UPPSALA
UNIVERSITET

Machine learning – Block 4

Måns Magnusson
Department of Statistics, Uppsala University

Autumn 2024

This week's lecture

- Previous assignment
- Convolutional Neural Networks
- Transfer Learning

Assignment 3: Evaluation

- A little bit to easy VG part
- Hassle to install the software
- Some unclarities

On this weeks assignment

- It takes long time to run the models this week. Start early!

Section 2

Introduction

Convolutional Neural Networks

- Acknowledgements: Anders Eklund, Linköping University.

Convolutional Neural Networks

- Acknowledgements: Anders Eklund, Linköping University.
- Convolutional Neural Networks are behind great progress in the 2010s.
- Revolutionized Computer Vision.

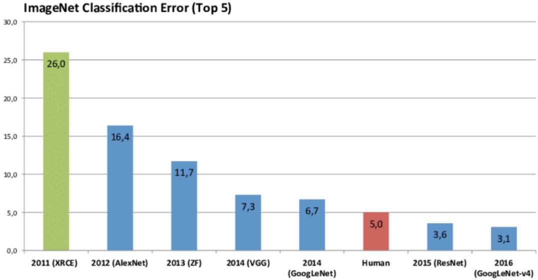
UPPSALA
UNIVERSITET

- Practicalities
- Introduction
- Convolution
- Convolutional Neural Networks
  - The Convolution Layer
  - The Pooling Layer
  - Regularization
  - Examples
- Transfer learning
- Practical Methodology

Convolutional Neural Networks

- Acknowledgements: Anders Eklund, Linköping University.
- Convolutional Neural Networks are behind great progress in the 2010s.
- Revolutionized Computer Vision.
- Also called: ConvNets, Convolutional nets, Convolutional networks

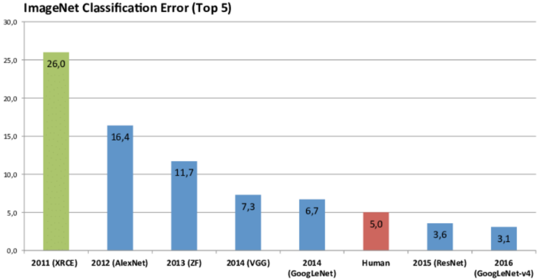
Figure: ImageNet performance (Roessler, 2019)



Convolutional Neural Networks

- Acknowledgements: Anders Eklund, Linköping University.
- Convolutional Neural Networks are behind great progress in the 2010s.
- Revolutionized Computer Vision.
- Also called: ConvNets, Convolutional nets, Convolutional networks

Figure: ImageNet performance (Roessler, 2019)



Convolutional Neural Networks

- Special architecture that works well for data with a grid structure
  1. 1D-grids: Time series

Convolutional Neural Networks

- Special architecture that works well for data with a grid structure
  1. 1D-grids: Time series
  2. 2D-grids: Gray-scale Images (pixels)

Convolutional Neural Networks

- Special architecture that works well for data with a grid structure
  1. 1D-grids: Time series
  2. 2D-grids: Gray-scale Images (pixels)
  3. 3D-grids: Color Images (pixels and chanels)

Convolutional Neural Networks

- Special architecture that works well for data with a grid structure
  1. 1D-grids: Time series
  2. 2D-grids: Gray-scale Images (pixels)
  3. 3D-grids: Color Images (pixels and chanels)
  4. 4D-grids: Color Video (pixels, chanels, frames)

Computer Vision

- Problems
  - Image Classification

UPPSALA UNIVERSITET

Computer Vision

- Problems
  - Image Classification
  - Image Segmentation

Computer Vision

- Problems
  - Image Classification
  - Image Segmentation
  - Object Detection
  - ...

UPPSALA UNIVERSITET

Computer Vision

- Problems
  - Image Classification
  - Image Segmentation
  - Object Detection
  - ...
- Focus: 2D and 3D data

UPPSALA UNIVERSITET

Computer Vision

- Problems
  - Image Classification
  - Image Segmentation
  - Object Detection
  - ...
- Focus: 2D and 3D data
- Very Large Datasets:
  - ImageNet: 14M Images, 20k classes, 1M bounding boxes

Computer Vision

UPPSALA UNIVERSITET

- Problems
  - Image Classification
  - Image Segmentation
  - Object Detection
  - ...
- Focus: 2D and 3D data
- Very Large Datasets:
  - ImageNet: 14M Images, 20k classes, 1M bounding boxes
- Many different pre-trained models (e.g. VGG16)

Example: Object Detection

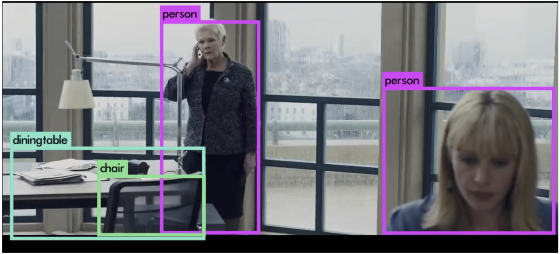


Figure: Object detection (see https://www.youtube.com/watch?v=VOC3huqHrss)

Example: Pneumonia detection

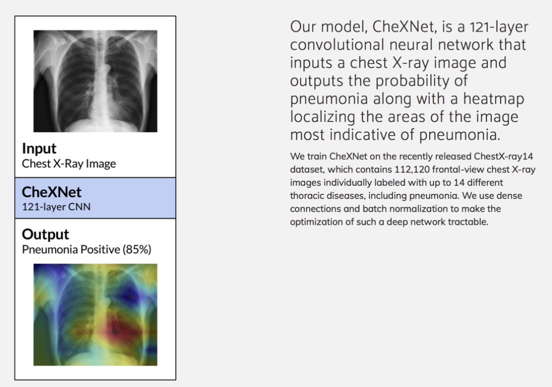


Figure: Rajpurkar et al. (2017). Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. arXiv preprint arXiv:1711.05225.

