A brief introduction to deep learning for generative modeling

Jakob Verbeek
INRIA, Grenoble, France

Breaking the Surface 2019
Biograd na Moru, Croatia

Plan for this presentation

1. Introduction to deep learning
    - Machine learning basics
    - Deep learning building blocks (MLP, convolution, back-propagation)
2. Deep generative models
    - Generative modeling basics
    - Generative adversarial networks
    - Variational autoencoders
    - Flow-based density estimation

Part I

Brief introduction to (deep) learning

Machine learning paradigms

Machine learning paradigms

- **Supervised Learning**: use of labeled training set
  - ex: email spam detector with training set of already labeled emails

Machine learning paradigms

- **Supervised Learning**: use of labeled training set
  - ex: email spam detector with training set of already labeled emails
- **Unsupervised Learning**: discover patterns in unlabeled data
  - ex: cluster similar documents based on text content

Machine learning paradigms

- **Supervised Learning**: use of labeled training set
  - ex: email spam detector with training set of already labeled emails
- **Unsupervised Learning**: discover patterns in unlabeled data
  - ex: cluster similar documents based on text content
- **Reinforcement Learning**: learning sequential decision making based on feedback or reward
  - ex: learning to play a game by winning or losing

What is Deep Learning

What is Deep Learning

- Part of the ML field of learning representations of data

What is Deep Learning

- Part of the ML field of learning representations of data
- Learning algorithms derive meaning out of data by using a **hierarchy of multiple layers** of units (*neurons*)

What is Deep Learning

- Part of the ML field of learning representations of data
- Learning algorithms derive meaning out of data by using a **hierarchy of multiple layers** of units (*neurons*)
- Each layer computes linear function of its inputs, which is passed through a non linear function

What is Deep Learning

- Part of the ML field of learning representations of data
- Learning algorithms derive meaning out of data by using a **hierarchy of multiple layers** of units (*neurons*)
- Each layer computes linear function of its inputs, which is passed through a non linear function
- Learning = find optimal model parameters from data
    - ex: deep speech transcription system has 10-20M of parameters

A very brief history

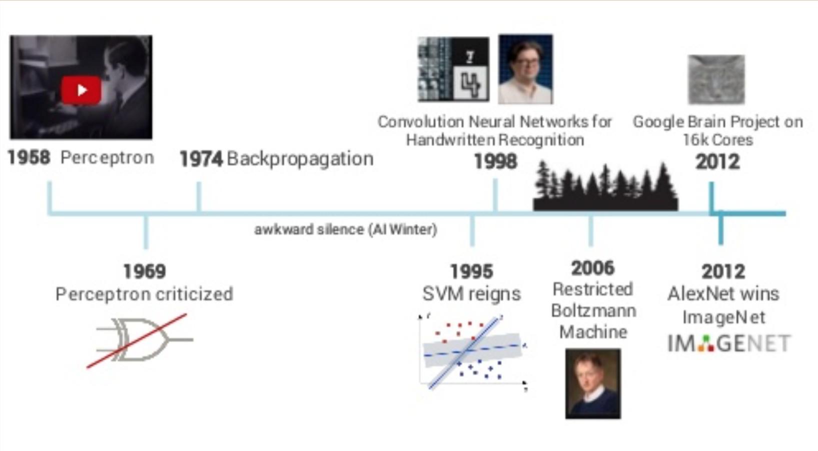


Figure from https://www.slideshare.net/LuMa921/deep-learning-a-visual-introduction

A very brief history

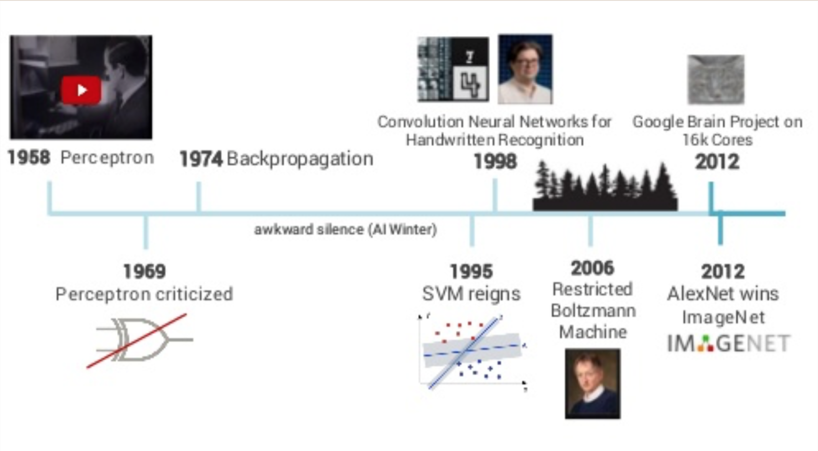


Figure from https://www.slideshare.net/LuMa921/deep-learning-a-visual-introduction

- 2012 breakthrough due to
  - Lots of labeled data (ex: ImageNet)
  - Computation (ex: GPU)
  - Algorithmic & architectural progresses (ex: SGD, ReLU)

Success stories of deep learning in recent years

Success stories of deep learning in recent years

- Convolutional neural networks

Success stories of deep learning in recent years

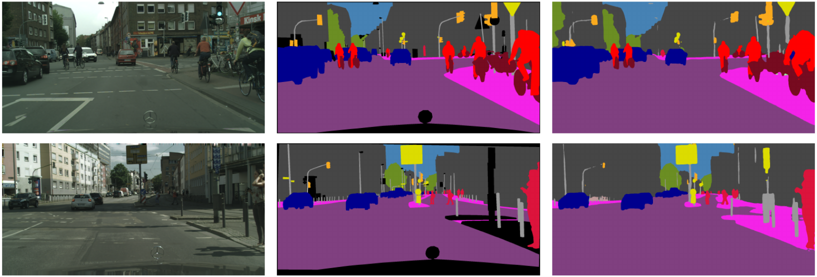
- Convolutional neural networks
- For stationary signals such as audio, images, and video, sampled on regular grid structure

Success stories of deep learning in recent years

- Convolutional neural networks
- For stationary signals such as audio, images, and video, sampled on regular grid structure
- Applictions: Object detection, semantic segmentation, image retrieval, pose estimation, action recognition, . . .

Success stories of deep learning in recent years

- Convolutional neural networks
- For stationary signals such as audio, images, and video, sampled on regular grid structure
- Applictions: Object detection, semantic segmentation, image retrieval, pose estimation, action recognition, . . .



RGB Input Ground-truth Predictions

Semantic segmentation pixel labeling [Lin et al., 2017]

Success stories of deep learning in recent years

Success stories of deep learning in recent years

- Recurrent neural networks

Success stories of deep learning in recent years

- Recurrent neural networks
- For variable length sequence data, **e.g**. in natural language

Success stories of deep learning in recent years

- Recurrent neural networks
- For variable length sequence data, **e.g**. in natural language
- Applications: Machine translation, image captioning, speech recognition, ...

Success stories of deep learning in recent years

- Recurrent neural networks
- For variable length sequence data, **e.g**. in natural language
- Applications: Machine translation, image captioning, speech recognition, . . .

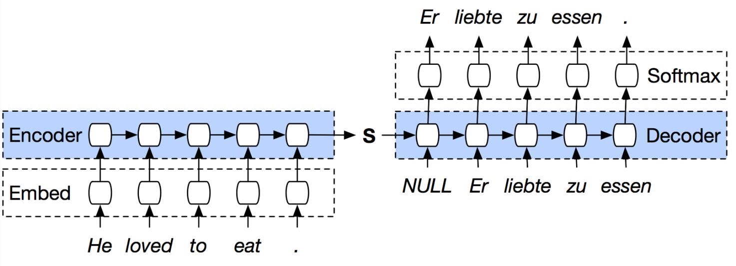


Figure from: https://smerity.com/media/images/articles/2016/

It’s all about the features!

It's all about the features!

- Conventional vision / audio processing approach

It's all about the features!

- Conventional vision / audio processing approach
    1. Features extraction (engineered) : SIFT, MFCC, ...

It's all about the features!

- Conventional vision / audio processing approach
  1. Features extraction (engineered) : SIFT, MFCC, ...
  2. Feature pooling (unsupervised): bag-of-words, Fisher vectors, ...

It's all about the features!

- Conventional vision / audio processing approach
    1. Features extraction (engineered) : SIFT, MFCC, ...
    2. Feature pooling (unsupervised): bag-of-words, Fisher vectors, ...
    3. Image recognition (supervised): linear/kernel classifier, ...

It’s all about the features!

- Conventional vision / audio processing approach
  1. Features extraction (engineered) : SIFT, MFCC, . . .
  2. Feature pooling (unsupervised): bag-of-words, Fisher vectors, . . .
  3. Image recognition (supervised): linear/kernel classifier, . . .

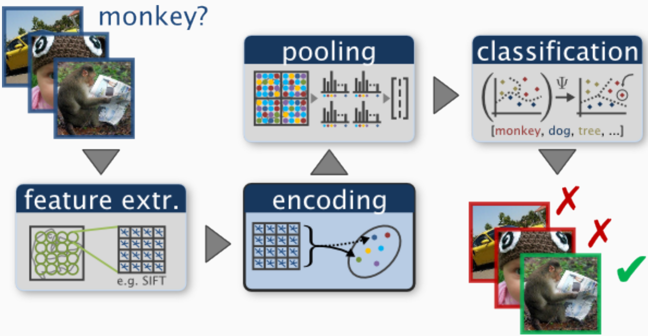


Image from [Chatfield et al., 2011]

It's all about the features

- **Deep learning blurs boundary feature / classifier**

It's all about the features

- **Deep learning blurs boundary feature / classifier**
  - Starts from raw input signal, **e.g**. image pixels

It's all about the features

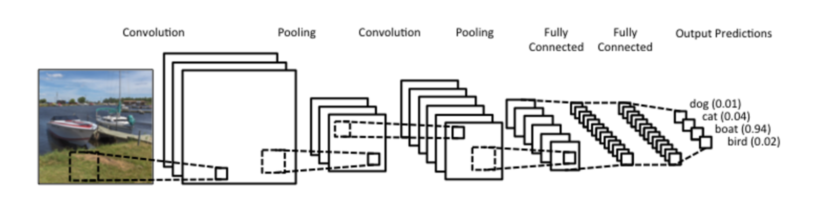
- **Deep learning blurs boundary feature / classifier**
  - Starts from raw input signal, **e.g**. image pixels
  - Stacks simple linear transformations with non-linearities in between

It's all about the features

- **Deep learning blurs boundary feature / classifier**
  - Starts from raw input signal, **e.g**. image pixels
  - Stacks simple linear transformations with non-linearities in between
  - Learns progressively more abstract representation

It's all about the features

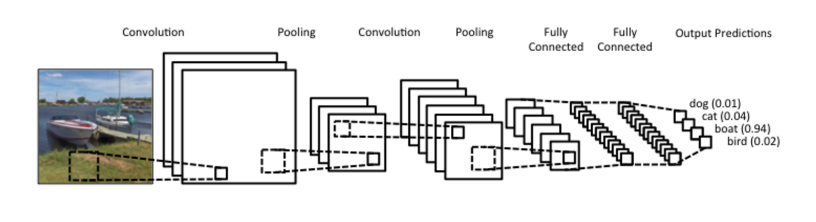
- **Deep learning blurs boundary feature / classifier**
  - Starts from raw input signal, **e.g**. image pixels
  - Stacks simple linear transformations with non-linearities in between
  - Learns progressively more abstract representation



It's all about the features

- **Deep learning blurs boundary feature / classifier**
  - Starts from raw input signal, **e.g**. image pixels
  - Stacks simple linear transformations with non-linearities in between
  - Learns progressively more abstract representation

- End-to-end training of entire pipeline minimizing specific loss

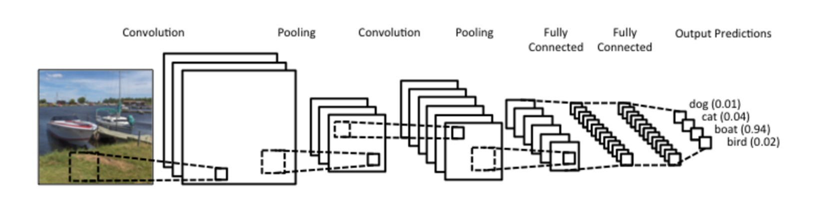


It's all about the features

- **Deep learning blurs boundary feature / classifier**
  - Starts from raw input signal, **e.g**. image pixels
  - Stacks simple linear transformations with non-linearities in between
  - Learns progressively more abstract representation

- End-to-end training of entire pipeline minimizing specific loss

- Supervised learning from lots of labeled data



Training a model by empirical risk minimization

- Given labeled training data $(x_i, y_i)_{i=1...N}$ with $x_i \in \mathcal{X}, y_i \in \mathcal{Y}$
- Learn a prediction function $f : \mathcal{X} \to \mathcal{Y}$.

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}}$$

Training a model by empirical risk minimization

- Given labeled training data $(x_i, y_i)_{i=1\ldots N}$ with $x_i \in \mathcal{X}, y_i \in \mathcal{Y}$
- Learn a prediction function $f : \mathcal{X} \to \mathcal{Y}$.

$$\min_{f\in\mathcal{F}} \underbrace{\frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda\Omega(f)}_{\text{regularization}}$$

- The targets y_i can be in
  - $\{-1, +1\}$: binary classification
  - $\{1, \ldots, K\}$: multi-class classification
  - $\mathbb{R}$: regression
  - $\mathbb{R}^n$: multivariate regression

Training a model by empirical risk minimization

- Given labeled training data $(x_i, y_i)_{i=1 \ldots N}$ with $x_i \in \mathcal{X}, y_i \in \mathcal{Y}$
- Learn a prediction function $f : \mathcal{X} \rightarrow \mathcal{Y}$.

$$\underset{f \in \mathcal{F}}{min} \; \underbrace{\frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}}$$

- The targets y_i can be in
    - { -1, + 1}: binary classification
    - {1, . . . , K}: multi-class classification
    - $\mathbb{R}$: regression
    - $\mathbb{R}^n$: multivariate regression
- Loss function L evaluates predictions, often convex

Training a model by empirical risk minimization

- Given labeled training data $(x_i, y_i)_{i=1...N}$ with $x_i \in \mathcal{X}, y_i \in \mathcal{Y}$
- Learn a prediction function $f : \mathcal{X} \to \mathcal{Y}$.

$$\underset{f \in \mathcal{F}}{min} \underbrace{\frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}}$$

- Not just risk minimization
  - Need to generalize to unseen examples
  - Occam's razor (favor simplicity)
  - Regularization: control the complexity of solutions

Training a model by empirical risk minimization

- Given labeled training data $(x_i, y_i)_{i=1\dots N}$ with $x_i \in \mathcal{X}, y_i \in \mathcal{Y}$
- Learn a prediction function $f : \mathcal{X} \to \mathcal{Y}$.

