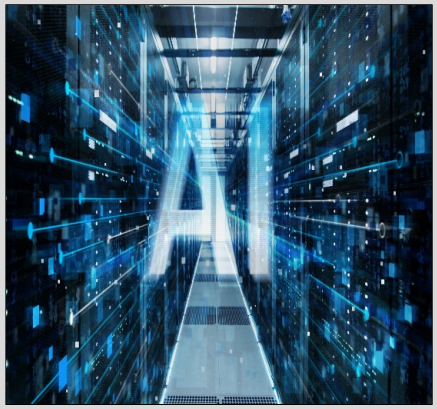


Artificial Intelligence Data Analysis (AIDA)

1st School for Heliophysicists

Prof. Dr. – Ing. Morris Riedel
Associated Professor
School of Engineering and Natural Sciences, University of Iceland, Reykjavik, Iceland
Research Group Leader, Juelich Supercomputing Centre, Forschungszentrum Juelich, Germany

@Morris Riedel @MorrisRiedel @MorrisRiedel

LECTURE 9

Performance & Tuning of Machine Learning

January 22, 2020
CINECA, Bologna, Italy

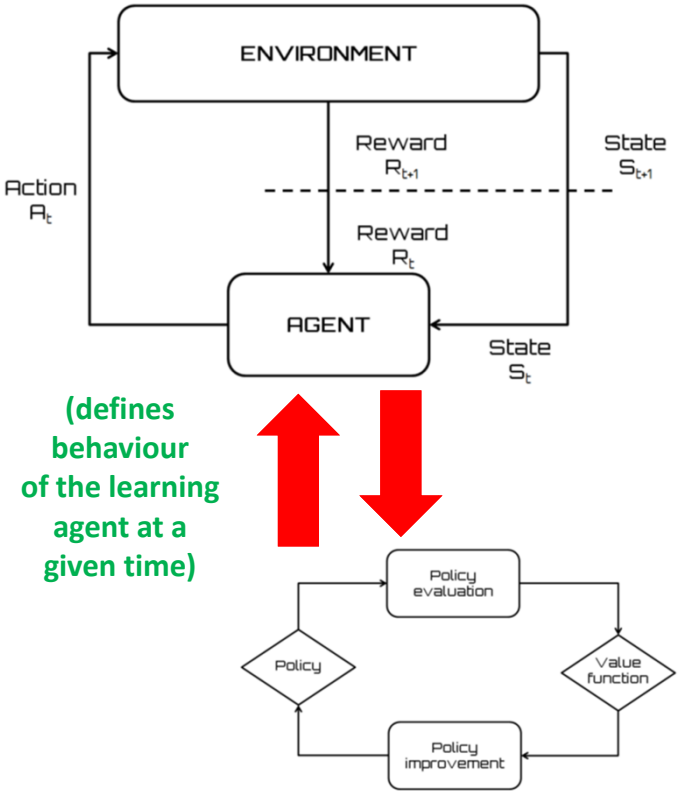
UNIVERSITY OF ICELAND
SCHOOL OF ENGINEERING AND NATURAL SCIENCES
FACULTY OF INDUSTRIAL ENGINEERING, MECHANICAL ENGINEERING AND COMPUTER SCIENCE

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

DEEP Projects

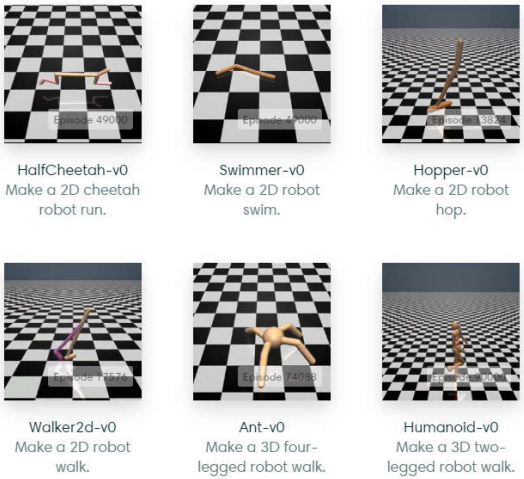
HELMHOLTZ AI | ARTIFICIAL INTELLIGENCE COOPERATION UNIT

Review of Lecture 8 – Reinforcement Learning – Short Introduction



[1] Keras Reinforcement Learning

```
import numpy as np
import gym

from keras.models import Sequential
from keras.layers import Dense, Activation, Flatten
from keras.optimizers import Adam

from rl.agents.dqn import DQNAgent
from rl.policy import BoltzmannQPolicy
from rl.memory import SequentialMemory

ENV_NAME = 'CartPole-v0'

env = gym.make(ENV_NAME)

nb_actions = env.action_space.n

model = Sequential()
model.add(Flatten(input_shape=(1,) + env.observation_space.shape))
model.add(Dense(16))
model.add(Activation('relu'))
model.add(Dense(16))
model.add(Activation('relu'))
model.add(Dense(16))
model.add(Activation('relu'))
model.add(Dense(nb_actions))
model.add(Activation('linear'))
print(model.summary())

memory = SequentialMemory(limit=50000, window_length=1)
policy = BoltzmannQPolicy()

dqn = DQNAgent(model=model, nb_actions=nb_actions, memory=memory,
               nb_steps_warmup=10,target_model_update=1e-2,
               policy=policy)

dqn.compile(Adam(lr=1e-3), metrics=['mae'])

dqn.fit(env, nb_steps=1000, visualize=True, verbose=2)

dqn.save_weights('dqn_{}_weights.h5f'.format(ENV_NAME), overwrite=True)

dqn.test(env, nb_episodes=5, visualize=True)
```



[2] OpenAI Gym Toolkit

Outline of the School

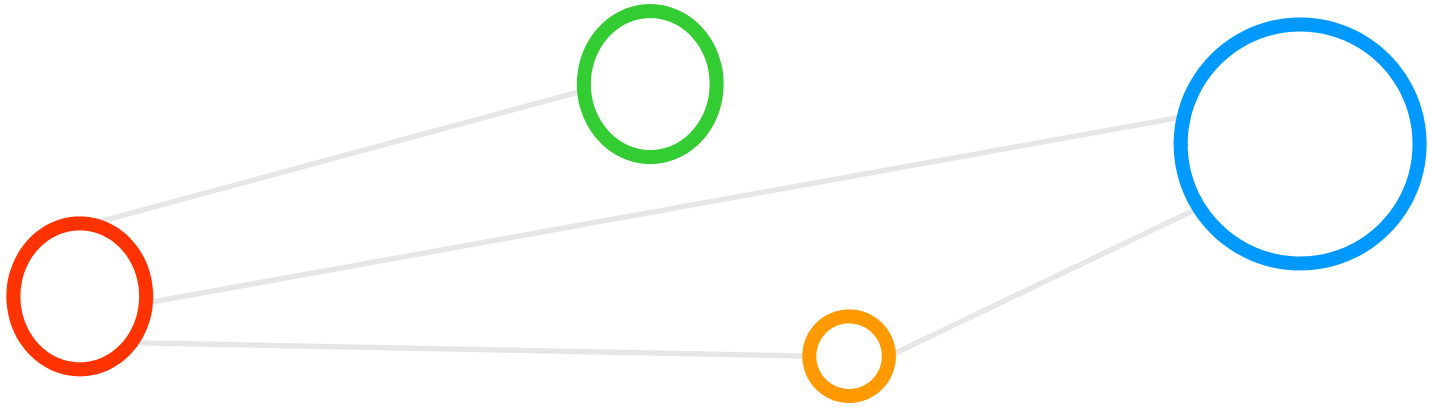
Time	**Day 1**	**Day 2**	**Day 3**
9 - 10	Welcome and intro to the school (Giovanni Lapenta, Jorge Amaya)	Space missions data acquisition (Hugo Breuillard)	Review of ML applied to heliophysics (Peter Wintoft)
10 - 11	Introduction and differences between AI, ML, NN and Big Data (Morris Riedel)	Data manipulation in python with pandas, xarray, and additional python tools (Geert Jan Bex)	Review of ML applied to heliophysics (Peter Wintoft)
	Coffee break	Coffee break	Coffee break
11:30 - 12:30	Unsupervised learning (Morris Riedel)	Feature engineering and data reduction (Geert Jan Bex)	Reinforcement learning (Morris Riedel)
	Lunch	Lunch	Lunch
14 - 15	Unsupervised learning (Morris Riedel)	Data reduction and visualization (Geert Jan Bex)	Physics informed ML (Romain Dupuis)
15 -16	Supervised learning (Morris Riedel)	CNN, DNN (Morris Riedel)	Explainable AI (Jorge Amaya)
	Coffee break	Coffee break	Coffee break
16:30 - 18:00	Supervised learning (Morris Riedel)	CNN, DNN (Morris Riedel)	Performance and tuning of ML (Morris Riedel)

Outline

- Performance & Tuning of Machine Learning
  - Overfitting & Validation Revisited & N-Fold Validation Approach
  - Traditional Gridsearch in 10x Cross-Validation Example
  - Application Examples of Remote Sensing in Context
  - Understanding Regularization Parameter Tunings & Data Contamination
  - Other Tunings with Evolutionary Algorithms & Particle Swarm Optimization

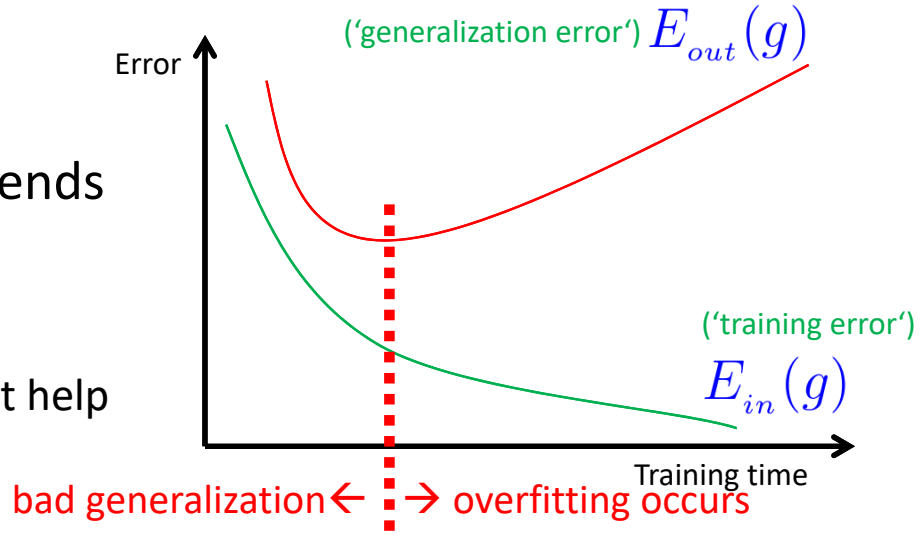
- (Semi-)Automated Machine Learning & Architecture Tuning
  - Key challenges & Motivations of Model Tunings Revisited
  - Neural Architecture Search (NAS) Approach
  - Reinforcement Learning Approach for NAS
  - Examples of Instance-Aware NAS
  - Environment & Trainings at the Juelich Supercomputing Centre

Performance & Tuning of Machine Learning



Problem of Overfitting – Clarifying Terms – Revisited

- Overfitting & Errors
  - $E_{in}(g)$ goes down
  - $E_{out}(g)$ goes up
- 'Bad generalization area' ends
  - Good to reduce $E_{in}(g)$
- 'Overfitting area' starts
  - Reducing $E_{in}(g)$ does not help
  - Reason 'fitting the noise'

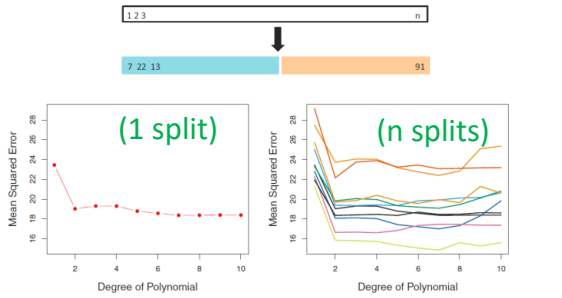


- **A good model must have low training error (E_{in}) and low generalization error (E_{out})**
- **Model overfitting is if a model fits the data too well (E_{in}) with a poorer generalization error (E_{out}) than another model with a higher training error (E_{in})**
- **The two general approaches to prevent overfitting are (1) validation and (2) regularization**

Validation & Model Selection – Terminology – Revisited

- ‘Training error‘
  - Calculated when learning from data (i.e. dedicated training set)
- ‘Test error’
  - Average error resulting from using the model with ‘new/unseen data‘
  - ‘new/unseen data‘ was not used in training (i.e. dedicated test set)
  - In many practical situations, a dedicated test set is not really available
- ‘Validation Set‘
  - Split data into training & validation set
- ‘Variance‘ & ‘Variability‘
  - Result in different random splits (right)

(split creates a two subsets of comparable size)



- **The ‘Validation technique‘ should be used in all machine learning or data mining approaches**
- **Model assessment is the process of evaluating a models performance**
- **Model selection is the process of selecting the proper level of flexibility for a model**

Validation Technique – Formalization & Goal – Revisited

- Regularization & Validation
  - Approach: introduce a 'overfit penalty' that relates to model complexity
  - Problem: Not accurate values: 'better smooth functions'

(regularization uses a term that captures the overfit penalty)

$$E_{out}(h) = E_{in}(h) + \textbf{overfit penalty}$$

(minimize both to be better proxy for E_{out})

(validation estimates this quantity)

(regularization estimates this quantity)

(measuring E_{out} is not possible as this is an unknown quantity, another quantity is needed that is measurable that at least estimates it)

- Validation
  - Goal 'estimate the out-of-sample error' (establish a quantity known as validation error)
  - Distinct activity from training and testing (testing also tries to estimate the E_{out})

- Validation is a very important technique to estimate the out-of-sample performance of a model
- Main utility of regularization & validation is to control or avoid overfitting via model selection