Example: Fracture detection

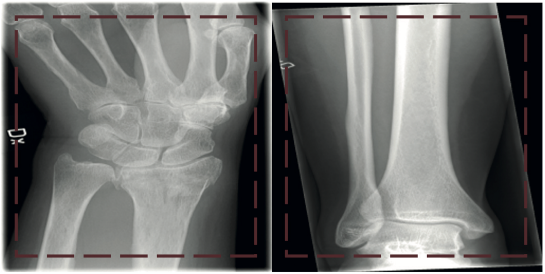


Figure 1. 2 images from the dataset. The area within the red box is the section presented to the network in order to classify the image. The left image is of a wrist fracture while the right image is without any apparent fracture.

Figure: Olczak et al, (2017) Artificial intelligence for analyzing orthopedic trauma radiographs, Acta Orthopaedica, 88:6, 581-586

Example: Image Segmentation

Figure: Bishop and Bishop (2024) Fig 10.26

What is an Image?

- 2-dimensional object
- Each pixel has:
  1. a coordinate
  2. a value (light intensity)

What is an Image?

- 2-dimensional object
- Each pixel has:
  1. a coordinate
  2. a value (light intensity)
- Grayscale: single channel
- Color: three channel (RGB)

What is an Image?

- 2-dimensional object
- Each pixel has:
  1. a coordinate
  2. a value (light intensity)
- Grayscale: single channel
- Color: three channel (RGB)
- Spatial and hiearchical correlation structures

UPPSALA UNIVERSITET

MNIST example

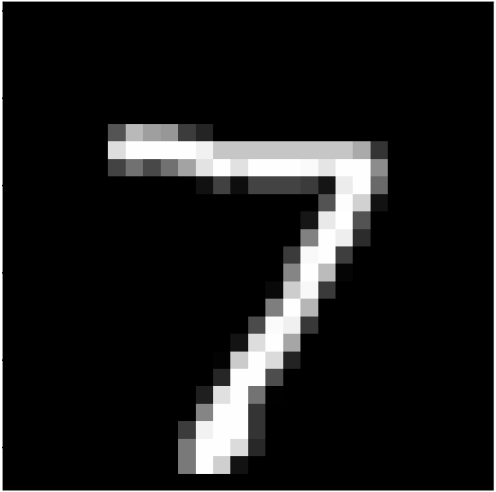


Figure: Example from the MNIST dataset (28 by 28 pixels)

How to train models for images?

- We want to learn representations of parts of images

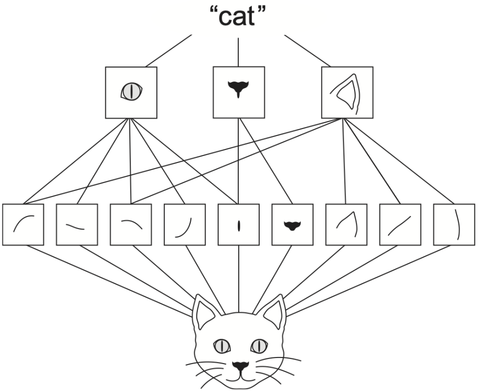


Figure: The representations of a cat (Chollet and Allair, 2018, Fig 5.2)

How to train models for images?

- We want to learn representations of parts of images

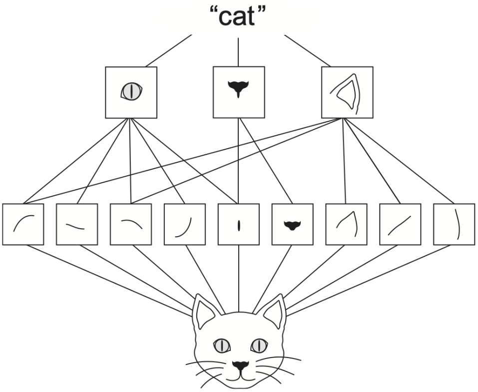


Figure: The representations of a cat (Chollet and Allair, 2018, Fig 5.2)

- CNN uses Convolutional Layers to learn parameter efficient representations

Learning Representations for Images (again)

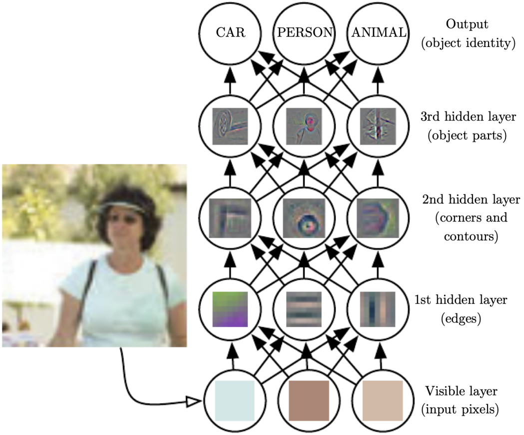


Figure: Learning representations for images (Goodfellow et al, 2017, Fig. 1.2)

