INTRODUCTION

INF5860 — Machine Learning for Image Analysis

Anne Solberg, Tollef Jahren, Ole-Johan Skrede

16.01.2019

University of Oslo

Plan for today

- Introduction about the course
- Motivation: what deep learning can do
- Short background on image classification
- Short introduction to image convolution
- Information about exercises and jupyter/python

INTRODUCTION AND ABOUT THE COURSE

Lecturers

Tollef Jahren
tollefsj@ifi.uio.no

Ole-Johan Skrede
olejohas@ifi.uio.no

Anne Schistad Solberg
anne@ifi.uio.no

Overview

Lectures

- Machine learning
- Deep learning and image analysis
- Getting deep learning to work in practice
- Applications

Groups

- Jupyter notebooks, numpy and image analysis
- Exercises on image classification, image captioning, image segmentation
- Basic skills in Python
- Pytorch for more complex deep learning
- Students get time-limited access to GPU-servers at USIT for selected exercises (from March)

Schedule

Lectures

- Wednesdays 14.15-16, Lille Aud (Check room on course web page)

Mandatory exercises

- 2 Mandatory exercises, one in basic python and one in PyTorch

Groups

- Tuesday 10.15-12, Thursday 10.15-12 Room Modula
- 3 Group lecturers Lucas Charpentier (l.g.g.charpentier@fys.uio.no), Samuel Korsan Knudsen (s.k.knudsen@fys.uio.no), Herman Netskar (herman.netskar at gmail.com) (corrector only)
- Up to 4 hours weeks to complete labs, you can attend both sessions if needed
- Access to GPU-servers given additonaly for 1-2 hours a week (selected periods)

Course resources

- Course web page: https://www.uio.no/studier/emner/matnat/ifi/IN5400/v19/index.html
- Curriculum: lecture notes, weekly exercises, mandatory exercises
- No book, but links to relevant literature/notes given
- Links to relevant online lectures given for each week
- Piazza for discussion https://piazza.com/class/jq82079l9sg5xy
- Devilry for submitting mandatory exercises
- Digital exam, focus on understanding the topics

Who is this course for?

- Core focus: Master and PhD students working on image analysis problems using deep learning
- Will be useful for others with a good background in Python and linear algebra wanting to learn deep learning
- Demanding programming exercises: do all of them!
- Learning is about loss functions and derivation - you are expected to do calculations and compute derivatives by hand
- Expertize wanted for many companies - good job prospects, but your motivation must be high!

Prelimary schedule

- 16.1 Introduction
- 23.1 From regression to softmax classification
- 30.1 Feedforward nets and backpropagation
- 6.2 PyTorch
- 13.2 Convolutional nets
- 27.2 Guidelines for training and architectures
- 6.3 Generalization
- 13.3 Localization and segmentation
- 20.3 Adverserial fooling and interpretability
- 27.3 Recurrent nets
- 3.4 Unsupervised
- 10.4 GAn
- 24.4 Reinforcement learning
- 22.5 Repetition
- 31.5 Exam

Web page

- https://www.uio.no/studier/emner/matnat/ifi/IN5400/v19/index.html
- Lectures with links to lecture foils, curriculum, weekly exercises, and mandatory exercises.
- Piazza
- Messages on the web page
- Urgent messages emailed to studenter.in5400@ifi.uio.no and studenter.in9400@ifi.uio.no. Links to your official uio-email - read this!

Group sessions

- Tuesday and thursdag 10.15-12
- Group teachers:
  Lucas Charpentier (l.g.g.charpentier at fys.uio.no),
  Samuel Korsan Knudsen (s.k.knudsen at fys.uio.no),
  Herman Netskar (herman.netskar at gmail.com) (corrector only)
- 2x2 hours to give you time to solve programming errors with help
- Programming exercises mostly as jupyter notebooks
- Theory exercises

Mandatory exercises

- Required: use Python and PyTorch
- Exercise 1: implement vectorized backpropagation and layers in a feedforward and convolutional net, available around February 1.
- Exercise 2: Image captioning in PyTorch

Groups first week

- First group tomorrow, thursday 17.1 10.15-12
- Notebook on math operations in python
- Notebook on indexing
- Notebook on image convolution
- Download zip-file with exercises

Using Jupyter notebooks

- Use the linux computer in modula (/opt/ifi/anaconda3/bin/jupyter-notebook)
- Download anaconda/python3 to your own computer
- Later in the course: access GPUs on servers

Curriculum

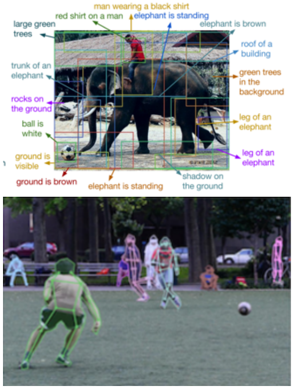
- Lecture notes, weekly exercises, and mandatory exercises define the curriculum
- The field is so new and rapidly changing that no good book exists
- Links to useful notes and online lectures will be given.

