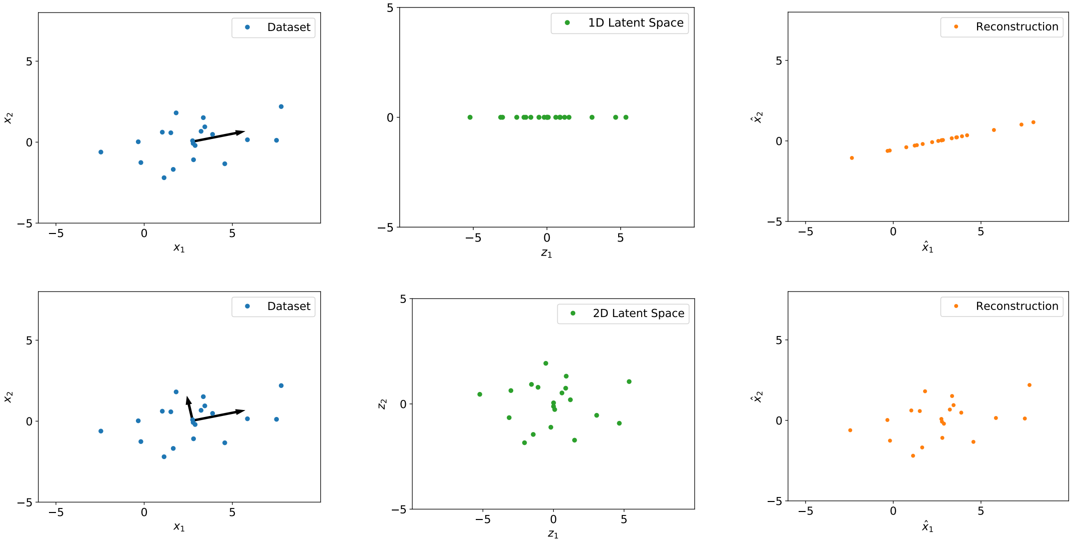
Examples

Results on Synthetic 2D Dataset

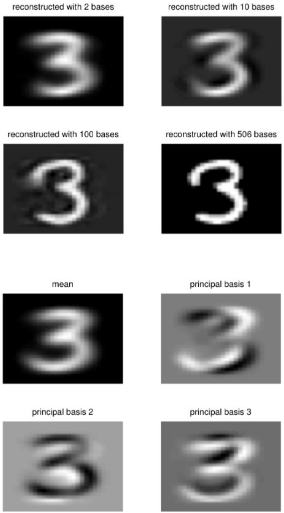


- PCA on a dataset with $N = 20$, $D = 2$ and $Q = 1$ (projection onto 1D subspace)
- The two eigenvectors $\mathbf{v}_j$ shown in black are scaled by $\sqrt{\lambda_j}$

Results on Synthetic 2D Dataset



Results on MNIST



Results on Faces

PCA on Face Images:

- PCA on 2429 19x19 grayscales images (CBCL data)
- Yields good reconstructions with only 3 components:

- We can apply a classifier directly on this latent representation
- PCA with 3 components obtains 79 % accuracy on face/non-face discrimination on test data vs. 76.8 % for a Gaussian Mixture Model with 84 states
- Also commonly used for analyzing the latent properties of a dataset

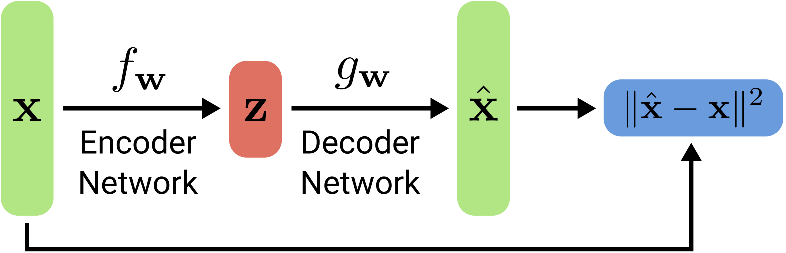
Turk and Pentland: Face recognition using eigenfaces. CVPR, 1991.

Learned Basis (Eigenfaces)

Turk and Pentland: Face recognition using eigenfaces. CVPR, 1991.

11.3
Autoencoders

Autoencoders



- An autoencoder is a **neural network** whose outputs are its own inputs
- The input $\mathbf{x} \in \mathbb{R}^D$ is compressed to a latent code $\mathbf{z} \in \mathbb{R}^Q$
- The goal is to **minimize the reconstruction error** (as in PCA)

PCA as Special Case of Autoencoders

Let $f_{\mathbf{w}} : \mathbf{x} \mapsto \mathbf{z}$ denote the encoder and $g_{\mathbf{w}} : \mathbf{z} \mapsto \hat{\mathbf{x}}$ the decoder network.

Let's assume both mappings to be **linear** (without activation function):

$$\mathbf{z} = f_{\mathbf{w}}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{a} \qquad \hat{\mathbf{x}} = g_{\mathbf{w}}(\mathbf{z}) = \mathbf{B}\mathbf{z} + \mathbf{b}$$

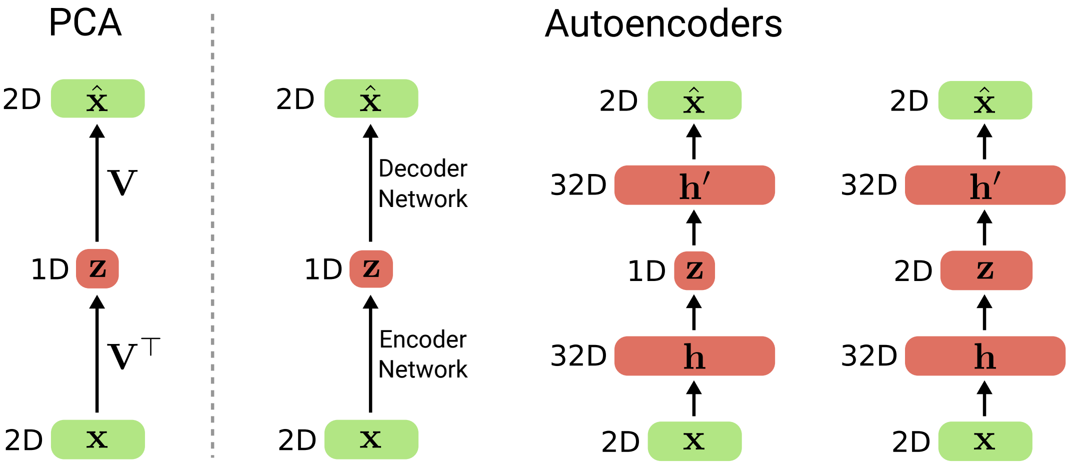
In this case, we have:

$$\hat{\mathbf{x}} = g_{\mathbf{w}}(f_{\mathbf{w}}(\mathbf{x})) = \underbrace{\mathbf{B}\mathbf{A}}_{=\mathbf{C}}\mathbf{x} + \underbrace{\mathbf{a} + \mathbf{b}}_{=\mathbf{c}}$$