Validation Technique – Model Selection Process – Revisited

Hypothesis Set

$\mathcal{H} = \{h\};\ g \in \mathcal{H}$

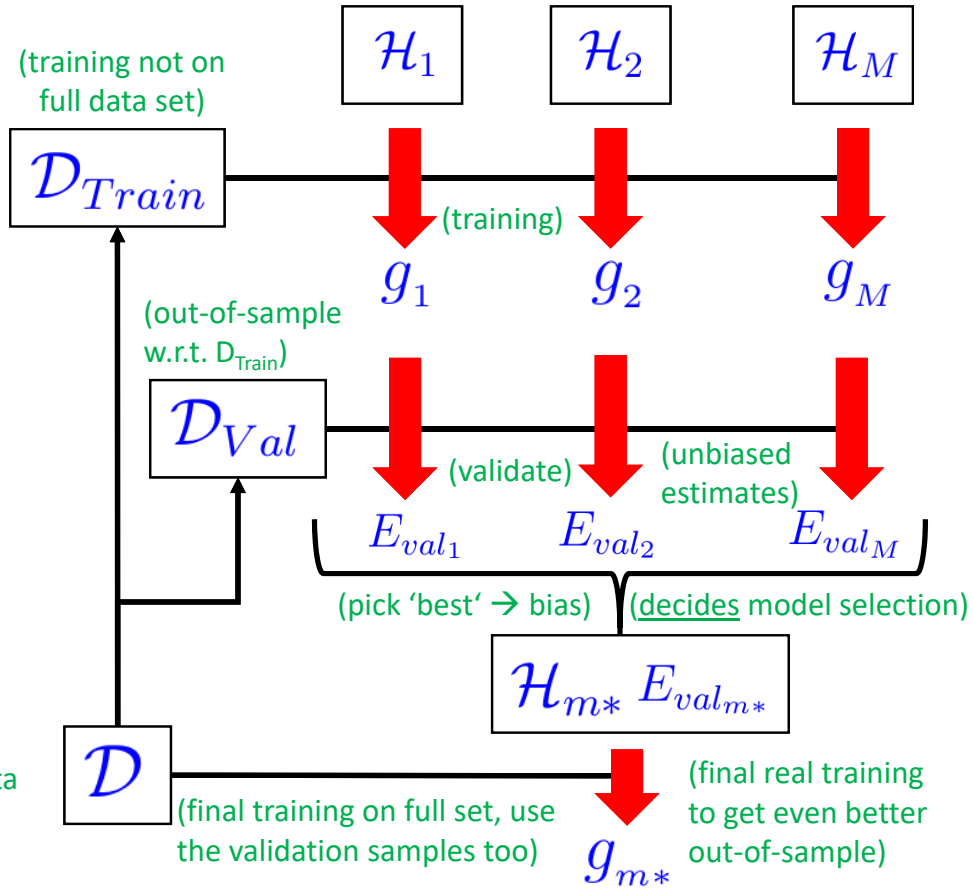
(set of candidate formulas across models)

- Many different models Use validation error to perform select decisions
- Careful consideration:
  - 'Picked means decided' hypothesis has already bias (→ contamination)
  - Using $\mathcal{D}_{Val}$ M times

Final Hypothesis

$g_{m*} \approx f$

(test this on unseen data good, but depends on availability in practice)



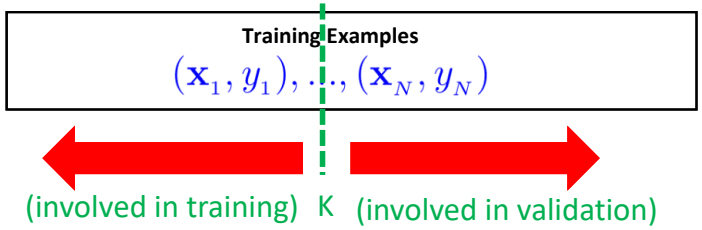
- **Model selection is choosing (a) different types of models or (b) parameter values inside models**
- **Model selection takes advantage of the validation error in order to decide → 'pick the best'**

Validation Technique – Cross-Validation – Trick

- **Cross-validation the technique of choice in practical situations to perform model selection**
- **Different techniques exist for cross-validation such as leave-one-out, leave-more-out**

(every time a data point is used for validation it is taken away from training)

- Goal (validation data not given on top of training data)
  - Target issue 'choosing K' out of N
  - Issue: K needs to be small & large
  - (cf. Lecture 1 feasibility of learning)



(conflicting requirements on K)

(chain of reasoning so far) $E_{out}(g) \approx E_{out}(g^-) \approx E_{val}(g^-)$ (idea: is there a solution over time?)

(small K for large N – K training delivering good out-of-sample performance)

(large K for validation delivering a good estimate for out-of-sample performance)

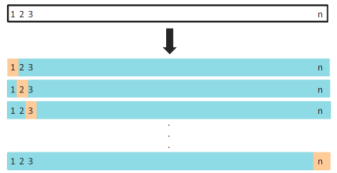
- Apply trick: repeat the number of trainings on different subsets
  - Train multiple times using e.g. leave-one-out or leave-more-out (practice)

- **Cross-validation 'trick' achieves to use N points for training and N points for validation (big gain!)**

Validation Technique – Cross-Validation – Leave-one-out

- Simplest form of cross-validation
  - Use N – 1 data points for training
  - Potential issue: only 1 data point for validation (bad estimate for E_{out})
  - Creates a very 'small validation' set, but very 'large training' set

(split data NOT just in two subsets of comparable size)



$$\mathcal{D}_n = (\mathbf{x}_1, y_1), (\mathbf{x}_{n-1}, y_{n-1}), (\mathbf{x}_n, y_n), (\mathbf{x}_{n+1}, y_{n+1}), \ldots, (\mathbf{x}_N, y_N)$$

(reduced dataset, but as training set very close to N missing just one)

(one data point left out for validation)

  - Final hypothesis to be 'selected' after training with $\mathcal{D}_n$ is g_n^-

(not trained on full set & depends on point left out)

(the hypothesis was trained not involving this point)

(check error on the point 'left out' → out-of-sample)

$$e_n = E_{val}(g_n^-) = e(g_n^-(\mathbf{x}_n), y_n)$$

(validate hypothesis on that data point taken out)

(one point in validation set brings bad estimate for E_{out})

- Apply 'resampling' trick: Repeat for different n with $\mathcal{D}_n$

$$E_{CV} = \frac{1}{N}\sum_{n=1}^{N} e_n$$

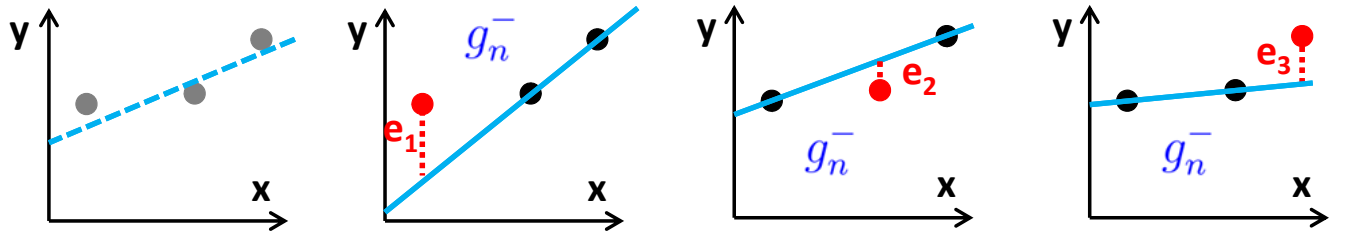
(works well with increasing N)

(cross-validation error on validation set with N points)

Validation Technique – Cross-Validation Error Example

- Example: Create a linear model
  - Assuming there is noise in the target function
  - Cross-validation: evaluate out-of-sample error to choose a model (later)

(red points are validation sets in each run) (black points are training sets in each run)



(the full dataset = 3 points) (n = left point out) (n = middle point out) (n = right point out)

$$E_{CV} = \frac{1}{N}\sum_{n=1}^{N} e_n$$

(simply compute average of all errors, e.g. using squared distance)

$$E_{CV} = \frac{1}{3}(e_1 + e_2 + e_3)$$

(cross-validation error as indication of how well 'the linear model' fits the data → out-of-sample)

(impact on N = small (3) is enormous, but if N = large average works very well)

- **Cross-validation is a 'resampling method' that obtains more information than 'fitting model once'**
- **Compute cross-validation error is possible (via 'in-sample') & a systematic way for model selection**

Validation Technique – Cross-Validation & Model Selection

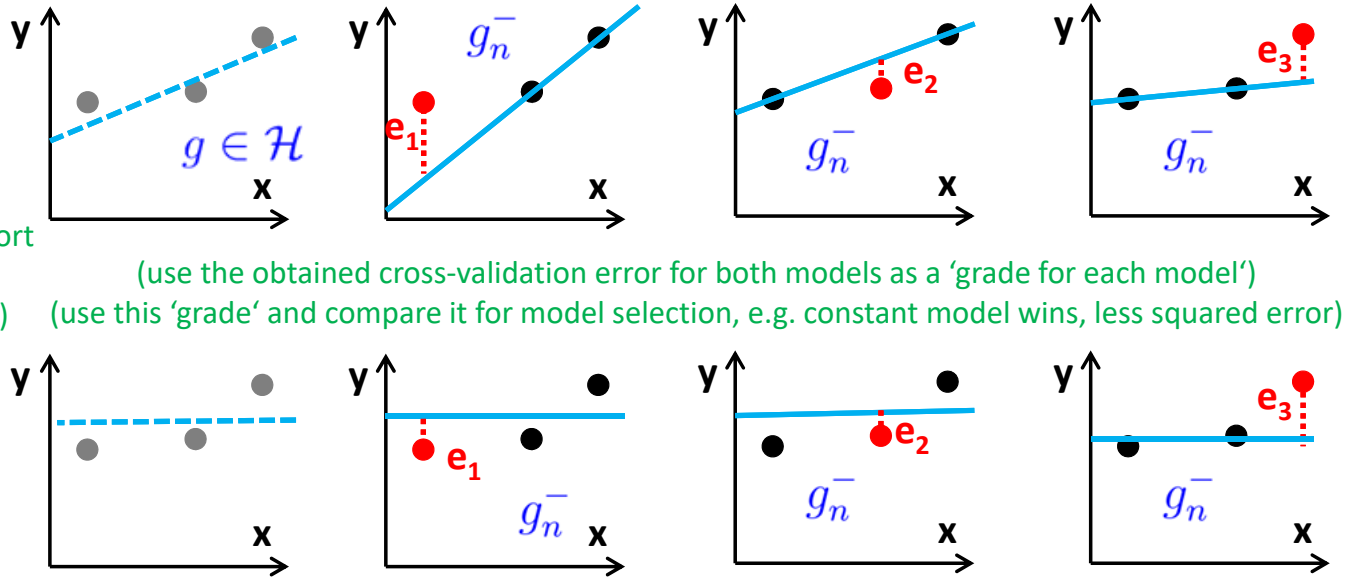
- Model selection: Perform a ‘decision‘
  - Cross-validation: evaluate out-of-sample error to choose a model (avoiding e.g. heuristics)
  - Example: Decide whether Linear Model or Constant Model is better

Hypothesis Set
$\mathcal{H} = \{h\};\ g \in \mathcal{H}$
(set of candidate formulas)

$\mathcal{H} = \{h\};\ g \in \mathcal{H}$
(LINEAR MODEL)

(decision support required, think N huge)

$\mathcal{H} = \{h\};\ g \in \mathcal{H}$
(CONSTANT MODEL)



(use the obtained cross-validation error for both models as a ‘grade for each model‘)
(use this ‘grade‘ and compare it for model selection, e.g. constant model wins, less squared error)

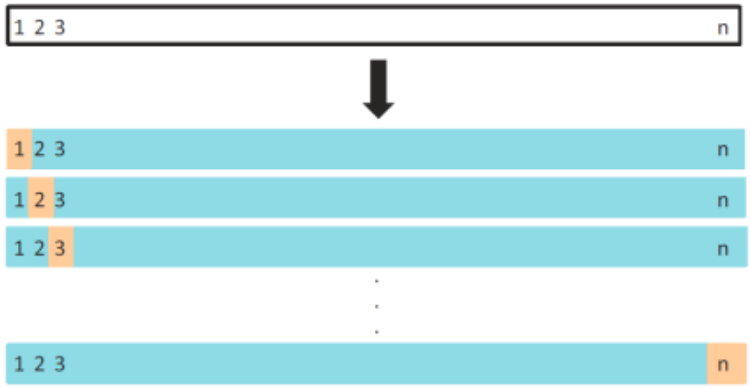
- **Main utility of cross-validation is model selection supporting a decision to choose a model**

Validation Technique – Cross-Validation – K-Fold Approach

(random strategy, works not particularly well)

(split creates a two subsets of comparable size)

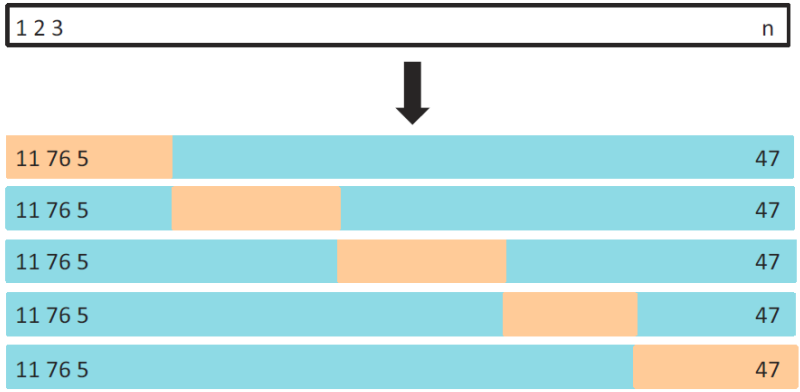
Leave-One-Out Cross-Validation (LOOCV) Example



(picking strategy, works well but possible long computing)

pick one point for validation
resulting in possible large
sets when number of points are high

5-fold Cross-Validation Example



(picking strategy, works well and reduces computing)

A set of data points is randomly split into
k non-overlapping groups ('k-folds')
of approximately equal size

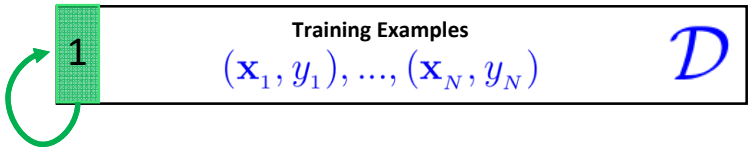
[3] Introduction to Statistical Learning

Recommendation in Practice

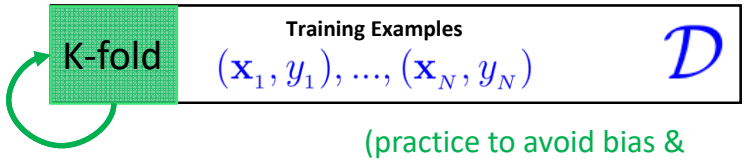
Validation Technique – Cross-Validation – Leave-more-out

- Leave-one-out
  - N training sessions on N – 1 points each time
- Leave-more-out
  - Break data into number of folds
  - N/K training sessions on N – K points each time (fewer training sessions than above)
  - Example: '10-fold cross-valdation' with K = N/10 multiple times (N/K)

(leave 1 point out at each run → many runs)