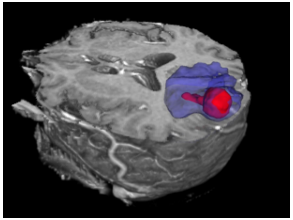
Required background

- Good knowledge in Python programming
- Experience in image analysis desired
- Good knowledge in mathematics (as a tool)
    - Linear algebra: vectors, matrix operations, dot products
    - Derivation of loss functions
    - Simple optimization

MOTIVATION

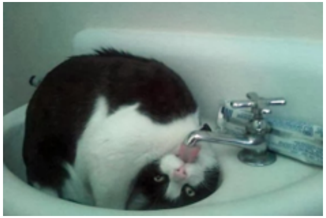
Many tasks require information from images



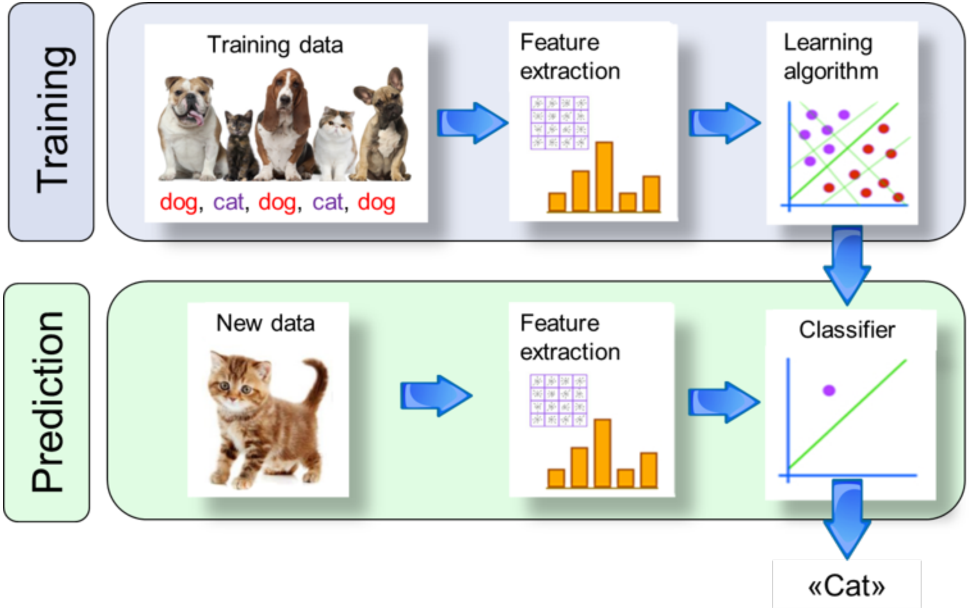


Why deep learning in image analysis?

- The human visual system is so good in recognizing objects
- It is almost impossible to specify all variations of objects that can occurr
- Writing new code for each variation is a lot of work

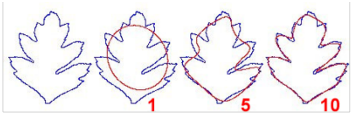


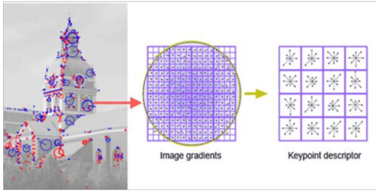
Classical machine learning

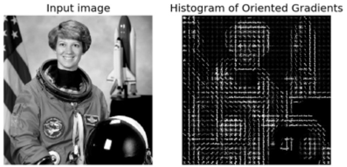


Good features are essential

- Features = Data representation
- Good features capture posterior belief about explanatory causes and underlying factors of variation
- Multitude of hand-designed and fixed features currently in use

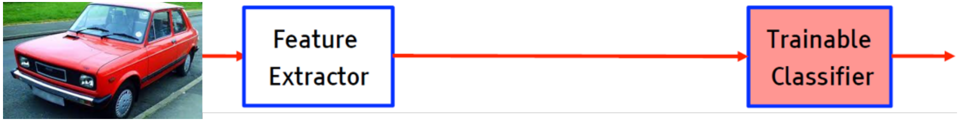




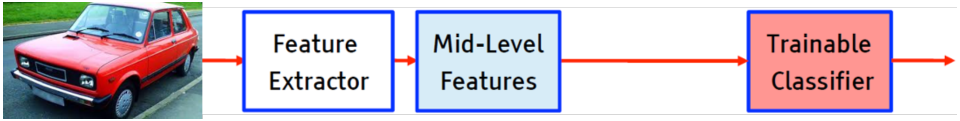


Architectures: image recognition systems

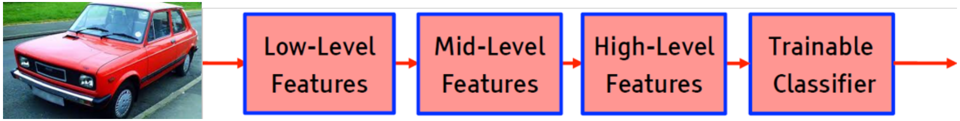
Tradional: Handcrafted features + supervised classifier