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda\Omega(f)}_{\text{regularization}}$$

- Linear regression example.
  - Assume linear relation between y and features $x \in \mathbb{R}^p$
  - $f(x) = w^T x + b$, parametrized by w, b in $\mathbb{R}^{p+1}$
  - L is often convex, $\Omega(f)$ often squared l_2-norm $||w||^2$.
  - Optimize by gradient descent: follow the steepest direction.
  - The problem is convex: local optimum is global.
  - Features and classification are decoupled

Training a model by empirical risk minimization

- Given labeled training data $(x_i, y_i)_{i=1 \dots N}$ with $x_i \in \mathcal{X}, y_i \in \mathcal{Y}$
- Learn a prediction function $f : \mathcal{X} \to \mathcal{Y}$.

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}}$$

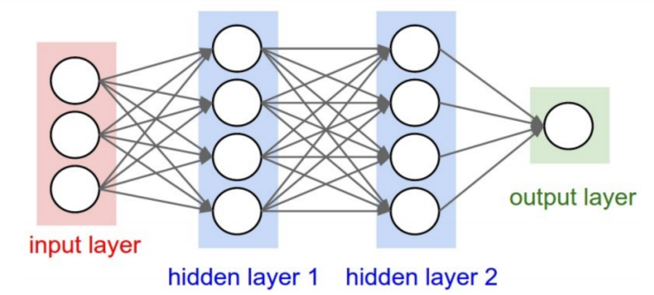
- Deep learning example.
  - Composition of linear transformations and non-linearities
  - Parametrization of deep models

$$\mathcal{F} : f(x) = \sigma_k(A_k \sigma_{k-1}(A_{k-1} \dots \sigma_2(A_2 \sigma_1(A_1 x))))$$

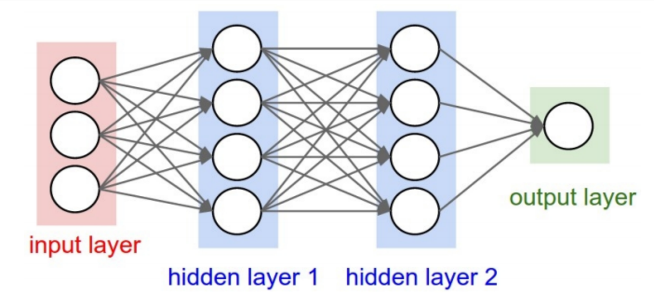
  - Adaptive features, universal approximation theorem
  - Hard to optimize: non-convex, high-dimensional

Multi-layer perceptron, or "fully connected network"

Multi-layer perceptron, or “fully connected network”

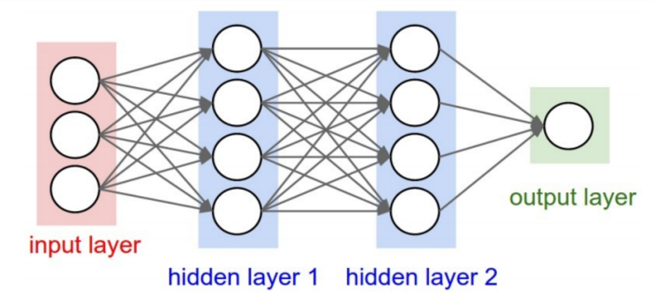


Multi-layer perceptron, or "fully connected network"



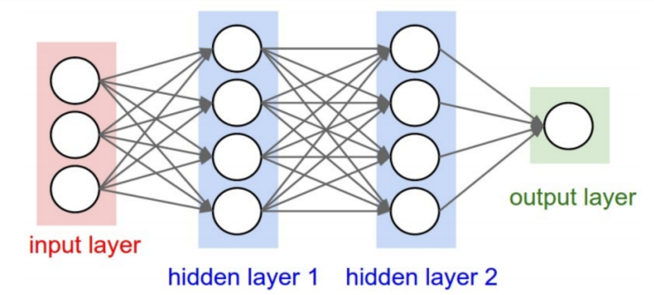
- Stack of linear operations $y = Wx + b$

Multi-layer perceptron, or “fully connected network”



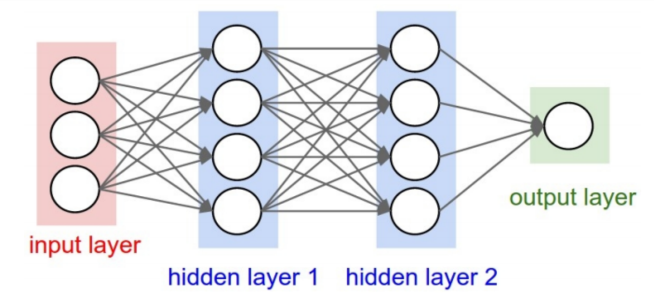
- Stack of linear operations $y = Wx + b$
- Non-linearities in between, **e.g**. $ReLU(x) = \max(0, x)$

Multi-layer perceptron, or “fully connected network”



- Stack of linear operations $y = Wx + b$
- Non-linearities in between, **e.g**. $ReLU(x) = \max(0, x)$
- One connection = one parameter

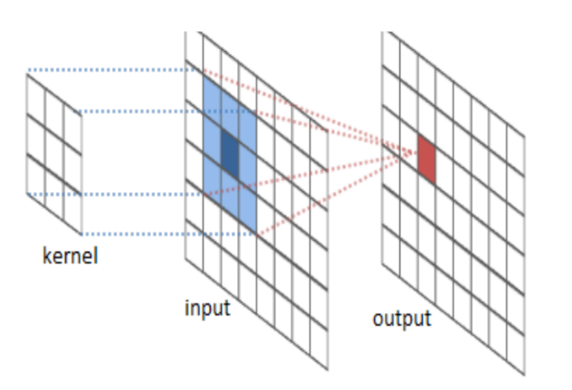
Multi-layer perceptron, or "fully connected network"



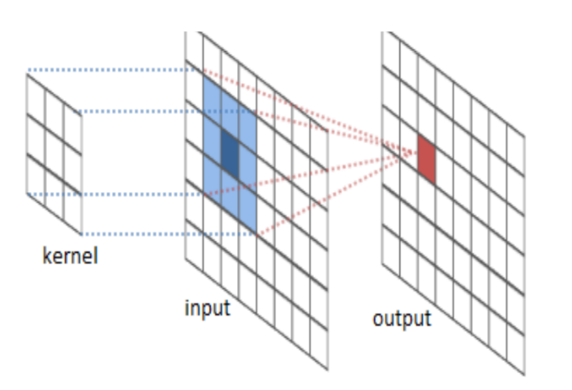
- Stack of linear operations $y = Wx + b$
- Non-linearities in between, **e.g**. $ReLU(x) = \max(0, x)$
- One connection = one parameter
- Limitations: No invariances, poor scaling of nr parameters

Convolutional networks

Convolutional networks

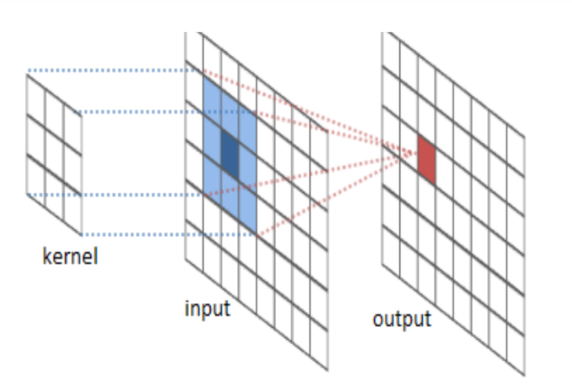


Convolutional networks



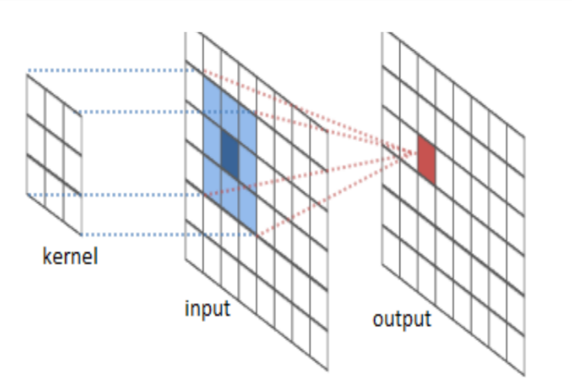
- Very sparse weight matrix W

Convolutional networks



- Very sparse weight matrix W
- Weights shared across positions, translation equivariant processing

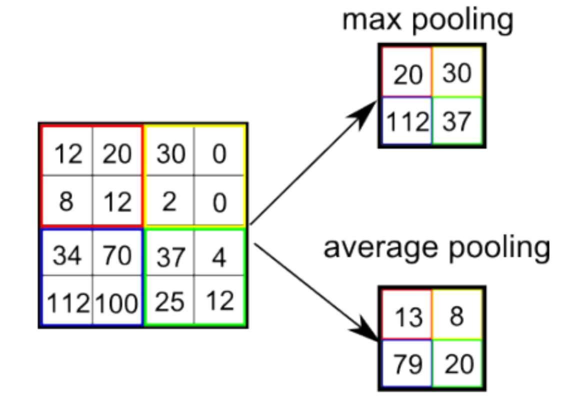
Convolutional networks



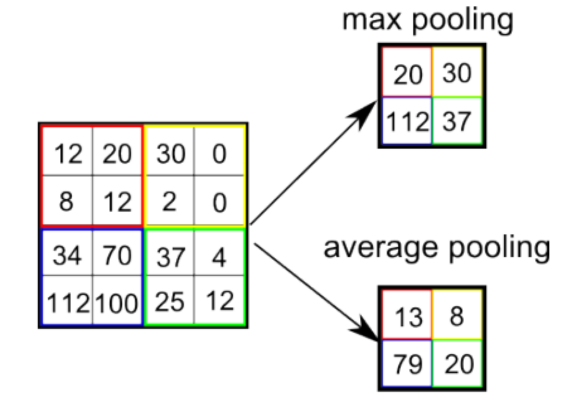
- Very sparse weight matrix W
- Weights shared across positions, translation equivariant processing
- Computations single instruction multiple data (SIMD): GPU

Pooling operations

Pooling operations

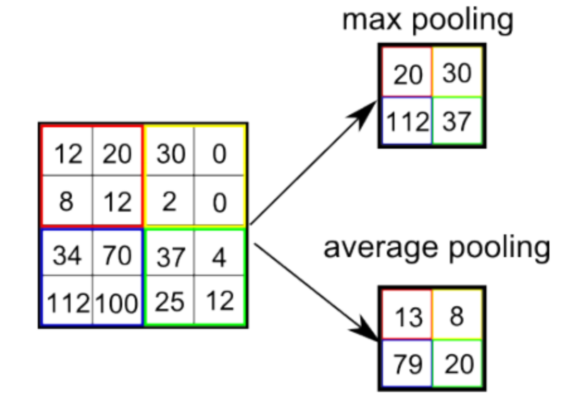


Pooling operations



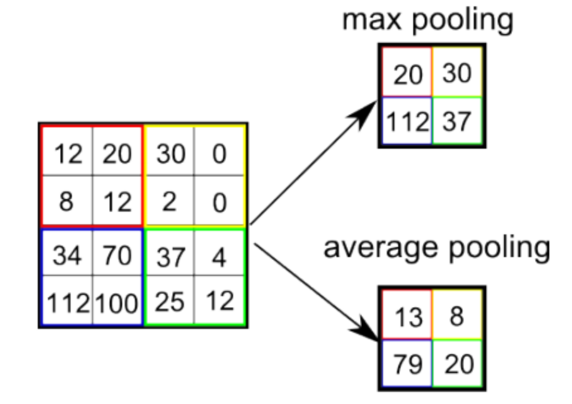
- Reduce spatial dimension

Pooling operations



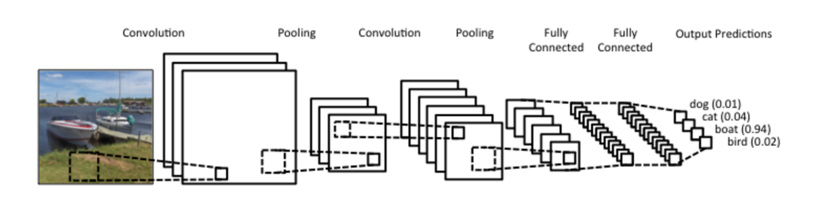
- Reduce spatial dimension
- Increase receptive field

Pooling operations



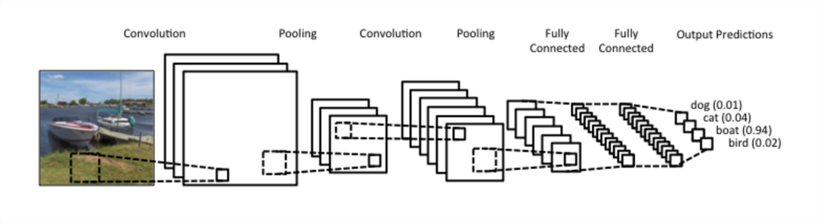
- Reduce spatial dimension
- Increase receptive field
- Or just down-sample after convolution ("strided convolution")

Training deep networks



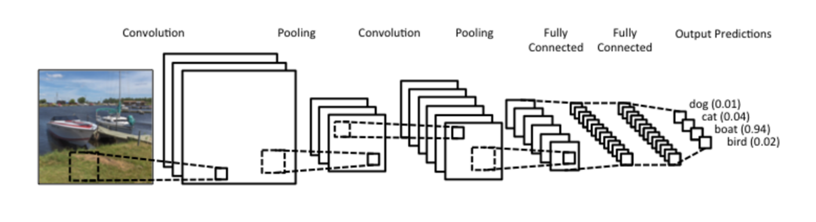
- Stack convolutions, pooling, and non-linearities