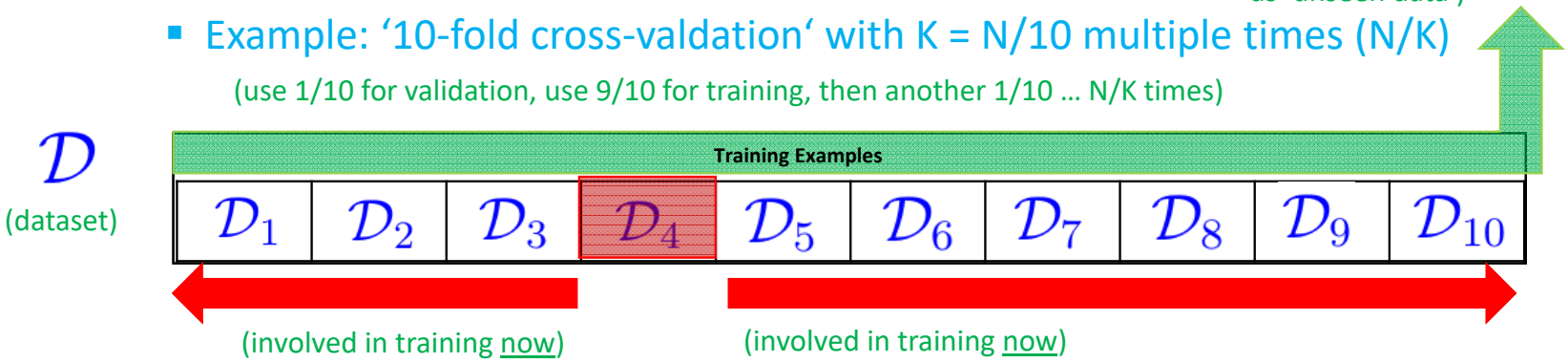


(generalization to leave k points out at each run)



(practice to avoid bias & contamination: some rest for test as 'unseen data')

(use 1/10 for validation, use 9/10 for training, then another 1/10 … N/K times)

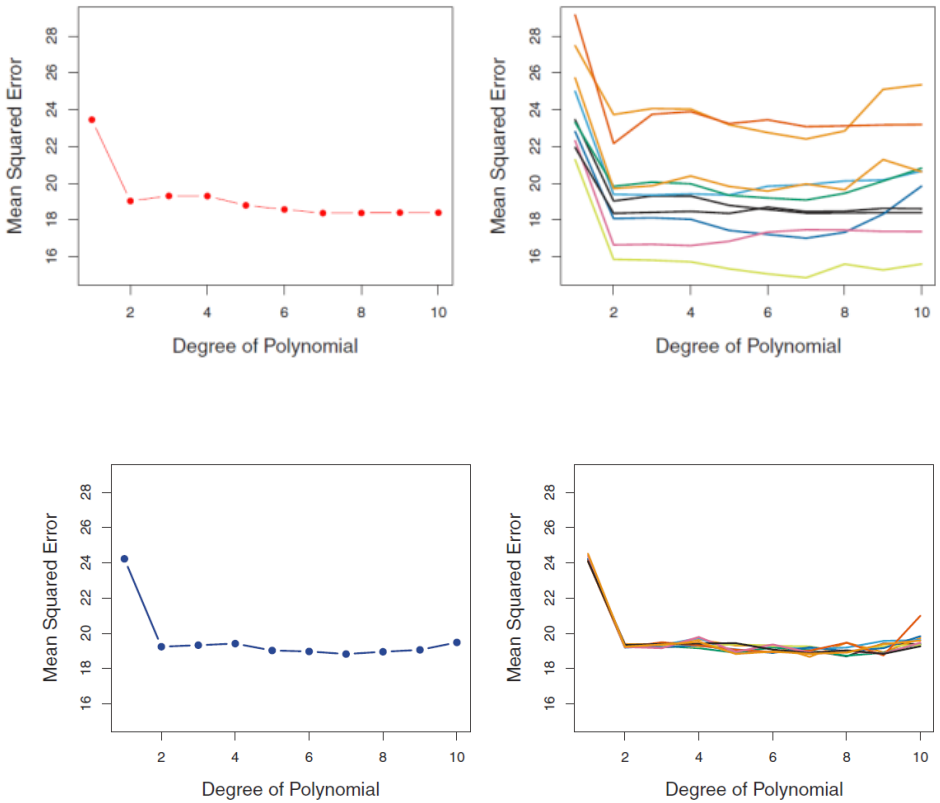


- **10-fold cross validation is mostly applied in practical problems by setting K = N/10 for real data**
- **Having N/K training sessions on N – K points each leads to long runtimes (→ use parallelization)**

Validation Technique – 10 fold Cross-Validation Example

- 10 times resampling
  - Validation set with 10 x 2 comparable sizes
  - ‘Random splits‘
  - High variability/variance

- 10-fold cross validation
  - Validation set with 10-fold x 2 strategy
  - No ‘random splits‘
  - Lower variability/variance



modified from [3] Introduction to Statistical Learning

Term Support Vector Machines

- Term detailed refinement into ‘three separate techniques’
  - Practice: applications mostly use the SVMs with kernel methods
- ‘Maximal margin classifier‘
  - A simple and intuitive classifier with a ‘best‘ linear class boundary
  - Requires that data is ‘linearly separable‘
- ‘Support Vector Classifier‘
  - Extension to the maximal margin classifier for non-linearly seperable data
  - Applied to a broader range of cases, idea of ‘allowing some error‘
- ‘Support Vector Machines‘ → Using Non-Linear Kernel Methods
  - Extension of the support vector classifier
  - Enables non-linear class boundaries & via kernels;

- **Support Vector Machines (SVMs) are a classification technique developed ~1990**
- **SVMs perform well in many settings & are considered as one of the best ‘out of the box classifiers‘**

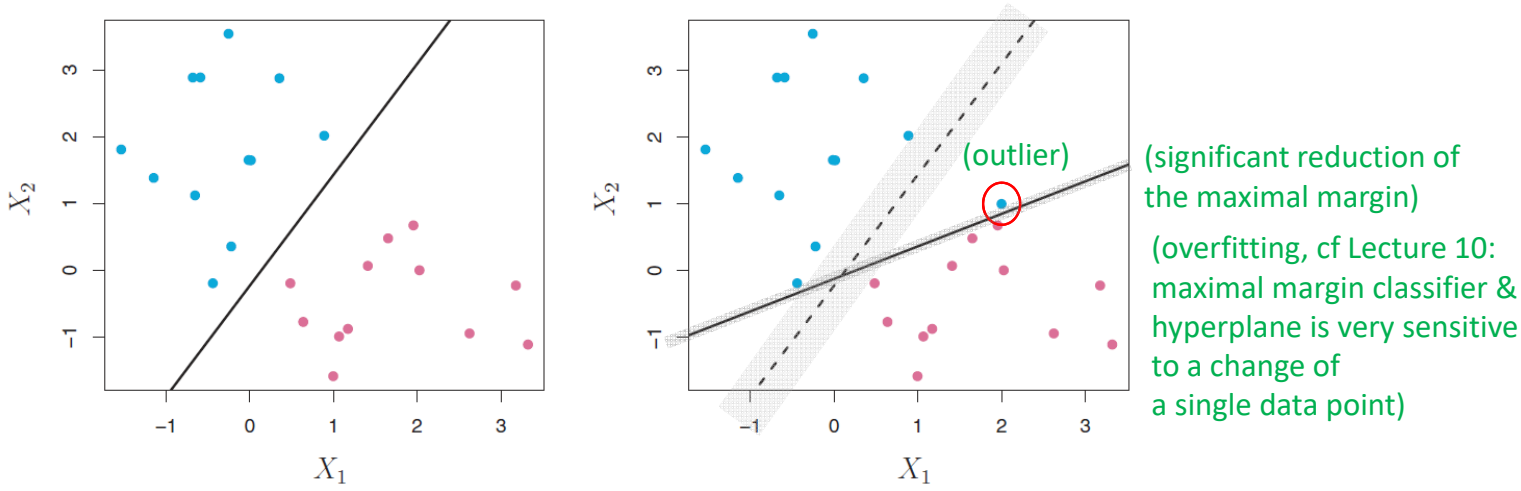
[3] An Introduction to Statistical Learning

Support Vector Classifiers – Motivation in Context Validation & Regularization

- **Support Vector Classifiers develop hyperplanes with soft-margins to achieve better performance**
- **Support Vector Classifiers aim to avoid being sensitive to individual observations (e.g. outliers)**

[3] An Introduction to Statistical Learning

- Approach
  - Generalization of the ‘maximal margin classifier‘ (get most & but not all training data correcly classified)
  - Include non-seperable cases with a soft-margin (almost instead of exact)
  - Being more robust w.r.t. extreme values (e.g. outliers) allowing small errors



(significant reduction of the maximal margin)

(overfitting, cf Lecture 10: maximal margin classifier & hyperplane is very sensitive to a change of a single data point)

Support Vector Classifier – Impact of Regularization with C

- Approach: Maximizing the margin
  - Equivalent to minimize objective function

$$\min_{w,} \left\{ \frac{1}{2} \| \mathbf{w} \|^2 \right\}$$

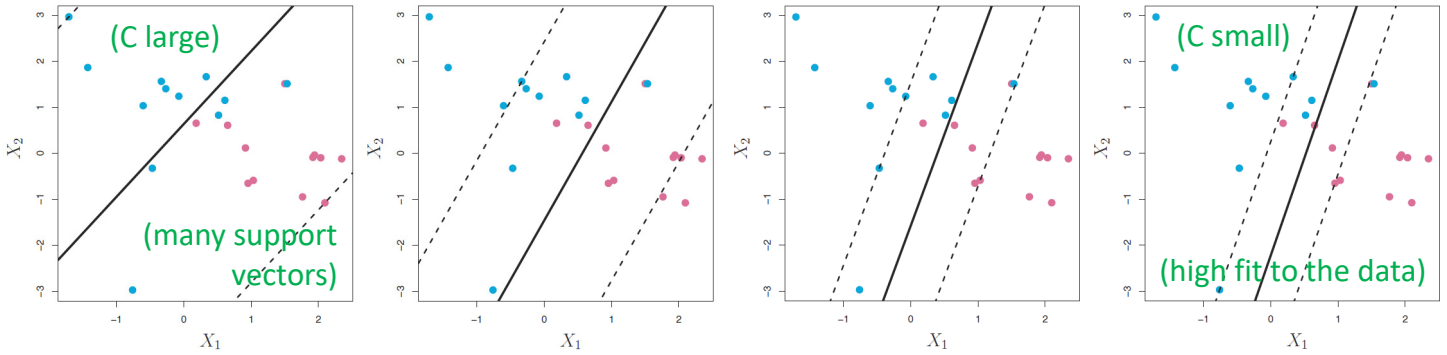
(maximal margin classifier)

- Support Vector Classifier
  - Same approach for solving
  - Adding slacks variables
  - C parameter that enables regularization (i.e. size of margin)

$$\min_{w,\xi_i} \left\{ \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_i \xi_i \right\}$$

(C is used here to multiply the sum of errors for increased severity, because # errors should be kept small, not a budget here)

[3] An Introduction to Statistical Learning



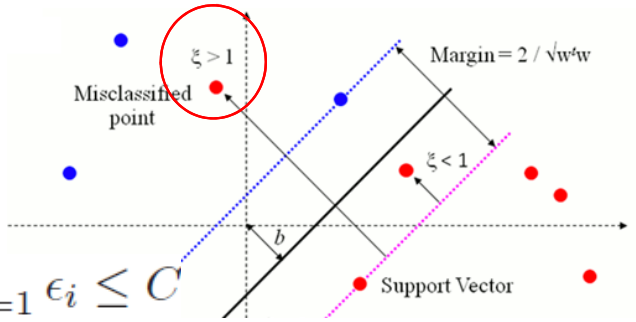
(fewer support Vectors: just #8)

Regularization Revisited & Kernel Parameter Gamma

- C = 0 (too rectrictive, potentially bad for E_{out})
  - No budget/costs for violations: comparable to maximal margin classifier
  - Further constraint: only works in linearly seperable cases (less in practice)

$$\epsilon_1 = \ldots = \epsilon_n = 0$$

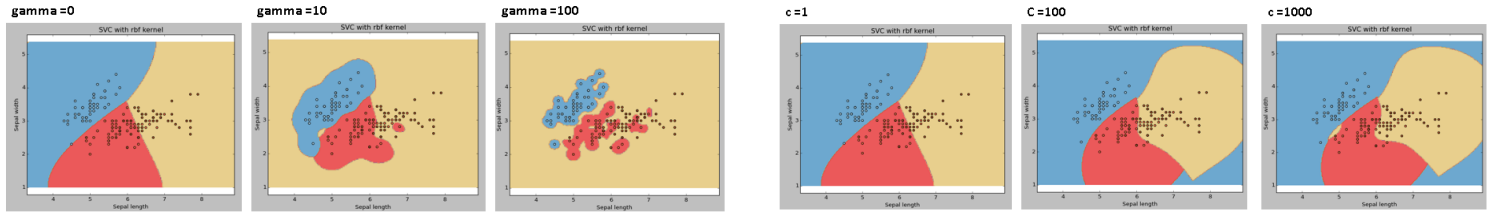
- C > 0 (flexible option, better for E_{out})
  - No more than C data points can be on the wrong side of the hyperplane ('how much misclassifications allowed')
  - Reasoning: if an observation is on the wrong side then $\epsilon_i > 1$



$$\sum_{i=1}^{n} \epsilon_i \leq C$$

- Kernel Parameter Gamma
  - RBF Kernel element, example Iris Dataset

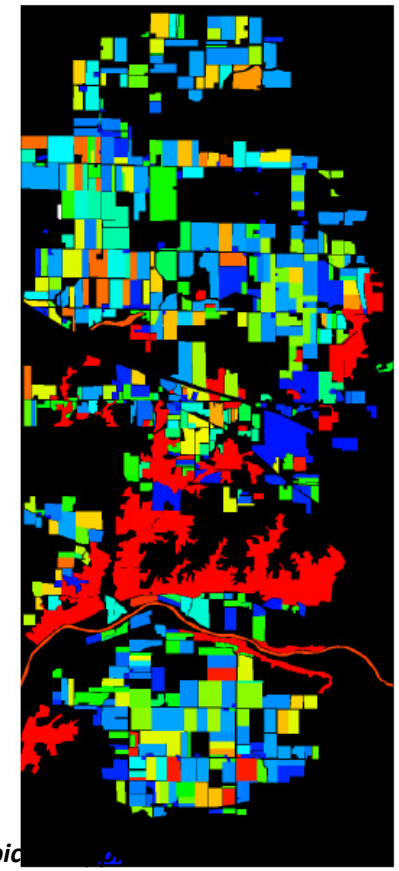
[6] SVM and Parameters



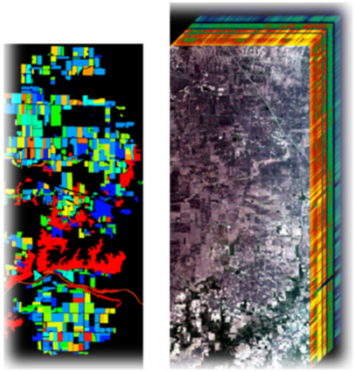
Remote Sensing Application Example using SWM – Dataset