Mainstream approach until recently for image (and speech) recognition:

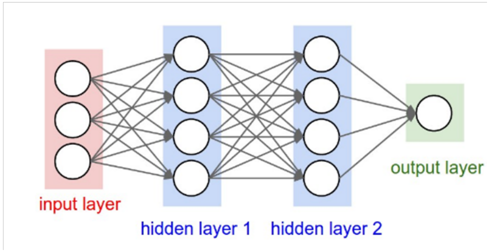


Deep learning: Train multiple stages end-to-end



How does a fully connected net see the world?

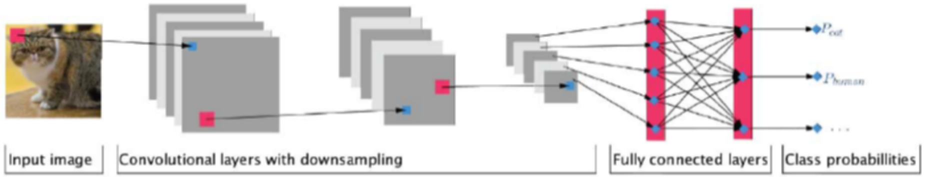
Unwrap 2D image a 1D vector. One input node per pixel



Fully connected neural network on images

- Most image applications are absolute position invariant
- A fully connected network will have too many parameters and not be able to scale to normal size images and generalize

Convolutional networks enforce similar response to neighboring pixels



What do the layers learn?

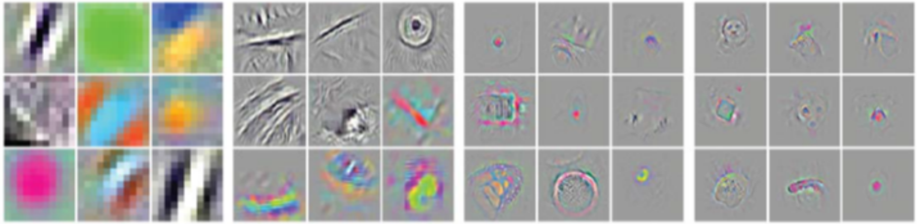
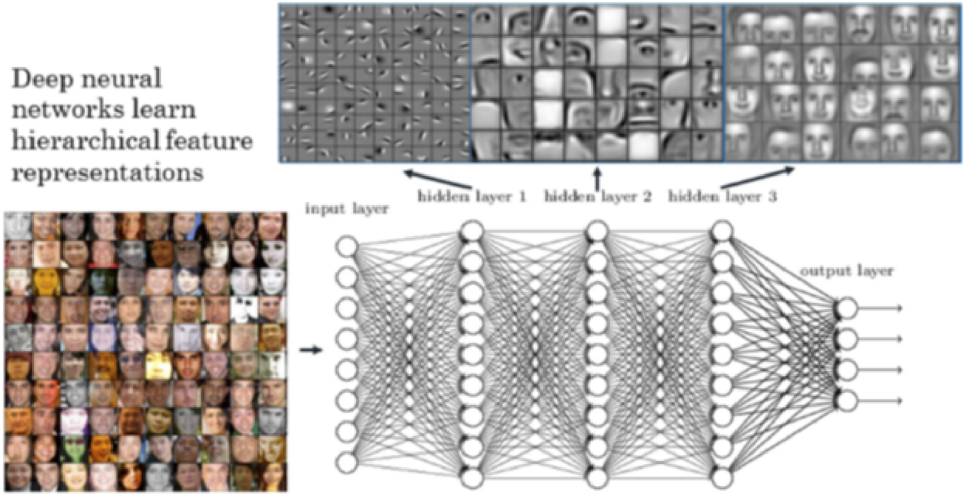


Figure 2: Visualized filter responses for some of the nodes of the first four layers of Alex-net (Krizhevsky et al. (2012)). The layers 1-4 are displayed from left to right. Since the filter responses cannot be directly related to the input image (due to the non-linearity in the network) the visualization technique by Zeiler and Fergus (2014) was used. The filter responses in this figure originally appeared in Zeiler and Fergus (2014) and is printed with permission from the authors of the original article.

HOW DOES DEEP NETWORKS CREATE INVARIANCE?