Training deep networks



- Stack convolutions, pooling, and non-linearities
  - Forward propagation from input x to output y

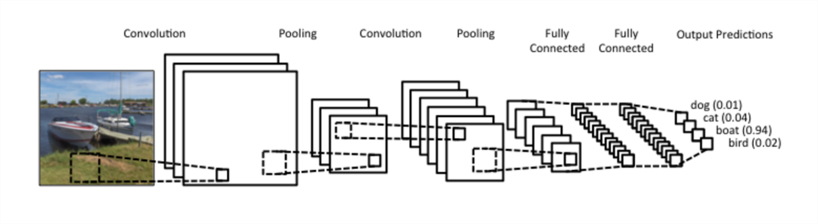
Training deep networks



- Stack convolutions, pooling, and non-linearities
  - Forward propagation from input x to output y
- Train by stochastic gradient descent, using small batches of data

$$\frac{1}{n}\sum_{i=1}^{n} \nabla_\theta L(y_i, f_\theta(x_i)), \qquad \text{with } n \ll N$$

Training deep networks

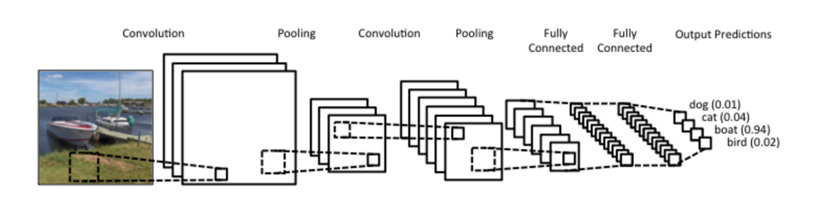


- Stack convolutions, pooling, and non-linearities
    - Forward propagation from input x to output y
- Train by stochastic gradient descent, using small batches of data

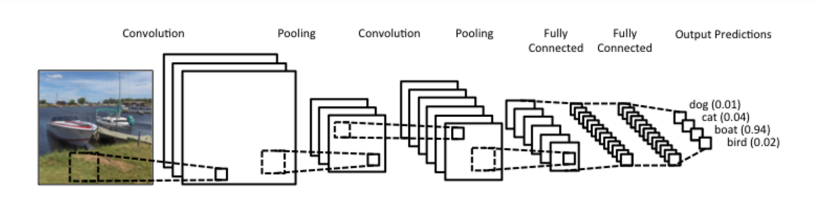
$$\frac{1}{n}\sum_{i=1}^{n}\nabla_\theta L(y_i, f_\theta(x_i)), \qquad \text{with } n \ll N$$

- Efficient gradient computations via backpropagation algorithm

Feature visualization



Feature visualization



Features visualisation

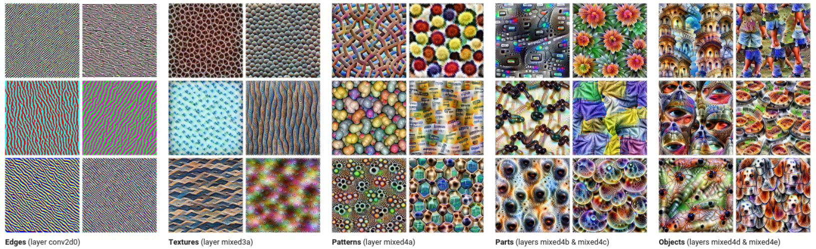


Figure from distill.pub

Take-home messages

- Core idea
  - Many processing layers from raw input to output
  - Learning features and classifier jointly

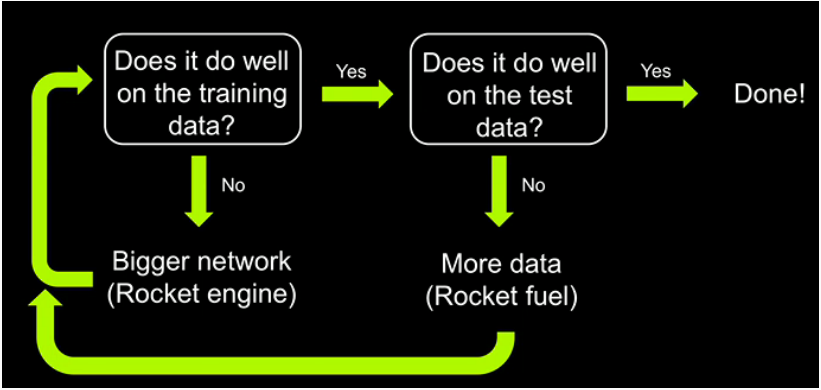
Take-home messages

- Core idea
  - Many processing layers from raw input to output
  - Learning features and classifier jointly
- In practice
  - Strategy efficient across disciplines (vision, speech, NLP, games etc.)
  - Large-scale applications widely adopted in industry
  - Computation and labeled (!) data hungry

Take-home messages

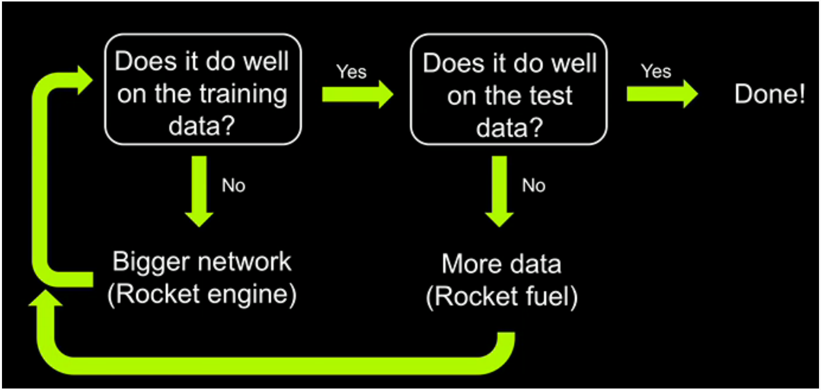
- Core idea
    - Many processing layers from raw input to output
    - Learning features and classifier jointly
- In practice
    - Strategy efficient across disciplines (vision, speech, NLP, games etc.)
    - Large-scale applications widely adopted in industry
    - Computation and labeled (!) data hungry
- In theory
    - Optimization still poorly understood
    - Generalization still poorly understood
    - Experimental results 'ahead' of theory

The Limits to Growth



From Andrew Ng's Keynote at Nvidia's GPU Technology Conf. 2015

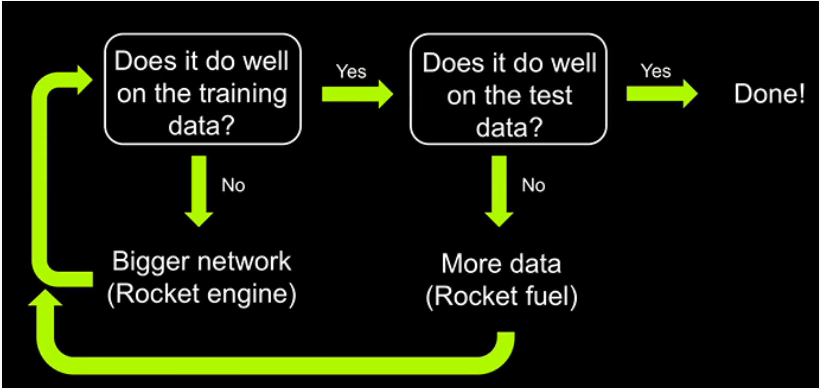
More labeled data



From Andrew Ng's Keynote at Nvidia's GPU Technology Conf. 2015

More labeled data

A sustainable approach?



From Andrew Ng's Keynote at Nvidia's GPU Technology Conf. 2015

Part II

Unsupervised deep learning

Motivations for unsupervised deep learning

Motivations for unsupervised deep learning

1. **Improve supervised learning from few samples**

Motivations for unsupervised deep learning

1. **Improve supervised learning from few samples**
    - Unlabeled data often abundantly available

Motivations for unsupervised deep learning

1. **Improve supervised learning from few samples**
    - Unlabeled data often abundantly available
    - Learn representations/features from unlabeled data

Motivations for unsupervised deep learning

1. **Improve supervised learning from few samples**
   - Unlabeled data often abundantly available
   - Learn representations/features from unlabeled data

2. **Generative models for image and other complex data**

Motivations for unsupervised deep learning

1. **Improve supervised learning from few samples**
   - Unlabeled data often abundantly available
   - Learn representations/features from unlabeled data
2. **Generative models for image and other complex data**
   - Unconditional density estim. $p_\theta(\mathbf{x})$, sampling, outlier detection, ...

Motivations for unsupervised deep learning

1. **Improve supervised learning from few samples**
   - Unlabeled data often abundantly available
   - Learn representations/features from unlabeled data
2. **Generative models for image and other complex data**
   - Unconditional density estim. $p_\theta(\mathbf{x})$, sampling, outlier detection, ...
   - Conditional density estim. $p_\theta(\mathbf{x}|y)$: text-to-speech, image colorization, video forecasting, **etc**.

Motivations for unsupervised deep learning

1. **Improve supervised learning from few samples**
   - Unlabeled data often abundantly available
   - Learn representations/features from unlabeled data

2. **Generative models for image and other complex data**
   - Unconditional density estim. $p_\theta(\mathbf{x})$, sampling, outlier detection, ...
   - Conditional density estim. $p_\theta(\mathbf{x}|y)$: text-to-speech, image colorization, video forecasting, **etc**.

Image colorization [Royer et al., 2017]

Gaussian mixture models

Gaussian mixture models

$$p(z = k) = \pi_k \quad (1)$$

$$p(\mathbf{x}|z = k) = \mathcal{N}(x; \mu_k, \sigma I_D) \quad (2)$$

$$p(\mathbf{x}) = \sum_z p(z)p(\mathbf{x}|z) \quad (3)$$

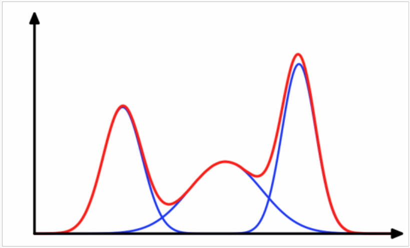


Figure from [Bishop, 2006]

Gaussian mixture models

$$p(z = k) = \pi_k \tag{1}$$
$$p(\mathbf{x}|z = k) = \mathcal{N}(x; \mu_k, \sigma I_D) \tag{2}$$
$$p(\mathbf{x}) = \sum_z p(z)p(\mathbf{x}|z) \tag{3}$$

- Estimation: Expectation-Maximization (EM) algorithm

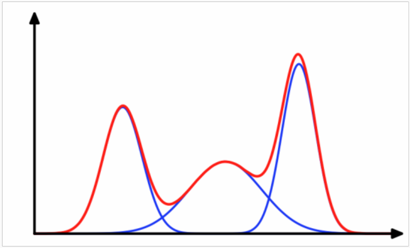


Figure from [Bishop, 2006]

Gaussian mixture models

$$p(z = k) = \pi_k \quad (1)$$

$$p(\mathbf{x}|z = k) = \mathcal{N}(x; \mu_k, \sigma I_D) \quad (2)$$

$$p(\mathbf{x}) = \sum_z p(z)p(\mathbf{x}|z) \quad (3)$$

- Estimation: Expectation-Maximization (EM) algorithm
- Sampling: pick component from prior distribution $p(z)$, then draw sample from conditional distribution $p(\mathbf{x}|z)$

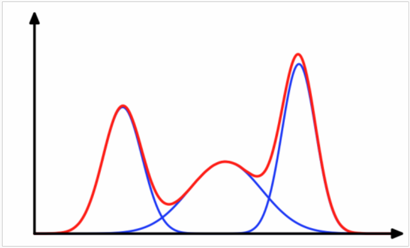


Figure from [Bishop, 2006]

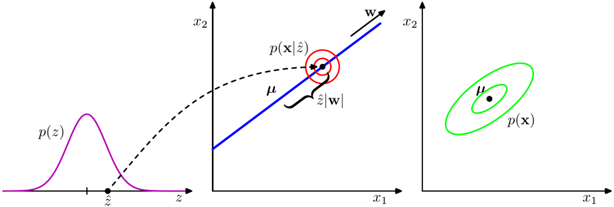
Linear latent variable models

- Probabilistic Principal Component Analysis [Roweis, 1997, Tipping and Bishop, 1999]

$$p(z) = \mathcal{N}(z; 0, I_d) \tag{4}$$

$$p(\mathbf{x}|z) = \mathcal{N}(\mathbf{x}; \mu + Wz, \sigma I_D) \tag{5}$$

$$p(\mathbf{x}) = \int_z p(z)p(\mathbf{x}|z) \tag{6}$$



Linear latent variable models

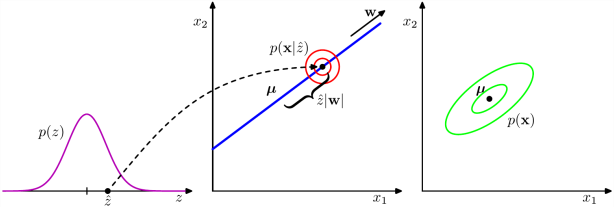
- Probabilistic Principal Component Analysis [Roweis, 1997, Tipping and Bishop, 1999]

$$p(z) = \mathcal{N}(z; 0, I_d) \tag{4}$$

$$p(\mathbf{x}|z) = \mathcal{N}(\mathbf{x}; \mu + Wz, \sigma I_D) \tag{5}$$

$$p(\mathbf{x}) = \int_z p(z)p(\mathbf{x}|z) \tag{6}$$

- Estimation: SVD or EM algorithm



Linear latent variable models

- Probabilistic Principal Component Analysis
  [Roweis, 1997, Tipping and Bishop, 1999]