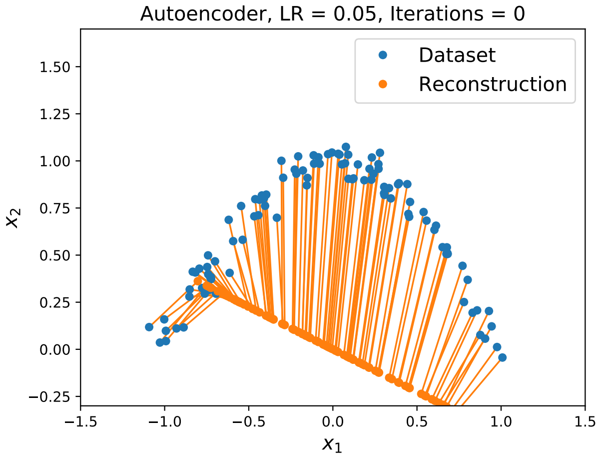
Thus, the optimal solution $\mathbf{w}^*$ is given by **PCA:**

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{N} \| \underbrace{\mathbf{C}\mathbf{x}_i + \mathbf{c}}_{=\hat{\mathbf{x}}_i} - \mathbf{x}_i \|^2$$

Results on Synthetic 2D Dataset

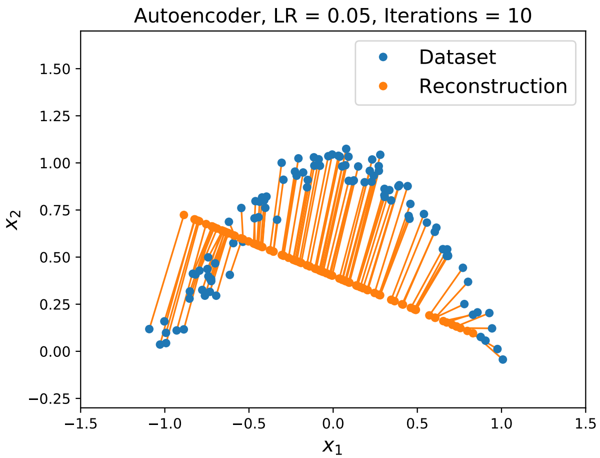


Results on Synthetic 2D Dataset



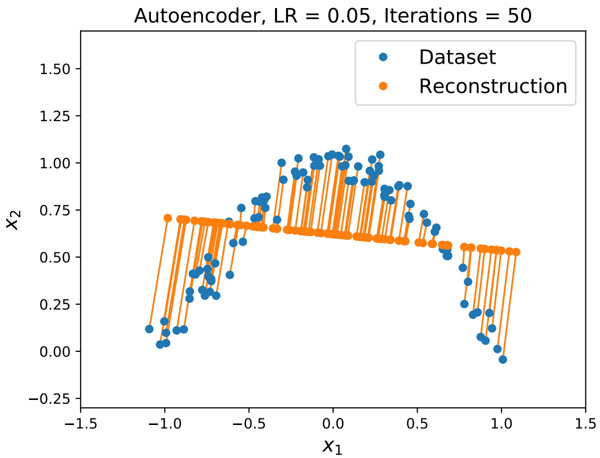
- **Autoencoder reconstructions** using one linear encoder/decoder layer with $Q = 1$

Results on Synthetic 2D Dataset



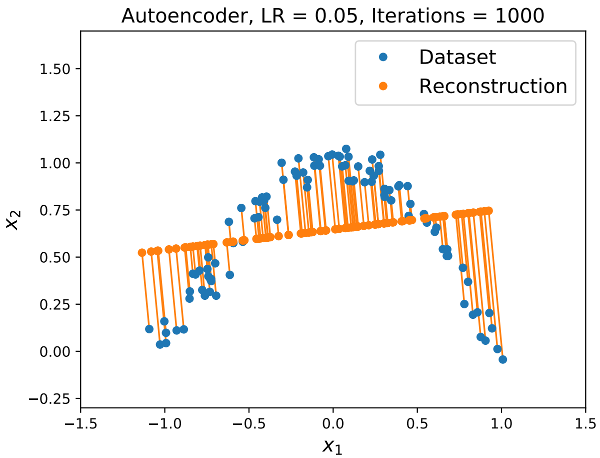
- **Autoencoder reconstructions** using one linear encoder/decoder layer with $Q = 1$

Results on Synthetic 2D Dataset



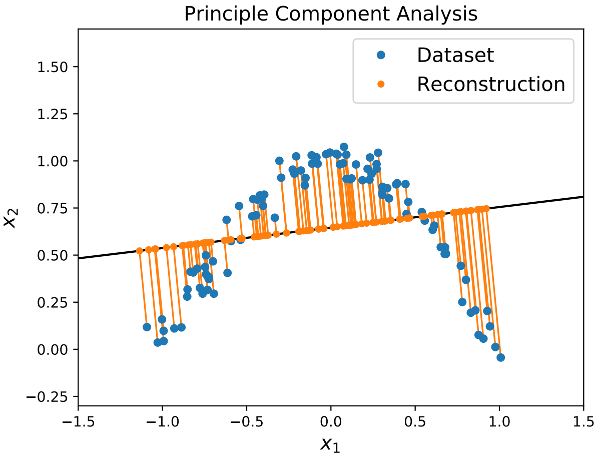
- **Autoencoder reconstructions** using one linear encoder/decoder layer with $Q = 1$