Section 3

Convolution

Convolution

- Different definitions are common, one example:

$$y(t) = \int x(\tau)k(t-\tau)d\tau = (x * k)(t)$$

- Inutuition: "Weighting together two functions"
- In a convolutional layer:
  1. $x(t)$: Input
  2. $k(t)$: Kernel, filter, "feature"
  3. $y(t)$: Output, feature map

DEMO

Discrete Convolution

- If t is discrete (as in a grid):

$$y(t) = (x * k)(t) = \sum_{\tau=-\infty}^{\infty} x(\tau)k(t-\tau)$$

UPPSALA UNIVERSITET

Discrete Convolution

- If t is discrete (as in a grid):

$$y(t) = (x * k)(t) = \sum_{\tau=-\infty}^{\infty} x(\tau)k(t-\tau)$$

- In the case of images we have 2 discrete dimensions

$$Y(i,j) = (X * K)\sum_{m}\sum_{n} X(m,n)K(i-m,j-n)$$

Discrete Convolution

- If t is discrete (as in a grid):

$$y(t) = (x * k)(t) = \sum_{\tau=-\infty}^{\infty} x(\tau)k(t-\tau)$$

- In the case of images we have 2 discrete dimensions

$$Y(i,j) = (X * K) \sum_{m} \sum_{n} X(m,n)K(i-m, j-n)$$

- Sometimes the cross-correlation is called convolution:

$$Y(i,j) = (X * K) \sum_{m} \sum_{n} X(m,n)K(i+m, j+n)$$

1. $X(i,j)$: Input (2D)
2. $K(i,j)$: Kernel, filter, "feature" (2D)
3. $Y(i,j)$: Output, feature map (2D)

Convolution of Images: 2D

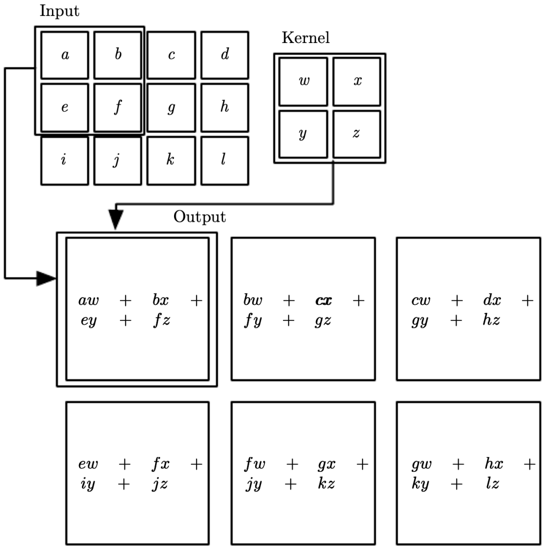


Figure: Convolution for an Image (Goodfellow et al, 2017, Fig. 9.1)

UPPSALA UNIVERSITET

Convolution of images: Example

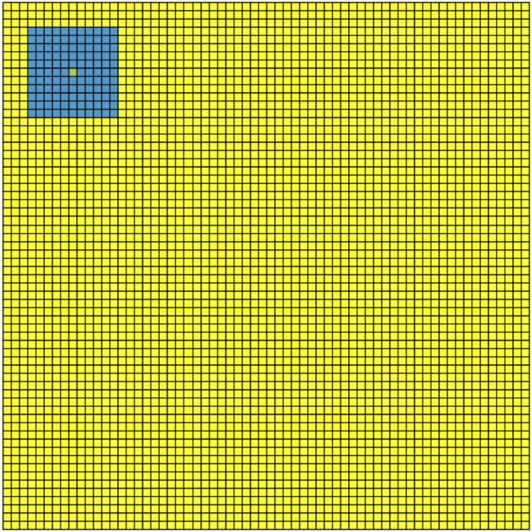


Figure: Convolution example.

Convolution of images: Examples

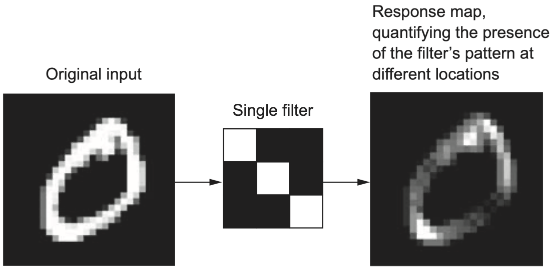


Figure: Convolution for an Image (Chollet and Allaire, 2018, Fig. 5.3)

Convolution of images: Example

$$X = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}, K = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$$

Convolution of images: Example

$$X = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}, K = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$$

$$Y = \begin{bmatrix} 0 & 0 & 2 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 1 & 2 & 0 \end{bmatrix},$$

