Machine Learning:
Convolutional Neural Networks and Autoencoders

PRINCETON
UNIVERSITY

Computer Vision

A huge subfield of deep learning dealing with image classification, object detection, segmentation, tracking, depth estimation, 3d reconstruction etc.

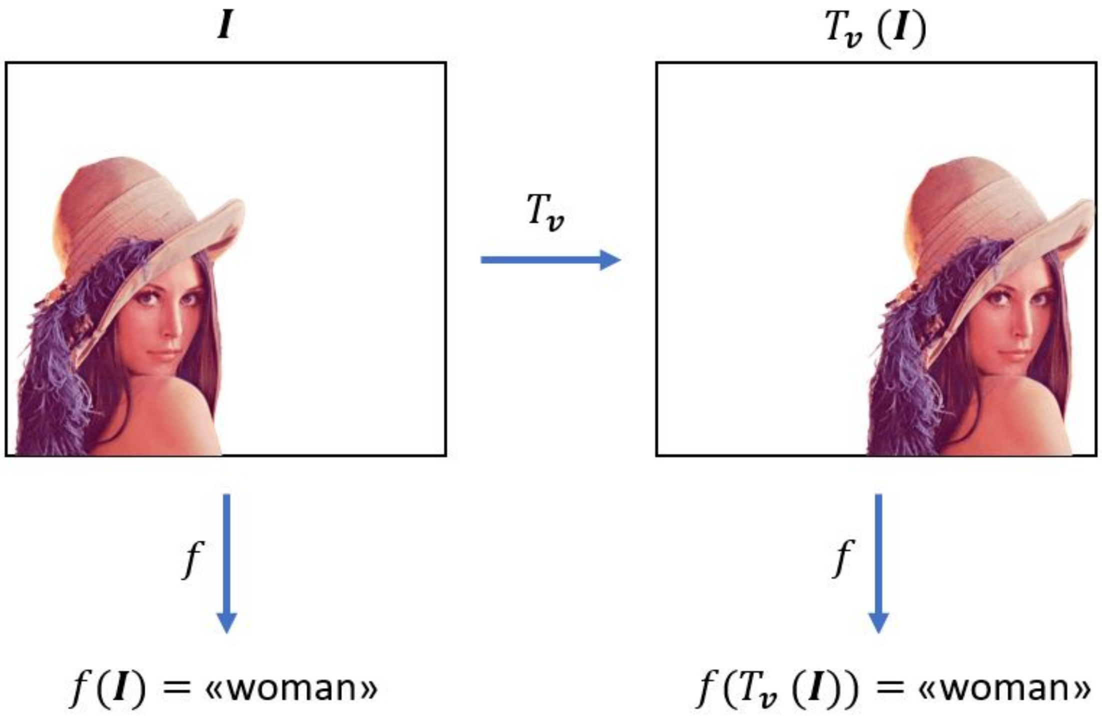
Challenges:

- Large dimensionality of the input, e.g. HD image has close to 1M pixels.
- Results must be invariant to translation, rotation, illumination etc.
- Images can contain several objects from multiple categories or multiple instances from one.

Convolutional Neural Networks (CNNs)

- We can boost the performance of neural networks by including inductive bias.
- When working with images, the model should be translation invariant, i.e. a representation meaningful at a certain location can / should be used everywhere.
- CNN idea: **sliding window**, i.e. slide a filter across the input to check for a specific pattern, i.e. activations will be high.
- Hand-engineered filters are difficult to define - use backprop to learn them.

Equivariance to Translation



Convolution Operation

- The result of a convolution is now equivalent to the dot product between every filter and every **receptive field** location (portion of the image).

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input

1	0	1
0	1	0
1	0	1

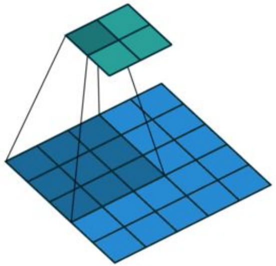
Filter / Kernel

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

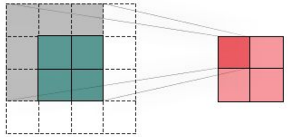
4		

Convolution Operation: Stride and Padding

- Stride: by how many pixels to move the filter at each step.