Deep neural networks learn hierarchical feature representations



GENERAL IMAGE ANALYSIS APPLICATIONS

Object detection

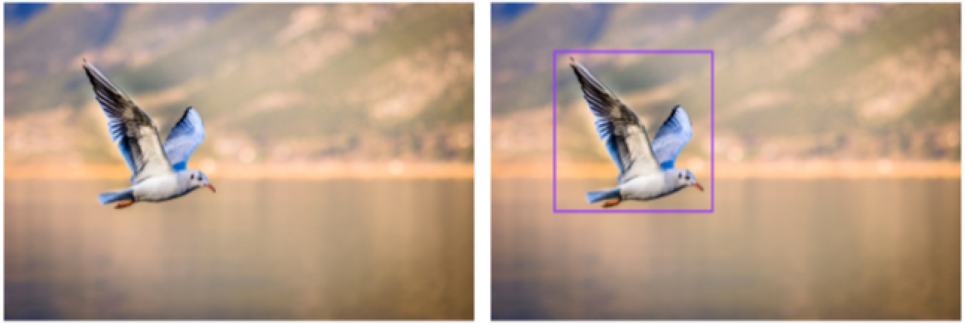
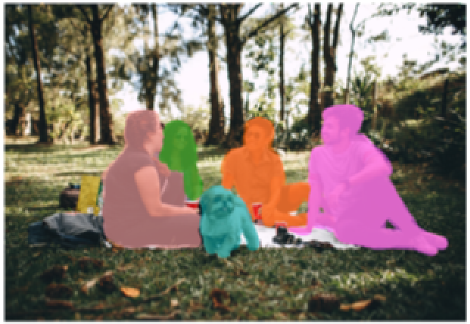
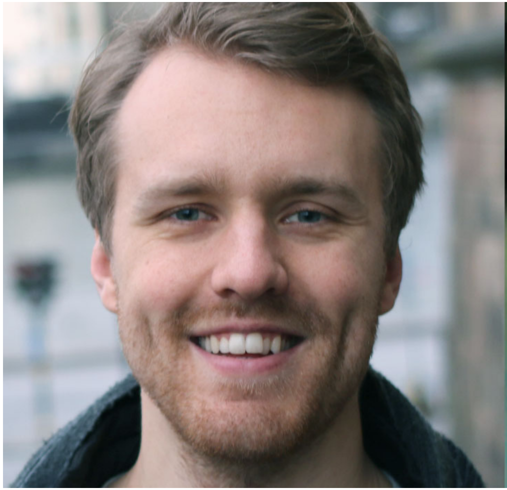


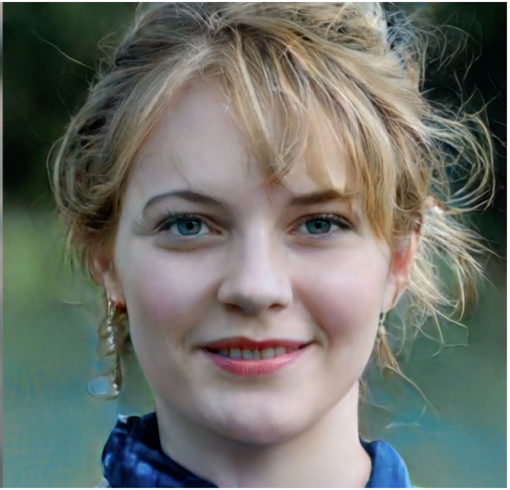
Figure 1: Seagull. Image source: https://www.pixabay.com

Instance segmentation

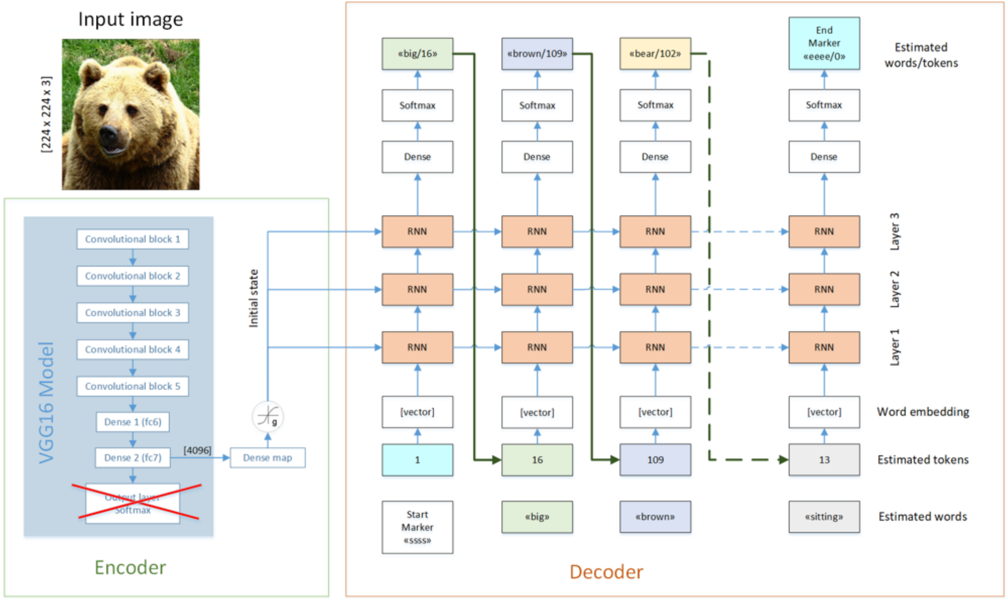


Variational autoencoders - which face is fake?

