Don't believe the hype? A hands-on introduction to machine-learning in Python - Part III

Open Workshops on Computer Based Systems Modelling

Johannes Schmidt, Johann Baumgartner

Institute for Sustainable Economic Development, BOKU, Vienna

7.05.2019

Contents

- A word on the blackbox
- Pattern recognition with neural networks: some work in progress
- Q-Learning

A word on the blackbox

- Partial dependence plot: Average effect of one variable on output, over complete parameter space. Shows functional relationship between input and output. Assumption: inputs are all uncorrelated.
- Other methods are available (e.g. Accumulated Local Effects plot, Individual Conditional Expectation)
- LSTM-Vis: visualize hidden states of LSTMs. Interesting to understand, what the hidden states may represent. See also this for an example of what hidden states represent in hydrology.

Pattern recognition with neural networks: work in progress

- Validating the Global Power Plant Database with satellite fotos...
- Work in progress. Results not extremely promising. Recommendations very much welcome.

Global Power Plant Database: Need for Validation

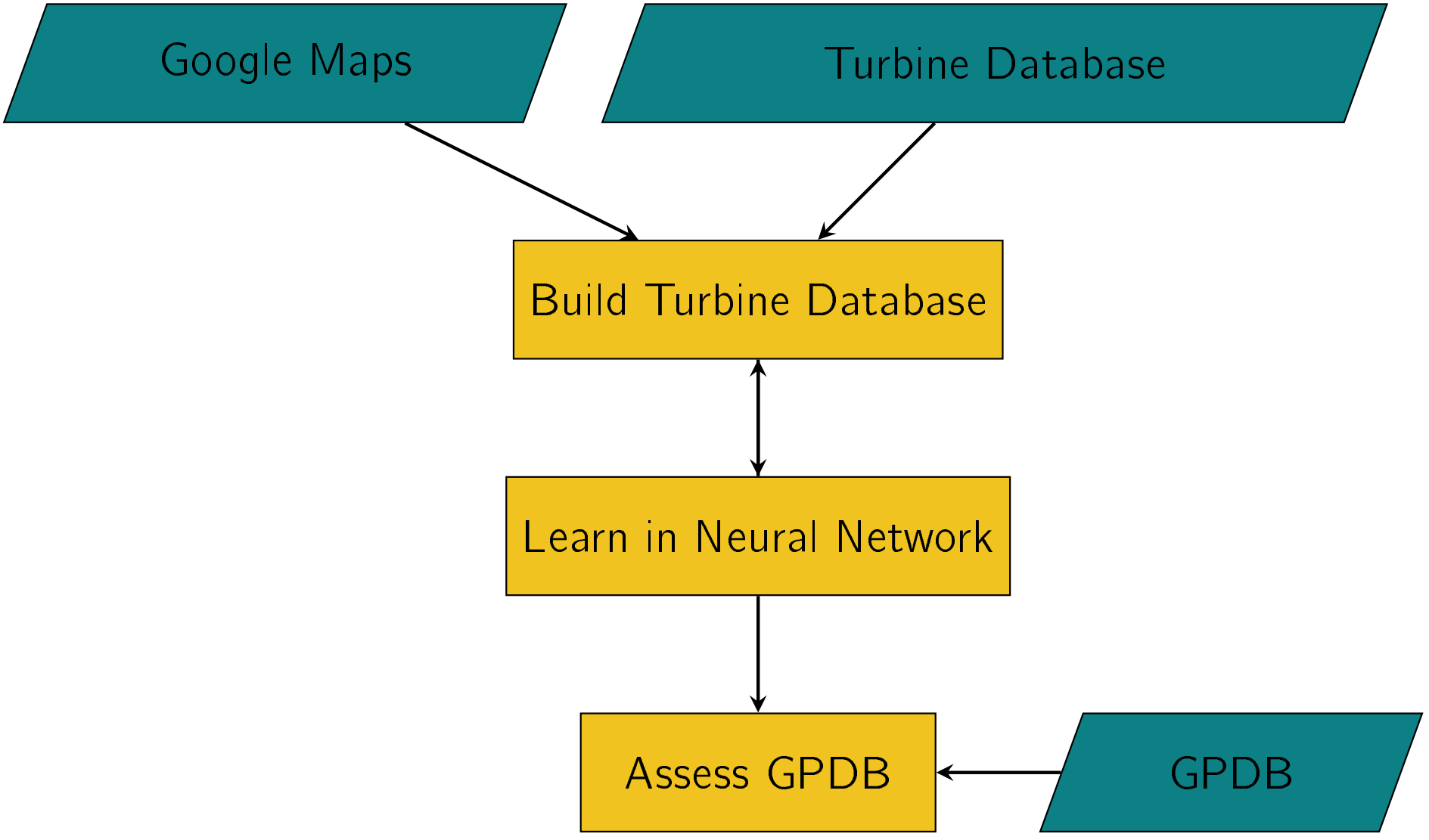
Satellite data for validation

Sentinel-2 (10 meter resolution), free

Satellite data for validation

WorldView-4, Quickbird (0.31-0.65 meter resolution) used by e.g. Google Maps

Approach


Google Maps
Turbine Database
Build Turbine Database
Learn in Neural Network
Assess GPDB
GPDB

Software

- Downloading of data from google maps: R (RGdal, tidyverse, raster)
- Machine Learning: Python (keras, scikit-image, gdal)
- Mixing R and Python: not a brilliant idea. Just lazy.

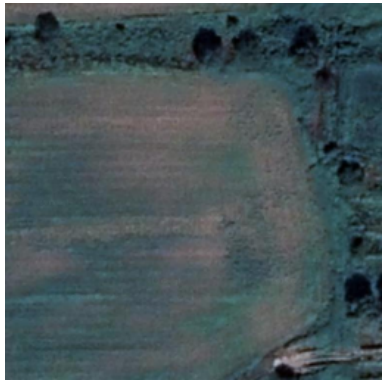
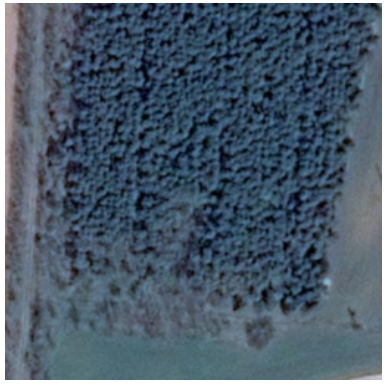
Create Sample Database