Results on Synthetic 2D Dataset



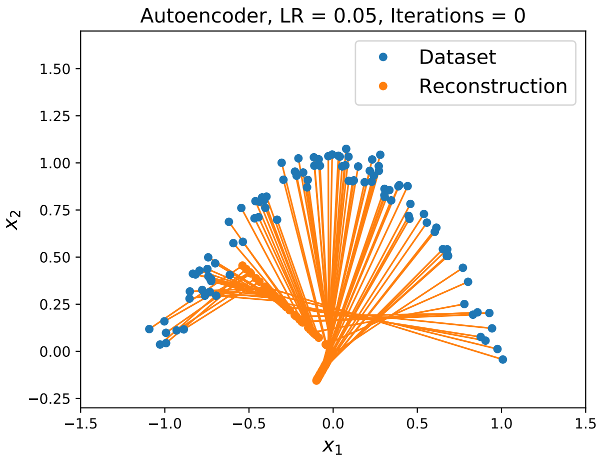
- **Autoencoder reconstructions** using one linear encoder/decoder layer with $Q = 1$

Results on Synthetic 2D Dataset



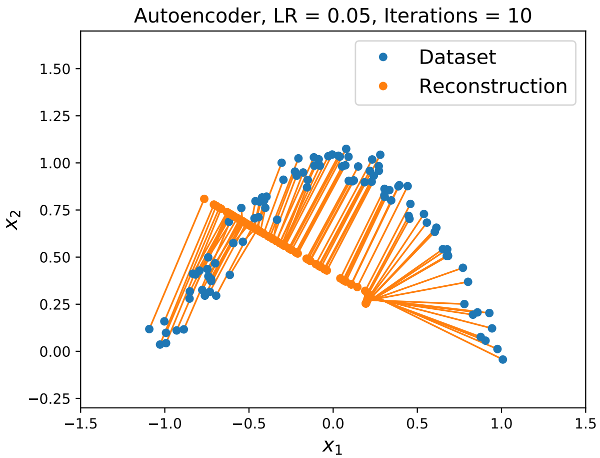
- **PCA reconstructions** on the same dataset with $Q = 1$

Results on Synthetic 2D Dataset



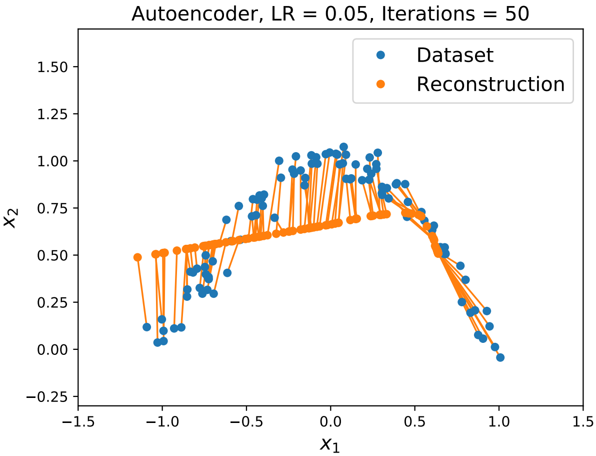
- **Autoencoder reconstructions** using 32 dimensional hidden layers and $Q = 1$

Results on Synthetic 2D Dataset



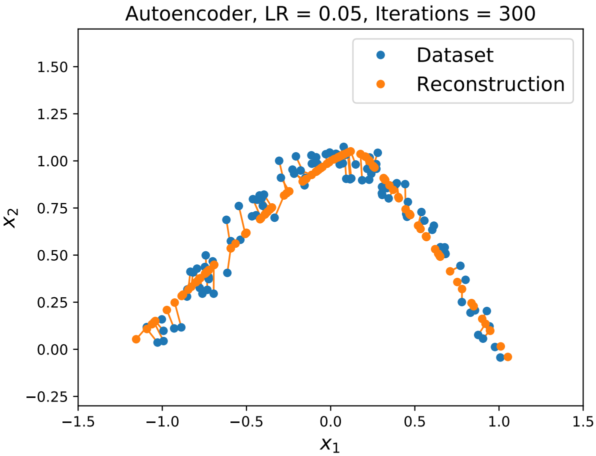
- **Autoencoder reconstructions** using 32 dimensional hidden layers and $Q = 1$

Results on Synthetic 2D Dataset



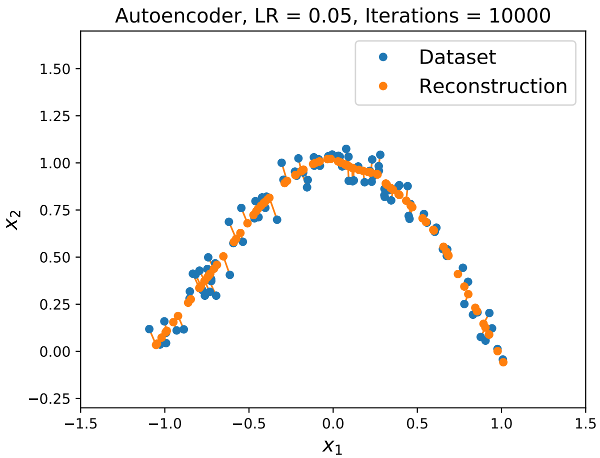
- **Autoencoder reconstructions** using 32 dimensional hidden layers and $Q = 1$

Results on Synthetic 2D Dataset



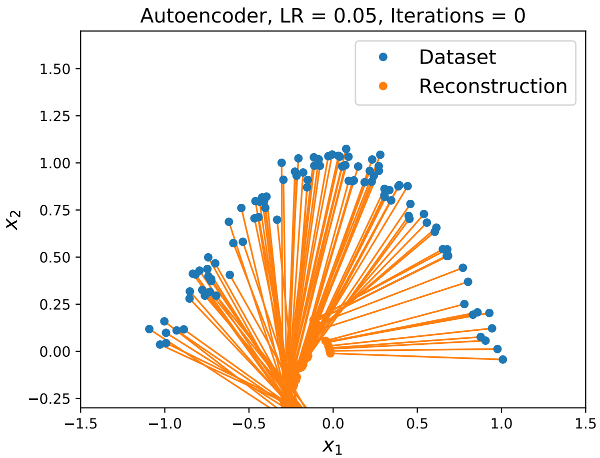
- **Autoencoder reconstructions** using 32 dimensional hidden layers and $Q = 1$