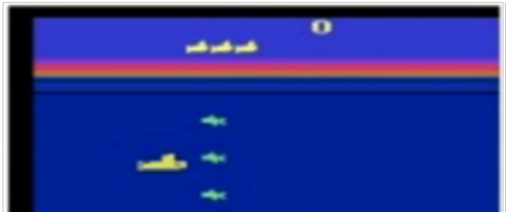




Combining recurrent and convolutional networks for image captioning



Reinforcement learning for games

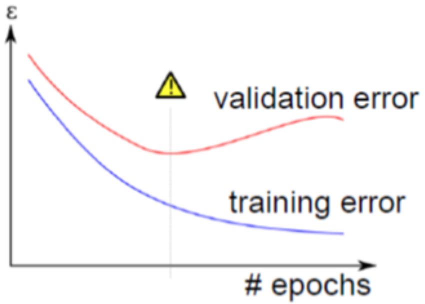


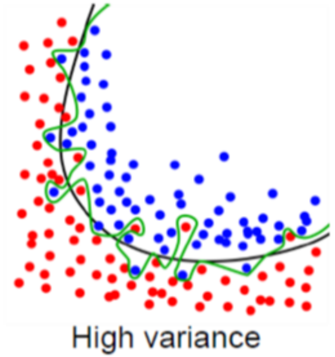
CHALLENGES WITH DEEP LEARNING

Training a machine learning system

- Training data: data with known objects used to find the weights in the net
- Validation data: data with known objects used to select architectures and hyperparameters
- Test data: data with known objects used once to estimate the error rate of the system

The danger of overfitting to training data



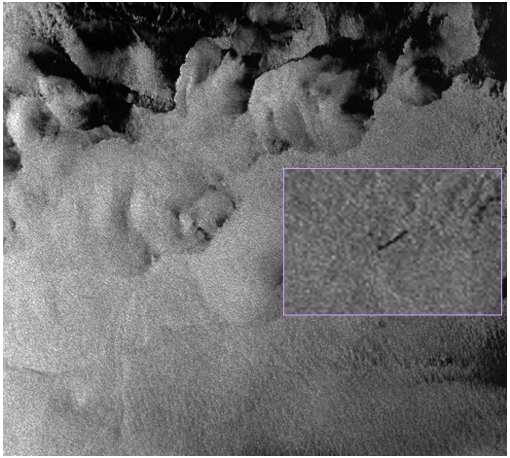


Working with skewed data

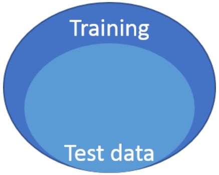
- Challenge: mostly data from “normal” classes, few samples from other classes (e.g. cancer, pollution etc.)
- Problem: How can we avoid that the networks fits the normal class and ignores the small but important class.

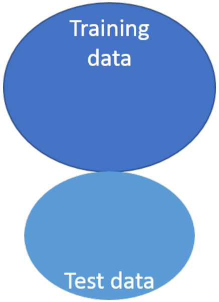
Working with skewed data - cont.

- Challenge: mostly data from "normal" classes, few samples from other classes (e.g. cancer, pollution etc.)
- Problem: How can we avoid that the networks fits the normal class and ignores the small but important class.



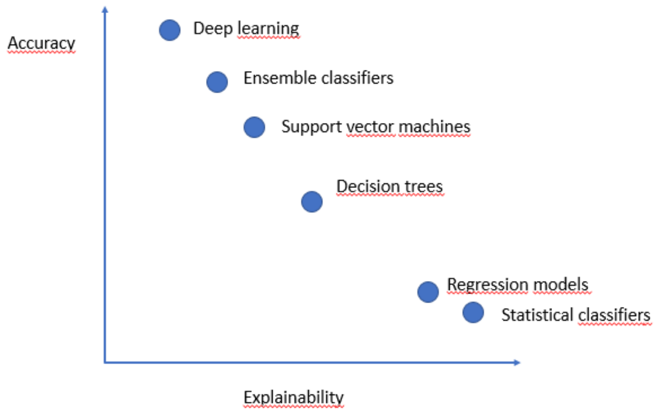
Domain shift





Ohen ghoor!

Explaining the results



With power comes responsibility!

- Models with millions of parameters can fit to almost anything!
- Seeing inside the black box: why did the computer decide the given class?
- How do we evaluate the results?
- What is the significance of the results depening on the data?
- How well can we expect the model to generalize?
- What are the ethical sides of the problem and the desicions?
- Know what we have not learned yet.