- 1417×617 pixels (∼600 MB)
- 200 bands (20 discarded, with low SNR)
- 58 classes (6 discarded, with ≤ 100 samples)

Class		Number of samples		Class		Number of samples	
number	name	training	test	number	name	training	test
1	Buildings	1720	15475	27	Pasture	1039	9347
2	Corn	1778	16005	28	pond	10	92
3	Corn?	16	142	29	Soybeans	939	8452
4	Corn-EW	51	463	30	Soybeans?	89	805
5	Corn-NS	236	2120	31	Soybeans-NS	111	999
6	Corn-CleanTill	1240	11164	32	Soybeans-CleanTill	507	4567
7	Corn-CleanTill-EW	2649	23837	33	Soybeans-CleanTill?	273	2453
8	Corn-CleanTill-NS	3968	35710	34	Soybeans-CleanTill-EW	1180	10622
9	Corn-CleanTill-NS-Irrigated	80	720	35	Soybeans-CleanTill-NS	1039	9348
10	Corn-CleanTilled-NS?	173	1555	36	Soybeans-CleanTill-Drilled	224	2018
11	Corn-MinTill	105	944	37	Soybeans-CleanTill-Weedy	54	489
12	Corn-MinTill-EW	563	5066	38	Soybeans-Drilled	1512	13606
13	Corn-MinTill-NS	886	7976	39	Soybeans-MinTill	267	2400
14	Corn-NoTill	438	3943	40	Soybeans-MinTill-EW	183	1649
15	Corn-NoTill-EW	121	1085	41	Soybeans-MinTill-Drilled	810	7288
16	Corn-NoTill-NS	569	5116	42	Soybeans-MinTill-NS	495	4458
17	Fescue	11	103	43	Soybeans-NoTill	216	1941
18	Grass	115	1032	44	Soybeans-NoTill-EW	253	2280
19	Grass/Trees	233	2098	45	Soybeans-NoTill-NS	93	836
20	Hay	113	1015	46	Soybeans-NoTill-Drilled	873	7858
21	Hay?	219	1966	47	Swampy Area	58	525
22	Hay-Alfalfa	226	2032	48	River	311	2799
23	Lake	22	202	49	Trees?	58	522
24	NotCropped	194	1746	50	Wheat	498	4481
25	Oats	174	1568	51	Woods	6356	57206
26	Oats?	34	301	52	Woods?	14	130



(non-linearly separable) dataset



[5] G. Cavallaro, M. Riedel, J.A. Benediktsson et al., Journal of Selected Topic
Earth Observation and Remote Sensing, 2015

Parallelization Benefit: Parallel 10-Fold Cross-Validation 'GridSearch'

- Example: 2 Parameters, 10-fold cross-validation
  - 2 x benefits of parallelization possible in a so-called 'gridsearch'
  - (1) Compute parallel; (2) Do all cross-validation runs in parallel (all cells)
  - Evaluation between Matlab (aka 'serial laptop') & parallel (80 cores)

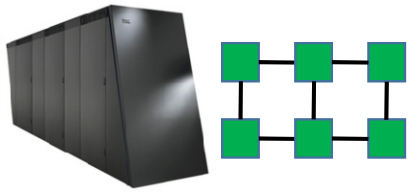
- **10-fold cross-validation achieves parallelization benefits (1) in each grid cell and (2) across all cells**

(2) Scenario 'pre-processed data', 10xCV **serial**: accuracy (min)

γ/C	1	10	100	1000	10 000
2	48.90 (18.81)	65.01 (19.57)	73.21 (20.11)	75.55 (22.53)	74.42 (21.21)
4	57.53 (16.82)	70.74 (13.94)	75.94 (13.53)	76.04 (14.04)	74.06 (15.55)
8	64.18 (18.30)	74.45 (15.04)	**77.00** (14.41)	75.78 (14.65)	74.58 (14.92)
16	68.37 (23.21)	76.20 (21.88)	76.51 (20.69)	75.32 (19.60)	74.72 (19.66)
32	70.17 (34.45)	75.48 (34.76)	74.88 (34.05)	74.08 (34.03)	73.84 (38.78)

(1) First Result: best parameter set from 14.41 min to 1.02 min

(2) Second Result: all parameter sets from ~9 hours to ~35 min



(2) Scenario 'pre-processed data', 10xCV **parallel**: accuracy (min)

γ/C	1	10	100	1000	10 000
2	75.26 (1.02)	65.12 (1.03)	73.18 (1.33)	75.76 (2.35)	74.53 (4.40)
4	57.60 (1.03)	70.88 (1.02)	75.87 (1.03)	76.01 (1.33)	74.06 (2.35)
8	64.17 (1.02)	74.52 (1.03)	**77.02 (1.02)**	75.79 (1.04)	74.42 (1.34)
16	68.57 (1.33)	76.07 (1.33)	76.40 (1.34)	75.26 (1.05)	74.53 (1.34)
32	70.21 (1.33)	75.38 (1.34)	74.69 (1.34)	73.91 (1.47)	73.73 (1.33)

'(1) each cell inherent parallel'

'(2) all cells in parallel'

[5] G. Cavallaro, M. Riedel, J.A. Benediktsson et al., Journal of Selected Topics in Applied Earth Observation and Remote Sensing, 2015

Parallelization Summary – aka ‘GridSearch‘ for Model Tuning

- Parallelization benefits are enormous for complex problems
  - Enables feasibility to tackle extremely large datasets & high dimensions
  - Provides functionality for a high number of classes (e.g. #k SVMs)
  - Achieves a massive reduction in time → lower time-to-solution

(1) Scenario ‘unprocessed data‘, 10xCV **serial**: accuracy (min)

γ/C	1	10	100	1000	10 000
2	27.30 (109.78)	34.59 (124.46)	39.05 (107.85)	37.38 (116.29)	37.20 (121.51)
4	29.24 (98.18)	37.75 (85.31)	38.91 (113.87)	38.36 (119.12)	38.36 (118.98)
8	31.31 (109.95)	**39.68** (118.28)	39.06 (112.99)	39.06 (190.72)	39.06 (872.27)
16	33.37 (126.14)	39.46 (171.11)	39.19 (206.66)	39.19 (181.82)	39.19 (146.98)
32	34.61 (179.04)	38.37 (202.30)	38.37 (231.10)	38.37 (240.36)	38.37 (278.02)

(2) Scenario ‘pre-processed data‘, 10xCV **serial**: accuracy (min)

γ/C	1	10	100	1000	10 000
2	48.90 (18.81)	65.01 (19.57)	73.21 (20.11)	75.55 (22.53)	74.42 (21.21)
4	57.53 (16.82)	70.74 (13.94)	75.94 (13.53)	76.04 (14.04)	74.06 (15.55)
8	64.18 (18.30)	74.45 (15.04)	**77.00** (14.41)	75.78 (14.65)	74.58 (14.92)
16	68.37 (23.21)	76.20 (21.88)	76.51 (20.69)	75.32 (19.60)	74.72 (19.66)
32	70.17 (34.45)	75.48 (34.76)	74.88 (34.05)	74.08 (34.03)	73.84 (38.78)

(1) Scenario ‘unprocessed data‘‘10xCV **parallel**: accuracy (min)

γ/C	1	10	100	1000	10 000
2	27.26 (3.38)	34.49 (3.35)	39.16 (5.35)	37.56 (11.46)	37.57 (13.02)
4	29.12 (3.34)	37.58 (3.38)	38.91 (6.02)	38.43 (7.47)	38.43 (7.47)
8	31.24 (3.38)	**39.77 (4.09)**	39.14 (5.45)	39.14 (5.42)	39.14 (5.43)
16	33.36 (4.09)	39.61 (4.56)	39.25 (5.06)	39.25 (5.27)	39.25 (5.10)
32	34.61 (5.13)	38.37 (5.30)	38.36 (5.43)	38.36 (5.49)	38.36 (5.28)

First Result: best parameter set from 118.28 min to 4.09 min
Second Result: all parameter sets from ~3 days to ~2 hours

(2) Scenario ‘pre-processed data‘, 10xCV **parallel**: accuracy (min)

γ/C	1	10	100	1000	10 000
2	75.26 (1.02)	65.12 (1.03)	73.18 (1.33)	75.76 (2.35)	74.53 (4.40)
4	57.60 (1.03)	70.88 (1.02)	75.87 (1.03)	76.01 (1.33)	74.06 (2.35)
8	64.17 (1.02)	74.52 (1.03)	**77.02 (1.02)**	75.79 (1.04)	74.42 (1.34)
16	68.57 (1.33)	76.07 (1.33)	76.40 (1.34)	75.26 (1.05)	74.53 (1.34)
32	70.21 (1.33)	75.38 (1.34)	74.69 (1.34)	73.91 (1.47)	73.73 (1.33)

First Result: best parameter set from 14.41 min to 1.02 min
Second Result: all parameter sets from ~9 hours to ~35 min

[5] G. Cavallaro, M. Riedel, J.A. Benediktsson et al., Journal of Selected Topics in Applied Earth Observation and Remote Sensing, 2015

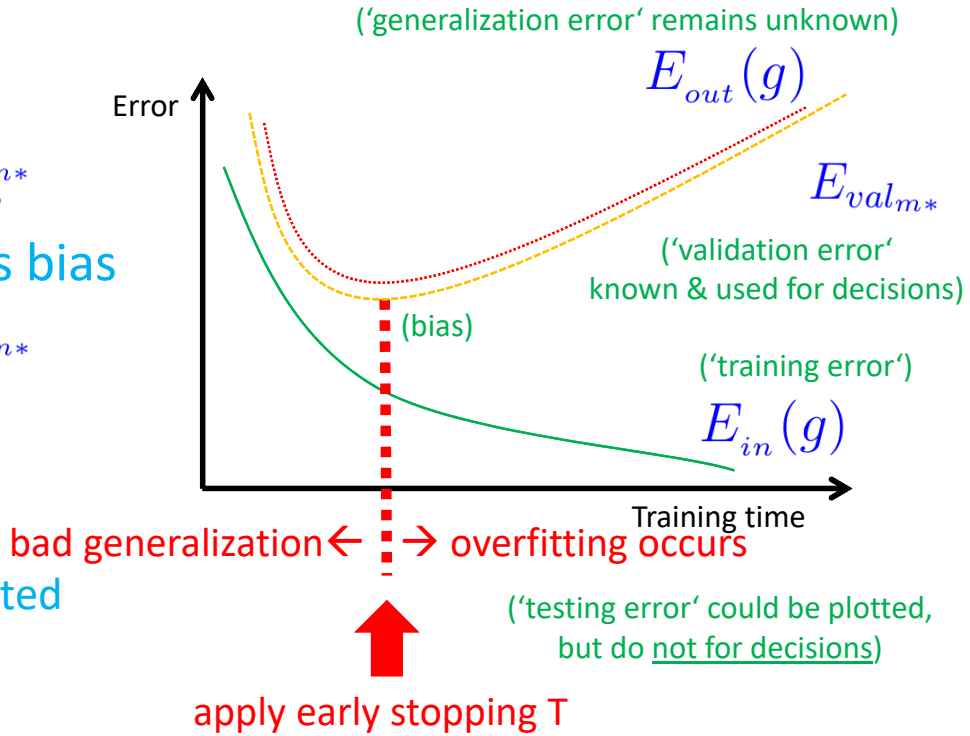
Model Performance – Validation Enables Early Stopping (1)

- Problem of overfitting
  - Issue is that $E_{out}(g)$ is unknown to perform 'early stopping'
  - Apply validation = 'perform decision to make a choice' based on $\mathcal{D}_{Val}$

$\mathcal{D}_{Val}$ (out-of-sample w.r.t. D_{Train})

  - Use validation error $E_{val_{m*}}$ to perform early stopping

- Validation decision brings bias
  - When the estimate $E_{val_{m*}}$ of $E_{out}(g)$ affects the learning process decision
  - Optimistic bias impact brings accuracy higher than in reality (cf. Associated use case with testset)



Model Performance – Validation Enables Early Stopping (2)

- ‘Bias‘ reviewed as ‘data contamination‘
  - Training set is biased and contaminated (i.e. ‘used for train model change‘)
  - Test set is unbiased and clean (i.e. ‘waiting to be used in the final end‘)
  - Validation set has an optimistic bias (i.s. ‘use in model selection decisions‘)

(‘slightly contaminated since only few choices‘)

$$E_{val_1} \quad E_{val_2} \quad E_{val_M}$$

(pick ‘best‘ → bias) (decides model selection)

$$\mathcal{H}_{m*} \; E_{val_{m*}}$$

$$\mathbb{E}[E_{val}(h)] = \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[e(h(\mathbf{x})_k, y_k)] = E_{out}$$

(reasoning of bias relates to the probability and estimated value of validation errors since ‘one is picked‘ as the minimum of all)

(e is a min function of E_{val1}, E_{val2}, etc.)

$$\mathbb{E}[e(h(\mathbf{x}), y)] = E_{out}(h)$$

(aka the long-run average value of repetitions of the experiment)

(cf. picking the ‘best time‘ in early stopping, also brings optimistic bias since minimum on model creation)

- **Optimistic bias means that there is ‘a belief‘ that the error is smaller as it is actually going to be**
- **Optimistic bias is minor and thus accepted in learning, but perform reporting with unbiased testset**
- **Important in validation is that the validation set stays only ‘slightly contaminated‘ (few choices)**
- **In practice several validation sets can be used for n parameter choices to keep reliable estimate**

Problem of Overfitting – Impacts on Learning & Model Tuning

- Understanding deterministic noise & target complexity
  - Increasing target complexity increases deterministic noise (at some level)
  - Increasing the number of data N decreases the deterministic noise
- Finite N case: $\mathcal{H}$ tries to fit the noise
  - Fitting the noise straightforward (e.g. Perceptron Learning Algorithm)
  - Stochastic (in data) and deterministic (simple model) noise will be part of it
- Two 'solution methods' for avoiding overfitting
  - Regularization: 'Putting the brakes in learning', e.g. early stopping (more theoretical, hence 'theory of regularization')
  - Validation: 'Checking the bottom line', e.g. other hints for out-of-sample (more practical, methods on data that provides 'hints')

- **The higher the degree of the polynomial (cf. model complexity), the more degrees of freedom are existing and thus the more capacity exists to overfit the training data**