Results on Synthetic 2D Dataset



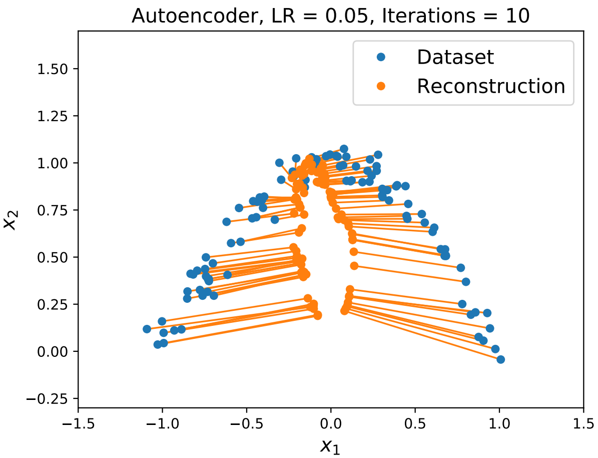
- **Autoencoder reconstructions** using 32 dimensional hidden layers and $Q = 1$

Results on Synthetic 2D Dataset



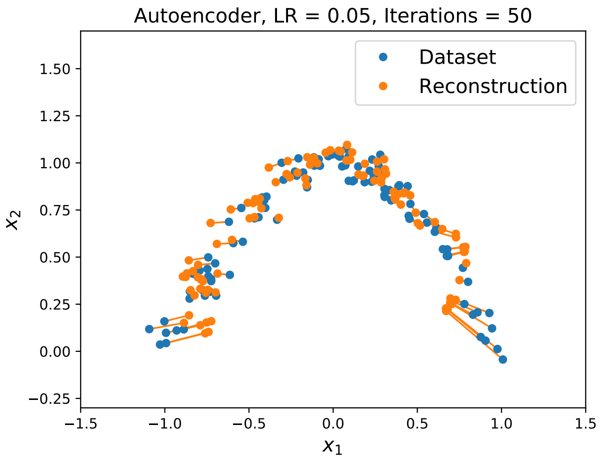
- **Autoencoder reconstructions** using 32 dimensional hidden layers and $Q = 2$

Results on Synthetic 2D Dataset



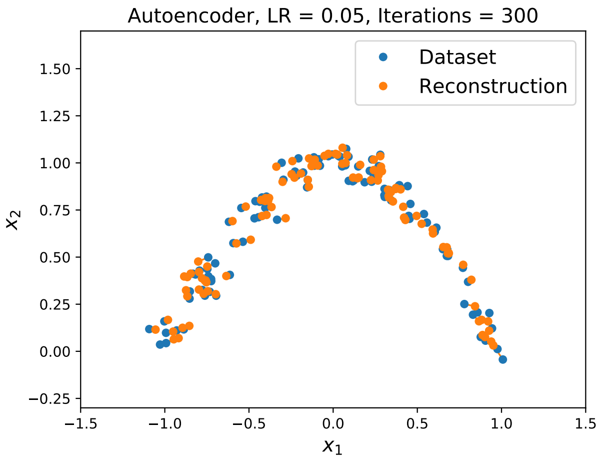
- **Autoencoder reconstructions** using 32 dimensional hidden layers and $Q = 2$

Results on Synthetic 2D Dataset



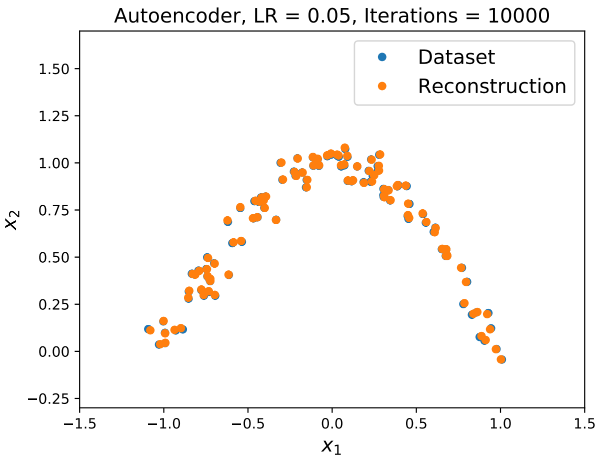
- **Autoencoder reconstructions** using 32 dimensional hidden layers and $Q = 2$

Results on Synthetic 2D Dataset



- **Autoencoder reconstructions** using 32 dimensional hidden layers and $Q = 2$

Results on Synthetic 2D Dataset



- **Autoencoder reconstructions** using 32 dimensional hidden layers and $Q = 2$

Comparing Reconstructions

- Deep autoencoders learn **non-linear** latent embeddings
- They often have **smaller reconstruction errors** compared to PCA with same Q
- In contrast, PCA always learns the best linear mapping

Denoising Autoencoders

- **Denoising Autoencoders** take noisy inputs and predict the original noise-free data
- Higher level representations are relatively stable and robust to input corruption
- Encourages the model to **generalize better** and capture useful structure
- Similar to data augmentation (except that the “label” is the noise-free input)
- `https://blog.keras.io/building-autoencoders-in-keras.html`

11.4
Variational Autoencoders

A Bayesian Generative Latent Variable Model

So far, we have discussed deterministic latent variables. We will now take a **probabilistic perspective** on **latent variable models** with autoencoding properties. Consider the following **Bayesian model** of the data $\mathbf{x} \in \mathbb{R}^D$:

$$p(\mathbf{x}) = \int_{\mathbf{z}} \underbrace{p(\mathbf{z})p(\mathbf{x}|\mathbf{z})}_{\text{Generative Process}} \, d\mathbf{z} = \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})} p(\mathbf{x}|\mathbf{z})$$

- $p(\mathbf{z})$ is the **prior** over the **latent variable** $\mathbf{z} \in \mathbb{R}^Q$
- $p(\mathbf{x}|\mathbf{z})$ is the **likelihood**
- $p(\mathbf{x})$ is the **marginal** of the joint distribution $p(\mathbf{x}, \mathbf{z})$

A Bayesian Generative Latent Variable Model

Assumptions:

- We assume the **prior** model $p(\mathbf{z})$ to be samplable and computable
- We assume the **likelihood** model $p(\mathbf{x}|\mathbf{z})$ to be computable
- In other words, we can **sample** from $p(\mathbf{z})$ and we can **compute** the probability mass/density of $p(\mathbf{z})$ and $p(\mathbf{x}|\mathbf{z})$ for any given $\mathbf{x}$ and $\mathbf{z}$
- These assumptions hold for autoregressive models (e.g., language models)
- However, they fail for loopy graphical models where approximations must be used
- We will choose **simple parameteric distributions** to achieve this

A Bayesian Generative Latent Variable Model

To find the model parameters $\mathbf{w}$, we want to minimize the **negative log likelihood:**