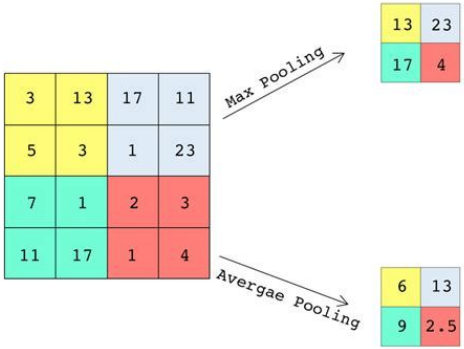


- Padding: add extra pixels around the input - performance near the edges.



Pooling Operation

- Pooling operation reduces the spatial size of the representations.
- Two types of operations: average or max.



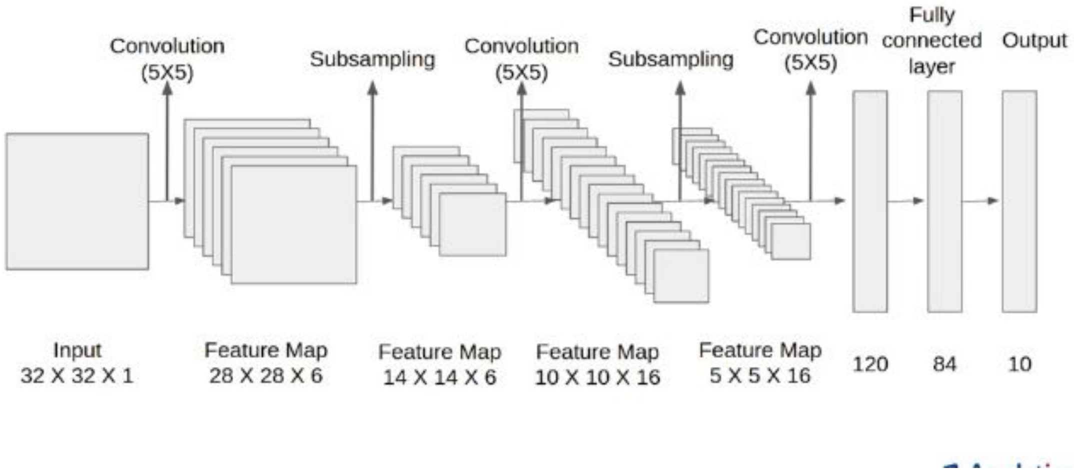
- Lowers computational load of training and inference.

Parameter Sharing

- Parameters are shared by each neuron producing in the activation map.
- This reduces number of weights: **parameter sharing**.
- Example:
  - Input: 256×256×3 (RGB image) and kernel: 3x3x3: 27 weights.
  - Equivalent for a dense network: $O(10^{10})$ weights.