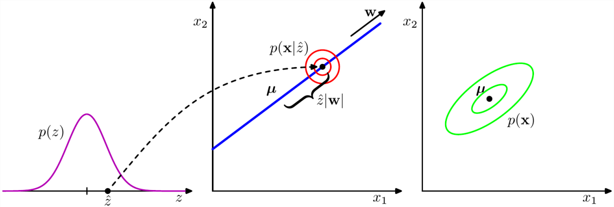
$$p(z) = \mathcal{N}(z; 0, I_d) \tag{4}$$

$$p(\mathbf{x}|z) = \mathcal{N}(\mathbf{x}; \mu + Wz, \sigma I_D) \tag{5}$$

$$p(\mathbf{x}) = \int_z p(z)p(\mathbf{x}|z) \tag{6}$$

- Estimation: SVD or EM algorithm

- Sampling: pick point in subspace from prior $p(z)$,
  then draw sample from conditional distribution $p(\mathbf{x}|z)$



Non-linear latent variable models

- Simple distribution $p(\mathbf{z})$ on latent variable $\mathbf{z}$,
  e.g. standard Gaussian

Non-linear latent variable models

- Simple distribution $p(\mathbf{z})$ on latent variable $\mathbf{z}$, **e.g**. standard Gaussian

- Non-linear function $\mathbf{x} = f_\theta(\mathbf{z})$ maps latent variable to data space, **e.g**. deep neural net

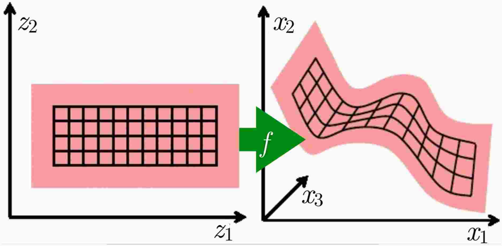


Figure from Aaron Courville

Non-linear latent variable models

- Simple distribution $p(\mathbf{z})$ on latent variable $\mathbf{z}$,
  e.g. standard Gaussian

- Non-linear function $\mathbf{x} = f_\theta(\mathbf{z})$ maps latent variable to data space,
  e.g. deep neural net

- Sampling: pick point in subspace from prior $p(z)$,
  then draw sample from conditional distribution $p(\mathbf{x}|z)$

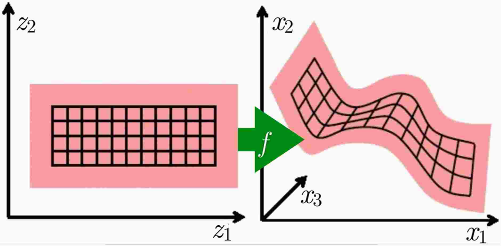


Figure from Aaron Courville

Learning deep latent variable models

- Marginal distribution on $\mathbf{x}$ obtained by integrating out $\mathbf{z}$

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I), \tag{7}$$

$$p_\theta(\mathbf{x}) = \int_z p(\mathbf{z}) p(\mathbf{x}|f_\theta(\mathbf{z})). \tag{8}$$

Learning deep latent variable models

- Marginal distribution on $\mathbf{x}$ obtained by integrating out $\mathbf{z}$

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I), \tag{7}$$

$$p_\theta(\mathbf{x}) = \int_z p(\mathbf{z}) p(\mathbf{x}|f_\theta(\mathbf{z})). \tag{8}$$

- Problem: Evaluation of $p_\theta(\mathbf{x})$ intractable due to integral involving flexible non-linear deep net $f_\theta(\cdot)$

Learning deep latent variable models

- Marginal distribution on $\mathbf{x}$ obtained by integrating out $\mathbf{z}$

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I), \tag{7}$$

$$p_\theta(\mathbf{x}) = \int_z p(\mathbf{z}) p(\mathbf{x} | f_\theta(\mathbf{z})). \tag{8}$$

- Problem: Evaluation of $p_\theta(\mathbf{x})$ intractable due to integral involving flexible non-linear deep net $f_\theta(\cdot)$

- Solutions by different unsupervised deep learning paradigms

Learning deep latent variable models

- Marginal distribution on $\mathbf{x}$ obtained by integrating out $\mathbf{z}$

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I), \tag{7}$$

$$p_\theta(\mathbf{x}) = \int_z p(\mathbf{z}) p(\mathbf{x}|f_\theta(\mathbf{z})). \tag{8}$$

- Problem: Evaluation of $p_\theta(\mathbf{x})$ intractable due to integral involving flexible non-linear deep net $f_\theta(\cdot)$
- Solutions by different unsupervised deep learning paradigms
  - Avoid integral: Generative adversarial networks (GAN)

Learning deep latent variable models

- Marginal distribution on $\mathbf{x}$ obtained by integrating out $\mathbf{z}$

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I), \tag{7}$$

$$p_\theta(\mathbf{x}) = \int_z p(\mathbf{z}) p(\mathbf{x} | f_\theta(\mathbf{z})). \tag{8}$$

- Problem: Evaluation of $p_\theta(\mathbf{x})$ intractable due to integral involving flexible non-linear deep net $f_\theta(\cdot)$

- Solutions by different unsupervised deep learning paradigms
    - Avoid integral: Generative adversarial networks (GAN)
    - Approximate integral: Variational autoencoders (VAE)

Learning deep latent variable models

- Marginal distribution on $\mathbf{x}$ obtained by integrating out $\mathbf{z}$

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I), \tag{7}$$

$$p_\theta(\mathbf{x}) = \int_z p(\mathbf{z}) p(\mathbf{x} | f_\theta(\mathbf{z})). \tag{8}$$

- Problem: Evaluation of $p_\theta(\mathbf{x})$ intractable due to integral involving flexible non-linear deep net $f_\theta(\cdot)$
- Solutions by different unsupervised deep learning paradigms
  - Avoid integral: Generative adversarial networks (GAN)
  - Approximate integral: Variational autoencoders (VAE)
  - Tractable integral: constrain f_θ to invertible "flow"

Learning deep latent variable models

- Marginal distribution on $\mathbf{x}$ obtained by integrating out $\mathbf{z}$

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I), \tag{7}$$

$$p_\theta(\mathbf{x}) = \int_z p(\mathbf{z})p(\mathbf{x}|f_\theta(\mathbf{z})). \tag{8}$$

- Problem: Evaluation of $p_\theta(\mathbf{x})$ intractable due to integral involving flexible non-linear deep net $f_\theta(\cdot)$

- Solutions by different unsupervised deep learning paradigms
    - Avoid integral: Generative adversarial networks (GAN)
    - Approximate integral: Variational autoencoders (VAE)
    - Tractable integral: constrain f_θ to invertible "flow"
    - Avoid latent variables: autoregressive models

Part III

Generative adversarial networks

Generative adversarial networks [Goodfellow et al., 2014]

- Sample $p(\mathbf{z})$, map it using deep net to $\mathbf{x} = G_\theta(\mathbf{z})$

Generative adversarial networks [Goodfellow et al., 2014]

- Sample $p(\mathbf{z})$, map it using deep net to $\mathbf{x} = G_\theta(\mathbf{z})$
- Instead of trying to evaluate $p(\mathbf{x})$, use classifier D_ϕ
  - $D_\phi(\mathbf{x}) \in [0, 1]$ probability $\mathbf{x}$ is real **vs**.synth. image

Generative adversarial networks [Goodfellow et al., 2014]

- Sample $p(\mathbf{z})$, map it using deep net to $\mathbf{x} = G_\theta(\mathbf{z})$
- Instead of trying to evaluate $p(\mathbf{x})$, use classifier D_ϕ
  - $D_\phi(\mathbf{x}) \in [0, 1]$ probability $\mathbf{x}$ is real **vs**.synth. image

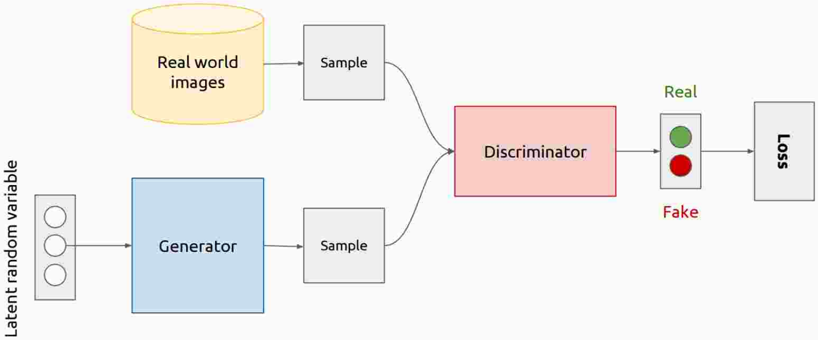


Figure from Kevin McGuinness

Discriminator architecture for images

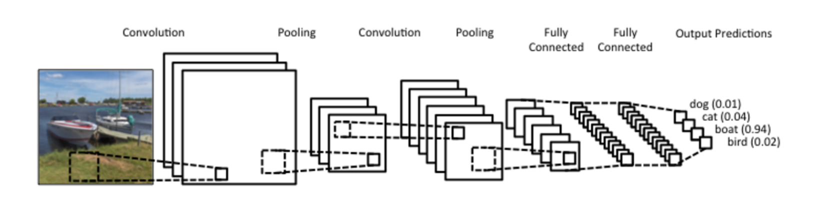


Figure from Kevin McGuinness

- Recognition CNN model, with sigmoid output layer
- Binary classification output: real / synthetic

Generator architecture for images

- Unit Gaussian prior on $\mathbf{z} \in \mathbb{R}^D$, typically 10^2 to 10^3 dimensions

Generator architecture for images

- Unit Gaussian prior on $\mathbf{z} \in \mathbb{R}^D$, typically 10^2 to 10^3 dimensions
- Up-convolutional deep network (reverse recognition CNN)
  - Pooling layers replaced with upsampling layers (nearest neighbor, bi-linear, or learned)

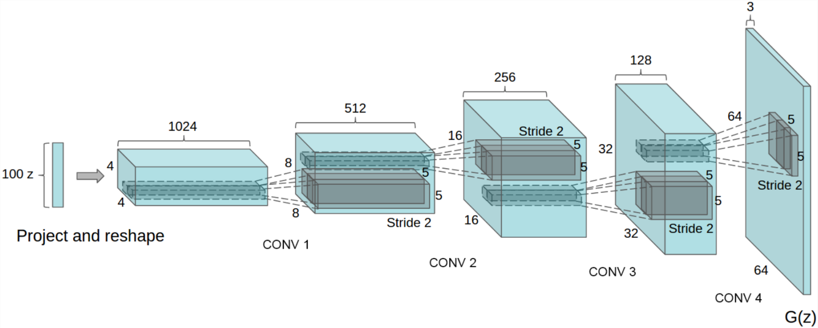


Figure from [Radford et al., 2016]

Generator architecture for images

- Unit Gaussian prior on $\mathbf{z} \in \mathbb{R}^D$, typically 10^2 to 10^3 dimensions
- Up-convolutional deep network (reverse recognition CNN)
  - Pooling layers replaced with upsampling layers (nearest neighbor, bi-linear, or learned)
  - Low-resolution layers induce long-range correlations
  - High-resolution layers induce short-range correlations

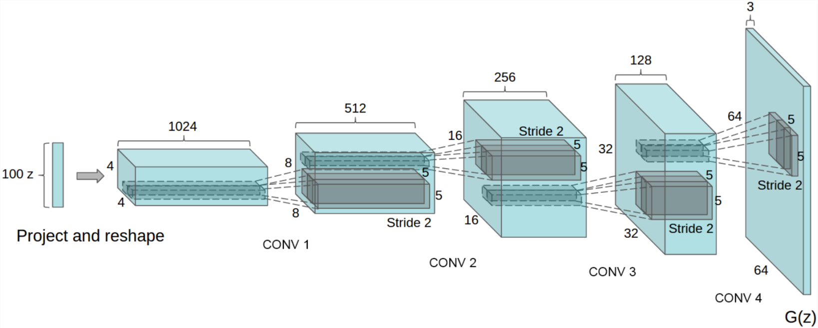
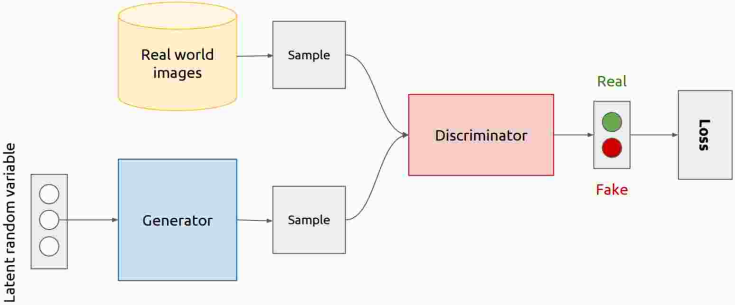


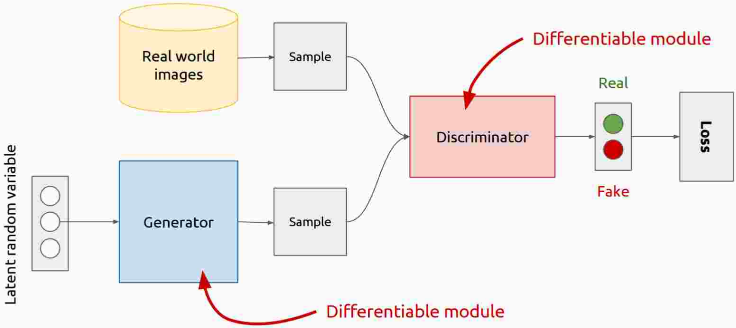
Figure from [Radford et al., 2016]

Training GANs



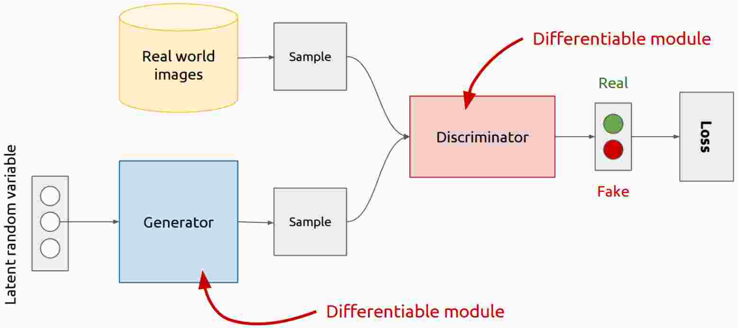
- Discriminator: maximize classification for a given generator
- Generator: degrade classification of a given discriminator