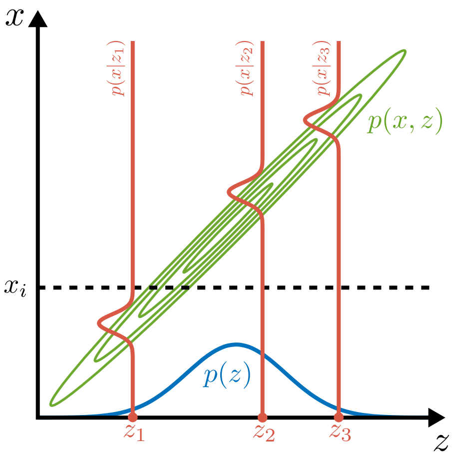
$$
\begin{aligned}
\mathbf{w}^* &= \underset{\mathbf{w}}{\operatorname{argmin}}\ \mathbb{E}_{\mathbf{x}\sim p_{data}}\left[-\log p_{\mathbf{w}}(\mathbf{x})\right] \\
&= \underset{\mathbf{w}}{\operatorname{argmin}}\ \mathbb{E}_{\mathbf{x}\sim p_{data}}\left[-\log \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{w}}(\mathbf{z})} p_{\mathbf{w}}(\mathbf{x}|\mathbf{z})\right] \\
&= \underset{\mathbf{w}}{\operatorname{argmin}}\ \sum_{i=1}^{N} -\log \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{w}}(\mathbf{z})} p_{\mathbf{w}}(\mathbf{x}_i|\mathbf{z})
\end{aligned}
$$

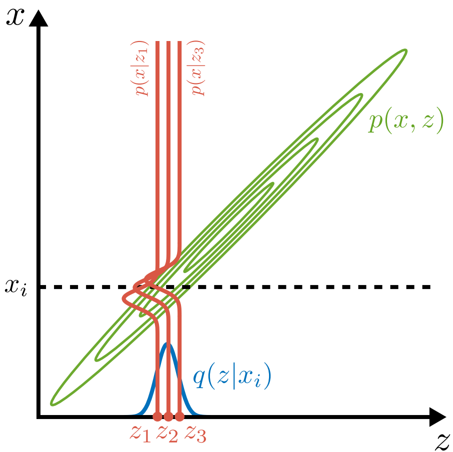
- Unfortunately, even given our assumption, computing $p_{\mathbf{w}}(\mathbf{x})$ is **intractable**
- VAEs side-step this by introducing another model component, a so-called **recognition model** $q_{\mathbf{w}}(\mathbf{z}|\mathbf{x})$ to approximate the true posterior $p_{\mathbf{w}}(\mathbf{z}|\mathbf{x})$

Intuition for Intractability

- Imagine the observations are sound waves and the latents are word sequences
- We are looking for a "theory" of sound waves that best explains them
- We want to minimize the cross entropy between the data distribution over sound waves and our model distribution of the sound waves
- However, the marginal $p_{\mathbf{w}}(\mathbf{x})$ is intractable due to the large search space
- If we listen to a song, it is sometimes unclear what the lyrics are
- If someone tells us the lyrics, we can suddenly hear it / verify it
- But the search over the word sequences that explain the sound waves is hard as there are many sequences and we might not think of the right one
- VAEs thus use a recognition model $q_{\mathbf{w}}(\mathbf{z}|\mathbf{x})$ that computes the word sequence
- This model does not need to be correct, it is an approximation to $p(\mathbf{z}|\mathbf{x})$

Intuition for Intractability





- Computing $p_{\mathbf{w}}(\mathbf{x}) = \mathbb{E}_{\mathbf{z}}\, p_{\mathbf{w}}(\mathbf{x}|\mathbf{z})$ is hard, in particular in high dimensions

The Evidence Lower Bound (ELBO)

We seek a **tractable lower bound** to the likelihood:

$$
\begin{aligned}
\log p(\mathbf{x}) &= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})} \log \frac{p(\mathbf{x})p(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})} \\
&= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})} \log \frac{p(\mathbf{z},\mathbf{x})}{q(\mathbf{z}|\mathbf{x})} + \log \frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})} \\
&= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})} \log \frac{p(\mathbf{z},\mathbf{x})}{q(\mathbf{z}|\mathbf{x})} + \underbrace{KL(q(\mathbf{z}|\mathbf{x}), p(\mathbf{z}|\mathbf{x}))}_{\geq 0} \\
&\geq \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})} \log \frac{p(\mathbf{z},\mathbf{x})}{q(\mathbf{z}|\mathbf{x})} \qquad \text{(ELBO)}
\end{aligned}
$$

- In practice, $q(\mathbf{z}|\mathbf{x})$ is a **variational approximation** to the true posterior $p(\mathbf{z}|\mathbf{x})$
- Therefore, the ELBO is sometimes also called **variational lower bound**
- The divergence term $KL(q(\mathbf{z}|\mathbf{x}), p(\mathbf{z}|\mathbf{x}))$ measures the **approximation error**

The Evidence Lower Bound (ELBO)

The **negative log likelihood** is thus **upper bounded** by:

$$
\begin{aligned}
-\log p(\mathbf{x}) &\leq \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})} \log \frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{z},\mathbf{x})} \\
&= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})} \log \frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})p(\mathbf{x}|\mathbf{z})} \\
&= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})} \log \frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})} - \log p(\mathbf{x}|\mathbf{z}) \\
&= KL(q(\mathbf{z}|\mathbf{x}), p(\mathbf{z})) + \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})} \left[-\log p(\mathbf{x}|\mathbf{z})\right]
\end{aligned}
$$

- The bound comprises a **KL divergence** and a **conditional log likelihood**
- Note how $q(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{x}|\mathbf{z})$ act as an **autoencoder** model: $\mathbf{x} \xrightarrow{q} \mathbf{z} \xrightarrow{p} \mathbf{x}$

Variational Autoencoder (VAE)

Let $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}_{i=1}^N$ with $\mathbf{x}_i \in \mathbb{R}^D$ be a dataset and $\mathbf{w}$ the model parameters.
The **Variational Autoencoder** minimizes this bound to the **negative log likelihood:**

$$\begin{aligned}
\mathbf{w}^* &= \underset{\mathbf{w}}{\operatorname{argmin}}\ \mathbb{E}_{\mathbf{x}\sim p_{data}}\left[-\log p_{\mathbf{w}}(\mathbf{x})\right] \\
&= \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{N} \left[-\log p_{\mathbf{w}}(\mathbf{x}_i)\right] \\
&\approx \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{N} \underbrace{KL(q_{\mathbf{w}}(\mathbf{z}|\mathbf{x}_i), p(\mathbf{z}))}_{\text{Approx. Posterior = Prior}} + \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\mathbf{w}}(\mathbf{z}|\mathbf{x}_i)}\left[-\log p_{\mathbf{w}}(\mathbf{x}_i|\mathbf{z})\right]}_{\text{Reconstruction Term}}
\end{aligned}$$

- In a VAE, $q_{\mathbf{w}}(\mathbf{z}|\mathbf{x})$ is a **multivariate Gaussian** parameterized by a neural network
- It thus makes a **variational approximation** $q_{\mathbf{w}}(\mathbf{z}|\mathbf{x})$ to the true posterior $p(\mathbf{z}|\mathbf{x})$

Diederik P. Kingma, Max Welling: Auto-Encoding Variational Bayes. ICLR, 2014.

