UPPSALA
UNIVERSITET

Machine learning – Block 4

Måns Magnusson
Department of Statistics, Uppsala University

Autumn 2022

UPPSALA UNIVERSITET

This week's lecture

- Previous assignments
- The Mini-Project and Master Thesis Projects
- Convolutional Neural Networks
- Transfer Learning

Assignment 3: Evaluation

- A little too simple?
- Problems with Tensorflow and R

On this weeks assignment

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

UPPSALA UNIVERSITET

Mini-project

- Time to start think about the project.

UPPSALA UNIVERSITET

Mini-project

- Time to start think about the project.
- Supervised problem of choice on real data.

Mini-project

- Time to start think about the project.
- Supervised problem of choice on real data.
- 2-3 students.

UPPSALA UNIVERSITET

Mini-project

- Time to start think about the project.
- Supervised problem of choice on real data.
- 2-3 students.
- Supply a project proposal of data and problem at the end of *15th of December 23.59*.
- *Hint!* Submit page 1-1.5 of the project as project proposal.
- Deadline is after all lectures on supervised learning

UPPSALA UNIVERSITET

Mini-project

- Time to start think about the project.
- Supervised problem of choice on real data.
- 2-3 students.
- Supply a project proposal of data and problem at the end of *15th of December 23.59*.
- *Hint!* Submit page 1-1.5 of the project as project proposal.
- Deadline is after all lectures on supervised learning
- Feel free to combine it with your master thesis project!

Mini-project

- Time to start think about the project.
- Supervised problem of choice on real data.
- 2-3 students.
- Supply a project proposal of data and problem at the end of *15th of December 23.59*.
- *Hint!* Submit page 1-1.5 of the project as project proposal.
- Deadline is after all lectures on supervised learning
- Feel free to combine it with your master thesis project!
- Check with me if you have questions.
- The project should result in a 4 page report (PDF) using the ICML LaTeX template.

Section 3

Introduction

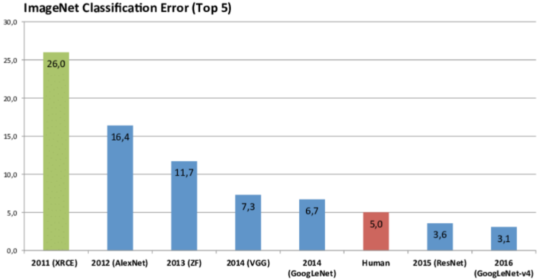
Convolutional Neural Networks

- Acknowledgements: Anders Eklund, Linköping University.

Convolutional Neural Networks

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

Figure: ImageNet performance (Roessler, 2019)

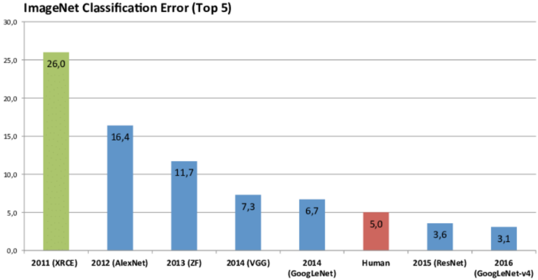


UPPSALA UNIVERSITET

Convolutional Neural Networks

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

UPPSALA UNIVERSITET

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

Computer Vision

- Problems
  - Image Classification
  - Image Segmentation

Computer Vision

- Problems
  - Image Classification
  - Image Segmentation
  - Object Detection

Computer Vision

- Problems
  - Image Classification
  - Image Segmentation
  - Object Detection
  - Object Localization

Computer Vision

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

UPPSALA UNIVERSITET

Computer Vision

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

Computer Vision

- Problems
  - Image Classification
  - Image Segmentation
  - Object Detection
  - Object Localization
- 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

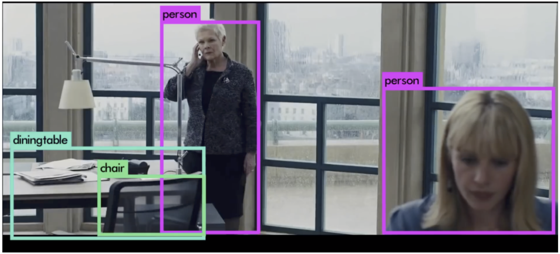


Figure: Object detection (see https://www.youtube.com/watch?v=VOC3huqHrss)