Regularization – Two Approaches

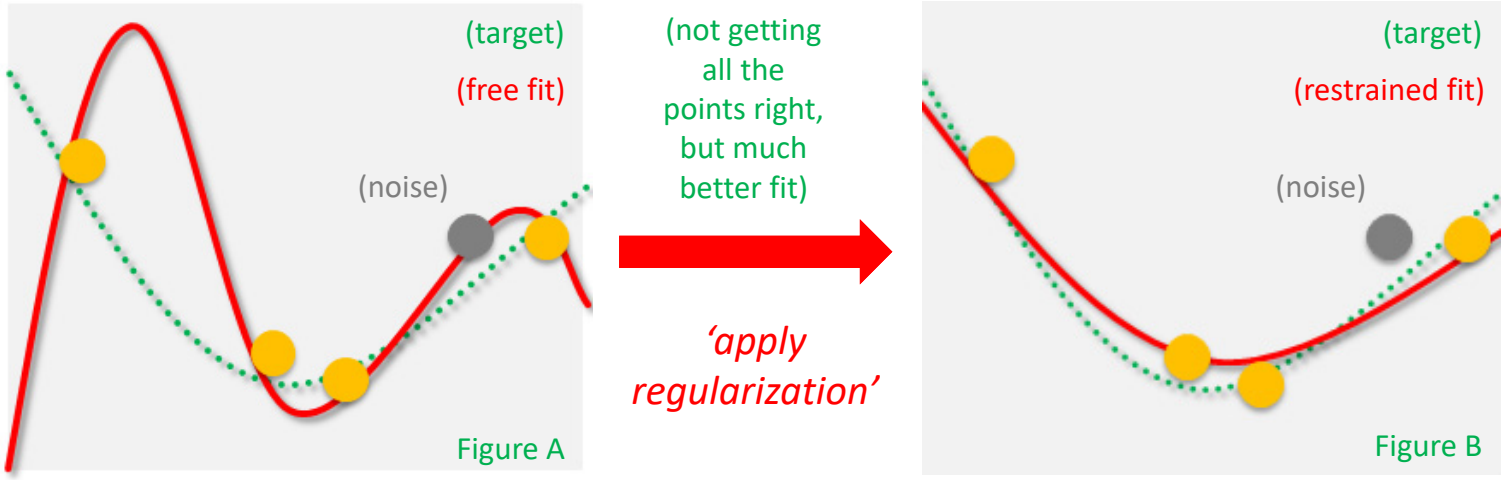
- Used in almost every machine learning situation
  - Two approaches to regularization: mathematics & heuristics
- Mathematics (from ill-posed problems in function approximation)
  - Goal: approximate a function
  - 'Ill-posed Problem': there are many functions that fit this function
  - Approach: Impose 'smoothness constraints' to solve the problem
  - Learning practice: assumptions in mathematical approach not realistic
  - Solution: use mathematical approach intuitively to create 'rules of thumb'
- Heuristics
  - Handy-capping the minimization of the in-sample-error $E_{in}(g)$
  - Idea of 'putting the brakes' – partly based on mathematics
  - Not a random process and oriented towards preventing overfitting

- **Regularization creates simpler models & thus in-line with 'Occams Razor' (simple model better)**

Regularization – Key Principle ‘Do not take data too serious‘

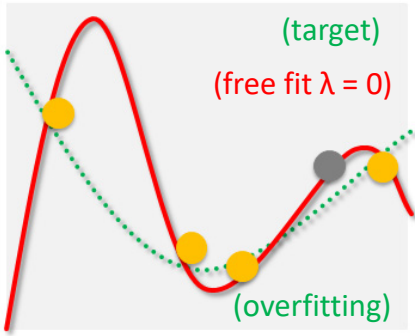
- Initial setup → Rather memorizing (not learning to generalize)
  - Considered as ‘free fit‘ – ‘fit as far as the model can do‘
  - E.g. use 4th order polynomial model and fit the 5 data points (cf. Figure A)
- ‘Putting the brakes‘ regularization to avoid overfitting
  - Apply a ‘restrained fit‘ – ‘preventing to fit the data perfectly‘
  - E.g. use 4th order polynomial model but use ‘minimal brakes‘ (cf. Figure B)

(equivalent meaning of explicitly forbidding some of the hypothesis to be considered in learning)

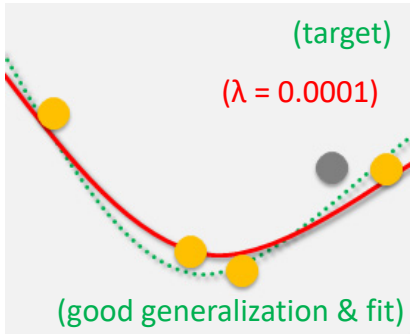


Regularization – Regularization Example & Impact

- Example: Minimize $E_{in}(\mathbf{w}) + \frac{\lambda}{N}\mathbf{w}^T\mathbf{w}$ (penalizing models with extreme parameter values)

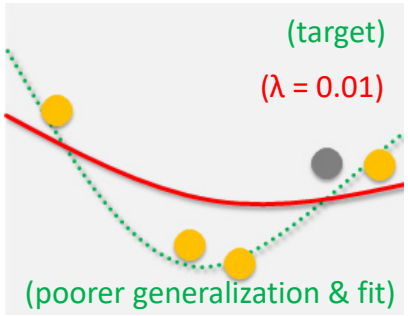


'apply little regularization'

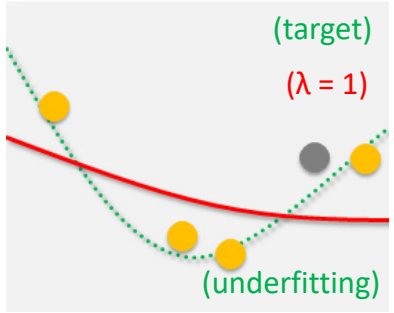


(regularization coefficient λ controls tradeoff: how much it is important to fit the data well vs. simple hypothesis)

'apply more regularization'



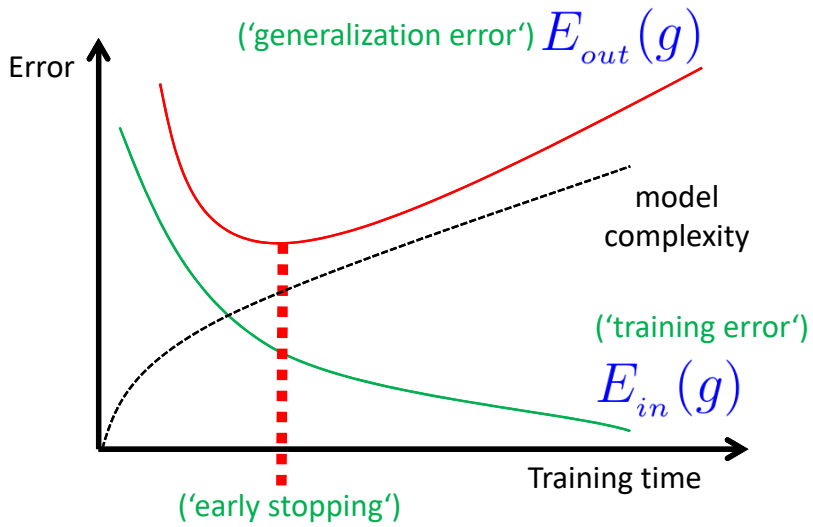
'apply too much regularization'



- **Choice of λ is extremely critical and can lead with high values to underfitting (as bad as overfitting)**
- **Choice of type of regularizers is heuristic, but choice of λ will be extremely principled (validation)**

Regularizer Methods – Early Stopping & Big Data

- Idea: regularization through the optimizer
  - Apply early stopping algorithm as part of the 'training time'
  - Based on practical validation methods
  - Point in time to stop will be choosen by validation



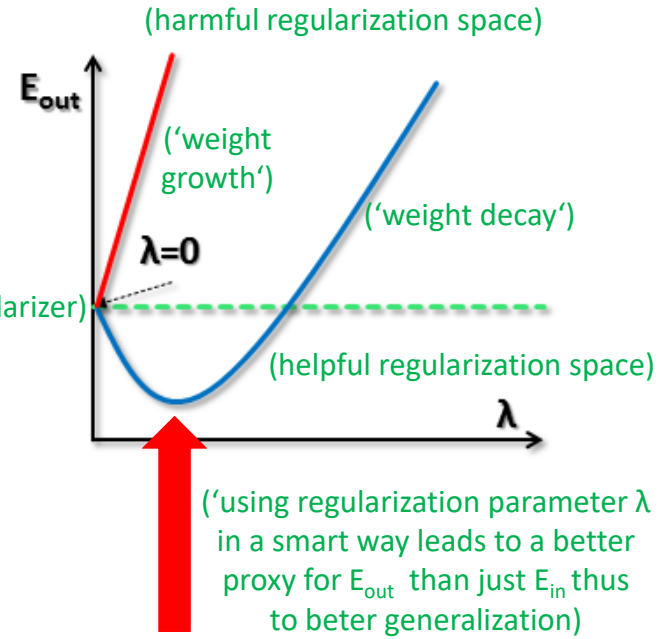
- **Big data has a bigger dataset N and more likely also 'more stochastic noise'**
- **Preventing overfitting with big data tends to require stricter regularization**
- **Early stopping offers a pragmatic way for big data to prevent overfitting**
- **Given large quantities of dataset N, the training time is massively reduced too**

Regularization Methods – Choosing Guidelines

- **Choosing a regularizer in learning from data is a must have & leads to better generalization**
- **Choosing not a regularizer will definitely lead to overfitting the data in practical situations**

- Selected ‘rules of thumb‘
  - Choosing constraints with ‘weight growth‘ gives terrible results
  - Choosing constraints towards ‘smoother hypothesis‘ gives good results
- Reasoning noise is not smooth
  - Stochastic noise → ‘high frequency‘
  - Determinisic noise → ‘non-smooth‘
  - Smaller weights correspond to smoother hypothesis

(‘hectic dog & calm man have a walk‘ analogy)



- **The rule of thumb in regularization is to constrain the learning with smoother/simpler hypothesis**
- **Regularization forbids some hypothesis & thus reduces the VC dimension: improving generalization**

Model Tuning in Neural Networks: ANN 2 Hidden - 1/5 Validation Splits

- If there is enough data available one rule of thumb is to take 1/5 (0.2) 20% of the datasets for validation only
- Validation data is used to perform model selection (i.e. parameter / topology decisions)

```
# parameter setup
NB_EPOCH = 20
BATCH_SIZE = 128
NB_CLASSES = 10 # number of outputs = number of digits
OPTIMIZER = SGD() # optimization technique
VERBOSE = 1
N_HIDDEN = 128 # number of neurons in one hidden layer
VAL_SPLIT = 0.2 # 1/5 for validation rule of thumb
```

- The validation split parameter enables an easy validation approach during the model training (aka fit)
- Expectations should be a higher accuracy for unseen data since training data is less biased when using validation for model decisions (check statistical learning theory)
- VALIDATION_SPLIT: Float between 0 and 1
- Fraction of the training data to be used as validation data
- The model fit process will set apart this fraction of the training data and will not train on it
- Intead it will evaluate the loss and any model metrics on the validation data at the end of each epoch.

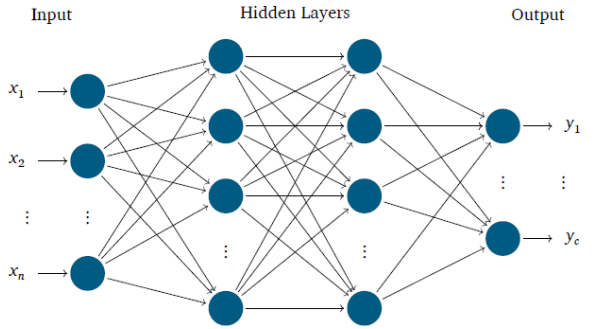
```
# model training
history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NB_EPOCH, verbose=VERBOSE, validation_split = VAL_SPLIT)
```

```
Train on 48000 samples, validate on 12000 samples
```

Model Tuning in Neural Networks: ANN 2 Hidden – DropOut Regularization

```
# parameter setup
NB_EPOCH = 20
BATCH_SIZE = 128
NB_CLASSES = 10 # number of outputs = number of digits
OPTIMIZER = SGD() # optimization technique
VERBOSE = 1
N_HIDDEN = 128 # number of neurons in one hidden layer
VAL_SPLIT = 0.2 # 1/5 for validation rule of thumb
DROPOUT = 0.3 # regularization
```

```
# modeling step
# 2 hidden layers each N_HIDDEN neurons
model.add(Dense(N_HIDDEN, input_shape=(RESHAPED,)))
model.add(Activation('relu'))
model.add(Dropout(DROPOUT))
model.add(Dense(N_HIDDEN))
model.add(Activation('relu'))
model.add(Dropout(DROPOUT))
model.add(Dense(NB_CLASSES))
```

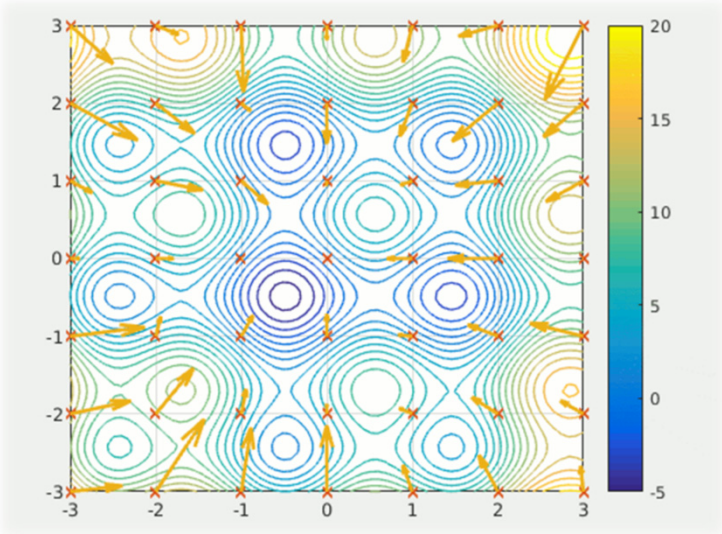


- **A Dropout() regularizer randomly drops with ist dropout probability some of the values propagated inside the Dense network hidden layers improving accuracy again**
- **Our standard model is already modified in the python script but needs to set the DROPOUT rate**
- **A Dropout() regularizer randomly drops with ist dropout probability some of the values propagated inside the Dense network hidden layers improving accuracy again**

```
model.add(Activation('relu'))
model.add(Dropout(DROPOUT))
```