Limitations of deep learning

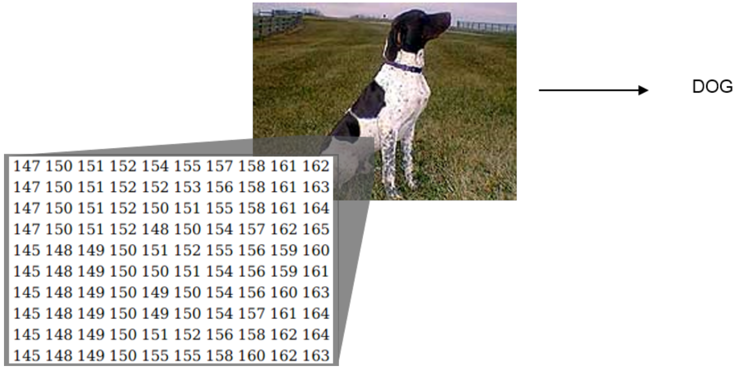
- Complex tasks can be difficult to learn, particularly memory-intensive tasks
- Sometimes easy to fool the network
- Som tasks are untrainable
- You have only learnt your training data Deep learning requires good knowledge of technical details, and how to train efficiently

INTRODUCTION TO IMAGE CLASSIFICATION

Reading material

- Link to note on introduction to classification of images: here
- Convolution: (in Norwegian):Links to two lectures from INF 2310 Image Processing
    - Lecture1
    - Lecture2

Supervised Image classification - introduction

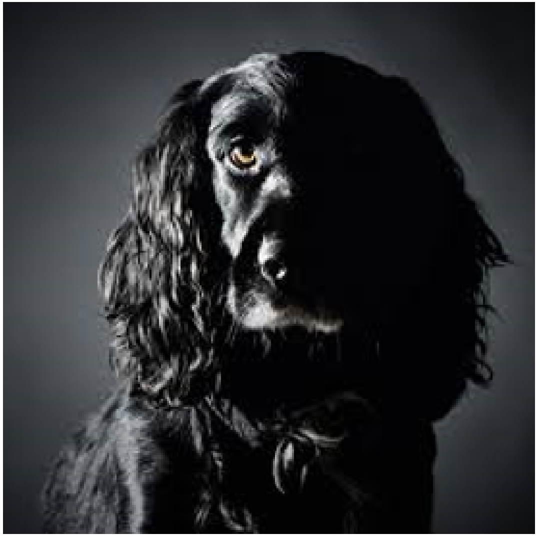


147	150	151	152	154	155	157	158	161	162
147	150	151	152	152	153	156	158	161	163
147	150	151	152	150	151	155	158	161	164
147	150	151	152	148	150	154	157	162	165
145	148	149	150	151	152	155	156	159	160
145	148	149	150	150	151	154	156	159	161
145	148	149	150	149	150	154	156	160	163
145	148	149	150	149	150	154	157	161	164
145	148	149	150	151	152	156	158	162	164
145	148	149	150	155	155	158	160	162	163

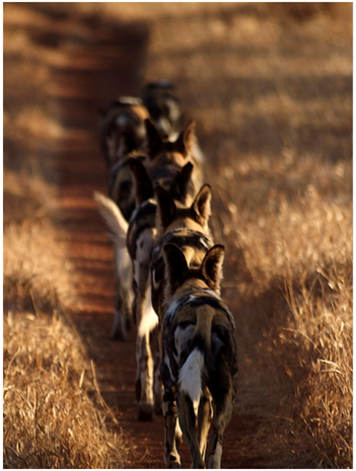
Task: use the entire image to classify the image into one of a set of known classes.