Manual quality control

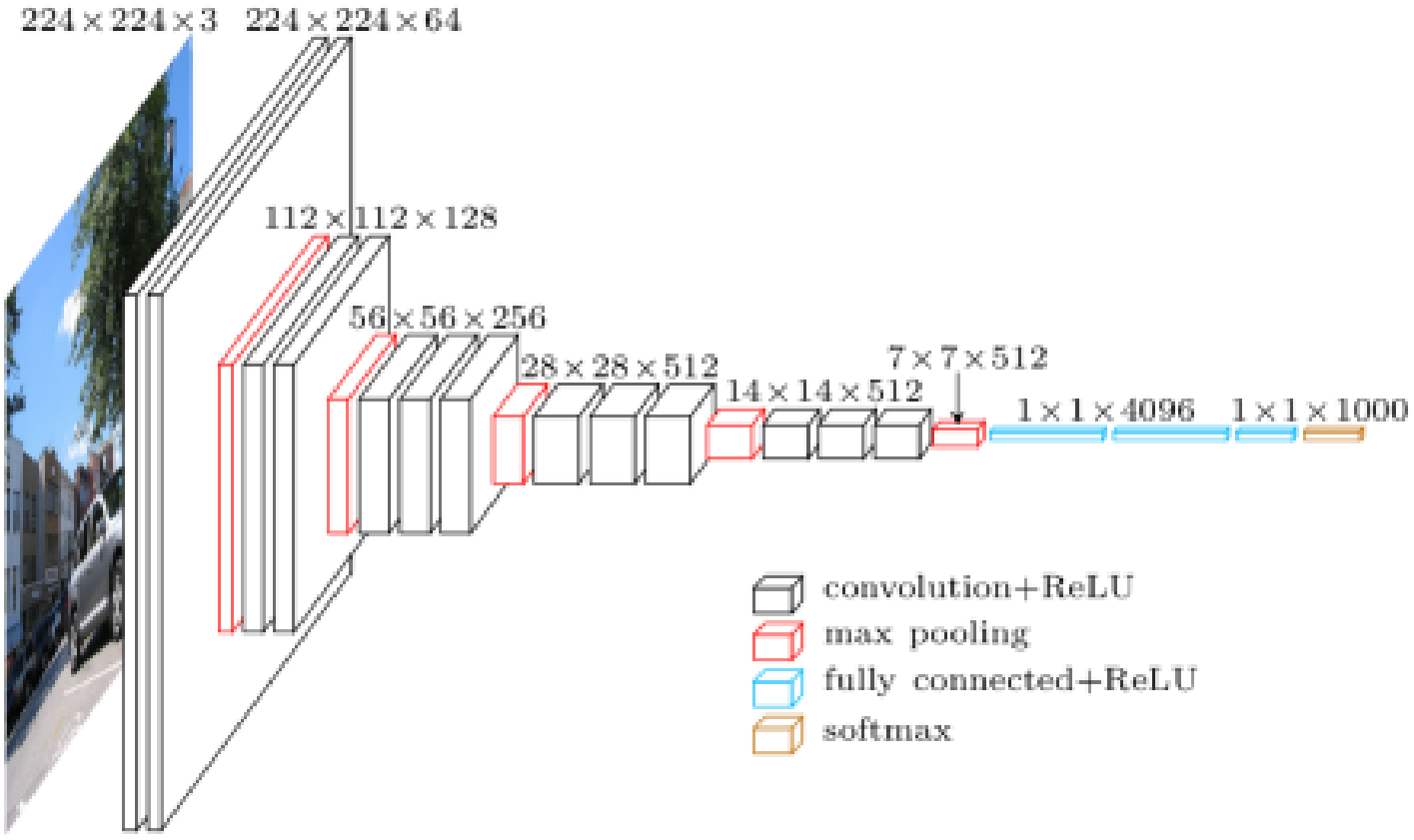
Positives




Negatives


Learn with Convolutional layers



Layered feature learning

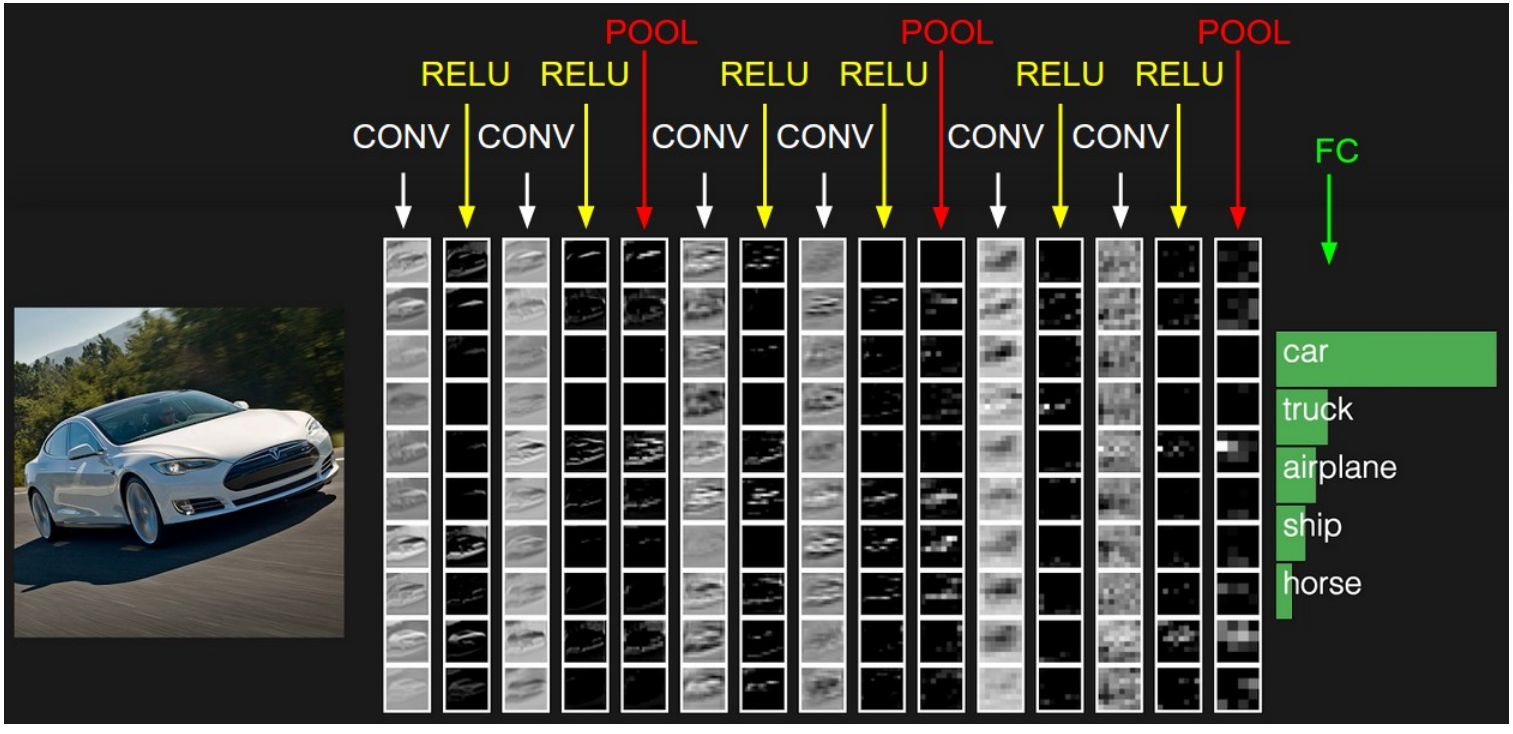


Figure: Taken from here

Terminology

- Convolution operation