Other Model Tunings with Evolutionary Algorithms

- Particle Swarm Optimization (PSO)
  - Optimizes a problem by iteratively trying to improve a candidate solution w.r.t. given measure of quality
  - Tools support sometimes limited
  - More efficient than multi-dimensional ‘gridsearch‘ since this becomes with multi-dimensional increase compute-intensive



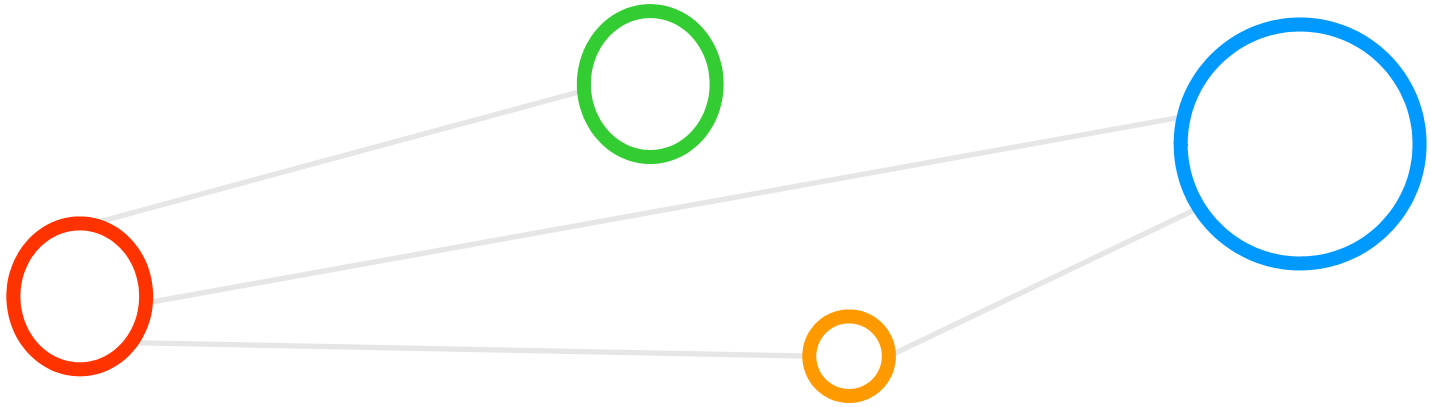
[7] Particle Swarm Optimization

- **Basic idea of Particle Swarm Optimization (PSO) is in having a population (swarm) of candidate solutions (called particles)**
- **The particles are moved around in the search-space according to a few simple formulae**
- **Movements of particles are guided by their own best konwn position in the search-space**
- **Movements of particles also guided by the entire swarm‘s best known position**
- **When improved positions are being discovered these will then come to guide the movements of the swarm**
- **Process is repeated and by doing to (not guaranteed) finds a better satisfactory solution**

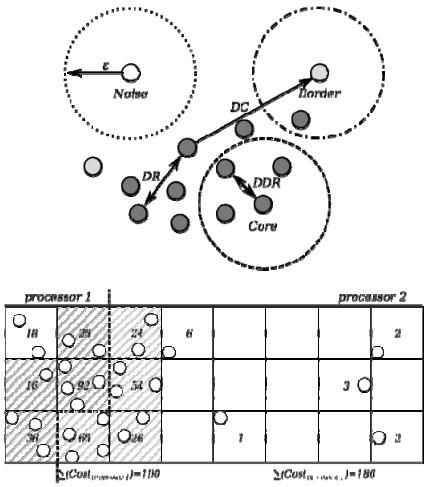
[Video] Summary Data Contamination & Validation & Regularization

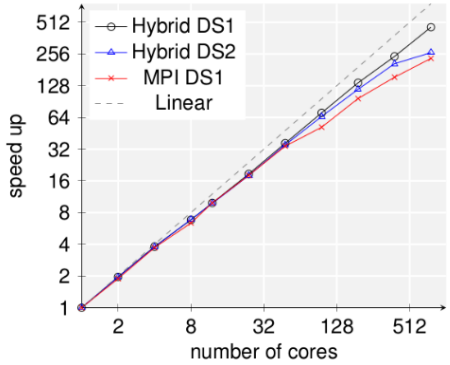
[4] YouTube Video, Machine Learning : Model Selection & Cross Validation

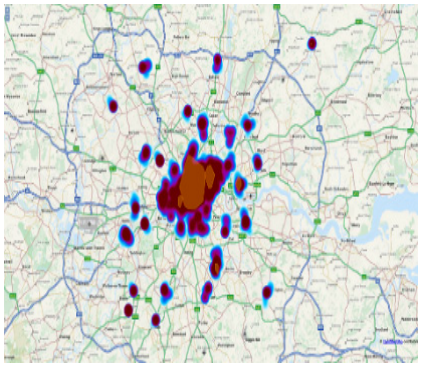
(Semi-)Automated Machine Learning & Architecture Tuning

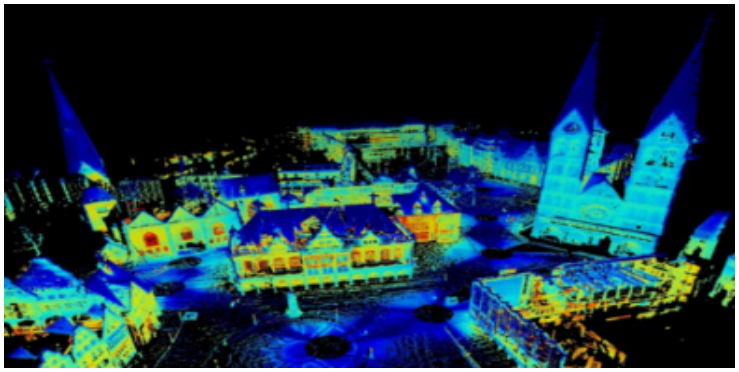


Motivation for Automated Model Tuning: Parallel & Scalable DBSCAN



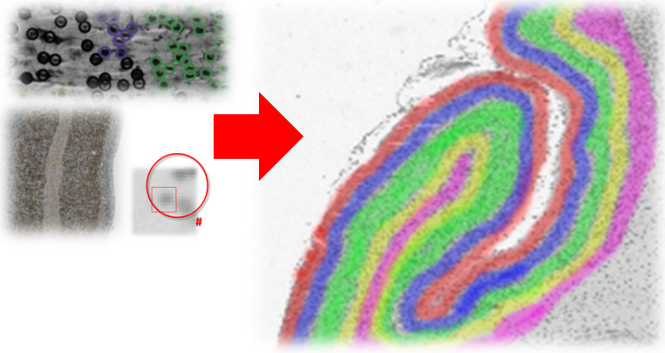


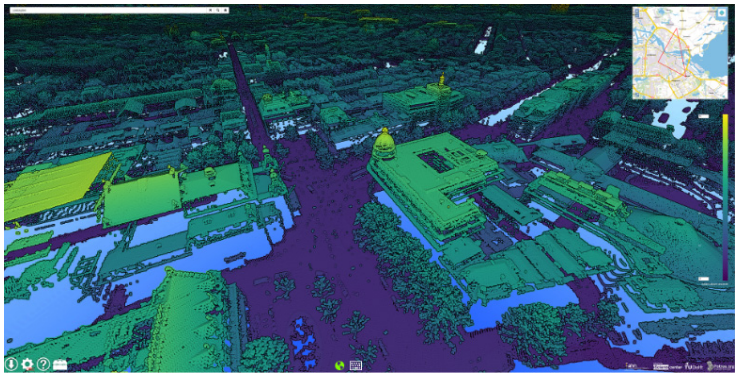




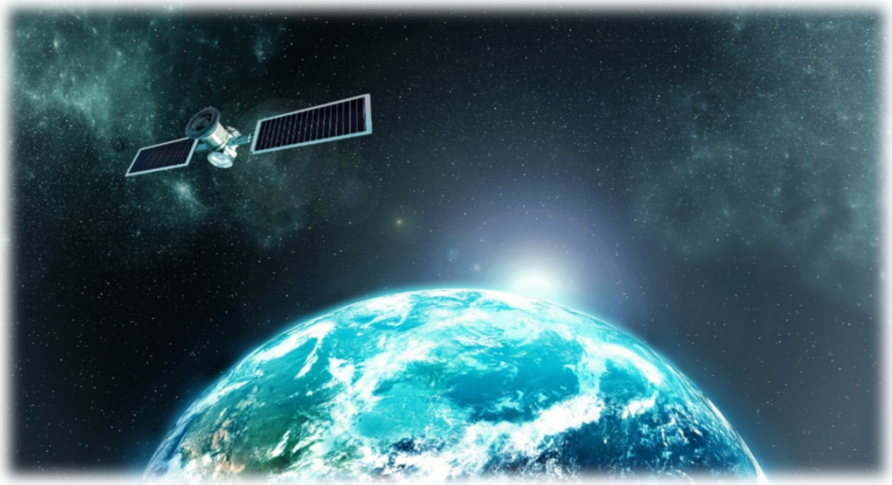
[8] M. Goetz and M. Riedel et al, Proceedings IEEE Supercomputing Conference, 2015

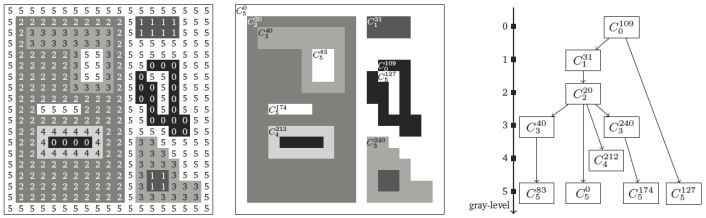
- Find right set of 2 parameters for application
- 1 Parameter: Minimum number of points
- 2 Parameter: Epsilon Neighbourhood
- Needs already HPC to be efficient in searching the right set of parameters, e.g. gridsearch, particle swarm optimization (evolutionary algorithm)



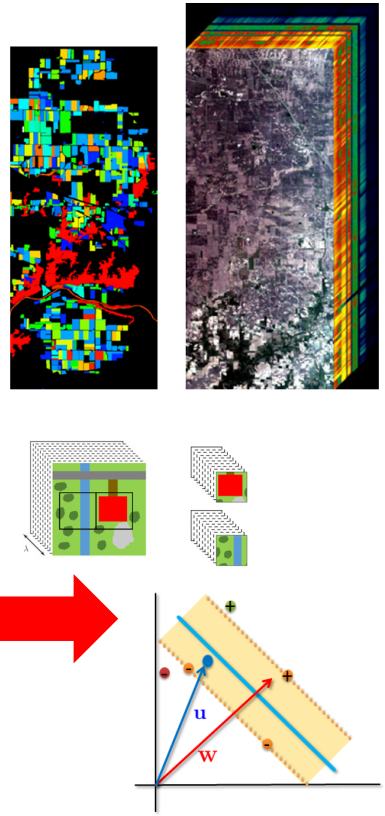


Motivation for Automated Model Tuning: Parallel & Scalable SVM





[16] M. Goetz and M. Riedel et al., Journal of Transactions on Parallel and Distributed Systems, 2018



[5] G. Cavallaro and M. Riedel et al., Journal of Selected Topics in Applied Earth Observation and Remote Sensing, 2015

- **Find right set of 2 parameters for application**
- **1 Parameter: RBF Parameter Gamma**
- **2 Parameter: Cost of Error allowed for soft margin**
- **Needs already HPC to be efficient in searching the right set of parameters, e.g. gridsearch using partly also community experience in applications**

Scenario 'pre-processed data', 10xCV **serial**: accuracy (min)

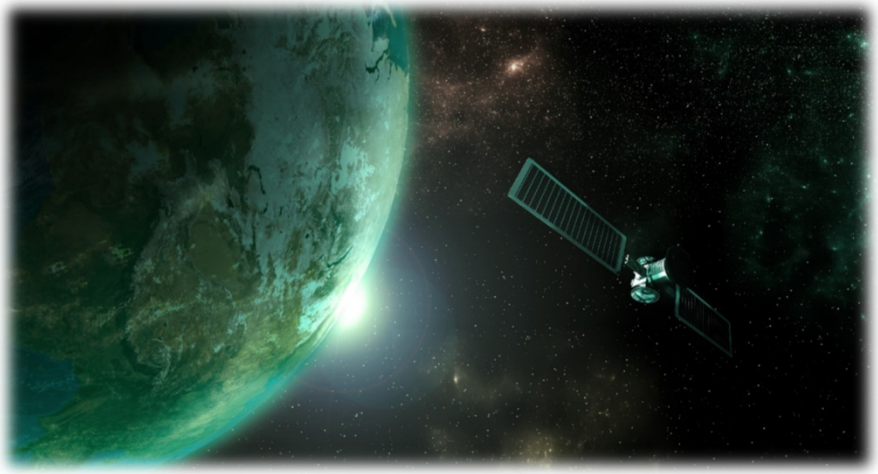
γ/C	1	10	100	1000	10 000
2	48.90 (18.81)	65.01 (19.57)	73.21 (20.11)	75.55 (22.53)	74.42 (21.21)
4	57.53 (16.82)	70.74 (13.94)	75.94 (13.53)	76.04 (14.04)	74.06 (15.55)
8	64.18 (18.30)	74.45 (15.04)	**77.00** (14.41)	75.78 (14.65)	74.58 (14.92)
16	68.37 (23.21)	76.20 (21.88)	76.51 (20.69)	75.32 (19.60)	74.72 (19.66)
32	70.17 (34.45)	75.48 (34.76)	74.88 (34.05)	74.08 (34.03)	73.84 (38.78)

Scenario 'pre-processed data', 10xCV **parallel**: accuracy (min)

γ/C	1	10	100	1000	10 000
2	75.26 (1.02)	65.12 (1.03)	73.18 (1.33)	75.76 (2.35)	74.53 (4.40)
4	57.60 (1.03)	70.88 (1.02)	75.87 (1.03)	76.01 (1.33)	74.06 (2.35)
8	64.17 (1.02)	74.52 (1.03)	**77.02 (1.02)**	75.79 (1.04)	74.42 (1.34)
16	68.57 (1.33)	76.07 (1.33)	76.40 (1.34)	75.26 (1.05)	74.53 (1.34)
32	70.21 (1.33)	75.38 (1.34)	74.69 (1.34)	73.91 (1.47)	73.73 (1.33)

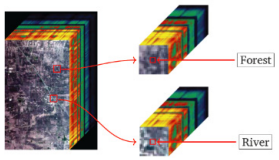
First Result: best parameter set from 14.41 min to 1.02 min
Second Result: all parameter sets from ~9 hours to ~35 min