Training GANs



- Discriminator: maximize classification for a given generator
- Generator: degrade classification of a given discriminator
- Samples $\mathbf{z}$ pass through two differentiable modules

Training GANs



- Discriminator: maximize classification for a given generator
- Generator: degrade classification of a given discriminator

- Samples **z** pass through two differentiable modules
- Discriminator acts as **trainable loss function**

GAN Optimization problem

- Objective function $V(\phi, \theta)$: performance of discriminator

$$V(\phi, \theta) = \mathbb{E}_{x \sim p_{\mathrm{data}}(x)}[\ln D_\phi(x)] + \mathbb{E}_{x \sim p(\mathbf{z})}[\ln(1 - D_\phi(G_\theta(\mathbf{z})))]$$

GAN Optimization problem

- Objective function $V(\phi, \theta)$: performance of discriminator

$$V(\phi, \theta) = \mathbb{E}_{x \sim p_{\mathrm{data}}(x)}[\ln D_\phi(x)] + \mathbb{E}_{x \sim p(\mathbf{z})}[\ln(1 - D_\phi(G_\theta(\mathbf{z})))]$$

$$\min_\theta \max_\phi V(\phi, \theta)$$

GAN Optimization problem

- Objective function $V(\phi, \theta)$: performance of discriminator

$$V(\phi, \theta) = \mathbb{E}_{x \sim p_{\mathrm{data}}(x)}[\ln D_\phi(x)] + \mathbb{E}_{x \sim p(\mathbf{z})}[\ln (1 - D_\phi(G_\theta(\mathbf{z})))]$$

$$\min_\theta \max_\phi V(\phi, \theta)$$

- Assuming infinite data and model capacity,
  and reaching optimal discriminator at each iteration

GAN Optimization problem

- Objective function $V(\phi, \theta)$: performance of discriminator

$$V(\phi, \theta) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\ln D_\phi(x)] + \mathbb{E}_{x \sim p(\mathbf{z})}[\ln(1 - D_\phi(G_\theta(\mathbf{z})))]$$

$$\min_\theta \max_\phi V(\phi, \theta)$$

- Assuming infinite data and model capacity, and reaching optimal discriminator at each iteration
    1. Unique global optimum for G at data distribution

GAN Optimization problem

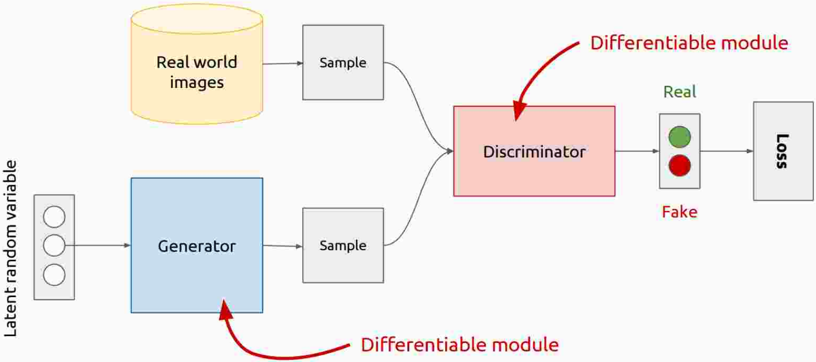
- Objective function $V(\phi, \theta)$: performance of discriminator

$$V(\phi, \theta) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\ln D_\phi(x)] + \mathbb{E}_{x \sim p(\mathbf{z})}[\ln (1 - D_\phi(G_\theta(\mathbf{z})))]$$

$$\min_\theta \max_\phi V(\phi, \theta)$$

- Assuming infinite data and model capacity, and reaching optimal discriminator at each iteration
  1. Unique global optimum for G at data distribution
  2. Convergence to optimum guaranteed

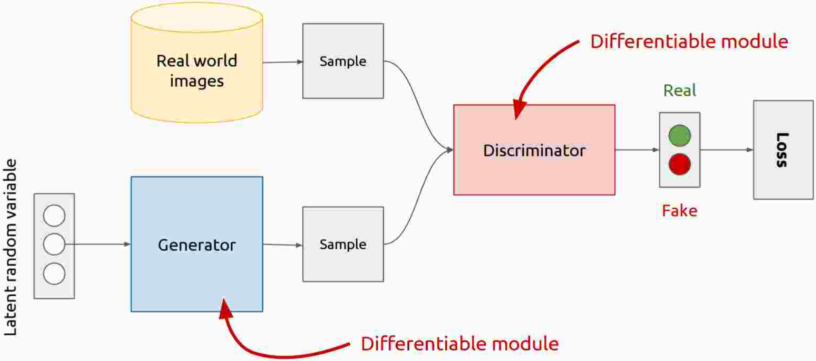
Training GANs in practice



$$V(\phi, \theta) = \mathbb{E}_{x \sim p_{\mathrm{data}}(x)}[\ln D_\phi(x)] + \mathbb{E}_{z \sim p(z)}[\ln(1 - D_\phi(f_\theta(z)))]$$

- Replace expectations with sample average in mini-batch

Training GANs in practice



$$V(\phi, \theta) = \mathbb{E}_{x \sim p_{\mathrm{data}}(x)}[\ln D_\phi(x)] + \mathbb{E}_{z \sim p(z)}[\ln(1 - D_\phi(f_\theta(z)))]$$

- Replace expectations with sample average in mini-batch
- Parallel stochastic gradient descent on ϕ and θ

Issues with GANs in practice

Issues with GANs in practice

- GANs known to be difficult to train in practice
  - Formulated as mini-max objective between two networks
  - Optimization can oscillate between solutions
  - Picking "compatible" generator and discriminator architectures
  - Training fails if the discriminator is too strong

Issues with GANs in practice

- GANs known to be difficult to train in practice
  - Formulated as mini-max objective between two networks
  - Optimization can oscillate between solutions
  - Picking "compatible" generator and discriminator architectures
  - Training fails if the discriminator is too strong
- Mode collapse: failure to capture parts of training data
  - Optimizes KL-divergence in the "wrong" direction, reverse from MLE [Lucas et al., 2019]

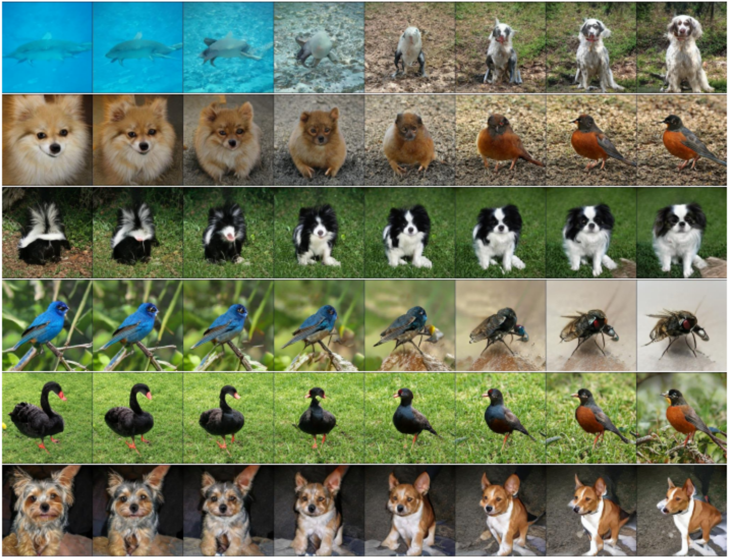
GANs offer outstanding sample quality

GANs offer outstanding sample quality

Class conditional ProGan [Karras et al., 2018] samples, for LSUN 256×256

GAN generalizes beyond training data

GAN generalizes beyond training data



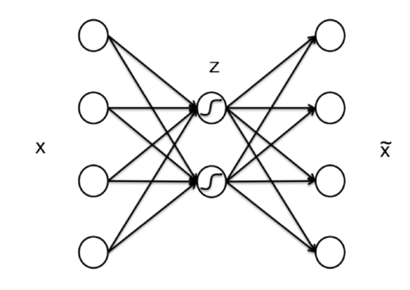
Examples taken from Brock et al. 2019

Part IV

Variational Autoencoders

Autoencoders

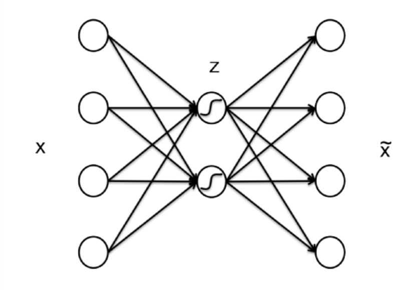
- Learn latent representation **z** via reconstruction of data **x**



Autoencoders

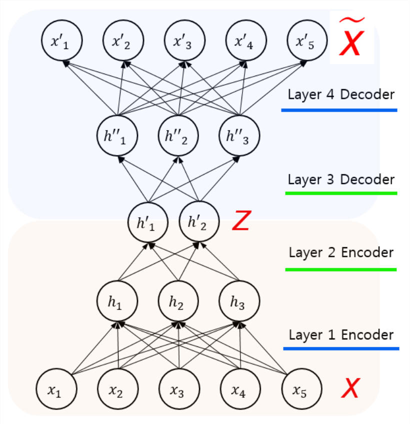
- Learn latent representation **z** via reconstruction of data **x**
- Autoencoder recovers PCA if [Baldi and Hornik, 1989]
  1. Encoder and decoder are both linear
  2. Optimizing ℓ_2 reconstruction loss

$$\min_{V,W} \sum_{n=1}^{N} ||x_n - VWx_n||^2 \tag{9}$$



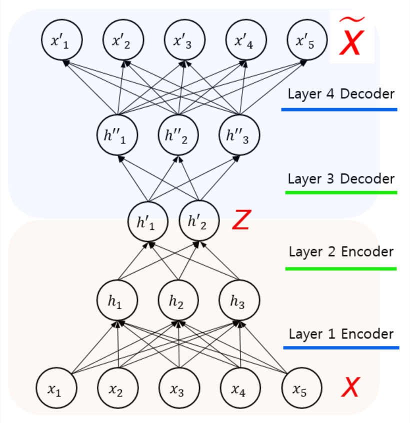
Deep non-linear autoencoders

- Stack many non-linear layers in encoder and decoder



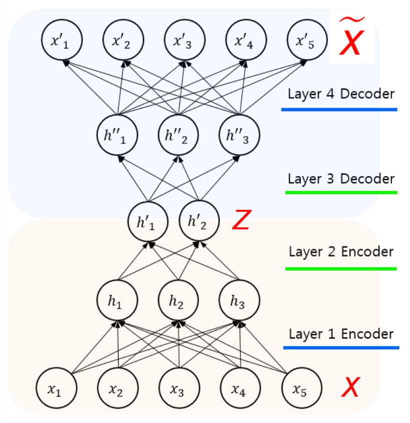
Deep non-linear autoencoders

- Stack many non-linear layers in encoder and decoder
- Non-linear representation learning



Deep non-linear autoencoders

- Stack many non-linear layers in encoder and decoder
- Non-linear representation learning
- Does not provide a generative model that can be sampled



Autoencoding variational Bayes [Kingma and Welling, 2014]

Autoencoding variational Bayes [Kingma and Welling, 2014]

- Encoder g compute approximate posterior distribution
  - Maps data $\mathbf{x}$ to latent code $\mathbf{z}$

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; g_\phi^\mu(\mathbf{x}), g_\phi^\sigma(\mathbf{x})) \tag{10}$$

Autoencoding variational Bayes [Kingma and Welling, 2014]

- Encoder g compute approximate posterior distribution
  - Maps data $\mathbf{x}$ to latent code $\mathbf{z}$

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; g_\phi^\mu(\mathbf{x}), g_\phi^\sigma(\mathbf{x})) \tag{10}$$

- Decoder f implements generative latent variable model
  - Maps latent code $\mathbf{z}$ to observation $\mathbf{x}$

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; f_\theta^\mu(\mathbf{z}), f_\theta^\sigma(\mathbf{z})) \tag{11}$$

Autoencoding variational Bayes [Kingma and Welling, 2014]

- Encoder g compute approximate posterior distribution
  - Maps data $\mathbf{x}$ to latent code $\mathbf{z}$

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; g_\phi^\mu(\mathbf{x}), g_\phi^\sigma(\mathbf{x})) \tag{10}$$

- Decoder f implements generative latent variable model
  - Maps latent code $\mathbf{z}$ to observation $\mathbf{x}$

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; f_\theta^\mu(\mathbf{z}), f_\theta^\sigma(\mathbf{z})) \tag{11}$$

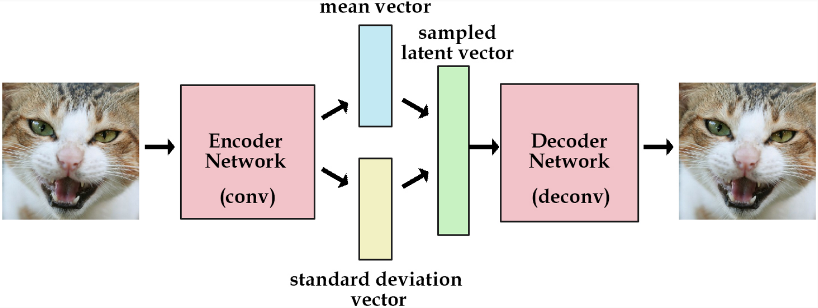


Figure from kvfrans@github

Objective function: Evidence lower bound (ELBO)

- Variational bound on data likelihood using Jensen inequality
  - Same bound that underlies the EM algorithm

Objective function: Evidence lower bound (ELBO)

- Variational bound on data likelihood using Jensen inequality
  - Same bound that underlies the EM algorithm

$$
\begin{aligned}
\ln p_\theta(\mathbf{x}) &\geq \ln p_\theta(\mathbf{x}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) && (12)\\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\ln(p_\theta(\mathbf{x}|\mathbf{z}))] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) && (13)
\end{aligned}
$$

Objective function: Evidence lower bound (ELBO)

- Variational bound on data likelihood using Jensen inequality
  - Same bound that underlies the EM algorithm

$$\ln p_\theta(\mathbf{x}) \geq \ln p_\theta(\mathbf{x}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) \tag{12}$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\ln(p_\theta(\mathbf{x}|\mathbf{z}))] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \tag{13}$$