Edge Detection

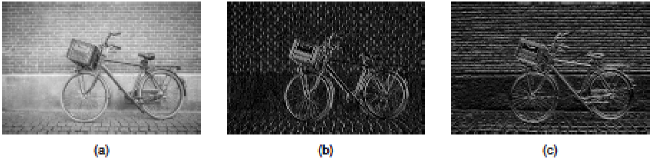


Figure: Edge Detections (Bishop and Bishop, 2024, Fig. 10.4)

Examples of filters

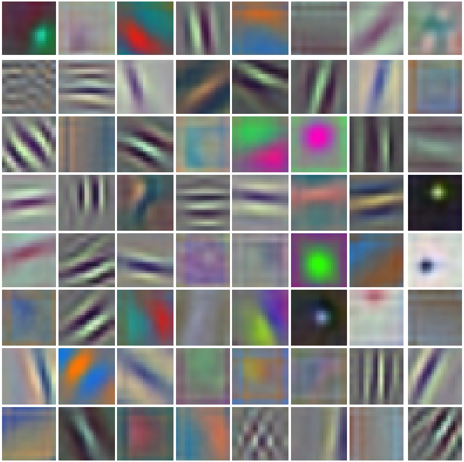


Figure: AlexNet filters (Bishop and Bishop, 2024, Fig. 10.12)

Section 4

Convolutional Neural Networks

Convolutional Neural Networks

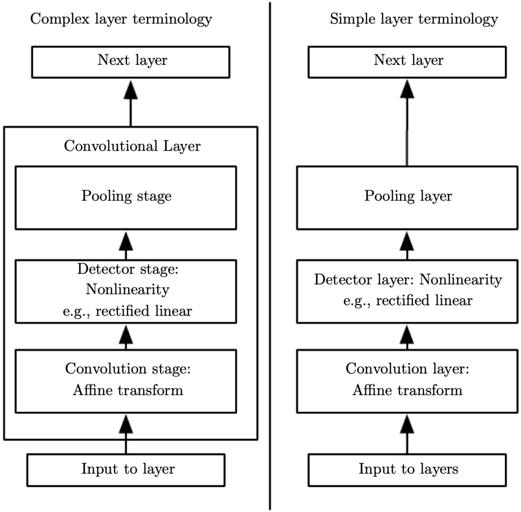


Figure: Convolution layer (Goodfellow et al, 2018, Fig. 9.7)

UPPSALA UNIVERSITET

Convolutional Neural Networks

- Most convolutional neural networks have:
  1. Many convolutional layers
  2. More kernels higher up in the network
  3. A classification head (usually a feed-forward neural network)

Convolutional Neural Networks

- Most convolutional neural networks have:
  1. Many convolutional layers
  2. More kernels higher up in the network
  3. A classification head (usually a feed-forward neural network)
- Benefits:
  1. Few(er) parameters (filters)
  2. Captures local structures
  3. Efficient computations

Convolutional Neural Networks

- Most convolutional neural networks have:
  1. Many convolutional layers
  2. More kernels higher up in the network
  3. A classification head (usually a feed-forward neural network)
- Benefits:
  1. Few(er) parameters (filters)
  2. Captures local structures
  3. Efficient computations
- How to choose filters?
  1. Before: manually handcrafted

Convolutional Neural Networks

- Most convolutional neural networks have:
  1. Many convolutional layers
  2. More kernels higher up in the network
  3. A classification head (usually a feed-forward neural network)
- Benefits:
  1. Few(er) parameters (filters)
  2. Captures local structures
  3. Efficient computations
- How to choose filters?
  1. Before: manually handcrafted
  2. Now: learn the filters

Convolution layer

- Input: Data or Feature Maps

UPPSALA UNIVERSITET

Convolution layer

- Input: Data or Feature Maps
- Parameters:
  - N filters/kernels of size $m \times m$
  - N bias terms (one per filter)

Convolution layer

- Input: Data or Feature Maps
- Parameters:
  - N filters/kernels of size $m \times m$
  - N bias terms (one per filter)
- Activation functions: Applied element wise on feature maps

Convolution layer

- Input: Data or Feature Maps
- Parameters:
  - N filters/kernels of size $m \times m$
  - N bias terms (one per filter)
- Activation functions: Applied element wise on feature maps
- Output: Feature Maps

UPPSALA UNIVERSITET

Convolution layer

- Input: Data or Feature Maps
- Parameters:
  - N filters/kernels of size $m \times m$
  - N bias terms (one per filter)
- Activation functions: Applied element wise on feature maps
- Output: Feature Maps
- In Keras:

```
layer_conv_2d(filters = 32, kernel_size =
c(3,3), activation = "relu", input_shape =
c(32,32,3))
```

Padding

- Handling edges
- *Padding*: add 0 around the image
- Nessecary to keep size of feature maps

Padding

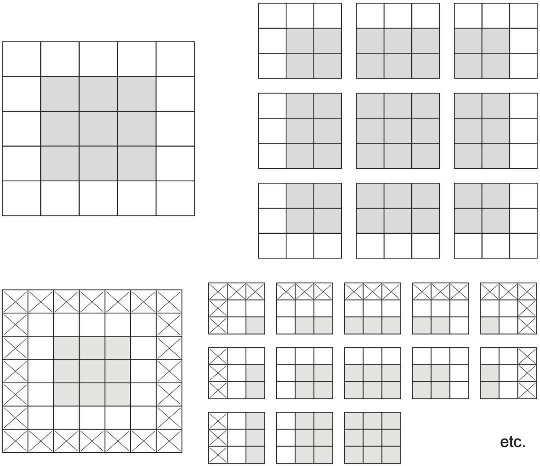


Figure: Padding and valid edge handling (Chollet and Allair (2018), Fig. 5.5, 5.6)