Which object does the image contain

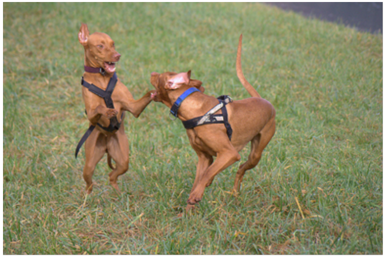
Challenges: Illumination

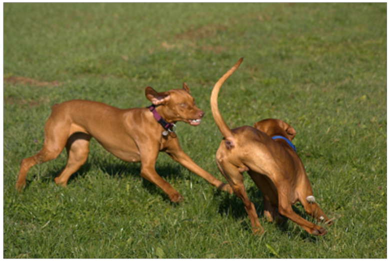


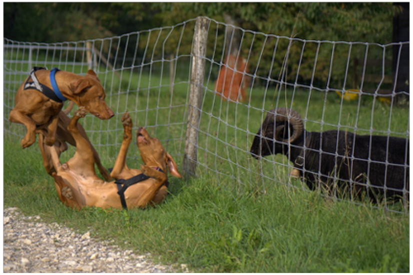
Challenges: Occlusion



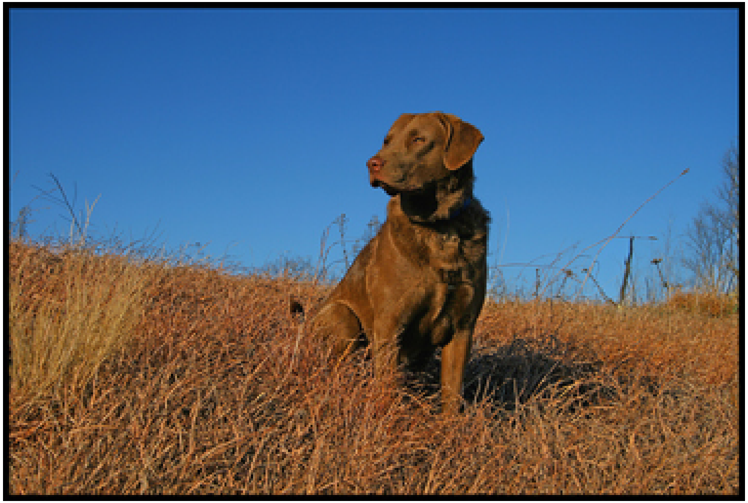
Challenges: Deformation



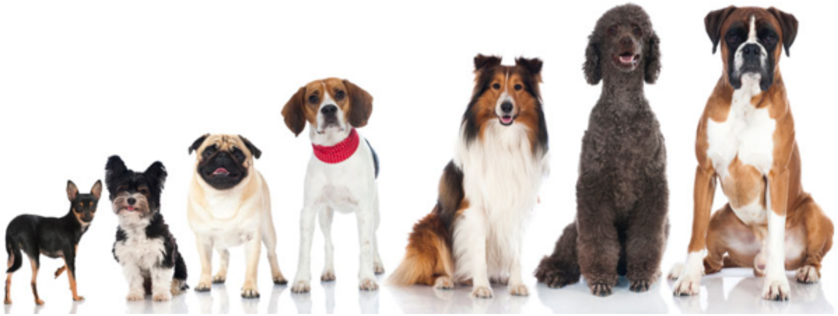




Challenges: Background clutter



Challenges: Intraclass variation

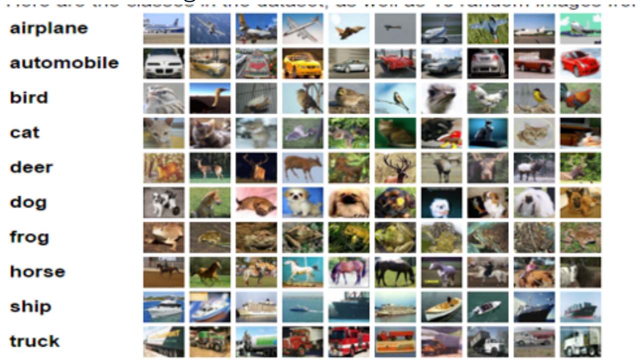


The CIFAR-10 image dataset

10 classes

50 000 training images of size 32$\times$ 32

10 000 test images



https://www.cs.toronto.edu/ kriz/learning-features-2009-TR.pdf

Measuring similarity between two images

- Image $F(k, i, j)$, pixel (i, j) in band k.
- **L1-distance:**
$$d_1(i,j) = \sum_k \sum_i \sum_j |F1(k,i,j) - F2(k,i,j)|$$
- **L2-distance:**
$$d_2(i,j) = \sum_k \sum_i \sum_j \sqrt{(F1(k,i,j)^2 - F2(k,i,j))^2}$$

L2 is called Euclidean distance

K-Nearest-Neighbor classification

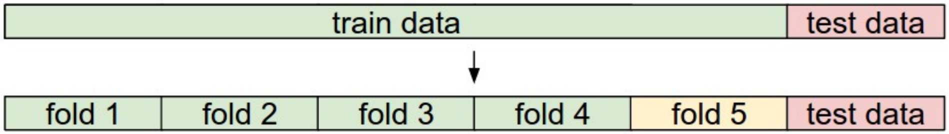
- Given a set of images m training images $F^{(i)}$ with true class labels $y^{(i)}, i \in \{1, m\}$
- Classification of a new sample $F^{(new)}$ is done as follows:
    - Out of the m training images, find the k images with smallest distance (measured by L1 or L2)
    - Out of these k samples, identify the most frequent class labels
    - Assign the class label the sample as the most frequent class labels among the k labels. Denote the predicted class as $\hat{y}^{(i)}$.
- k should be odd, and must be selected a priori (try different values of k and choose the one with the lowest classification error using crossvalidation)
- Classification error can be measured by counting the number of samples where the predicted class is equal to the true class $\hat{y}^{(i)} = y^{(i)}$

Selecting K using crossvalidation

For each value of k:

- Cross-validation: split the training data into d subset/folds
- Train on data from d-1 folds ,
- Estimate the accuracy/compute the number of correctly classified images on the last fold , store the accuracy.
- Repeat this nfold times and compute for the average of the accuracies.
- Repeat with different values of k, select the value that got the highest accuary.
- Using the best value of K, classify the test data set ONCE to get the accuracy of the classifier.

Crossvalidation example



- Example: 10000 training images, 5000 test images.
- Split training images into 10 folds of 1000 images.
- Train on 9 folds (9000 images), compute accuracy on the last 1000 images during cross-valiation.
- After finding k: train on all 10000, and estimate the reported accuracy on the 5000 test images.

