Image interpretation and Deep Learning

Signal Processing and Identification in Control and Monitoring of Mechatronic Devices

Ziemowit Dworakowski
AGH University of Krakow
zdw@agh.edu.pl

1

AI usage in image processing tasks

Can be done by shallow and deep learning:

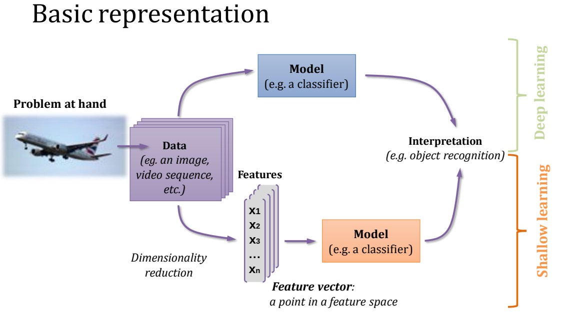
- **Classification / Labeling**
  Putting labels on the image or its parts
- **Intelligent global processing of images**
  Filter calibration, filtration, image segmentation, enhancing

Can be done by deep learning only:

- **Modeling – advanced interpretation of vision data**
  Understanding what is going on in the image (and why)
- **Artificial image generation**
  Building new images from prompts, augmenting image datasets

2

Basic representation

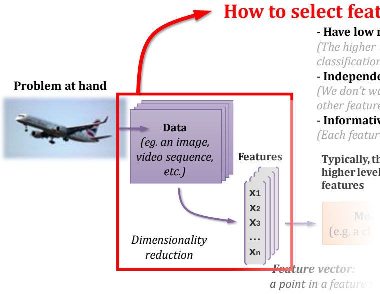


3

How to select features?

- **Have low number of them**
(The higher the numer is, the more advanced classification algorithms are needed)
- **Independent, uncorrelated**
(We don't want to copy information stored in other features)
- **Informative (meaningful)**
(Each feature should contribute to the solution)

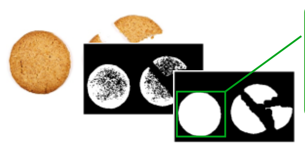
Typically, the more difficult the task is, the higher level of abstraction is required from features



4

Steps required for design of a shallow image-based classifier

1. Data acquisition (*many examples of objects from all the classes*)
2. Preprocesing to highlight class differences
3. Design of a good feature vector (*sensitive to damage*)
4. Design and training of a classifier based on **training** subset of data
5. (optional) Optimization of metaparameters using a **validation** subset of data
6. Test using a **testing** subset of data



Circumference to area ratio
Corner number
Moment invariants
Number of detected objects
Area
Object location
Object color

5

Steps required for design of a shallow image-based classifier

1. Data acquisition (*many examples of objects from all the classes*)
2. Preprocesing to highlight class differences
3. Design of a good feature vector (*sensitive to damage*)
4. Design and training of a classifier based on **training** subset of data
5. (optional) Optimization of metaparameters using a **validation** subset of data
6. Test using a **testing** subset of data

If problem is simple enough and features are good enough, even simple decision tree will get the job done. Otherwise, we need to try to improve classifier

Circumference to area ratio
Corner number
Moment invariants
Number of detected objects
Area
Object location
Object color

6

Assumptions:
- Task is so difficult that simple classifiers are not usable
- We've decided to use MLP

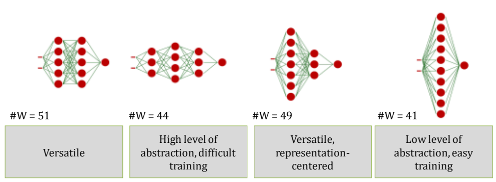
Decisions to be made:
- How to divide data among train/val/test datasets?
- What algorighm should be used for training?
- **How to structure our neural network?**
- **How to actually evaluate our decisions?**

7

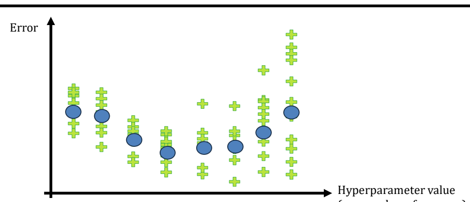
How to choose a MLP structure?

- How many neurons?
- How many layers?
- How to assign neurons to layers?