Stride

- Skip every nth pixel
- Reduces the computations

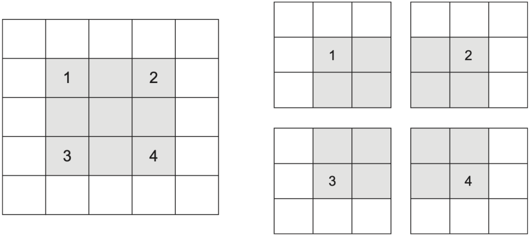


Figure: Strides (Chollet and Allair (2018), Fig. 5.5, 5.6)

Why Convolution Layers?

- Captures local spatial structure
- Reduces the number of parameters (parameter sharing)
  1. The number and size of filters
  2. We use the same filters everywhere

Why Convolution Layers?

- Captures local spatial structure
- Reduces the number of parameters (parameter sharing)
  1. The number and size of filters
  2. We use the same filters everywhere
- Example: a 1 megapixel image (1000×1000 pixels)
  1. Dense network with 100 nodes: 100M parameters
  2. CNN network with 100 3×3 filters: 1000 parameters (900 from filters, 100 bias terms)

Convolution Neural Nets

UPPSALA UNIVERSITET

- Practicalities
- Introduction
- Convolution
- Convolutional Neural Networks
  - The Convolution Layer
  - The Pooling Layer
  - Regularization
  - Examples
- Transfer learning
- Practical Methodology

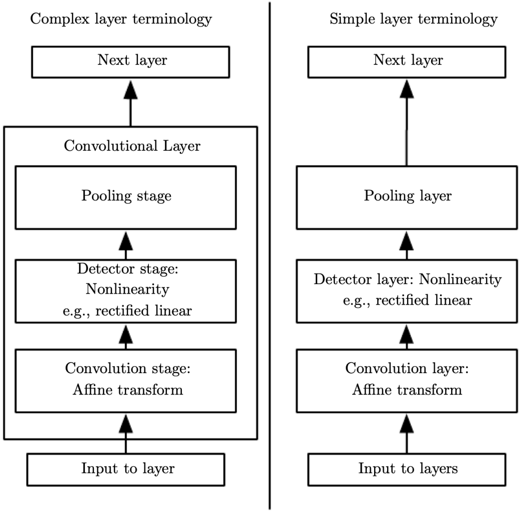


Figure: Convolution layer (Goodfellow et al, 2018, Fig. 9.7)

Detector stage

- Remember, in feed-forward networks: $\mathbf{h} = \sigma(\mathbf{XW} + b)$
- In CNN:
  1. $\mathbf{W}$ is the filter

Detector stage

- Remember, in feed-forward networks: $\mathbf{h} = \sigma(\mathbf{XW} + b)$
- In CNN:
  1. $\mathbf{W}$ is the filter
  2. $\mathbf{X}$ is the input feature map

Detector stage

- Remember, in feed-forward networks: $\mathbf{h} = \sigma(\mathbf{XW} + b)$
- In CNN:
  1. $\mathbf{W}$ is the filter
  2. $\mathbf{X}$ is the input feature map
  3. $\mathbf{XW}$ is the convolutional feature map

Detector stage

- Remember, in feed-forward networks: $\mathbf{h} = \sigma(\mathbf{XW} + b)$
- In CNN:
  1. $\mathbf{W}$ is the filter
  2. $\mathbf{X}$ is the input feature map
  3. $\mathbf{XW}$ is the convolutional feature map
  4. b is a bias (one per filter)

UPPSALA UNIVERSITET

Detector stage

- Remember, in feed-forward networks: $\mathbf{h} = \sigma(\mathbf{XW} + b)$
- In CNN:
  1. $\mathbf{W}$ is the filter
  2. $\mathbf{X}$ is the input feature map
  3. $\mathbf{XW}$ is the convolutional feature map
  4. b is a bias (one per filter)
  5. σ is the activation function (usually a ReLU)

Pooling layer

- We take a function f that return one value per pooling kernel

Pooling layer

- We take a function f that return one value per pooling kernel
- Most commonly $f = \max$
- Commonly a 2×2 pooling kernel with stride 2
- Why? Reduce the size of feature map, but keep the activation

Pooling layer

- We take a function f that return one value per pooling kernel
- Most commonly $f = \max$
- Commonly a 2×2 pooling kernel with stride 2
- Why? Reduce the size of feature map, but keep the activation
- In Keras: `layer_max_pooling_2d(pool_size = c(2, 2))(`

Max Pooling

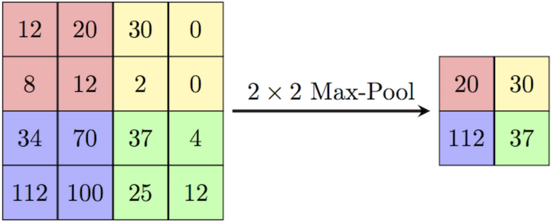


Figure: Strides (Computer Science Wikipedia)