- Filter
- Depth
- Stride
- Zero padding (wide and narrow convolution)

Pooling

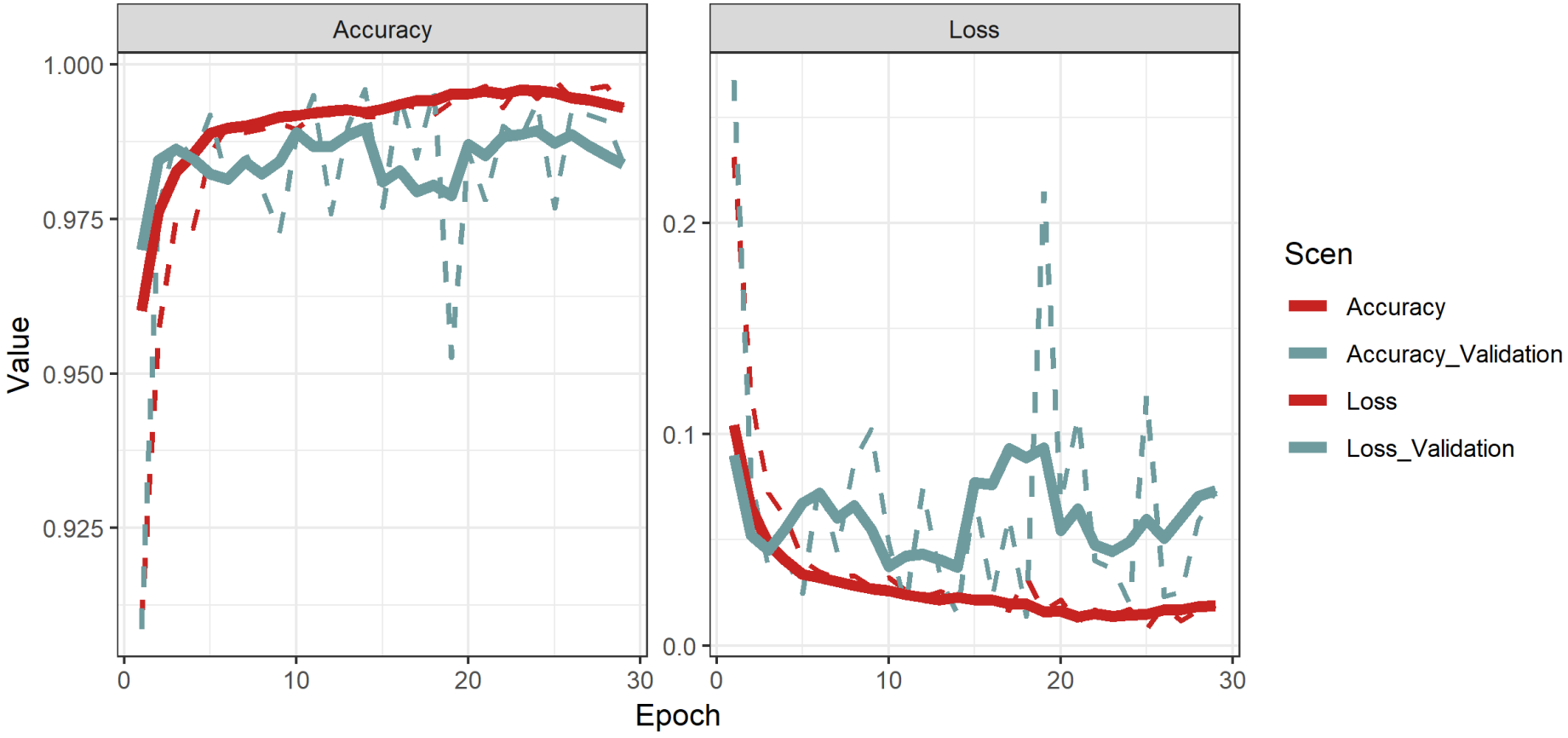
- Makes input representations (feature dimension) smaller and more manageable
- Reduces the number of parameters and computations in the network controlling overfitting
- Makes the network invariant to small transformations, distortions and translations in the input image
- Helps us arriving at an almost scale invariant representation of our image (the exact term is “equivariant”)
- taken from here