Its best to start from an arbitrary point based on experience and then proceed with changes from there. Because otherwise we would need to do full optimization…



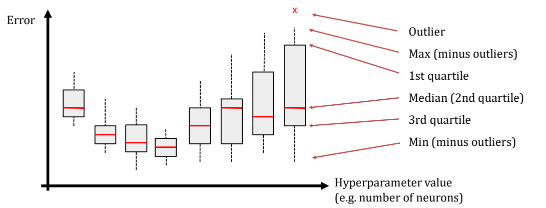
8



A „mean" approach

Average several tries for each of the methods. Usually only a few tries is required to observe „general tendencies" (e.g. in which area should we look for an actual minimum)

9



A „mean" approach — Average several tries for each of the methods. Usually only a few tries is required to observe „general tendencies" (e.g. in which area should we look for an actual minimum)

A „box-plot" approach — Often metaparameter value changes affect not only general error values but also their spread. Using many tries and generating box plots of the results allows for understanding of the meaning of the results.

10

General remarks:

1. **Classifier should be chosen on a basis of our knowledge of a feature space**
   (How many dimensions? How many features? Are the samples clustered?)
   The „task type", (e.g. do we classify cookies or vegetables) is much less relevant.

2. **Training, testing (and validation) datasets should be separate**
   Either we begin with random division of a data into training and testing subsets, or **(better!)** we gather new portion of data for testing purposes in another experiment.

3. **Number of degrees of freedom of a classifier (e.g. net weights) should depend on number of data samples**
   A good „rule of thumb" is that for each DOF of a classifier at least 10 data samples are required. If we can't do that, we make sure that overfitting is accounted for!

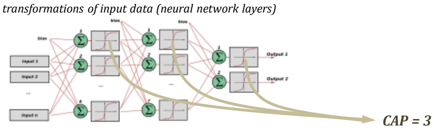
4. **Feature quality > Classifier**
   Good features allow for easy classification even with a simple classifier. Advanced classifier won't overcome weak features. It is better to spend more time on feature extraction than on classifier configuration.

11

	Structural difference	Functional difference
Shallow learning:	CAP depth <= 3	Features as inputs
Deep learning:	CAP depth > 3	Raw data as inputs

This distinction is more important...

Credit Assignment Path (CAP) *-> numer of consecutive nonlinear transformations of input data (neural network layers)*



12

So… What actually allows us learning path from raw data to high-level interpretation?

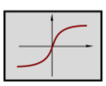
- **ReLU and Leaky ReLU activations**
- **Convolutional kernels**
- **Pooling**
- **Regularization**
- **Fine-tuning**

These are basic concepts – most of DL state-of-the-art engines use them in one form or another

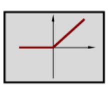
- Adversarial training
- Attention mechanisms
- Autoencoders
- Latent space
- Embeddings
- Reinforcement learning
- Transfer learning
- Transformers
- Recurrent neural networks (including Long-Short-Term-Memory)

These are more advanced and required to understand specialized applications, for instance **Midjourney** or **ChatGPT**

13



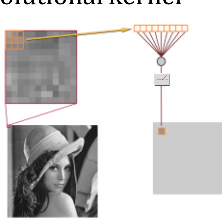
Sigmoid activation
+ works nice for small nets
- may cause gradient decay in large nets (rendering training to be inefficient)



Rectified Linear Unit (ReLU) activation
+ Combats gradient decay in large nets
+ Faster training
-/+ Can cause „neuron death" problem

14

Convolutional kernel



Provides:

*** Convolutional filtration**

*** Nonlinear filtration**

*** Morphological filtration**

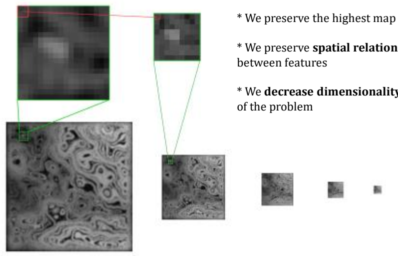
*** Extracts low-level features**
(recognition of „small objects")

* Thanks to ***weight sharing*** one net can do multiple tasks at once

* We can have **feature maps** as outputs

15

(Max) pooling



* We preserve the highest map values

* We preserve **spatial relations** between features

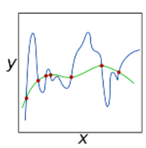
* We **decrease dimensionality** of the problem

16

Regularization

In order to prevent memorizing data we can use additional constraints – typically refering to admissible level of task complexity.