Using pooling to learn invariances

- pooling over spatial positions: invariant to translation
- pooling over different filters: invariant to transformations

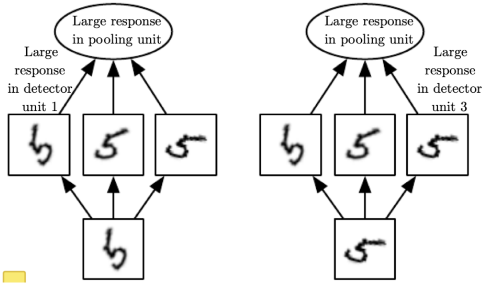


Figure: Learning invariances (Goodfellow et al., 2017, Fig. 9.9)

UPPSALA UNIVERSITET

Data Augmentation

Figure: Data Augmentation (Chollet and Allair, 2018, Fig 5.10)

Data Augmentation

Figure: Data Augmentation (Chollet and Allair, 2018, Fig 5.10)

- Can be done directly in Keras (data generator)

Popular CNN architectures

- AlexNet (2012), 5 convolutional layers
- VGG16 (2014), 16 convolutional layers
- ResNet (2015), 152 convolutional layers

VGG16

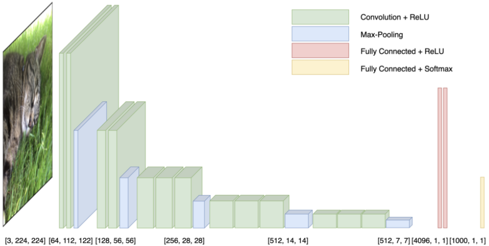


Figure 2.4: *Illustration of the data flow through the network VGG16. Data size, of the format* $[c, h, w]$*, is shown for the input image, output of each max-pooling layer, output after the first two fully connected layers, and the final network output.*

Jesper Westell, Multi-Task Learning using Road Surface Condition Classification and Road Scene Semantic Segmentation, LIU-IMT-TFK-A–19/570–SE

Section 5

Transfer learning

Transfer learning

- "Transfer knowledge between problems"
- Learning representations in P_1 will aid generalization in P_2

Transfer learning

- "Transfer knowledge between problems"
- Learning representations in P_1 will aid generalization in P_2
- A Bayesian perspective: A strong prior

Learning Representations for Images

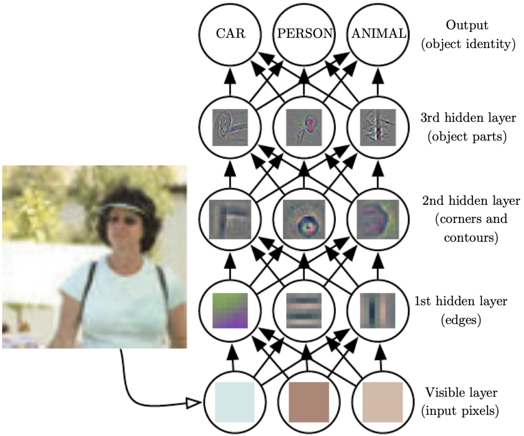


Figure: Learning representations can be crucial (Goodfellow et al, 2017, Fig. 1.2)

Transfer learning

- In practice: Transfer/reuse learned weights or rather some weights

Transfer learning

- In practice: Transfer/reuse learned weights or rather some weights
- Use (large) pre-trained models for smaller problems
- A reason for the success of CNN

Transfer learning

- In practice: Transfer/reuse learned weights or rather some weights
- Use (large) pre-trained models for smaller problems
- A reason for the success of CNN
- Two types of transfer learning in Neural Networks:
  - Feature extraction (use pre-trained networks for features)
  - Fine Tuning (adapt pre-trained features)

UPPSALA UNIVERSITET

Feature Extraction

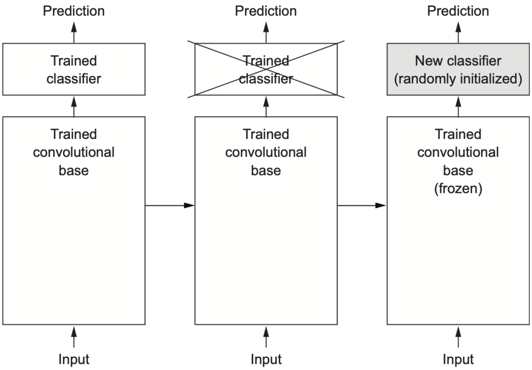


Figure: Using convnets as base for feature extraction (Chollet and Allair, 2018, Fig 5.12)

Fine-Tuning

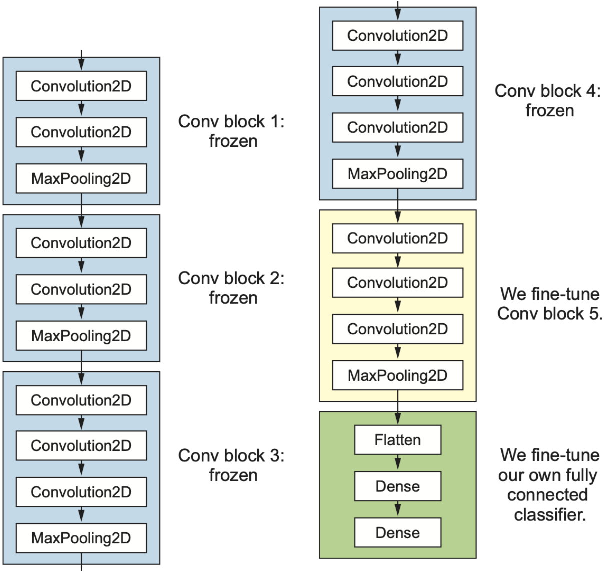


Figure: Finetuning a convolutional base (Chollet and Allair, 2018, Fig 5.15)