Variational Autoencoder (VAE)

Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}_{i=1}^N$ with $\mathbf{x}_i \in \mathbb{R}^D$ be a dataset and $\mathbf{w}$ the model parameters.

The **Variational Autoencoder** minimizes this bound to the **negative log likelihood:**

$$\begin{aligned}
\mathbf{w}^* &= \underset{\mathbf{w}}{\operatorname{argmin}} \; \mathbb{E}_{\mathbf{x} \sim p_{data}} \left[-\log p_{\mathbf{w}}(\mathbf{x}) \right] \\
&= \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{N} \left[-\log p_{\mathbf{w}}(\mathbf{x}_i) \right] \\
&\approx \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{N} \underbrace{KL(q_{\mathbf{w}}(\mathbf{z}|\mathbf{x}_i), p(\mathbf{z}))}_{\text{Approx. Posterior = Prior}} + \underbrace{\mathbb{E}_{\mathbf{z} \sim q_{\mathbf{w}}(\mathbf{z}|\mathbf{x}_i)} \left[-\log p_{\mathbf{w}}(\mathbf{x}_i|\mathbf{z}) \right]}_{\text{Reconstruction Term}}
\end{aligned}$$

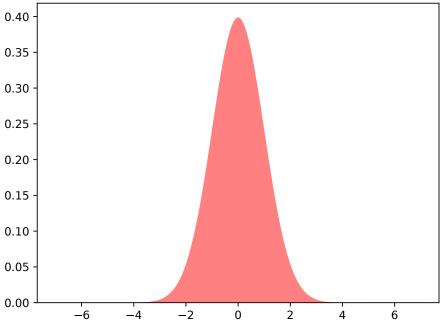
- The **likelihood model** $p_{\mathbf{w}}(\mathbf{x}_i|\mathbf{z})$ is also parameterized by a neural network
- For binary observations **Bernoulli**, for real observations **Gaussian** or **Laplacian**

Neural Network Parameterization

Let us consider a **Gaussian recognition model:**

$$q_{\mathbf{w}}(\mathbf{z}|\mathbf{x}) = \frac{1}{(2\pi)^{Q/2}} \frac{1}{|\boldsymbol{\Sigma}_{\mathbf{w}}(\mathbf{x})|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{z} - \boldsymbol{\mu}_{\mathbf{w}}(\mathbf{x}))^\top \boldsymbol{\Sigma}_{\mathbf{w}}(\mathbf{x})^{-1}(\mathbf{z} - \boldsymbol{\mu}_{\mathbf{w}}(\mathbf{x}))\right)$$

- The mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ are functions of $\mathbf{x}$ and with parameters $\mathbf{w}$
- In a VAE, these functions are implemented using a **neural network** (e.g., MLP)
- They often have a **shared backbone**
- Typically, we use $\boldsymbol{\Sigma}_{\mathbf{w}}(\mathbf{x}) = \mathrm{diag}(\boldsymbol{\sigma}^2_{\mathbf{w}}(\mathbf{x}))$



Diederik P. Kingma, Max Welling: Auto-Encoding Variational Bayes. ICLR, 2014.

Recognition Model and Prior

Assume a **Gaussian recognition model** $q(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ and **prior** $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$.
(note that for clarity, we drop the dependency of q, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ on $\mathbf{x}$ and $\mathbf{w}$)

$$\int q(\mathbf{z}) \log q(\mathbf{z})\, d\mathbf{z} = \int \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2) \log \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2)\, d\mathbf{z} = -\frac{Q}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(1 + \log \sigma_j^2)$$

$$\int q(\mathbf{z}) \log p(\mathbf{z})\, d\mathbf{z} = \int \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2) \log \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})\, d\mathbf{z} = -\frac{Q}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(\mu_j^2 + \sigma_j^2)$$

$$KL(q(\mathbf{z}), p(\mathbf{z})) = \int q(\mathbf{z})(\log q(\mathbf{z}) - \log p(\mathbf{z}))\, d\mathbf{z} = \frac{1}{2}\sum_{j=1}^{J}(\mu_j^2 + \sigma_j^2 - 1 - \log \sigma_j^2)$$

The KL term has a simple **analytical solution** in this case. This is the standard setup.
The **covariance matrix** $\boldsymbol{\Sigma} = \mathrm{diag}(\boldsymbol{\sigma}^2)$ of the recognition model is chosen **diagonal.**

Diederik P. Kingma, Max Welling: Auto-Encoding Variational Bayes. ICLR, 2014.

Learning a VAE

Variational Autoencoder Objective:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{N} \underbrace{KL(q_{\mathbf{w}}(\mathbf{z}|\mathbf{x}_i), p(\mathbf{z}))}_{\text{Approx. Posterior = Prior}} + \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\mathbf{w}}(\mathbf{z}|\mathbf{x}_i)}\left[-\log p_{\mathbf{w}}(\mathbf{x}_i|\mathbf{z})\right]}_{\text{Reconstruction Term}}$$