In practice we usually **iteratively slightly spoil the classifier**. Whenever generalization is achieved and the classifier begins fitting to noise, regularization factor starts do dominate over parameter-update routine.



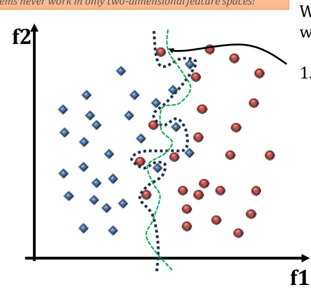
***For example**:*

• In gradient-based training, apart from weight update by the gradient-based policy we also randomly modify all or some weights

• Some net connections are deleted in-between net training cycles (a „brain damage" approach)

17

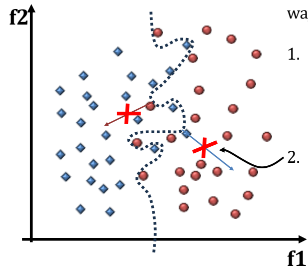
Graphical example – for explanation purposes only. Deep learning systems never work in only two-dimensional feature spaces!



We can stop this proces in two ways:

1. By iteratively adding random noise to weights – not allowing the algorithm to actually memorize data samples

18



We can stop this proces in two ways:

1. By iteratively adding random noise to weights – not allowing the algorithm to actually memorize data samples
2. By penalizing high weights (indirectly: not allowing high model complexity)

19

Fine tuning

In any case we'll need **a lot of data** and **large computing power**

In specialized cases there is often not enough data samples for training
It is often costly and time consuming to train a full model from scratch

Solution:
general training on general data, **fine-tuning** on the most relevant data

We do that generally by reinitializing weights of final layers of the neural network while freezing the remaining layers

20

Fine tuning - examples

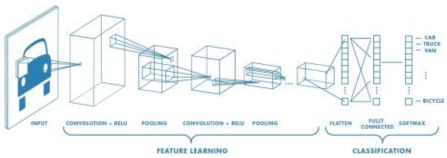
Train a Large Language Model to generate plausible text on a general text corpus (large text database) → Fine-tune it on texts by one author to allow mimicking his style

Train a Deep Neural Network for general object recognition → Fine-tune it on examples of two squirrel species to increase accuracy for a specific task

Train a Deep Neural Network for speech recognition → Fine-tune it on ultrasound inspection data to allow for distinguishing healthy from broken structures

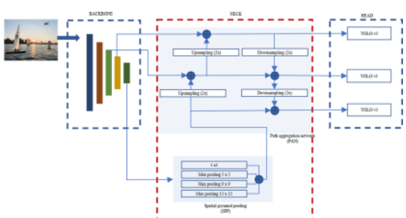
21

Examples: Matlab DCNN



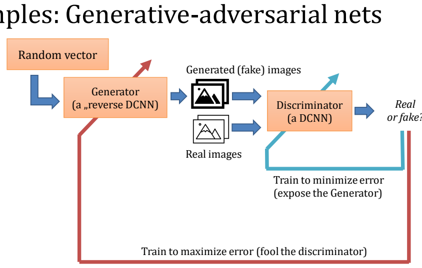
22

Examples: YOLO v4



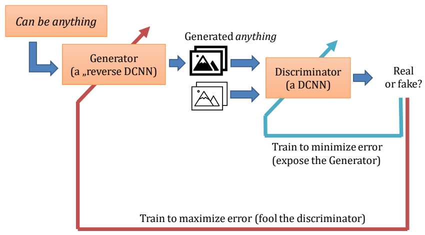
23

Examples: Generative-adversarial nets



24

Examples: Generative-adversarial nets



25

Summary (topics for test):

1) Examples of tasks in inteligent image processing
2) Processing path in deep and shallow learning (steps necessary, with example)
3) Functional and structural difference between shallow and deep learning
4) Differences between various MLP configurations
5) How can we optimize MLP structure? (Mean approach, box-plot approach)
6) Explain 4 general remarks for classifier training
7) Explain ReLU activation function, compare to sigmoid
8) Explain convolutional kernel
9) Explain max pooling
10) Explain regularization
11) Explain fine-tuning (with examples)

(*Detailed schemes of various deep learning systems won't be asked on tests*)

26