Some hints

- Compile high quality training set
- Do not overfit!
- Use pretrained networks to speed up learning process. Keras models can be found here
- Unfreeze last layers to learn your specific task, fixing the lower layers which extract basic features

Results Learning

Quality Measures



$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Number\ of\ Total\ Predictions}$$

$$Loss = -1\,\frac{1}{N}\sum_{i=1} Ny_i log(p(y_i)) + (1 - y_i) log(1 - p(y_i))$$

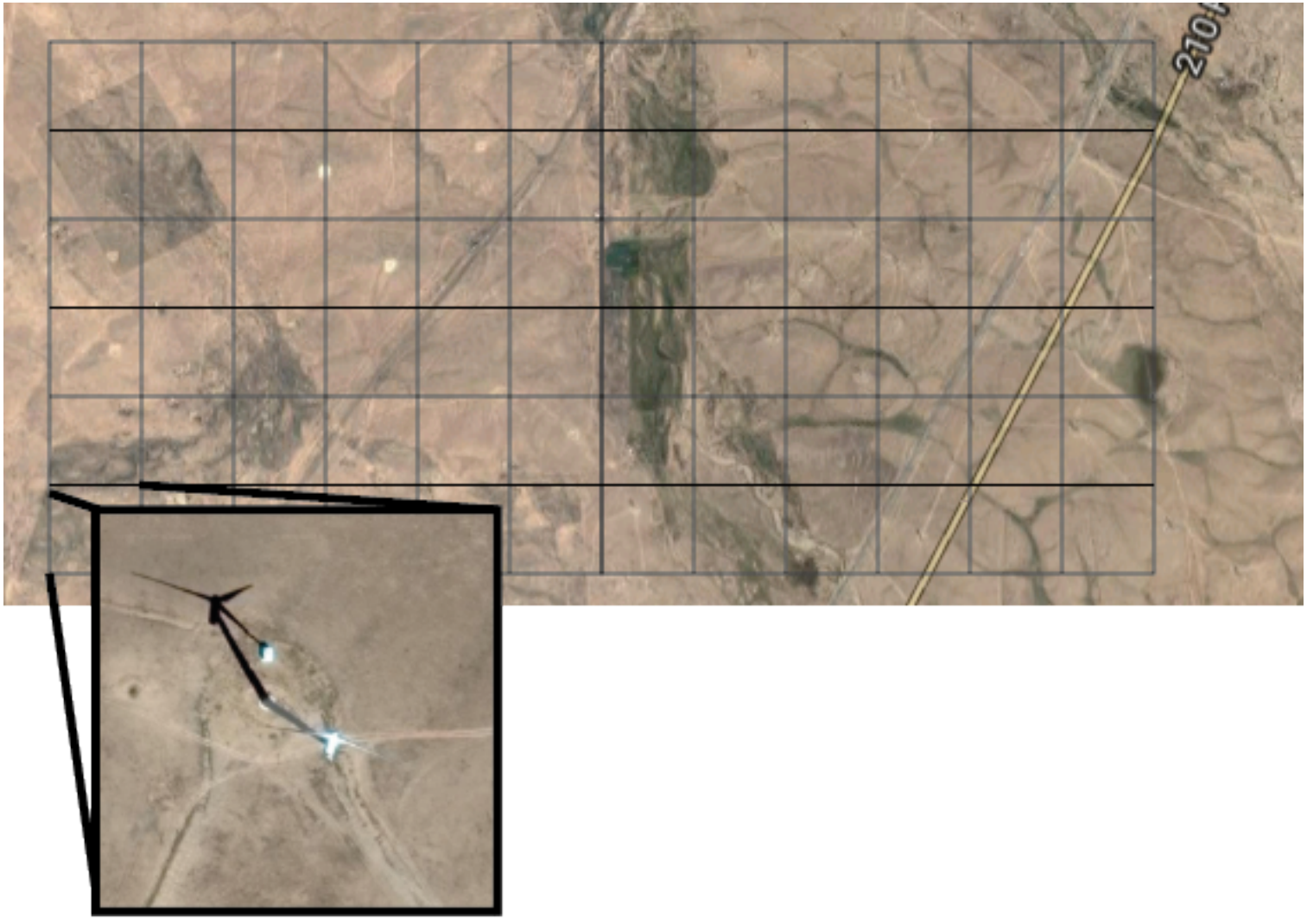
Results Learning

Class Activation Map

see notebook

Searching wind turbines

Create grid and check image by image



Application

French Wind Parks

France: 10 parks assessed.

Turbines correctly identified	Non-Turbines correctly identified	Turbines wrongly identified	Non-Turbines wrongly identified
6	3429	5	8

Discussion

- Beware: Shaky legal conditions. Also, date of satellite photos unknown.
- Runtime prohibitive on desktop computers. For a full global check would need cluster computing with high bandwidth (but see shaky legal conditions...)
- Improve classification: need for more negatives which are similiar to positives? How could we derive those? (Currently random sampling of negatives, therefore tiles do not contain a lot of cities or roads...)

Conclusions

- Full validation of GPDB due to missing temporal information not fully possible.
- Current best commercial satellite data allows identification of single turbines with good accuracy by using machine learning approaches
- Research limited by Data availability. Public domain data is low resolution (i.e. Sentinel-2) or small spatial domain (free ortho-photos, like basemmap.at).

And now for something completely different...