Transfer learning

- Catastophic forgetting

Transfer learning

- Catastophic forgetting
- Domain adaptation: Same problem but at different input dataset
  (e.g. language models for legal/medical/political data)

Transfer learning

- Catastophic forgetting
- Domain adaptation: Same problem but at different input dataset
  (e.g. language models for legal/medical/political data)
- Concept drift: Similar problem

Transfer learning

- Catastophic forgetting
- Domain adaptation: Same problem but at different input dataset
  (e.g. language models for legal/medical/political data)
- Concept drift: Similar problem
- Previously, popular with unsupervised pre-training.

Section 6

Practical Methodology

Practical Methodology

1. Determine your goals

Practical Methodology

1. Determine your goals
2. Setup your baseline
   (establish a working end-to-end pipeline as soon as possible)

UPPSALA UNIVERSITET

Practical Methodology

1. Determine your goals
2. Setup your baseline
   (establish a working end-to-end pipeline as soon as possible)
3. Diagnose your networks performance

Practical Methodology

1. Determine your goals
2. Setup your baseline
   (establish a working end-to-end pipeline as soon as possible)
3. Diagnose your networks performance
4. Make incremental improvements

- General idea: Increase data and model capacity until goal is reached

Practical Methodology

1. Determine your goals
2. Setup your baseline
   (establish a working end-to-end pipeline as soon as possible)
3. Diagnose your networks performance
4. Make incremental improvements

- General idea: Increase data and model capacity until goal is reached
- End goal: Good enough performance on test set

UPPSALA UNIVERSITET

1. Determine your goals

- Why are you building a model?

1. Determine your goals

- Why are you building a model?
- Set up the metric based on the overall goal of the system! This may need multiple metrics.

1. Determine your goals

- Why are you building a model?
- Set up the metric based on the overall goal of the system! This may need multiple metrics.
- What is good enough? Remember the Bayes error!

1. Determine your goals

- Why are you building a model?
- Set up the metric based on the overall goal of the system! This may need multiple metrics.
- What is good enough? Remember the Bayes error!
- What performance can you expect?

1. Determine your goals

- Why are you building a model?
- Set up the metric based on the overall goal of the system! This may need multiple metrics.
- What is good enough? Remember the Bayes error!
- What performance can you expect?
- Some errors are worse than others, e.g. spam filters.

1. Determine your goals

- Why are you building a model?
- Set up the metric based on the overall goal of the system! This may need multiple metrics.
- What is good enough? Remember the Bayes error!
- What performance can you expect?
- Some errors are worse than others, e.g. spam filters.
- Handling of uncertain predictions:
- Coverage: How large proportions can the system predict?

1. Determine your goals

- Why are you building a model?
- Set up the metric based on the overall goal of the system! This may need multiple metrics.
- What is good enough? Remember the Bayes error!
- What performance can you expect?
- Some errors are worse than others, e.g. spam filters.
- Handling of uncertain predictions:
- Coverage: How large proportions can the system predict?
- Manual curation can be faster and easier.

2. Setup your baseline

- Start with...
  - the most simple possible model (logistic regression)

2. Setup your baseline

- Start with...
  - the most simple possible model (logistic regression)
  - previous approaches/baselines

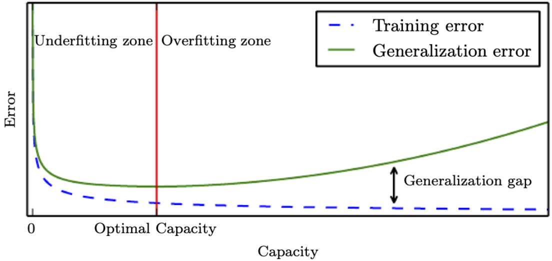
UPPSALA UNIVERSITET

2. Setup your baseline

- Start with...
  - the most simple possible model (logistic regression)
  - previous approaches/baselines
  - a simple neural network that is common in the domain/defaults
    (CNN for images, Adam as optimizer)

Remember: Statistical Learning Theory

Figure: Test, training, and model complexity (Goodfellow et al, 2017, Figure 5.3)



UPPSALA
UNIVERSITET

3. Diagnose and improve

- High training loss: Training data not fully used
  Neural network generally performs best when training error is low

3. Diagnose and improve

- High training loss: Training data not fully used
  Neural network generally performs best when training error is low
- High test loss: Low data quality, i.e. large Bayes error?