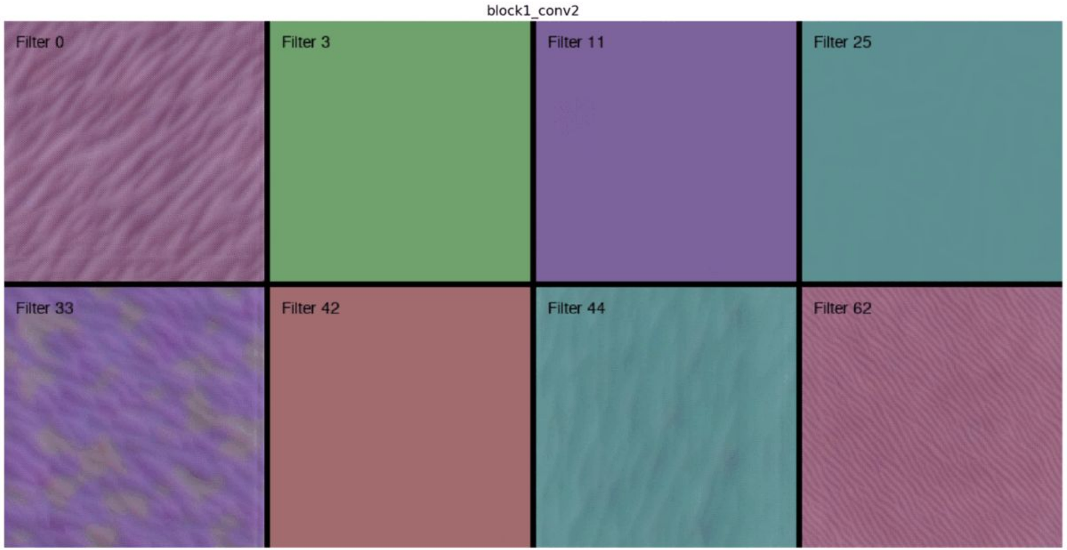
CNN Example

- Full architecture of a CNN is a mix of convolutional and pooling layers.



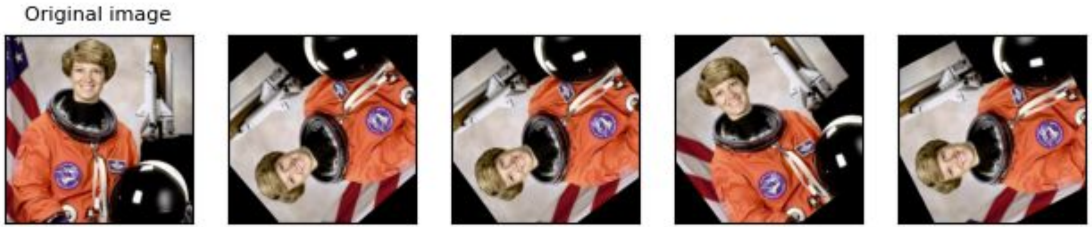
What CNNs Learn?

- Each filter picks up different aspects of the image, e.g. edges, complex patterns
- Each layer becomes more and more expressive.



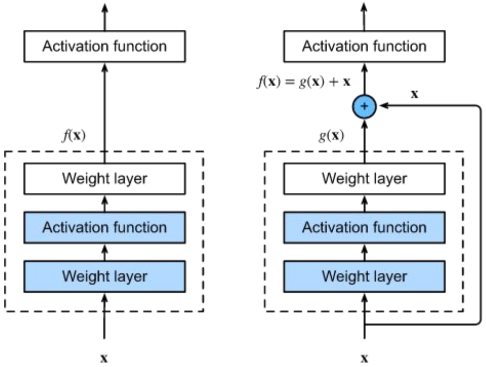
Data Augmentation

- The performance of CNNs improves with more data.
- If we can’t collect more data, we can artificially create new variants of existing **training** data with augmentations.
- Augmentations include operations such as rotations, shifts, flips, zooms, contrast, hue adjustments etc.
- Always consider domain-specific constraints.



ResNets: Residual Connections

- Deeper models have higher training errors due to vanishing gradient.
- ResNets introduced **residual connections** (also s**kip connections**):
    - Instead of hoping that each stacked layer directly learn the underlying mapping f(x) let them learn the residual mapping $g(\mathbf{x}) = f(\mathbf{x})\text{-}\mathbf{x}$
    - Intuition: if an identity mapping were optimal, it would be easier to push theresidual to zero than to fit an identity mapping by a stack of non-linear layers