- ELBO is function of inference net and generative net

$$F(\theta, \phi) = \mathbb{E}_{q_\phi}[\ln p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\big) \tag{14}$$

Objective function: Evidence lower bound (ELBO)

- Variational bound on data likelihood using Jensen inequality
  - Same bound that underlies the EM algorithm

$$\ln p_\theta(\mathbf{x}) \geq \ln p_\theta(\mathbf{x}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) \quad (12)$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\ln(p_\theta(\mathbf{x}|\mathbf{z}))] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \quad (13)$$

- ELBO is function of inference net and generative net

$$F(\theta, \phi) = \mathbb{E}_{q_\phi}[\ln p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\big) \quad (14)$$

- Optimize both networks jointly with SGD

Computation ELBO for variational autoencoder

$$F(\theta, \phi) = \underbrace{\mathbb{E}_{q_\phi}[\ln p_\theta(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction}} - \underbrace{D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\big)}_{\textbf{Regularization}} \tag{15}$$

- **Regularization term** keeps q from collapsing to single point $\mathbf{z}$

Computation ELBO for variational autoencoder

$$F(\theta, \phi) = \underbrace{\mathbb{E}_{q_\phi}[\ln p_\theta(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction}} - \underbrace{D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right)}_{\textbf{Regularization}} \tag{15}$$

- **Regularization term** keeps q from collapsing to single point $\mathbf{z}$
- Closed form if both terms are Gaussian, for $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I)$

$$D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right) = \frac{1}{2}\left[1 + \ln g_\phi^\sigma(\mathbf{x}) - g_\phi^\mu(\mathbf{x}) - g_\phi^\sigma(\mathbf{x})\right] \tag{16}$$

Computation ELBO for variational autoencoder

$$F(\theta, \phi) = \underbrace{\mathbb{E}_{q_\phi}[\ln p_\theta(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction}} - \underbrace{D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right)}_{\textbf{Regularization}} \tag{15}$$

- **Regularization term** keeps q from collapsing to single point $\mathbf{z}$
- Closed form if both terms are Gaussian, for $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I)$

$$D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) = \frac{1}{2}\left[1 + \ln g_\phi^\sigma(\mathbf{x}) - g_\phi^\mu(\mathbf{x}) - g_\phi^\sigma(\mathbf{x})\right] \tag{16}$$

- Differentiable function of inference net parameters

Computation ELBO for variational autoencoder

$$F(\theta, \phi) = \underbrace{\mathbb{E}_{q_\phi}[\ln p_\theta(\mathbf{x}|\mathbf{z})]}_{\textbf{Reconstruction}} - \underbrace{D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\big)}_{\text{Regularization}} \tag{17}$$

- **Reconstruction term**: to what extent can $\mathbf{x}$ be reconstructed from $\mathbf{z}$ following approximate posterior $q(\mathbf{z}|\mathbf{x})$

Computation ELBO for variational autoencoder

$$F(\theta, \phi) = \underbrace{\mathbb{E}_{q_\phi}[\ln p_\theta(\mathbf{x}|\mathbf{z})]}_{\textbf{Reconstruction}} - \underbrace{D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\big)}_{\text{Regularization}} \tag{17}$$

- **Reconstruction term**: to what extent can $\mathbf{x}$ be reconstructed from $\mathbf{z}$ following approximate posterior $q(\mathbf{z}|\mathbf{x})$
- Use unbiased sample approximation of intractable expectation $\mathbf{z_s} \sim q_\phi(\mathbf{z}|\mathbf{x})$

$$\mathbb{E}_{q_\phi}[\ln p_\theta(\mathbf{x}|\mathbf{z})] \approx \frac{1}{S}\sum_{s=1}^{S} \ln p_\theta(\mathbf{x}|\mathbf{z_s}) \tag{18}$$

Computation ELBO for variational autoencoder

$$F(\theta, \phi) = \underbrace{\mathbb{E}_{q_\phi}[\ln p_\theta(\mathbf{x}|\mathbf{z})]}_{\textbf{Reconstruction}} - \underbrace{D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\big)}_{\text{Regularization}} \tag{17}$$

- **Reconstruction term**: to what extent can $\mathbf{x}$ be reconstructed from $\mathbf{z}$ following approximate posterior $q(\mathbf{z}|\mathbf{x})$
- Use unbiased sample approximation of intractable expectation $\mathbf{z_s} \sim q_\phi(\mathbf{z}|\mathbf{x})$

$$\mathbb{E}_{q_\phi}[\ln p_\theta(\mathbf{x}|\mathbf{z})] \approx \frac{1}{S}\sum_{s=1}^{S} \ln p_\theta(\mathbf{x}|\mathbf{z_s}) \tag{18}$$

- Estimator is non-differentiable due to sampling operator

Re-parametrization trick

- Side-step non-differentiable sampling operator by re-parametrizing samples $\mathbf{z_s} \sim q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}\left(\mathbf{z}; g_\phi^\mu(\mathbf{x}), g_\phi^\sigma(\mathbf{x})\right)$

Re-parametrization trick

- Side-step non-differentiable sampling operator by re-parametrizing samples $\mathbf{z_s} \sim q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}\left(\mathbf{z}; g_\phi^\mu(\mathbf{x}), g_\phi^\sigma(\mathbf{x})\right)$
- Use inference net to modulate samples from a unit Gaussian

$$\mathbf{z}_s = g_\phi^\mu(\mathbf{x}) + g_\phi^\sigma(\mathbf{x}) \odot \epsilon_s, \qquad \epsilon_s \sim \mathcal{N}(\epsilon_s; 0, I) \tag{19}$$

Re-parametrization trick

- Side-step non-differentiable sampling operator by re-parametrizing samples $\mathbf{z_s} \sim q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}\left(\mathbf{z}; g_\phi^\mu(\mathbf{x}), g_\phi^\sigma(\mathbf{x})\right)$
- Use inference net to modulate samples from a unit Gaussian

$$\mathbf{z}_s = g_\phi^\mu(\mathbf{x}) + g_\phi^\sigma(\mathbf{x}) \odot \epsilon_s, \qquad \epsilon_s \sim \mathcal{N}(\epsilon_s; 0, I) \tag{19}$$

- Samples $\mathbf{z}_s$ differentiable function of inference net param. ϕ, given unit Gaussian samples ϵ_s

Re-parametrization trick

- Side-step non-differentiable sampling operator by re-parametrizing samples $\mathbf{z_s} \sim q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}\left(\mathbf{z}; g_\phi^\mu(\mathbf{x}), g_\phi^\sigma(\mathbf{x})\right)$

- Use inference net to modulate samples from a unit Gaussian

$$\mathbf{z}_s = g_\phi^\mu(\mathbf{x}) + g_\phi^\sigma(\mathbf{x}) \odot \epsilon_s, \qquad \epsilon_s \sim \mathcal{N}(\epsilon_s; 0, I) \tag{19}$$

- Samples $\mathbf{z}_s$ differentiable function of inference net param. ϕ, given unit Gaussian samples ϵ_s

- Unbiased differentiable approximation of ELBO

$$F(\theta, \phi) \approx \frac{1}{S} \sum_{s=1}^{S} \ln p_\theta\left(\mathbf{x}|g_\phi^\mu(\mathbf{x}) + g_\phi^\sigma(\mathbf{x}) \odot \epsilon_s\right) \tag{20}$$

$$-\frac{1}{2}\left[1 + \ln g_\phi^\sigma(\mathbf{x}) - g_\phi^\mu(\mathbf{x}) - g_\phi^\sigma(\mathbf{x})\right] \tag{21}$$

Re-parametrization trick in a cartoon

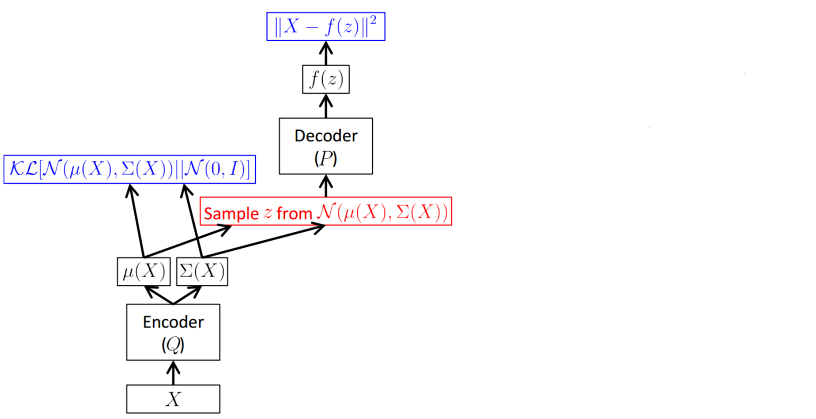


Figure from [Doersch, 2016]

Re-parametrization trick in a cartoon

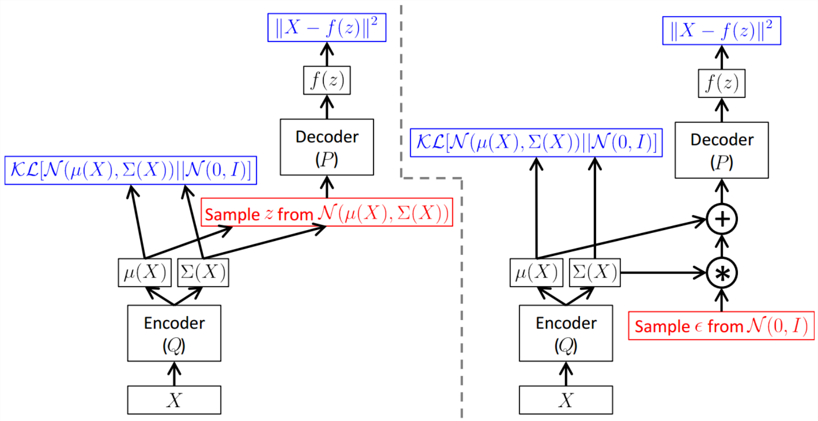


Figure from [Doersch, 2016]

Autoencoding variational Bayes training algorithm

- For each data point $\mathbf{x}$ in a mini-batch
  1. Sample one or multiple values $\{\epsilon_s\}$
  2. Use back-propagation to compute
     $g_\theta = \nabla_\theta F(\theta, \phi, \{\epsilon_s\})$
     $g_\phi = \nabla_\phi F(\theta, \phi, \{\epsilon_s\})$
  3. Gradient-based parameter update

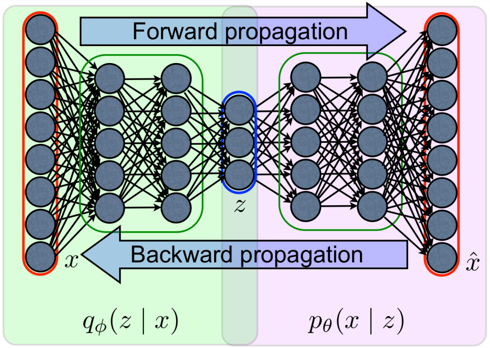


Figure from Aaron Courville

VAE compared to GAN

- VAE does not suffer from GAN training instability
- GANs typically have higher sample quality than VAE
- VAE defines likelihood $p(\mathbf{x})$ for all data $\mathbf{x}$,
  can **e.g**. be used for loss-less compression

Figure from [Hou et al., 2017], models trained on CelebA dataset

Part V

Deep invertible transformations

Modeling via the change of variable formula

- Learn invertible "flow", $f(\cdot)$, between latent and data space

Modeling via the change of variable formula

- Learn invertible "flow", $f(\cdot)$, between latent and data space
- Latent and data space have same dimensionality

Modeling via the change of variable formula

- Learn invertible "flow", $f(\cdot)$, between latent and data space
- Latent and data space have same dimensionality

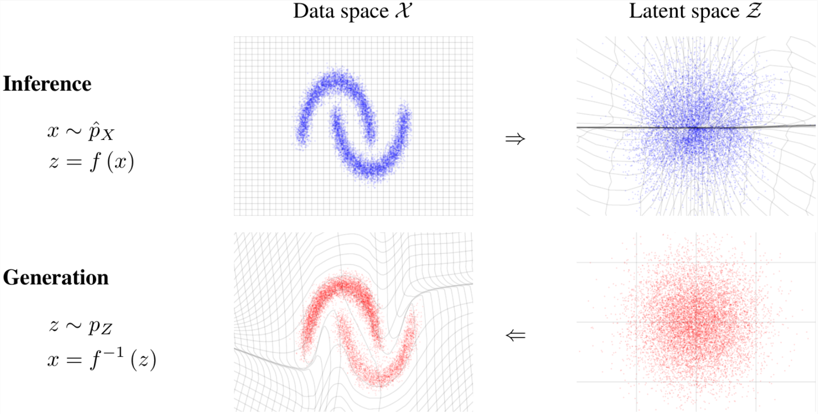


Figure from [Dinh et al., 2017]

Change of variables formula for invertible function

- Express density estimation in latent space

$$
\begin{aligned}
\mathbf{y} &= f(\mathbf{x}), && (22)\\
J_f &= \frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top}, && (23)\\
p_X(\mathbf{x}) &= p_Y(\mathbf{y}) \times |\det(J_f)| && (24)
\end{aligned}
$$

Change of variables formula for invertible function

- Express density estimation in latent space

$$\mathbf{y} = f(\mathbf{x}), \tag{22}$$

$$J_f = \frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top}, \tag{23}$$

$$p_X(\mathbf{x}) = p_Y(\mathbf{y}) \times \left|\det\left(J_f\right)\right| \tag{24}$$

- Place simple prior on latent variables, **e.g**. unit Gaussian

Change of variables formula for invertible function

- Express density estimation in latent space

$$\mathbf{y} = f(\mathbf{x}), \tag{22}$$

$$J_f = \frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top}, \tag{23}$$

$$p_X(\mathbf{x}) = p_Y(\mathbf{y}) \times |\det(J_f)| \tag{24}$$

- Place simple prior on latent variables, **e.g**. unit Gaussian
- Sampling: $\mathbf{y} \sim p(\mathbf{y})$, map through inverse $\mathbf{x} = f^{-1}(\mathbf{y})$