3. Diagnose and improve

- High training loss: Training data not fully used
  Neural network generally performs best when training error is low
- High test loss: Low data quality, i.e. large Bayes error?
- Low training error and high test error: Common situation

3. Diagnose and improve

- High training loss: Training data not fully used
  Neural network generally performs best when training error is low
- High test loss: Low data quality, i.e. large Bayes error?
- Low training error and high test error: Common situation
  - You can always improve by gather more data, or
  - Regularize to optimal capacity

3. Diagnose and improve

- High training loss: Training data not fully used
  Neural network generally performs best when training error is low
- High test loss: Low data quality, i.e. large Bayes error?
- Low training error and high test error: Common situation
  - You can always improve by gather more data, or
  - Regularize to optimal capacity
- How to know the improvements of additional data?

Learning curve

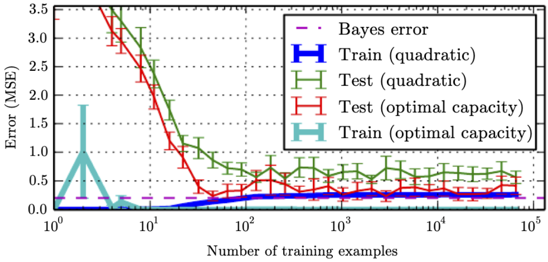


Figure: Learning curve to assess the need for more data (Goodfellow et al., 2017, Fig 5.4)

UPPSALA UNIVERSITET

3. Diagnosing and improving your model: Regularization

- Best performance: Larger model that is regularized well.
- Warning: Avoid the algorithm rabbit hole

3. Diagnosing and improving your model: Regularization

- Best performance: Larger model that is regularized well.
- Warning: Avoid the algorithm rabbit hole
- Adapt hyperparameters to get optimal capacity

3. Diagnosing and improving your model: Regularization

- Best performance: Larger model that is regularized well.
- Warning: Avoid the algorithm rabbit hole
- Adapt hyperparameters to get optimal capacity
- Hyperparameter Search Goal:
  adjust the model capacity to match the complexity of the task.

3. Diagnosing and improving your model: Regularization

- Best performance: Larger model that is regularized well.
- Warning: Avoid the algorithm rabbit hole
- Adapt hyperparameters to get optimal capacity
- Hyperparameter Search Goal:
  adjust the model capacity to match the complexity of the task.
- Marginal hyperparameter has a U-shaped error function (ideally)

3. Diagnosing and improving your model: Regularization

- Best performance: Larger model that is regularized well.
- Warning: Avoid the algorithm rabbit hole
- Adapt hyperparameters to get optimal capacity
- Hyperparameter Search Goal:
  adjust the model capacity to match the complexity of the task.
- Marginal hyperparameter has a U-shaped error function (ideally)
- Neural Networks Steps:
  1. Get good training error. E.g. by tune learning rate and increasing capacity

3. Diagnosing and improving your model: Regularization

- Best performance: Larger model that is regularized well.
- Warning: Avoid the algorithm rabbit hole
- Adapt hyperparameters to get optimal capacity
- Hyperparameter Search Goal:
  adjust the model capacity to match the complexity of the task.
- Marginal hyperparameter has a U-shaped error function (ideally)
- Neural Networks Steps:
  1. Get good training error. E.g. by tune learning rate and increasing capacity
  2. Tune hyperparameters (regularization):
     Requires monitoring both training and test error

Hyper-parameter optimization

- Neural networks has many hyperparameters
- What is a hyperparameter in a neural network?

UPPSALA UNIVERSITET

Hyper-parameter optimization

- Neural networks has many hyperparameters
- What is a hyperparameter in a neural network?
- Grid search:
  - Setup a grid of potential values
  - For 1-4 hyperparameter: grid search can work well
  - Usually iterative/repeated grid search is best: start with three values, and work iteratively
  - Grows exponentially with the number of parameters

Hyper-parameter optimization

- Neural networks has many hyperparameters
- What is a hyperparameter in a neural network?
- Grid search:
  - Setup a grid of potential values
  - For 1-4 hyperparameter: grid search can work well
  - Usually iterative/repeated grid search is best: start with three values, and work iteratively
  - Grows exponentially with the number of parameters
- Random search
  - Specify a marginal distribution for each hyperparameter (additional work)
  - Common: uniform on log scale
  - Can also be done iteratively
  - Can be exponentially more efficient
  - Generally reduce the error faster in setting with many hyper parameters

Diagnosis and Debugging

- Visualize the model predictions

Diagnosis and Debugging

- Visualize the model predictions
- Analyze the worst errors: why?

Diagnosis and Debugging

- Visualize the model predictions
- Analyze the worst errors: why?
- Use train and test error as a diagnostic: Can you overfit the data?

UPPSALA
UNIVERSITET

Diagnosis and Debugging

- Visualize the model predictions
- Analyze the worst errors: why?
- Use train and test error as a diagnostic: Can you overfit the data?
- Use test suites: Can you get known results on a toy data. Both training error and derivatives

Diagnosis and Debugging

- Visualize the model predictions
- Analyze the worst errors: why?
- Use train and test error as a diagnostic: Can you overfit the data?
- Use test suites: Can you get known results on a toy data. Both training error and derivatives
- Monitor the gradients

UPPSALA UNIVERSITET

Diagnosis and Debugging

- Visualize the model predictions
- Analyze the worst errors: why?
- Use train and test error as a diagnostic: Can you overfit the data?
- Use test suites: Can you get known results on a toy data. Both training error and derivatives
- Monitor the gradients
- Monitor activation function statistics. Are some never activated?