About KNN-classification

- If $k = 1$ (1NN-classification), each sample is assigned to the same class as the closest sample in the training data set.
- If k is very large, this is theoretically a very good classifier.
- This classifier involves no ”training time”, but the time needed to classify one pattern xi will depend on the number of training samples, as the distance to all points in the training set must be computed.
- ”Practical” values for k: $3 \leq k \leq 9$
- Classification performance should always be computed on the test data set.
- **For image classification using the original pixel values as feature values, the results is not invariant to object position, scale, contrast!**

IMAGE CONVOLUTION

Image convolution

- Common operation in image analysis
- Apply pre-defined filters to enhance or highlight certain features in an image
- Examples:
    - Image smoothing using e.g. an averaging filter or a Gaussian filter
    - Edge detection by computing the first derivatives
    - Point detection and edge locatization using second derivatives
- **Classical filters are predefined, but convolutional networks will estimate filters to locate various shapes in the image**

2D IMAGE CONVOLUTION

- Given a 2D filter $w(i, j)$ of size $h \times h$, $h = 2h_1 + 1$.
- Given an image $F(b, x, y)$, where b is band number and x and y spatial coordinates.
- To compute the result for **one pixel** in band b of the new image $G(b, x, y)$, compute

$$G(b, x, y) = \sum_{j=-h_1}^{h_1} \sum_{k=-h_1}^{h_1} w(j, k) F(b, x - j, y - k)$$

- The result is a weighted sum of the input pixels surrounding pixel (x, y).
- The value of the next pixel if G is found by moving the filter one position and computing again.
- To compute for an entire image with b bands, a straigthforward implementation will use 5 for-loops!

Convolution example

In this example, we compute the result for every location where the filter has some overlap with the image. Another choice could be to compute the result for every location where the entire filter fits inside the image.

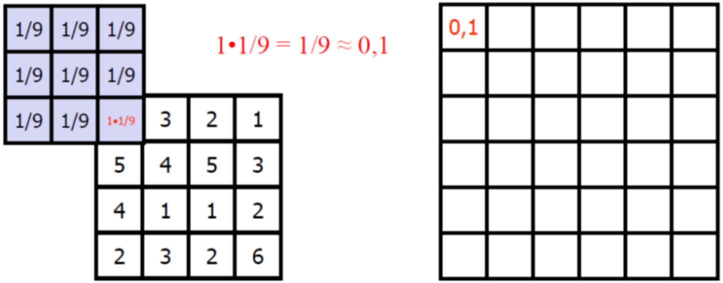
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Filter

1	3	2	1
5	4	5	3
4	1	1	2
2	3	2	6

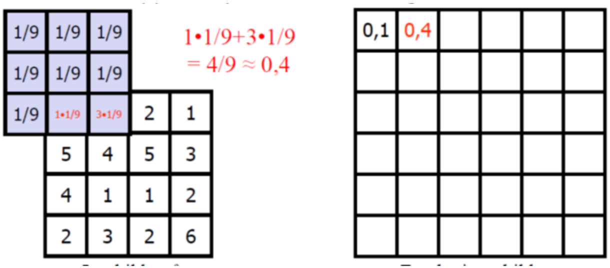
Image

Example continued



- Rotate the filter 180 degrees (due to the minus in the equation)
- Overlay on a the first pixel and multiply filter weights with image pixel values, then sum up over filter kernel

Example continued



- Move the filter to the next location and repeat
- **Note: in this example we compute the result for all pixels where the filter kernel overlaps the image. Since the result is assigned to the origin, this corresponds to zero-padding the input image with a frame of size $h1 + 1$ on each side**
- We can also chooose to only compute the results where the origin of the filter fits inside the image, or where the filter fits entirely inside the image. This will affect the size of the output image

Example of filter: average filter of size 5×5

Compute the average value in the neighborhood, normalize by dividing by the number of filter elements

$$\frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

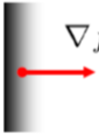
Digital gradient operators

- The gradient of *f(x)* is $\lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$

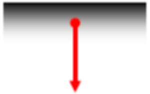
- The (intensity) gradient of an image:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

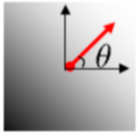
- The gradient points in the direction of most rapid (intensity) change



$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$



$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$



$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

2D GRADIENT OPERATORS

- Use the horisontal and vertical Sobel filters:

$$H_x(i,j) = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \ H_y(i,j) = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

- Let $G_x(b,x,y) = H_x * F(b,x,y)$, where $*$ is the convolution operator
- Let $G_y(b,x,y) = H_y * F(b,x,y)$,
- Compute the gradient magnitude as

$$\sqrt{(G_x(b,x,y)^2 + G_y(b,x,y)^2)}$$

- Compute the gradient direction as

$$\theta = \tan^{-1}\left(\frac{G_y(b,x,y)}{G_x(b,x,y)}\right)$$

Questions?