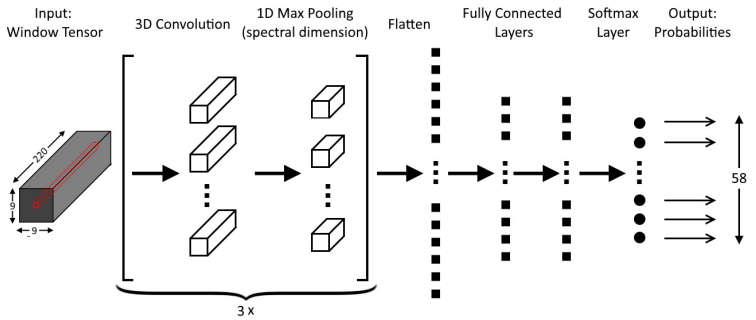
Motivation for Automated Model Tuning: CNN Deep Learning



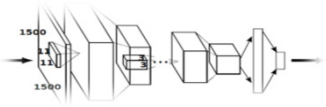
- **Using Convolutional Neural Networks (CNNs) with hyperspectral remote sensing image data**

[10] J. Lange and M. Riedel et al., IGARSS Conference, 2018





- **Find Hyperparameters & joint 'new-old' modeling & transfer learning given rare labeled/annotated data in science (e.g. 36,000 vs. 14,197,122 images ImageNet)**

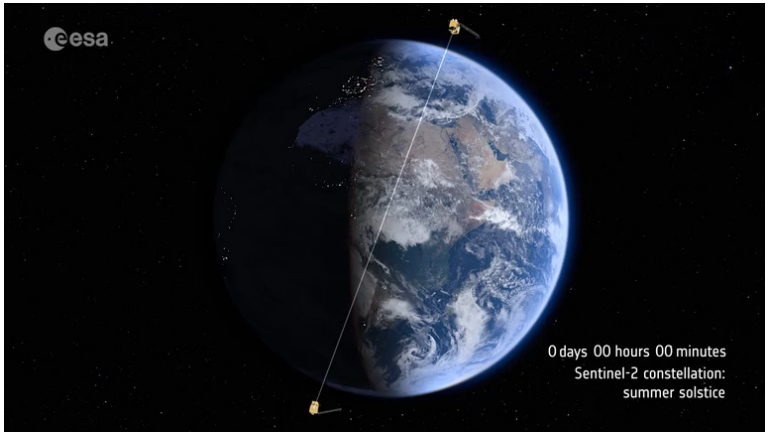


- What is the right optimization method?
- How many convolutional layers we need?
- How many neurons in dense layers?
- What is the right filter size?
- How do we train best?

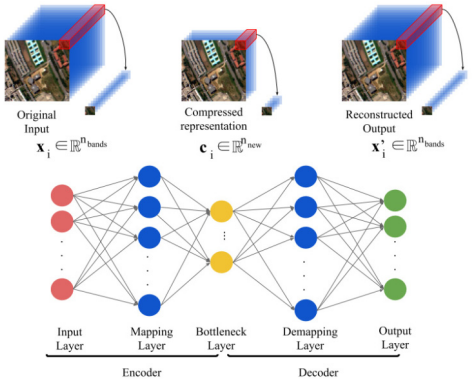
Feature	Representation / Value
Conv. Layer Filters	48, 32, 32
Conv. Layer Filter size	(3,3,5), (3,3,5), (3,3,5)
Dense Layer Neurons	128, 128
Optimizer	SGD
Loss Function	mean squared error
Activation Functions	ReLU
Training Epochs	600
Batch Size	50
Learning Rate	1
Learning Rate Decay	5×10^{-6}

- **Find right set of hyper-parameters and the right neural network architecture is a manual time-consuming and error-prone process**
- **Needs urgently HPC, but a systematic and automated way is required as trying out all options of hyper-parameters and architectures is computationally infeasible**

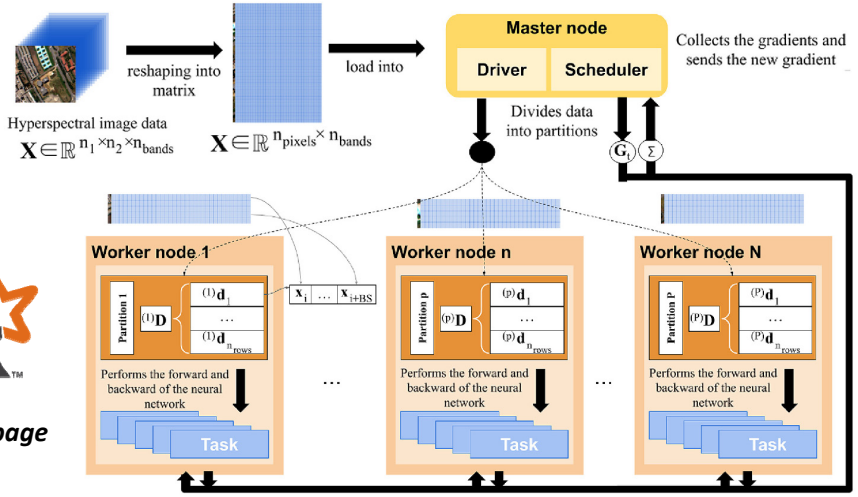
Motivation for Automated Model Tuning: Autoencoder Deep Learning



- **Find right set of hyper-parameters and the right neural network architecture for autoencoder is a manual time-consuming and error-prone process**
- **Needs urgently HPC, but a systematic and automated way is required as trying out all options of hyper-parameters and architectures is computationally infeasible**
- **As resolutions of sensors becomes better and more data is available it is likely that the learning model will be increasingly complex in the future that in turn raises demands for automated architecture search and meta-learning approaches**

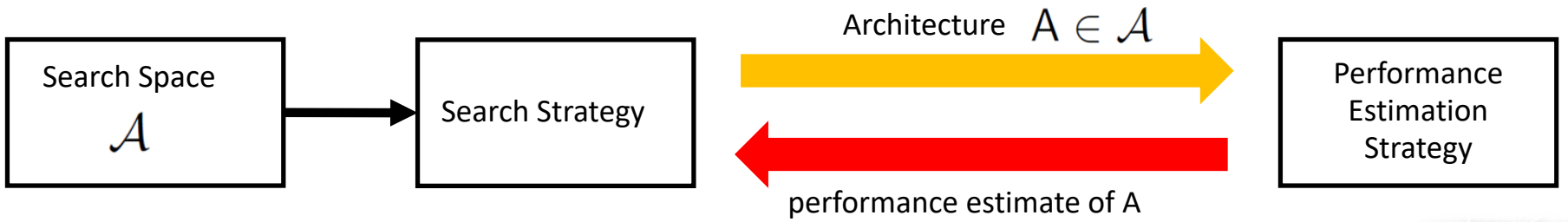


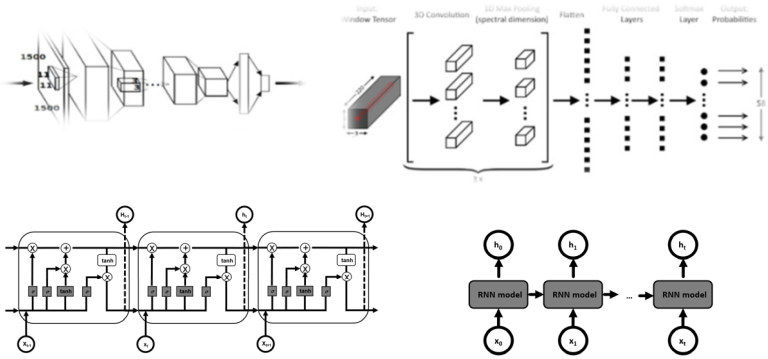
[11] J. Haut, G. Cavallaro and M. Riedel et al., IEEE Transactions on Geoscience and Remote Sensing, 2019



[12] Apache Spark Web page

Neural Architecture Search (NAS) – Overview



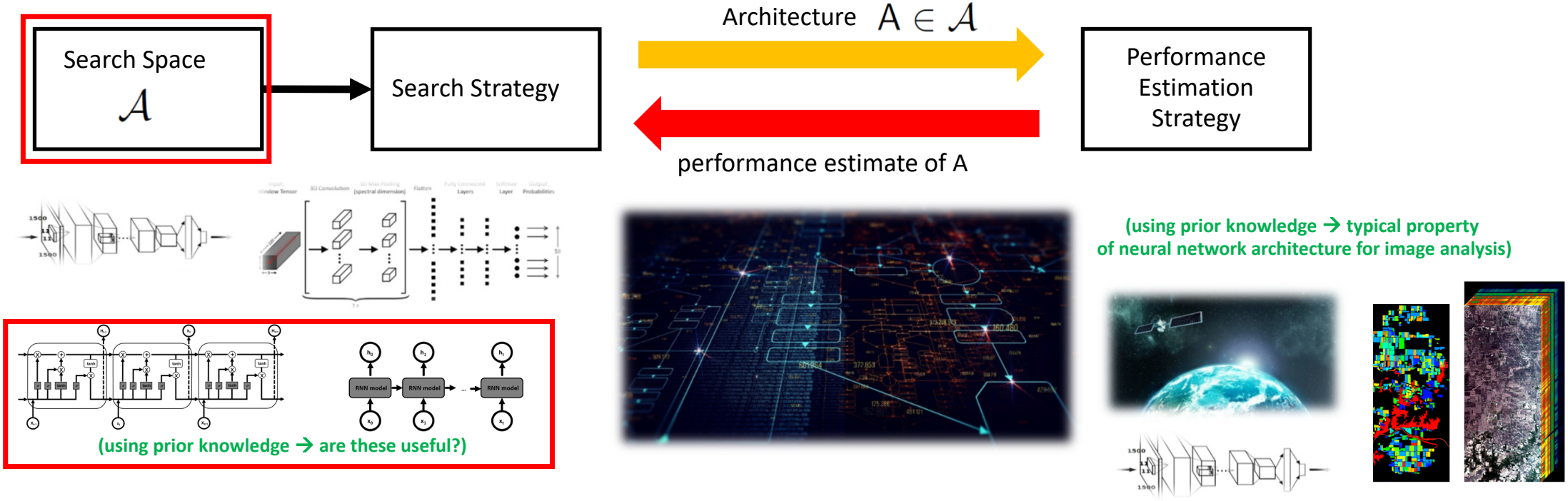


[13] Neural Architecture Search Research

[14] T. Elsken et al., Neural Architecture Search: A Survey, Journal of Machine Learning Research, 2019

- **Employed neural networks architectures are often developed manually by human experts that is time-consuming and error-prone**
- **Deep learning success has been accompanied by a rising demand for architecture engineering, where increasingly more complex neural architectures are designed manually**
- **Neural Architecture Search (NAS) methods can be categorized in (a) search space, (b) search strategy, and (c) performance estimation strategy**
- **Automated Neural Architecture (NAS) search methods aim to solve this problem as a process of automating Architecture engineering**

NAS – Understanding Search Space & Using Prior Knowledge Example



[14] T. Elsken et al., Neural Architecture Search: A Survey, Journal of Machine Learning Research, 2019

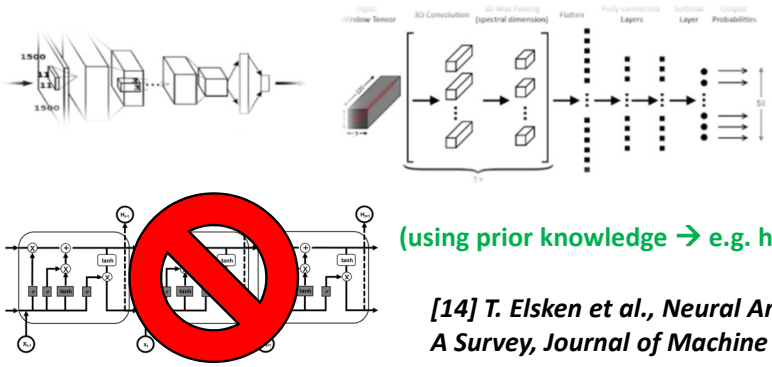
- **Search space defines which neural network architectures can be represented in principle**
- **Reduce the size of the search space to simplify the search by incorporating prior knowledge about typical properties of architectures**
- **Be aware that using prior knowledge also might introduce a human bias thus preventing finding novel neural network architectures**

NAS - Understanding the Search Space & Common Search Space for CNNs

Search Space

$\mathcal{A}$

?

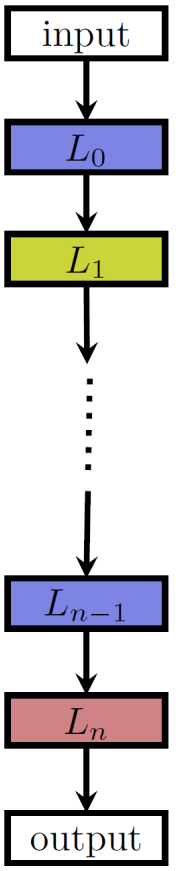


(using prior knowledge → e.g. here no time series)

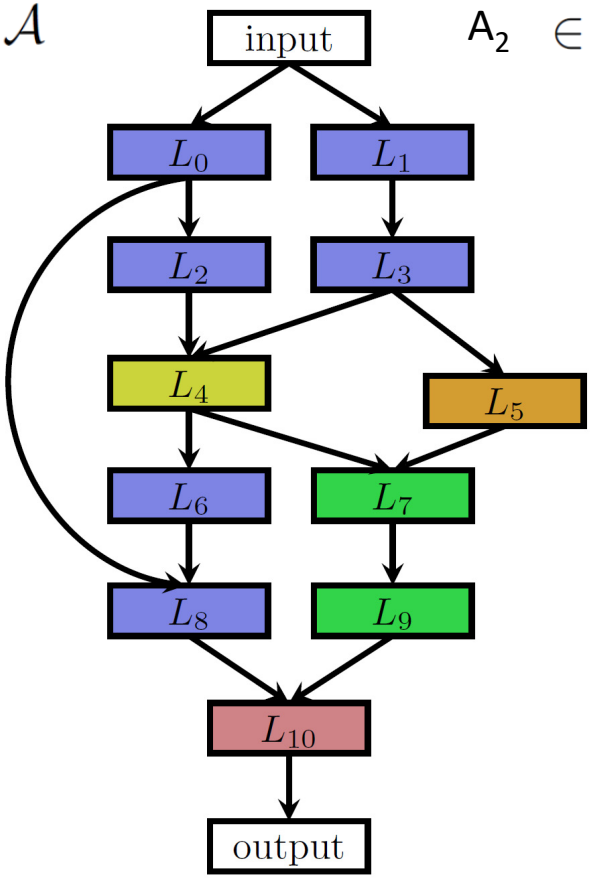
[14] T. Elsken et al., Neural Architecture Search: A Survey, Journal of Machine Learning Research, 2019

- Each node in the graphs corresponds to a layer in a convolutional neural network (CNN), e.g. convolutional or pooling layer
- Different layer types are visualized by different colors
- Example Architecture A_1: Element of a chain-structured space
- Example Architecture A_2: element of a more complex search space with additional layer types and multiple branches and skip connections

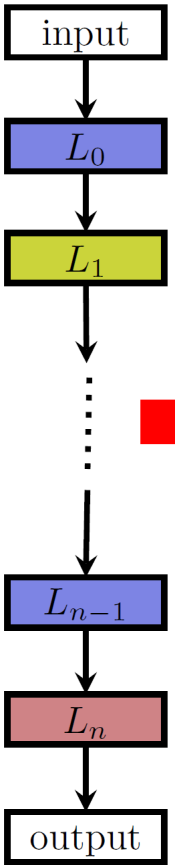
$A_1 \in \mathcal{A}$

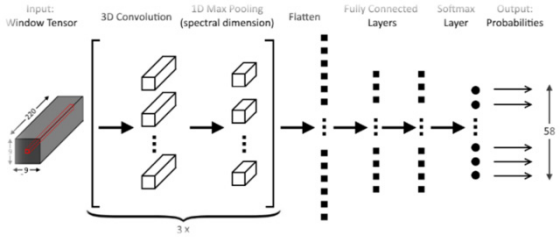


$A_2 \in \mathcal{A}$

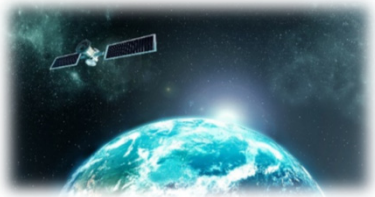


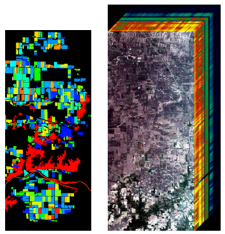
NAS - Understanding Layer Parameters & Complexity in Setting Parameter Values





?