Example: Pneumonia detection

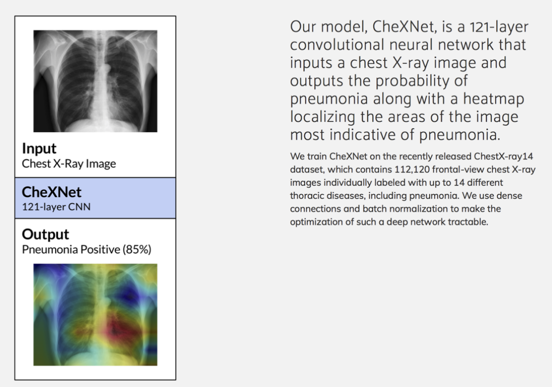


Figure: Rajpurkar et al. (2017). Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. arXiv preprint arXiv:1711.05225.

Example: Fracture detection

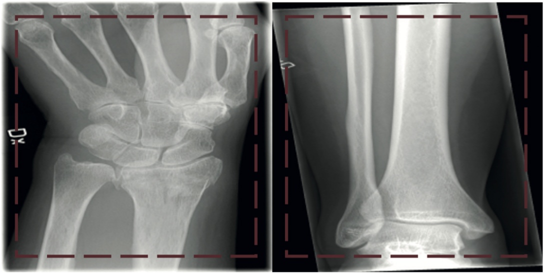


Figure 1. 2 images from the dataset. The area within the red box is the section presented to the network in order to classify the image. The left image is of a wrist fracture while the right image is without any apparent fracture.

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

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

- Previous assignments
- The Mini-Project
- Introduction
- Convolution
- Convolutional Neural Networks
  - The Convolution Layer
  - The Pooling Layer
  - Regularization
  - Examples
- Transfer learning
- Practical Methodology

MNIST example

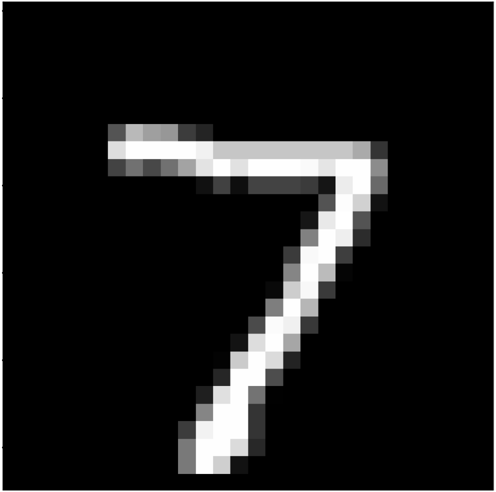


Figure: Example from the MNIST dataset (28 by 28 pixels)

How to train models for images?

- We want to learn representations of parts of images

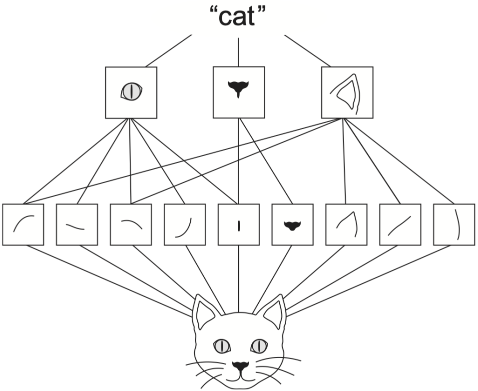


Figure: The representations of a cat (Chollet and Allair, 2018, Fig 5.2)

How to train models for images?