Contents

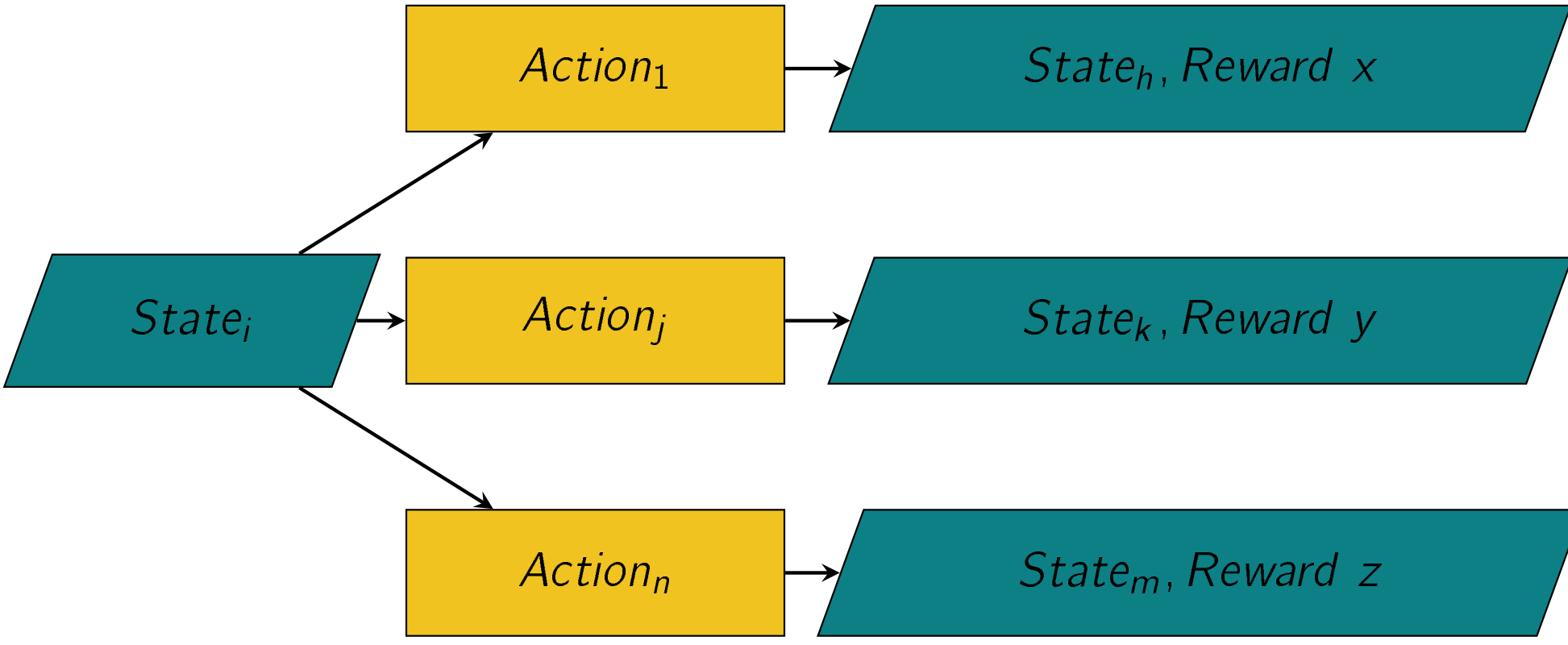
- The Q-learning algorithm
- A very simple example calculated by hand
- A simple example in python

Some examples

- Pan flipping robot
- The moment alphaGo wins against Lee Sedol
- Automatic recommendation systems

The Q-learning algorithm in a nutshell

Which action to pick?



The Q-Table

The Q-Table is updated in a learning by doing approach

Q-Updating rule introduced 1989 by Watson. Updates Q-Table according to current rewards and expected future rewards

$$q^{new}(s_t, a_t) = (1 - \alpha)q(s_t, a_t) + \alpha(r_t + \gamma \max_a q(s_{t+1}, a))$$