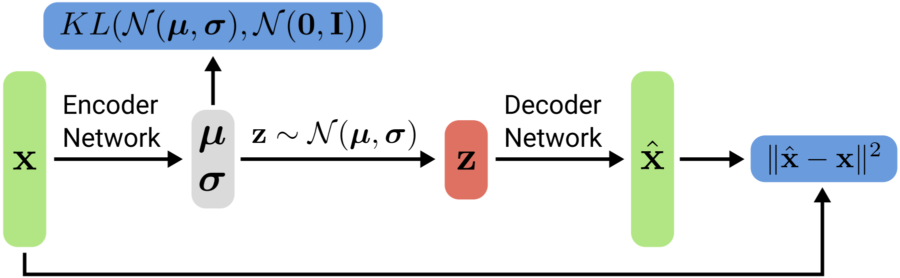
- The gradients for the **KL term** wrt. $\mathbf{w}$ are easily obtained using backpropagation
- For the **reconstruction term,** the forward pass can be computed using sampling, but the backward pass requires differentiating through a sampling operation
- Solved by the **reparameterization trick** which moves the sampling to the input:

$$\mathbb{E}_{\mathbf{z}\sim q_{\mathbf{w}}(\mathbf{z}|\mathbf{x}_i)}\left[-\log p_{\mathbf{w}}(\mathbf{x}_i|\mathbf{z})\right] = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[-\log p_{\mathbf{w}}(\mathbf{x}_i|\mathbf{z} = \boldsymbol{\mu}_{\mathbf{w}}(\mathbf{x}_i) + \boldsymbol{\sigma}_{\mathbf{w}}(\mathbf{x}_i) \odot \boldsymbol{\epsilon})\right]$$

- In practice, a single sample $\boldsymbol{\epsilon}$ per $\mathbf{x}$ often suffices (depends on minibatch size)

Diederik P. Kingma, Max Welling: Auto-Encoding Variational Bayes. ICLR, 2014.

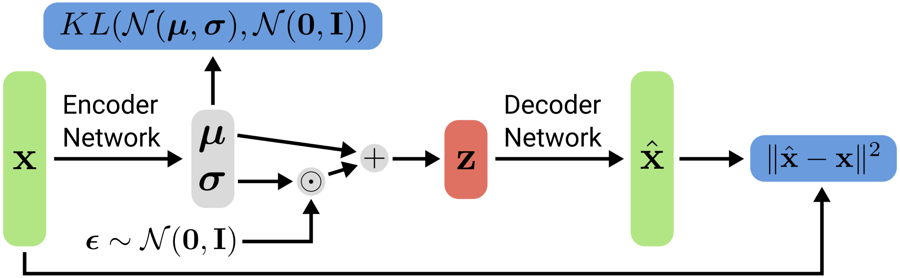
Learning a VAE



- **Vanilla VAE** formulation which is intractable due to sampling inside the network
- Remark: We assume a Gaussion likelihood $p_{\mathbf{w}}(\mathbf{x}|\mathbf{z})$ in this example

Diederik P. Kingma, Max Welling: Auto-Encoding Variational Bayes. ICLR, 2014.

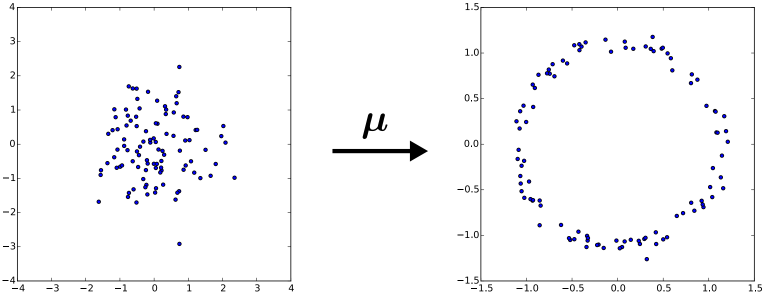
Learning a VAE



- **Reparameterized version** which is tractable as sampling has been moved to input
- This trick works for Gaussians and some other simple distributions (cf., Kingma)

Diederik P. Kingma, Max Welling: Auto-Encoding Variational Bayes. ICLR, 2014.

Expressiveness

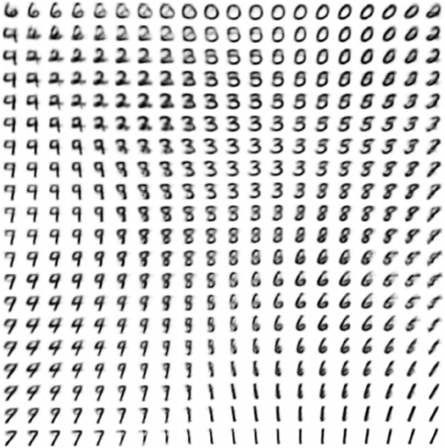


- VAEs are very **expressive:** Consider random samples $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Mapping the samples through $\boldsymbol{\mu}(\mathbf{z}) = \mathbf{z}/10 + \mathbf{z}/||\mathbf{z}||$ yields a complex distribution
- VAEs model $\boldsymbol{\mu}(\mathbf{z})$ as a **deterministic neural network** and learn its parameters

Diederik P. Kingma, Max Welling: Auto-Encoding Variational Bayes. ICLR, 2014.

Learned Manifold

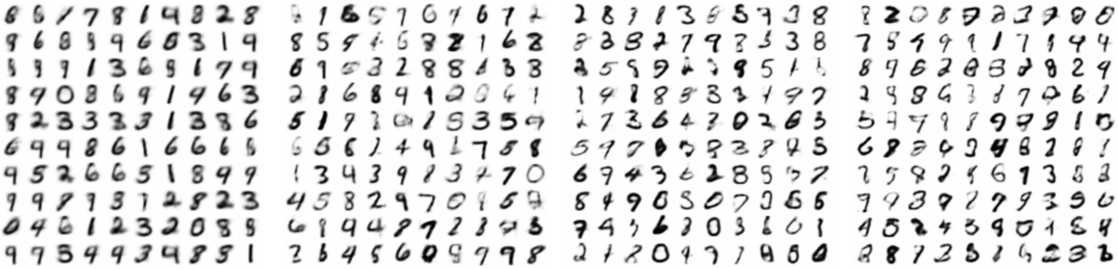
(a) Learned Frey Face manifold



(b) Learned MNIST manifold

Diederik P. Kingma, Max Welling: Auto-Encoding Variational Bayes. ICLR, 2014.

Random Samples



(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Diederik P. Kingma, Max Welling: Auto-Encoding Variational Bayes. ICLR, 2014.

DRAW: A Sequential Variational Autoencoder

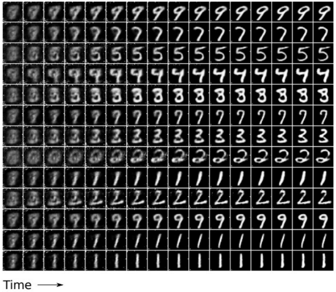


Figure 7. **MNIST generation sequences for DRAW without attention.** Notice how the network first generates a very blurry image that is subsequently refined.