- We want to learn representations of parts of images

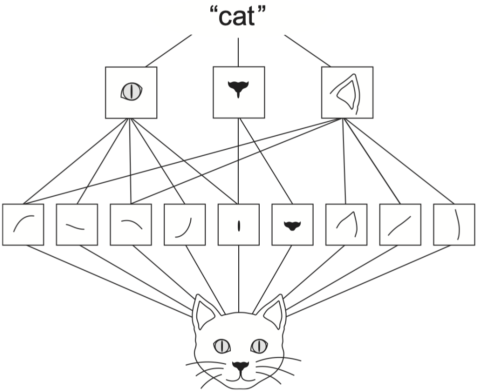


Figure: The representations of a cat (Chollet and Allair, 2018, Fig 5.2)

- CNN uses Convolutional Layers to learn parameter efficient representations

Learning Representations for Images (again)

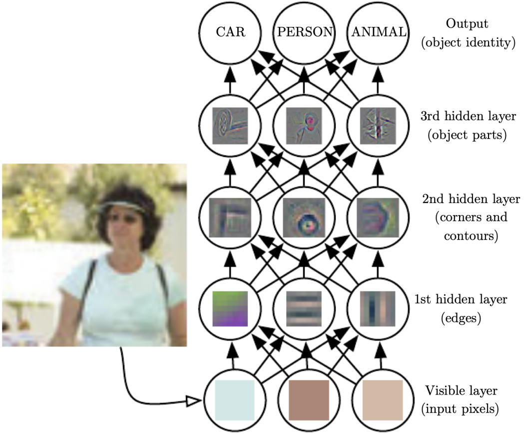


Figure: Learning representations for images (Goodfellow et al, 2017, Fig. 1.2)

Section 4

Convolution

UPPSALA UNIVERSITET

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

Discrete Convolution

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

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

- 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

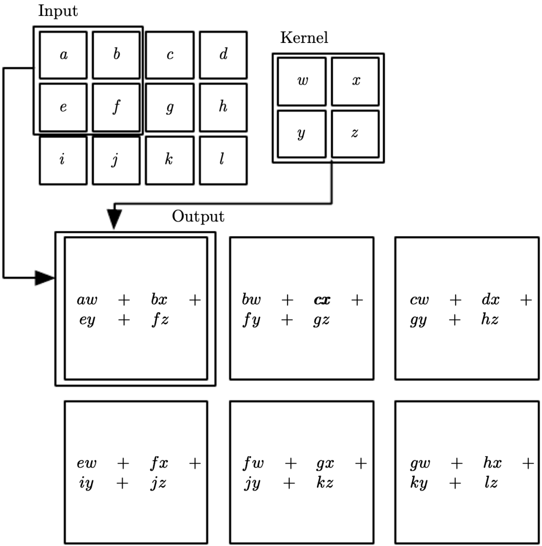


Figure: Convolution for an Image (Goodfellow et al, 2017, Fig. 9.1)

Convolution of images: Example

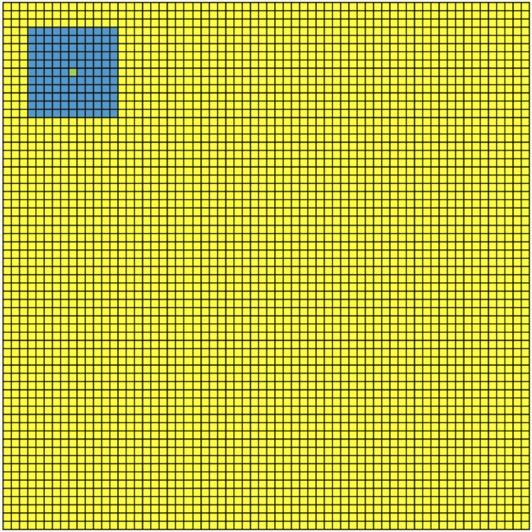


Figure: Convolution example.

Convolution of images: Examples

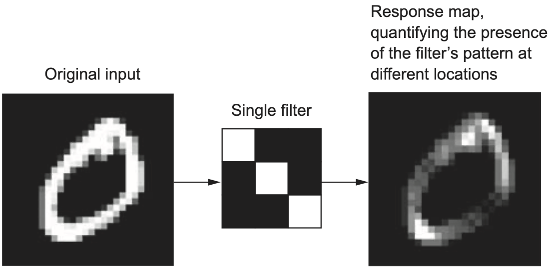


Figure: Convolution for an Image (Chollet and Allaire, 2018, Fig. 5.3)