Change of variables formula for invertible function

- Express density estimation in latent space

$$\mathbf{y} = f(\mathbf{x}), \tag{22}$$

$$J_f = \frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top}, \tag{23}$$

$$p_X(\mathbf{x}) = p_Y(\mathbf{y}) \times \left|\det\left(J_f\right)\right| \tag{24}$$

- Place simple prior on latent variables, **e.g**. unit Gaussian
- Sampling: $\mathbf{y} \sim p(\mathbf{y})$, map through inverse $\mathbf{x} = f^{-1}(\mathbf{y})$
- Naive computation of determinant costs $O(D^3)$

Change of variables formula for invertible function

- Express density estimation in latent space

$$\mathbf{y} = f(\mathbf{x}), \tag{22}$$

$$J_f = \frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top}, \tag{23}$$

$$p_X(\mathbf{x}) = p_Y(\mathbf{y}) \times \left|\det\left(J_f\right)\right| \tag{24}$$

- Place simple prior on latent variables, **e.g**. unit Gaussian
- Sampling: $\mathbf{y} \sim p(\mathbf{y})$, map through inverse $\mathbf{x} = f^{-1}(\mathbf{y})$
- Naive computation of determinant costs $O(D^3)$
- Impose structure on $f(\cdot)$ to make both operations efficient

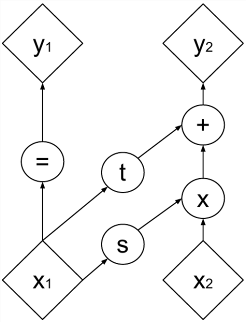
Non-volume Preserving transformations (NVP) [Dinh et al., 2017]

- Stack many invertible "coupling layers"
- Each has simple inverse and determinant

Non-volume Preserving transformations (NVP) [Dinh et al., 2017]

- Stack many invertible "coupling layers"
- Each has simple inverse and determinant

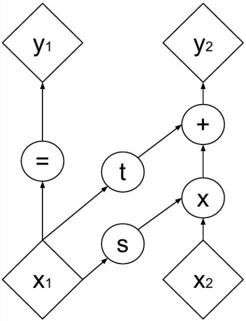
1. Partition variables in groups $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$.
   For example, half of pixels in one group



Non-volume Preserving transformations (NVP) [Dinh et al., 2017]

- Stack many invertible "coupling layers"
- Each has simple inverse and determinant

1. Partition variables in groups $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$.
   For example, half of pixels in one group
2. Keep group $\mathbf{x}_1$ unchanged



Non-volume Preserving transformations (NVP) [Dinh et al., 2017]

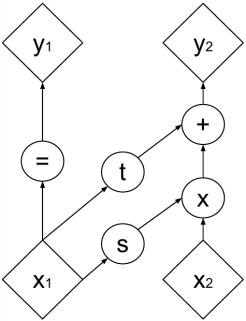
- Stack many invertible "coupling layers"
- Each has simple inverse and determinant

1. Partition variables in groups $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$.
   For example, half of pixels in one group

2. Keep group $\mathbf{x}_1$ unchanged

3. Let $\mathbf{x}_1$ transform $\mathbf{x}_2$ via translation and scaling

$$\begin{aligned} \mathbf{y}_1 &= \mathbf{x}_1 \\ \mathbf{y}_2 &= t(\mathbf{x}_1) + \mathbf{x}_2 \odot \exp(s(\mathbf{x}_1)) \end{aligned}$$

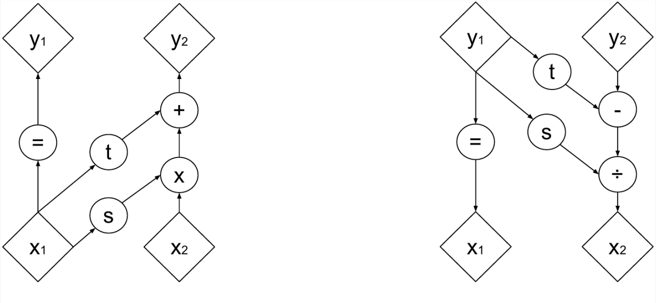


Properties: Efficient inversion

- Inverse transformation

$$\mathbf{x}_1 = \mathbf{y}_1 \tag{25}$$

$$\mathbf{x}_2 = (\mathbf{y}_2 - t(\mathbf{x}_1)) \odot \exp(-s(\mathbf{x}_1)) \tag{26}$$



(a) Forward propagation (b) Inverse propagation

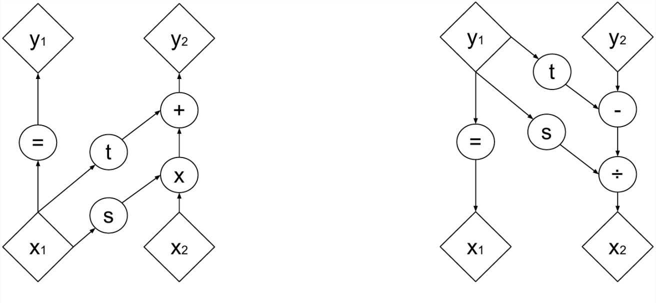
Properties: Efficient inversion

- Inverse transformation

$$\mathbf{x}_1 = \mathbf{y}_1 \tag{25}$$

$$\mathbf{x}_2 = (\mathbf{y}_2 - t(\mathbf{x}_1)) \odot \exp(-s(\mathbf{x}_1)) \tag{26}$$

- No need to invert $s(\cdot)$ and $t(\cdot)$
- Can use complex non-invertible functions, **e.g**. deep CNN



(a) Forward propagation (b) Inverse propagation

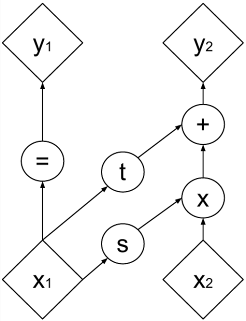
Properties: Efficient determinant computation

- Triangular structure of Jacobian

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}^\top} = \begin{bmatrix} I_d & 0 \\ \frac{\partial \mathbf{y}_2}{\partial \mathbf{x}_1^\top} & \mathrm{diag}(\exp(s(\mathbf{x}_1))) \end{bmatrix}$$

- Determinant given by product of Jacobian's diagonal terms

$$\ln \left| \det \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}^\top} \right) \right| = \mathbf{1}^\top s(\mathbf{x}_1)$$



Properties: Efficient determinant computation

- Triangular structure of Jacobian

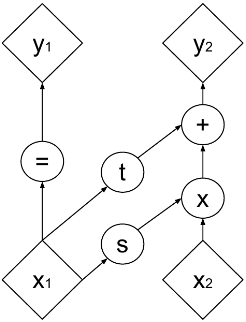
$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}^\top} = \begin{bmatrix} I_d & 0 \\ \frac{\partial \mathbf{y}_2}{\partial \mathbf{x}_1^\top} & \operatorname{diag}(\exp(s(\mathbf{x}_1))) \end{bmatrix}$$

- Determinant given by product of Jacobian's diagonal terms

$$\ln \left| \det \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}^\top} \right) \right| = \mathbf{1}^\top s(\mathbf{x}_1)$$

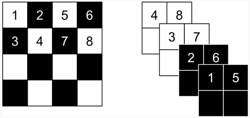
- Log-likelihood easily computed, optimize using stochastic gradient decent

$$\ln p_X(\mathbf{x}) = \ln p_Y(f(\mathbf{x})) + \mathbf{1}^\top s(\mathbf{x}_1)$$

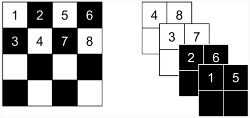


Implementation

- Layers cycle through various partitionings
  - Checkerboard mask
  - Channel-wise mask



Implementation

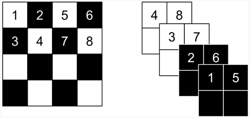


- Layers cycle through various partitionings
    - Checkerboard mask
    - Channel-wise mask

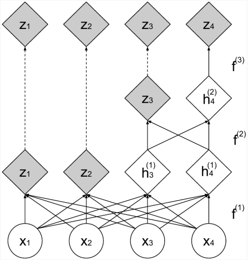
- Multi-scale architecture

Implementation

- Layers cycle through various partitionings
  - Checkerboard mask
  - Channel-wise mask

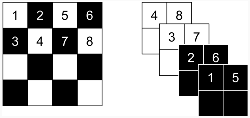


- Multi-scale architecture
  - Down sample at regular intervals

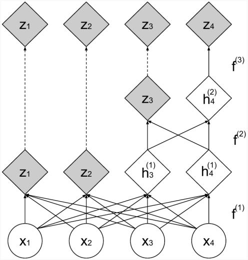


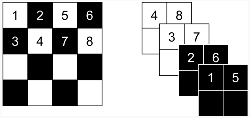
Implementation

- Layers cycle through various partitionings
  - Checkerboard mask
  - Channel-wise mask

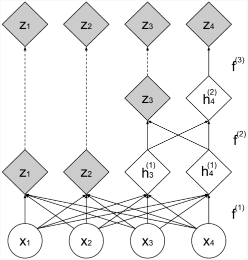


- Multi-scale architecture
  - Down sample at regular intervals
  - Squeeze $2h \times 2w \times c$ map into $h \times w \times 4c$





- Layers cycle through various partitionings
  - Checkerboard mask
  - Channel-wise mask



- Multi-scale architecture
  - Down sample at regular intervals
  - Squeeze $2h \times 2w \times c$ map into $h \times w \times 4c$
  - "Freeze" half the channels / latent vars.

Illustration multi-scale feature hierarchy

- Images obtained after re-sampling part of latent variables

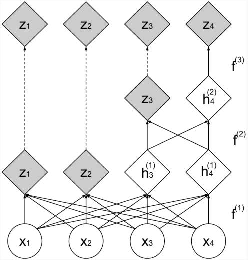


Illustration multi-scale feature hierarchy

- Images obtained after re-sampling part of latent variables
- From left to right: original, keeping $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}$

ImageNet 64×64

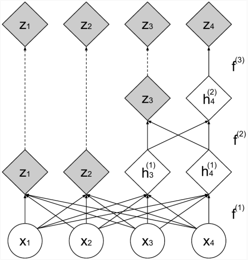
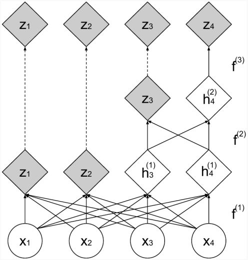


Illustration multi-scale feature hierarchy

- Images obtained after re-sampling part of latent variables
- From left to right: original, keeping $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}$

ImageNet 64×64

CelebA 64×64



Flow vs. VAE & GAN

- Flow offers stable training ($\neq$GAN) with exact likelihood ($\neq$VAE)
- VAE offers best likelihood on held-out data
- GAN may offer best samples, but flows can come very close

Flow vs. VAE & GAN

- Flow offers stable training ($\neq$GAN) with exact likelihood ($\neq$VAE)
- VAE offers best likelihood on held-out data
- GAN may offer best samples, but flows can come very close

Samples from flow model trained on CelebA 256$\times$256 [Kingma and Dhariwal, 2018]

Part VI

Autoregressive density estimation

Autoregressive modeling

- Avoid intractable integral over latent variables

Autoregressive modeling

- Avoid intractable integral over latent variables
- Consider generic factorization of joint probability

$$p(\mathbf{x}_{1:D}) = p(x_1) \prod_{i=2}^{D} p(x_i | \mathbf{x}_{<i}) \tag{27}$$

with $\mathbf{x}_{<i} = \mathbf{x}_1, \ldots, \mathbf{x}_{i-1}$

Autoregressive modeling

- Avoid intractable integral over latent variables
- Consider generic factorization of joint probability

$$p(\mathbf{x}_{1:D}) = p(x_1) \prod_{i=2}^{D} p(x_i | \mathbf{x}_{<i}) \tag{27}$$

with $\mathbf{x}_{<i} = \mathbf{x}_1, \ldots, \mathbf{x}_{i-1}$

- Use deep neural net to model complex conditionals $p(x_i | \mathbf{x}_{<i})$

Autoregressive modeling

- Avoid intractable integral over latent variables
- Consider generic factorization of joint probability

$$p(\mathbf{x}_{1:D}) = p(x_1) \prod_{i=2}^{D} p(x_i | \mathbf{x}_{<i}) \tag{27}$$

  with $\mathbf{x}_{<i} = \mathbf{x}_1, \ldots, \mathbf{x}_{i-1}$

- Use deep neural net to model complex conditionals $p(x_i|\mathbf{x}_{<i})$
- Tractable exact likelihood computations

Autoregressive modeling

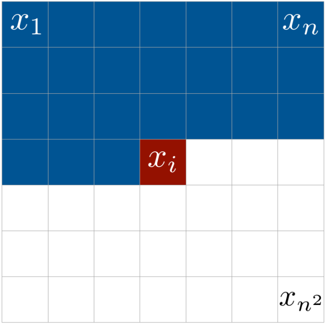
- Avoid intractable integral over latent variables
- Consider generic factorization of joint probability

$$p(\mathbf{x}_{1:D}) = p(x_1) \prod_{i=2}^{D} p(x_i|\mathbf{x}_{<i}) \tag{27}$$

  with $\mathbf{x}_{<i} = \mathbf{x}_1, \ldots, \mathbf{x}_{i-1}$
- Use deep neural net to model complex conditionals $p(x_i|\mathbf{x}_{<i})$
- Tractable exact likelihood computations
- Slow sequential one-by-one sampling of pixels
  - Cannot rely on latent variables to induce dependencies

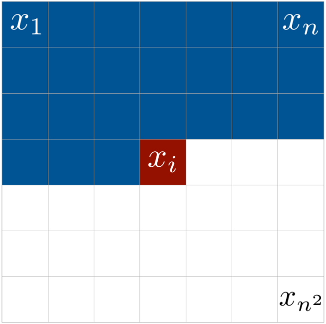
Pixel Convolutional Neural Networks [Oord et al., 2016a]

- Predict pixels one-by-one in row-major ordering



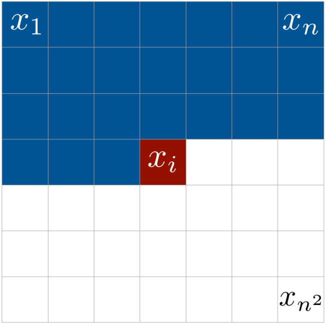
Pixel Convolutional Neural Networks [Oord et al., 2016a]