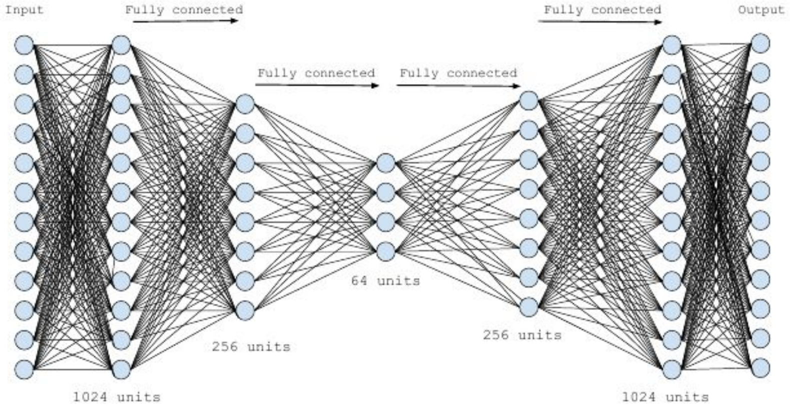
Unsupervised Learning

- Labels might be imperfect, have high cost to acquire or even don't exist.
- Very often you don't need the labels, e.g. anomaly detection, denoising, compression, clustering.
- Dimensionality reduction transforms original data from a high-dimensional space into a low-dimensional space ideally retaining meaningful properties of the original data.
- In dimensionality reduction we could:
  - compress the data to a latent space with smaller number of dimensions, *and*
  - recover the original data from this latent space representation.
- So the latent space must encode and retain the important information about the data.
- Dimensionality reduction with deep learning: autoencoders.

Autoencoders

- A deep autoencoder is composed of two neural networks:
    - encoder E that takes an input and maps it to a usually low-dimensional representation
    - decoder D that tries to reconstruct the original input from the representation vector:

$$\hat{x} = D(E(x)) \text{ where } \hat{x} \sim x.$$



Autoencoders

- A deep autoencoder is composed of two neural networks:
  - encoder E that takes an input and maps it to a usually low-dimensional representation
  - decoder D that tries to reconstruct the original input from the representation vector:

$$\hat{x} = D(E(x)) \text{ where } \hat{x} \sim x.$$

- Trained to perform an approximate identity mapping between their input and output layers.
- Trained with **reconstruction loss**: distance between $\hat{x}$ and x, e.g. mean squared error.
- The reconstruction criterion is insufficient for learning useful representation: often the capacity must be regularized by reduced hidden dimensionality, i.e. *undercomplete autoencoders*.

Autoencoders: Applications

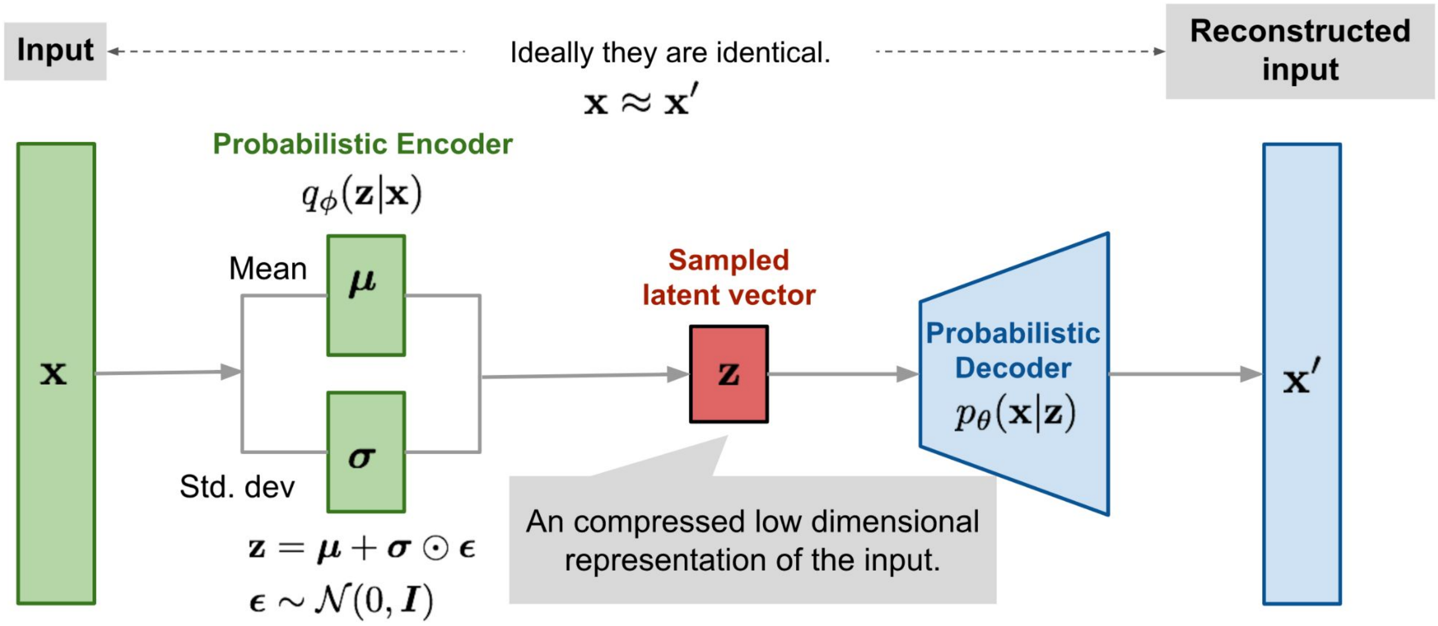
- Input denoising.