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

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

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

Section 5

Convolutional Neural Networks

Convolutional Neural Networks

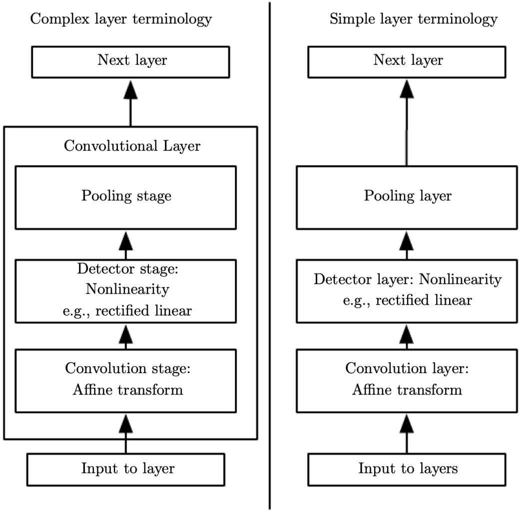


Figure: Convolution layer (Goodfellow et al, 2018, Fig. 9.7)

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

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

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

UPPSALA UNIVERSITET

Padding

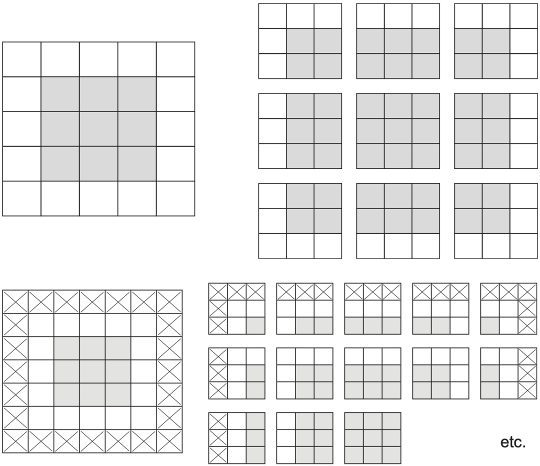


Figure: Padding and valid edge handling (Chollet and Allair (2018), Fig. 5.5, 5.6)

UPPSALA UNIVERSITET

Stride

- Skip every *n*th pixel
- Reduces the computations

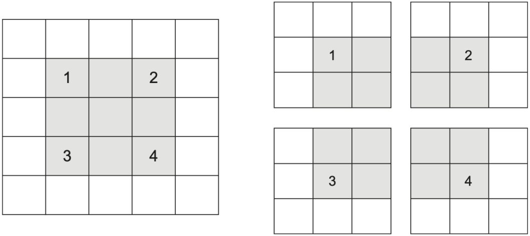


Figure: Strides (Chollet and Allair (2018), Fig. 5.5, 5.6)

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

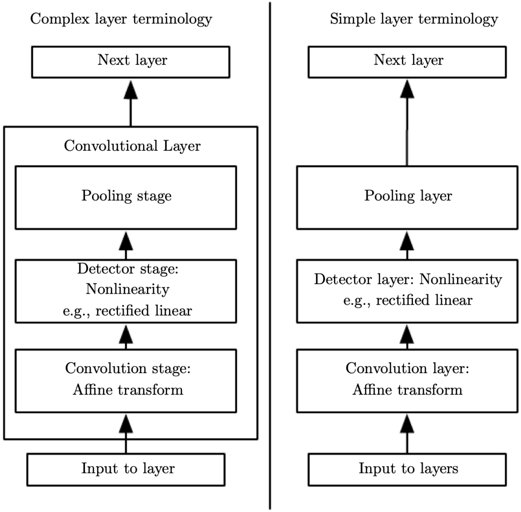


Figure: Convolution layer (Goodfellow et al, 2018, Fig. 9.7)

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

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

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)

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

UPPSALA UNIVERSITET

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

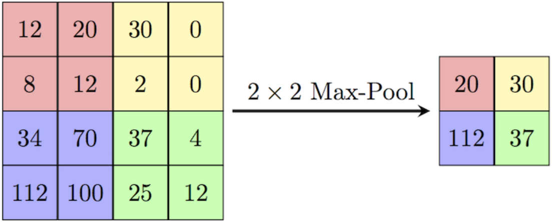


Figure: Strides (Computer Science Wikipedia)