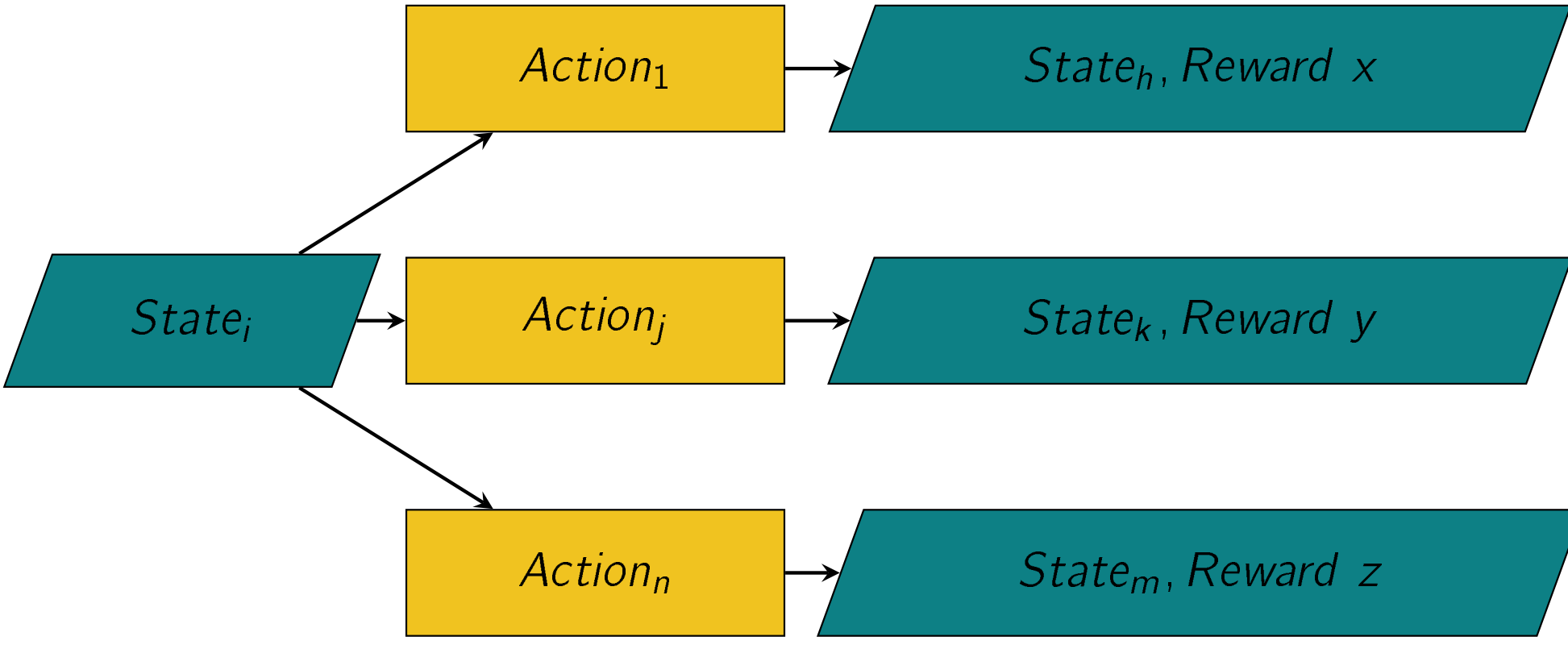
α is learning rate, γ is discount rate

r_t is reward, s_t is state, a_t is action

States	Actions			
	q_{s_1,a_1}	q_{s_1,a_2}	q_{s_1,a_3}	q_{s_1,a_4}
	q_{s_2,a_1}	q_{s_2,a_2}	q_{s_2,a_3}	q_{s_2,a_4}
	q_{s_3,a_1}	q_{s_3,a_2}	q_{s_3,a_3}	q_{s_3,a_4}
	q_{s_4,a_1}	q_{s_4,a_2}	q_{s_4,a_3}	q_{s_4,a_4}

The Q-learning algorithm in a nutshell

Which action to pick?



The Q-Table

The Q-Table is updated in a learning by doing approach

Q-Updating rule introduced 1989 by Watson. Updates Q-Table according to current rewards and expected future rewards

$$q^{new}(s_t, a_t) = (1 - \alpha)q(s_t, a_t) + \alpha(r_t + \gamma \max_a q(s_{t+1}, a))$$

α is learning rate, γ is discount rate

r_t is reward, s_t is state, a_t is action

States	Actions			
	q_{s_1,a_1}	q_{s_1,a_2}	q_{s_1,a_3}	q_{s_1,a_4}
	q_{s_2,a_1}	q_{s_2,a_2}	q_{s_2,a_3}	q_{s_2,a_4}
	q_{s_3,a_1}	q_{s_3,a_2}	q_{s_3,a_3}	q_{s_3,a_4}
	q_{s_4,a_1}	q_{s_4,a_2}	q_{s_4,a_3}	q_{s_4,a_4}

Choosing an action

- If random number $< \epsilon$, take a random action (exploration)
- Otherwise, take the action with the highest q-value (exploitation)

Examples

- A simple game on the blackboard
- Navigate a maze with q-learning

Thank you!

For slides and source-code, check
https://github.com/joph/Machine-Learning-Workshop
mail: johannes.schmidt@boku.ac.at