- Data compression, non-linear dimensionality reduction.
- Anomaly detection: train on *inlier* class.

Autoencoders: Generative Deep Learning

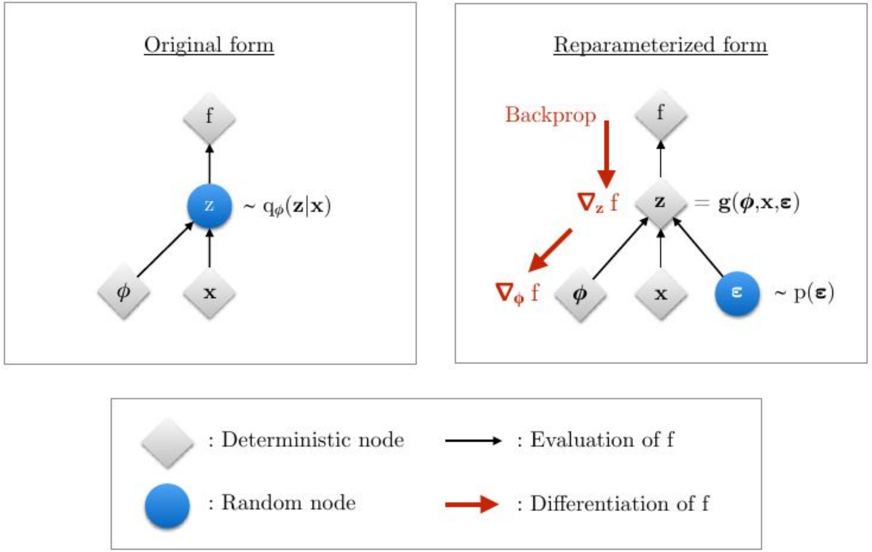
- Can we use autoencoders to learn $p(x)$ and generate new data?
- Sample a latent representation z from $p(z)$ and generate x using conditional distribution $p(x|z)$.
    - Use decoder for $p(x|z)$: describes the distribution of the decoded variable given the encoded one.
    - Use encoder for $q(z|x)$: describes the distribution of the encoded variable given the decoded one.
- We cannot use vanilla autoencoders for this because the latent space can't be interpolated.
- To learn the distribution we construct two neural networks p_θ and q_ϕ, such that:
    - $\mu_\theta(x)$, $\sigma_\theta(x) = q_\phi(x)$,
    - $x' = \mathrm{p}_\theta(z)$, where $z \sim N(\mu_\theta(x), \sigma_\theta(x))$.
- Jointly trained p_θ and q_ϕ is called a Variational Autoencoder (VAE)

Variational Autoencoders



Variational Autoencoders: Reparameterization Trick

- We train VAEs as any other neural network: using backpropagation.
- However, we cannot backpropagate through stochastic node.
- Reparameterization trick: externalize the randomness in *z* by introduction a new random variable $\mathcal{E}$.



Variational Autoencoders: Loss

- Kullback-Leibler (KL) divergence measures distance between two distributions:

$$D_{KL}(p(x)||q(x)) = \sum_{x \in X} p(x) \ln \frac{p(x)}{q(x)}$$

- KL divergence can be used as a regularization so that the posterior distribution tries to match the prior (in our case the Gaussian).
- The full learning objective of the VAE is then: $\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) = \log p_{\boldsymbol{\theta}}(\mathbf{x}) - D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))$ also called evidence lower bound (ELBO).
- The KL loss can be computed analytically: DKL = 0.5 $\sum$ [1 + $\log(\sigma_\theta^2(x)) - \mu_\theta^2(x) - \sigma_\theta^2(x)$]

Variational Autoencoders: Example

Hands On Session