Using pooling to learn invariances

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

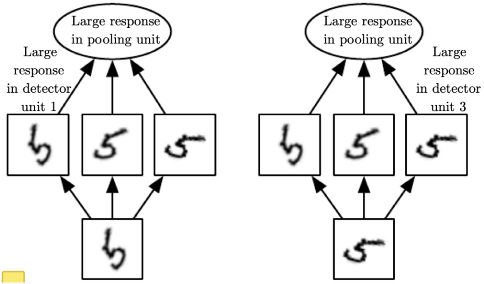


Figure: Learning invariances (Goodfellow et al., 2017, Fig. 9.9)

UPPSALA UNIVERSITET

- Previous assignments
- The Mini-Project
- Introduction
- Convolution
- Convolutional Neural Networks
  - The Convolution Layer
  - The Pooling Layer
  - Regularization
  - Examples
- Transfer learning
- Practical Methodology

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

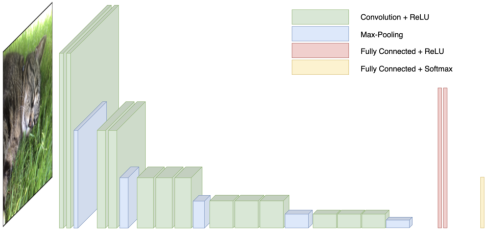


Figure 2.4: *Illustration of the data flow through the network VGG16. Data size, of the format $[c, h, w]$, is shown for the input image, output of each max-pooling layer, output after the first two fully connected layers, and the final network output.*

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

UPPSALA UNIVERSITET

Section 6

Transfer learning

Transfer learning

- Learning representations in P_1 will help generalizations in P_2
- Main idea: transfer learned representations to new settings where they might be useful
- "Transfer knowledge between problems"
- Transfer/reuse learned weights

Transfer learning

- Learning representations in P_1 will help generalizations in P_2
- Main idea: transfer learned representations to new settings where they might be useful
- "Transfer knowledge between problems"
- Transfer/reuse learned weights
- A Bayesian perspective: A strong prior
- Use (large) pre-trained models for smaller problems
- One of the main reasons for the success of (convolutional) neural networks

UPPSALA UNIVERSITET

Transfer learning

- Learning representations in P_1 will help generalizations in P_2
- Main idea: transfer learned representations to new settings where they might be useful
- "Transfer knowledge between problems"
- Transfer/reuse learned weights
- A Bayesian perspective: A strong prior
- Use (large) pre-trained models for smaller problems
- One of the main reasons for the success of (convolutional) neural networks
- The Risk: Catastophic forgetting
- Two types of transfer learning in Neural Networks:
  - Feature extraction
  - Fine Tuning
- 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. Now mainly resused models.

Learning Representations for Images

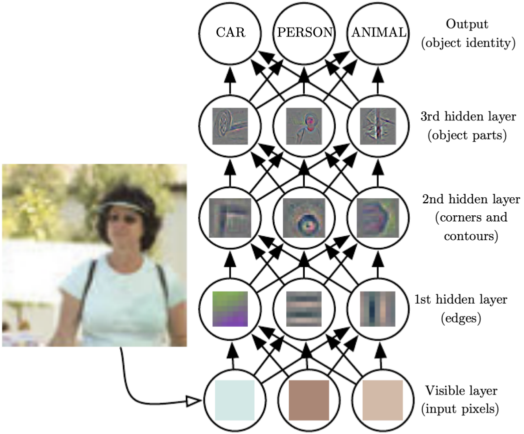


Figure: Learning representations can be crucial (Goodfellow et al, 2017, Fig. 1.2)

Feature Extraction

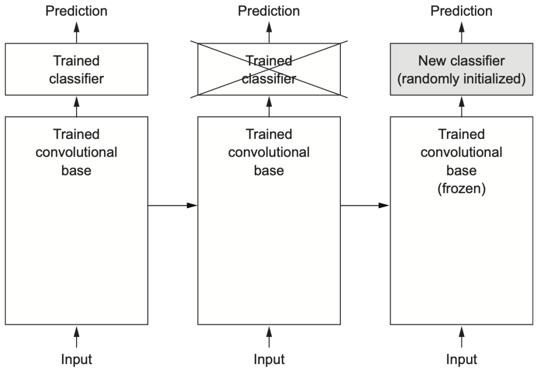


Figure: Using convnets as base for feature extraction (Chollet and Allair, 2018, Fig 5.12)