- Predict pixels one-by-one in row-major ordering
- Translation invariant definition of conditionals $p(x_i|\mathbf{x}_{<i})$



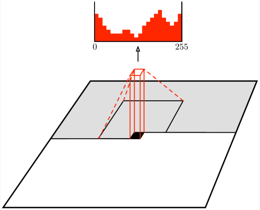
Pixel Convolutional Neural Networks [Oord et al., 2016a]

- Predict pixels one-by-one in row-major ordering
- Translation invariant definition of conditionals $p(x_i|\mathbf{x}_{<i})$
- Decouple number of pixels from number of parameters



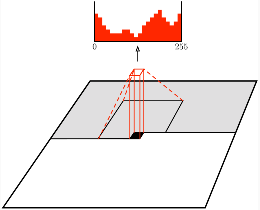
Pixel Convolutional Neural Networks

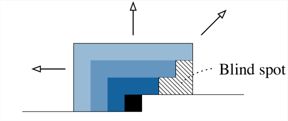
- Use limited context via CNN layers
    - Only local dependencies per layer
- Masked convolutions to ensure autoregressive property



Pixel Convolutional Neural Networks

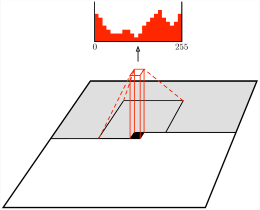
- Use limited context via CNN layers
  - Only local dependencies per layer
  - Adding layers increases context
- Masked convolutions to ensure autoregressive property

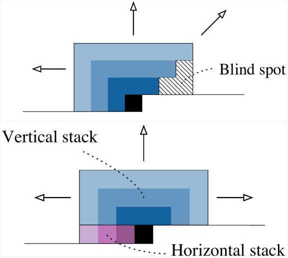




Pixel Convolutional Neural Networks

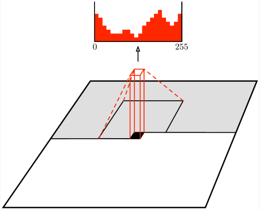
- Use limited context via CNN layers
  - Only local dependencies per layer
  - Adding layers increases context
- Masked convolutions to ensure autoregressive property
  - Block pixels below / right

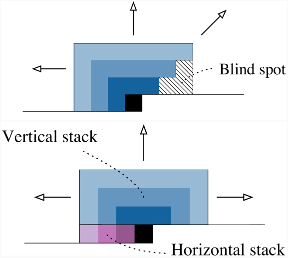




Pixel Convolutional Neural Networks

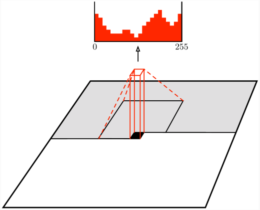
- Use limited context via CNN layers
  - Only local dependencies per layer
  - Adding layers increases context
- Masked convolutions to ensure autoregressive property
  - Block pixels below / right
  - Blind spot filled using two feature stacks

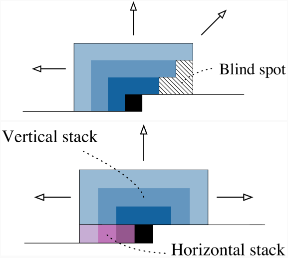




Pixel Convolutional Neural Networks

- Use limited context via CNN layers
  - Only local dependencies per layer
  - Adding layers increases context
- Masked convolutions to ensure autoregressive property
  - Block pixels below / right
  - Blind spot filled using two feature stacks
- Efficient parallel training, sampling remains slow





WaveNet: Autoregressive audio model

- Autoregressive CNN model in 1 dimension of raw waveform

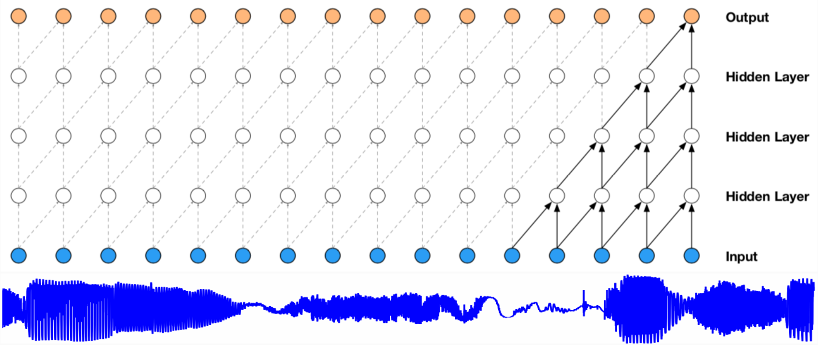


Figure from [Kalchbrenner et al., 2017]

Multiscale autoregressive modeling [Reed et al., 2017]

- Address the inherently limited sampling efficiency of autoregressive models

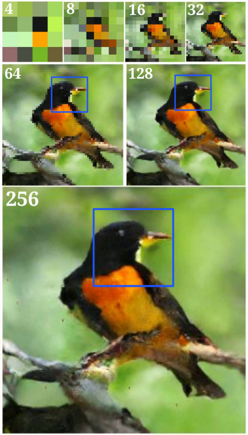
$$p(\mathbf{x}_{1:N}) = \prod_{i=1}^{N} p(x_i | \mathbf{x}_{<i})$$

Multiscale autoregressive modeling [Reed et al., 2017]

- Address the inherently limited sampling efficiency of autoregressive models

$$p(\mathbf{x}_{1:N}) = \prod_{i=1}^{N} p(x_i | \mathbf{x}_{<i})$$

- Sample image along a scale pyramid
  - Pixel-CNN for base resolution, **e.g**. 4×4
  - Autoregressive upsampling networks

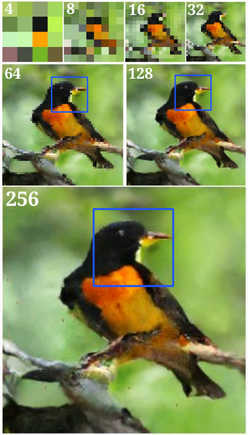


Multiscale autoregressive modeling [Reed et al., 2017]

- Address the inherently limited sampling efficiency of autoregressive models

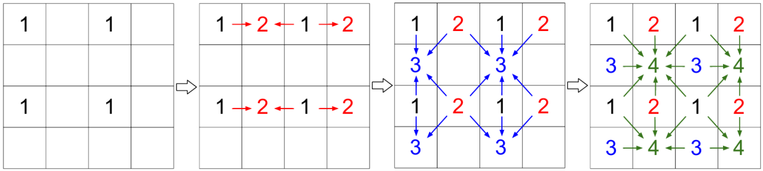
$$p(\mathbf{x}_{1:N}) = \prod_{i=1}^{N} p(x_i|\mathbf{x}_{<i})$$

- Sample image along a scale pyramid
  - Pixel-CNN for base resolution, **e.g**. 4×4
  - Autoregressive upsampling networks
- Impose group structure among pixels
  - Sample independent within group
  - Sample autoregressive across groups



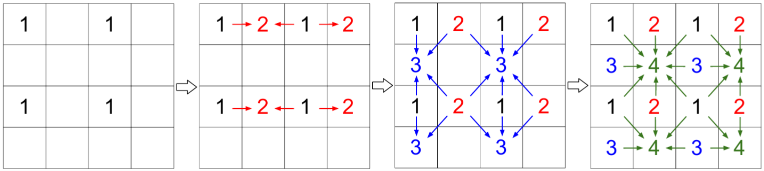
Sampling pixels in groups

- Group pixels along position in 2×2 blocks
  - Group 1 given from previous resolution
  - Sample remaining pixels in three steps

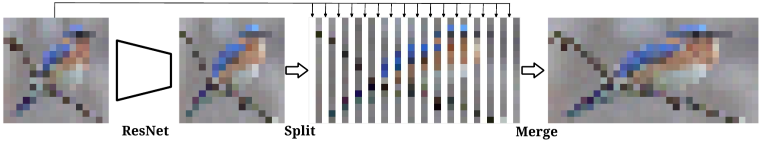


Sampling pixels in groups

- Group pixels along position in 2×2 blocks
  - Group 1 given from previous resolution
  - Sample remaining pixels in three steps



- Example network to predict group 2 from group 1
  - Use CNN without pooling to predict/sample new columns
  - Interleave pixel columns from group 1 and 2



Example results of upsampling real low-resolution images

- About 100× speed-up w.r.t. pixel-CNN sampling

Pixel CNN compared to VAE and GAN

- Exact likelihoods unlike VAE and GAN

Class-conditional pixelCNN 32×32 samples trained on ImageNet [Oord et al., 2016b]

Pixel CNN compared to VAE and GAN

- Exact likelihoods unlike VAE and GAN
- No latent variable representation learning

Class-conditional pixelCNN 32×32 samples trained on ImageNet [Oord et al., 2016b]

Pixel CNN compared to VAE and GAN

- Exact likelihoods unlike VAE and GAN
- No latent variable representation learning
- Convincing samples at low resolutions, too slow for high resolution

Class-conditional pixelCNN 32×32 samples trained on ImageNet [Oord et al., 2016b]

Take home message

- Deep learning forms the basis of state of the art in many domains

Take home message

- Deep learning forms the basis of state of the art in many domains
  - Speech recognition, image understanding, machine translation, advertising, remote sensing, 3D shape processing, finance, medical imaging, autonomous driving, ...

Take home message

- Deep learning forms the basis of state of the art in many domains
  - Speech recognition, image understanding, machine translation, advertising, remote sensing, 3D shape processing, finance, medical imaging, autonomous driving, ...
- Supervised deep learning flourishes with more data and compute

Take home message

- Deep learning forms the basis of state of the art in many domains
  - Speech recognition, image understanding, machine translation, advertising, remote sensing, 3D shape processing, finance, medical imaging, autonomous driving, . . .
- Supervised deep learning flourishes with more data and compute
  - Lots of work models that are efficient in memory, compute, and energy once trained (models probably running on your phone...)

Take home message

- Deep learning forms the basis of state of the art in many domains
  - Speech recognition, image understanding, machine translation, advertising, remote sensing, 3D shape processing, finance, medical imaging, autonomous driving, . . .
- Supervised deep learning flourishes with more data and compute
  - Lots of work models that are efficient in memory, compute, and energy once trained (models probably running on your phone...)
- Deep learning brought unprecedented progress in generative models
  - No need for labeled training data!

Take home message

- Deep learning forms the basis of state of the art in many domains
  - Speech recognition, image understanding, machine translation, advertising, remote sensing, 3D shape processing, finance, medical imaging, autonomous driving, ...
- Supervised deep learning flourishes with more data and compute
  - Lots of work models that are efficient in memory, compute, and energy once trained (models probably running on your phone...)
- Deep learning brought unprecedented progress in generative models
  - No need for labeled training data!
  - Semi-supervised learning, prediction of missing data, ...

Take home message

- Deep learning forms the basis of state of the art in many domains
  - Speech recognition, image understanding, machine translation, advertising, remote sensing, 3D shape processing, finance, medical imaging, autonomous driving, ...
- Supervised deep learning flourishes with more data and compute
  - Lots of work models that are efficient in memory, compute, and energy once trained (models probably running on your phone...)
- Deep learning brought unprecedented progress in generative models
  - No need for labeled training data!
  - Semi-supervised learning, prediction of missing data, ...
  - Generation of realistic (and varied) samples of speech, images, ...

Thanks for your attention!

Jakob Verbeek
INRIA, Grenoble, France

jakob.verbeek@inria.fr

References i

Baldi, P. and Hornik, K. (1989).
Neural networks and principal component analysis: Learning from examples without local minima.
Neural Networks.

Bishop, C. (2006).
Pattern recognition and machine learning.
Spinger-Verlag.

Chatfield, K., Lempitsky, V., Vedaldi, A., and Zisserman, A. (2011).
The devil is in the details: an evaluation of recent feature encoding methods.
In *BMVC*.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017).
Density estimation using real NVP.
In *ICLR*.

Doersch, C. (2016).
Tutorial on variational autoencoders.
arXiv:1606.05908.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014).
Generative adversarial nets.
In *NeurIPS*.

References ii

Hou, X., Shen, L., Sun, K., and Qiu, G. (2017).
Deep feature consistent variational autoencoder.
In *WACV*, volume abs/1610.00291.

Kalchbrenner, N., van den Oord, A., Simonyan, K., Danihelka, I., Vinyals, O., Graves, A., and Kavukcuoglu, K. (2017).
Video pixel networks.
In *ICML*.

Karras, T., Aila, T., and abd J. Lehtinen, S. L. (2018).
Progressive growing of GANSs for improved quality, stability, and variation.
In *ICLR*.

Kingma, D. and Dhariwal, P. (2018).
Glow: Generative flow with invertible 1x1 convolutions.
In *NeurIPS*.

Kingma, D. and Welling, M. (2014).
Auto-encoding variational Bayes.
In *ICLR*.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017).
Feature pyramid networks for object detection.
In *CVPR*.

Lucas, T., Shmelkov, K., Alahari, K., Schmid, C., and Verbeek, J. (2019).
Adaptive density estimation for generative models.
In *NeurIPS*.

Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. (2016a).
Pixel recurrent neural networks.
In *ICML*.

Oord, A. v. d., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. (2016b).
Conditional image generation with PixelCNN decoders.
In *NeurIPS*.

Radford, A., Metz, L., and Chintala, S. (2016).
Unsupervised representation learning with deep convolutional generative adversarial networks.
In *ICLR*.

Reed, S., van den Oord, A., Kalchbrenner, N., Colmenarejo, S. G., Wang, Z., Belov, D., and de Freitas, N. (2017).
Parallel multiscale autoregressive density estimation.
In *ICML*.

References iv

Roweis, S. (1997).
EM Algorithms for PCA and SPCA.
In *NeurIPS*.

Royer, A., Kolesnikov, A., and Lampert, C. (2017).
Probabilistic image colorization.
In *BMVC*.

Tipping, M. E. and Bishop, C. M. (1999).
Mixtures of probabilistic principal component analysers.
Neural Computation, 11(2):443–482.