Figure 8. **Generated MNIST images with two digits.**

Figure 9. **Generated SVHN images.**

- Sequential VAE for generating images (combines VAE with RNN)

Gregor, Danihelka, Graves, Rezende and Wierstra: DRAW: A Recurrent Neural Network For Image Generation. ICML, 2015.

Deep Convolutional Inverse Graphics Network

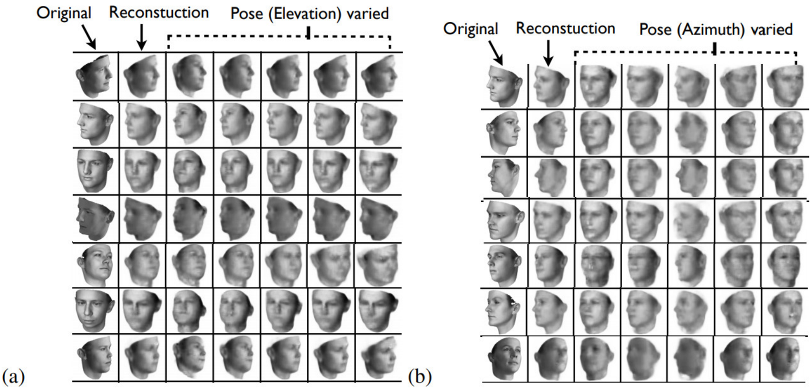
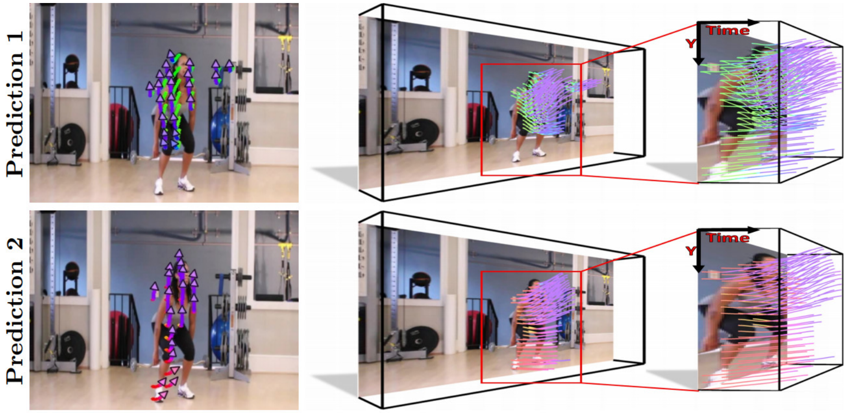


Figure 4: **Manipulating pose variables:** Qualitative results showing the generalization capability of the learned DC-IGN decoder to rerender a single input image with different pose directions.

- Forces disentangled latents (pose, light, texture, shape) through weak supervision

Kulkarni, Whitney, Kohli and Tenenbaum: Deep Convolutional Inverse Graphics Network. NIPS, 2015.

Motion Forecasting from Static Images



(a) Trajectories on Image (b) Trajectories in Space-Time

- Motion forecasting from static image by jointly encoding images and trajectories

Walker, Doersch, Gupta and Hebert: An Uncertain Future: Forecasting from Static Images Using Variational Autoencoders. ECCV, 2016.

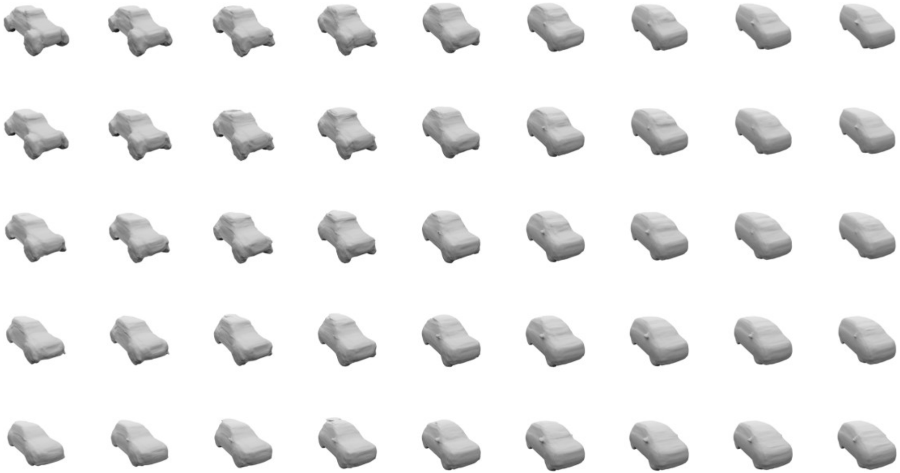
VQ-VAE-2: Vector Quantized Variational Autoencoder

Figure 1: Class-conditional 256x256 image samples from a two-level model trained on ImageNet.

- VQ-VAEs predict discrete codes and learn the prior distribution $\Rightarrow$ state-of-the-art

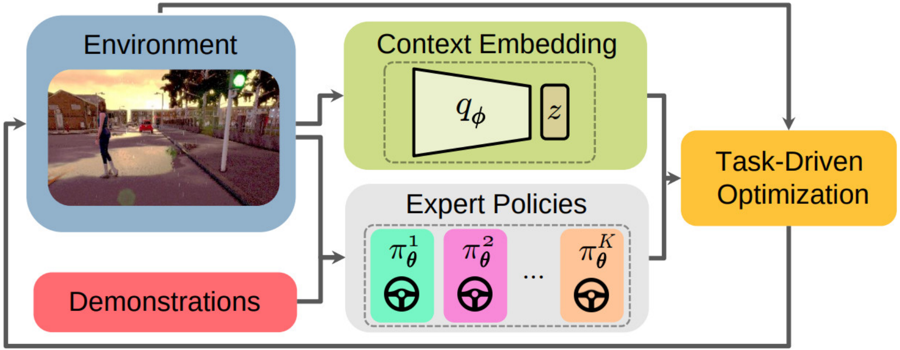
Razavi, Oord and Vinyals: Generating Diverse High-Fidelity Images with VQ-VAE-2. NeurIPS, 2019.

Occupancy Networks: Learning 3D Reconstruction in Function Space



Mescheder, Oechsle, Niemeyer, Nowozin and Geiger: Occupancy Networks: Learning 3D Reconstruction in Function Space. CVPR, 2019.

Learning Situational Driving



- Data-efficient reinforcement learning with a latent embedding of the environment

Ohn-Bar, Prakash, Behl, Chitta and Geiger: Learning Situational Driving. CVPR, 2020.