Fine-Tuning

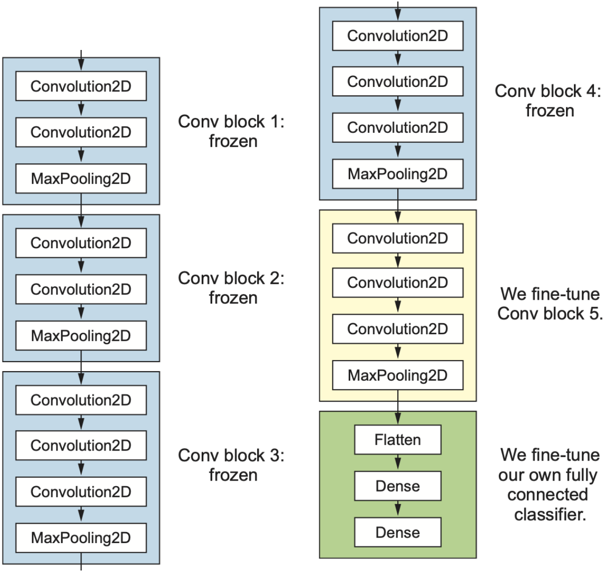


Figure: Finetuning a convolutional base (Chollet and Allair, 2018, Fig 5.15)

Section 7

Practical Methodology

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

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 target 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 target 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 target of the system! This may need multiple metrics.
- What is good enough? Remember the Bayes error!
- What performance can you expect?

UPPSALA UNIVERSITET

1. Determine your goals

- Why are you building a model?
- Set up the metric based on the overall target 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 target 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 target 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?
- It is not uncommon that manual curation is faster and easier in some settings.

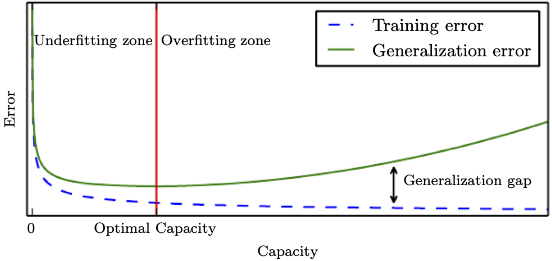
UPPSALA UNIVERSITET

2. Setup your baseline

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

Statistical Learning Theory

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



3. Diagnosing and improving your model: More data

- Gather more data: Usually a good solution
- Avoid focus on the "best" algorithm and the "ML-sickness"
- If high loss in training: The training data is not yet used fully
- If high loss on test: The quality of the training data can be poor. This means that the Bayes error is larger.
- If low training error, but high test error: Probably more data is a good way forward. If costly or difficult with more data -¿ regularization

Learning curve to assess if you need more data

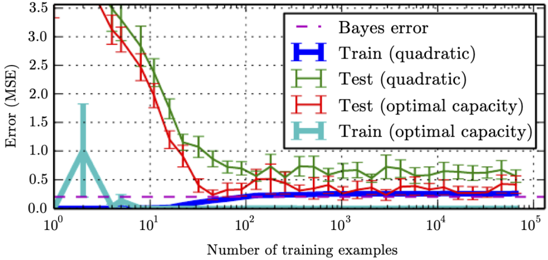


Figure: Learning curve to assess the need for more data (Goodfellow et al., 2017, Fig 5.4)

3. Diagnosing and improving your model: Regularization

- General idea: Increase data and model capacity until target is reached.
- Adapt hyperparameters to get optimal capacity
- Goal: The primary goal of manual hyperparameter search is to adjust the effective capacity of the model to match the complexity of the task.
- Ideally, each marginal hyperparameter has a U-shaped form
- Step 1: Get good training error. Tune learning rate. U-shaped for training error. Neural network performs best when training error is low.
- Step 2: Tuning hyperparameters (regularization) requires monitoring both training and test error to diagnose whether your model is overfitting or underfitting
- Best performance: Larger model that is regularized well.
- End goal: Good performance on test set
- With low training error you can always improve by adding more data

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
  - random search can be exponentially more efficient
  - reduce the error faster

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?