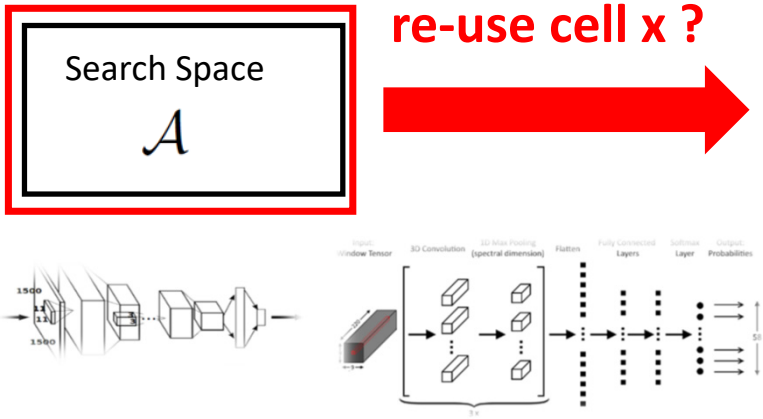
?

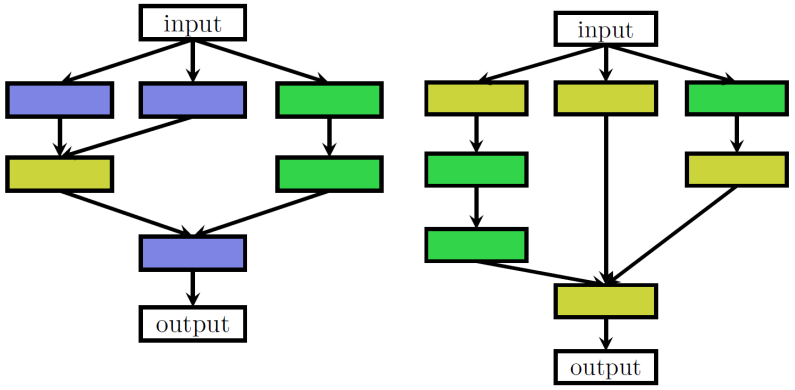
Layer Type	Layer Parameters	Parameter Values
Convolution (C)	$i \sim$ Layer depth $f \sim$ Receptive field size $\ell \sim$ Stride $d \sim$ # receptive fields $n \sim$ Representation size	< 12 Square. $\in \{1, 3, 5\}$ Square. Always equal to 1 $\in \{64, 128, 256, 512\}$ $\in \{(\infty, 8], (8, 4], (4, 1]\}$
Pooling (P)	$i \sim$ Layer depth $(f, \ell) \sim$ (Receptive field size, Strides) $n \sim$ Representation size	< 12 Square. $\in \{(5, 3), (3, 2), (2, 2)\}$ $\in \{(\infty, 8], (8, 4]$ and $(4, 1]\}$
Fully Connected (FC)	$i \sim$ Layer depth $n \sim$ # consecutive FC layers $d \sim$ # neurons	< 12 < 3 $\in \{512, 256, 128\}$
Termination State	$s \sim$ Previous State $t \sim$ Type	Global Avg. Pooling/Softmax

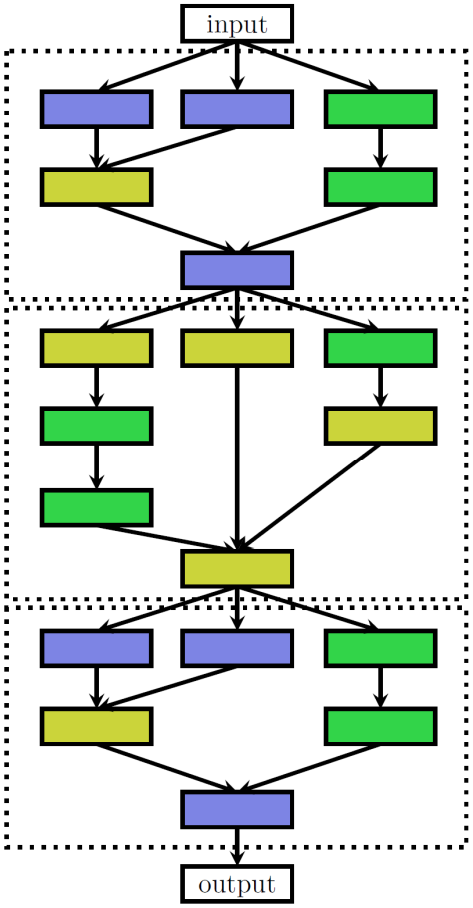
[14] T. Elsken et al., Neural Architecture Search: A Survey, Journal of Machine Learning Research, 2019

[15] B. Baker et al., 'Designing Neural Network Architectures using Reinforcement Learning', 2017

NAS - Understanding the Search Space & Cells instead of whole Architectures







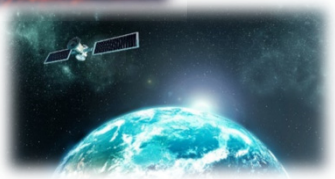
[14] T. Elsken et al., Neural Architecture Search: A Survey, Journal of Machine Learning Research, 2019

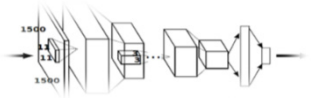
- **Example illustrations of the cell search space with two different cells (not whole neural network architectures) → many design choices for the overall 'macro architecture' of the network**
- **Approach: a whole neural network architecture can be built by stacking the cells sequentially**
- **Complex approach: Cells can also be combined in a more complex approach: e.g., in multi-branch spaces by simply replacing layers**

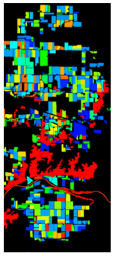
- Simplicity: Search space is drastically reduced (less layers → ~ 7 x speed-up vs. full architectures in some recent work examples)
- Reusability: Architectures built from cells can more easily be transferred or adapted to other datasets
- Repitition: Creating architectures by repeating building blocks has proven a useful design (e.g. CNN with N x convolution, pooling layers, etc.)

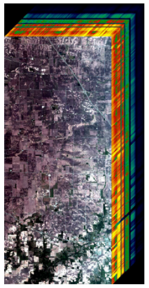
NAS - Computational Complexity & Number of Parameters of Architectures

Model	Depth	Parameters	Error rate (%)
Network in Network (Lin et al., 2013)	-	-	8.81
All-CNN (Springenberg et al., 2014)	-	-	7.25
Deeply Supervised Net (Lee et al., 2015)	-	-	7.97
Highway Network (Srivastava et al., 2015)	-	-	7.72
Scalable Bayesian Optimization (Snoek et al., 2015)	-	-	6.37
FractalNet (Larsson et al., 2016)	21	38.6M	5.22
with Dropout/Drop-path	21	38.6M	4.60
ResNet (He et al., 2016a)	110	1.7M	6.61
ResNet (reported by Huang et al. (2016c))	110	1.7M	6.41
ResNet with Stochastic Depth (Huang et al., 2016c)	110	1.7M	5.23
	1202	10.2M	4.91
Wide ResNet (Zagoruyko & Komodakis, 2016)	16	11.0M	4.81
	28	36.5M	4.17
ResNet (pre-activation) (He et al., 2016b)	164	1.7M	5.46
	1001	10.2M	4.62
DenseNet ($L = 40, k = 12$) Huang et al. (2016a)	40	1.0M	5.24
DenseNet($L = 100, k = 12$) Huang et al. (2016a)	100	7.0M	4.10
DenseNet ($L = 100, k = 24$) Huang et al. (2016a)	100	27.2M	3.74
DenseNet-BC ($L = 100, k = 40$) Huang et al. (2016b)	190	25.6M	3.46
Neural Architecture Search v1 no stride or pooling	15	4.2M	5.50
Neural Architecture Search v2 predicting strides	20	2.5M	6.01
Neural Architecture Search v3 max pooling	39	7.1M	4.47
Neural Architecture Search v3 max pooling + more filters	39	37.4M	3.65





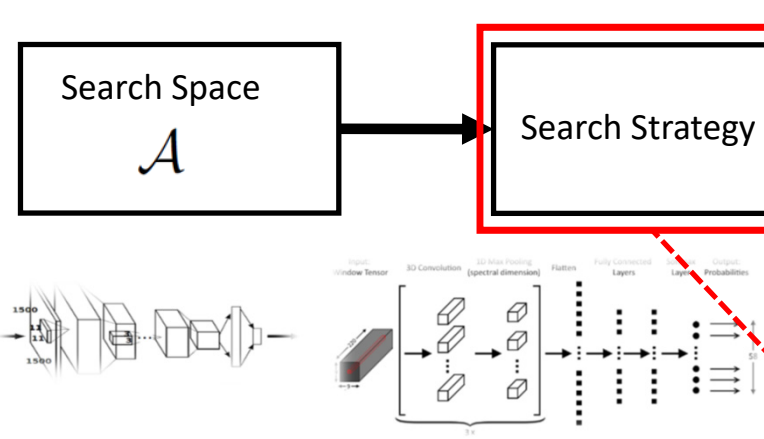




- **Further complexity in choosing HPC system features (e.g. Network Attached Memory, parallel file-systems, etc.)**
- **Computing Impact in various GPU architectures like Kepler, Pascal, Volta**

[16] B. Zoph et al., 'InstaNAS: Instance-aware Neural Architecture Search', 2018

NAS - Understanding the Search Strategy



[14] T. Elsken et al., Neural Architecture Search: A Survey, Journal of Machine Learning Research, 2019

- **Search strategy details how to explore the search space (which is often exponentially large or even unbounded)**
- **(a) it is desirable to find well-performing architectures quickly**
- **(b) premature convergence to a region of suboptimal architectures should be avoided**
- **Search strategy encompasses the classical exploration-exploitation trade-off: (a) vs (b)**

(different search strategies to explore the space of neural network architectures in addition to random search)

Reinforcement Learning

- Generation of an architecture can be considered as agent's action with space identical with neural architecture search space and rewards are used to guide the process

Gradient-Based Methods

- Optimize network weights & architecture by alternating gradient descent steps on training data for weights & validation data for architectural parameters

Bayesian Optimization

- Central idea of Bayesian optimization is to build a neural network architecture that can be updated and queried to drive optimization decisions, also assuming e.g. distributions (i.e., priors)

Evolutionary Methods

- Evolve a population of neural architecutres & in every evolution step a model from the population is sampled and serves as parent to generate offsprings & mutations

Learning Approaches – Reinforcement Learning – Revisited (cf. Lecture 8)

- Each observation of the predictor measurement(s) has some associated response measurement:
  - Input $\mathbf{x} = x_1, \ldots, x_d$
  - Some output & grade of the output
  - Data $(\mathbf{x}_1), \ldots, (\mathbf{x}_N)$
- Goal: Learn through iterations
  - Guided by output grade: check learning and compare with grade
- Challenge:
  - Iterations may require lots of CPU time (e.g. backgammon playing rounds to learn 'experience')
  - Often requires specific toolkits and ('not commonly') installed environments (e.g. OpenAI Gym)

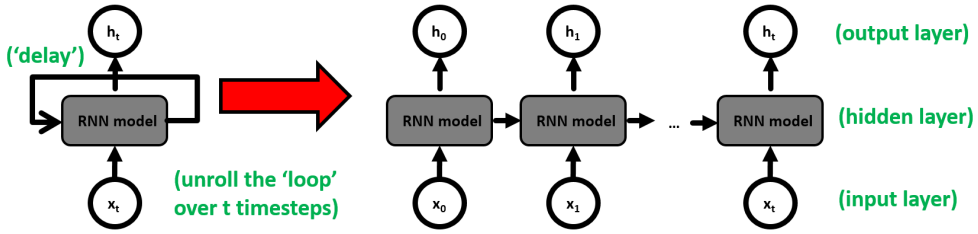
- **Reinforcement learning approaches learn through iterations using the grading output as guide**
- **Reinforcement learning approaches are used in playing game algorithms (e.g backgammon)**
- **Unupervised learning works with data = [input, some output, grade for this output]**

[17] An Introduction to Statistical Learning

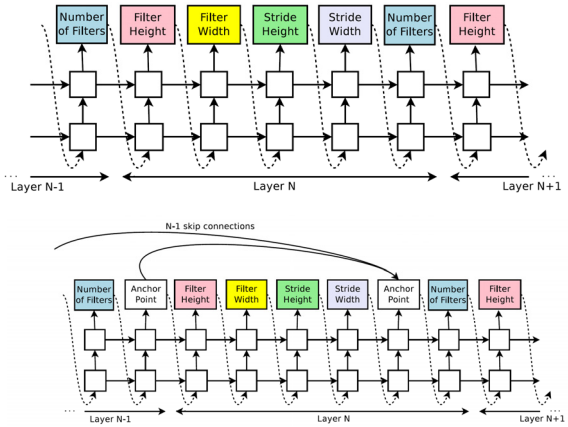
[18] OpenAI Gym Toolkit

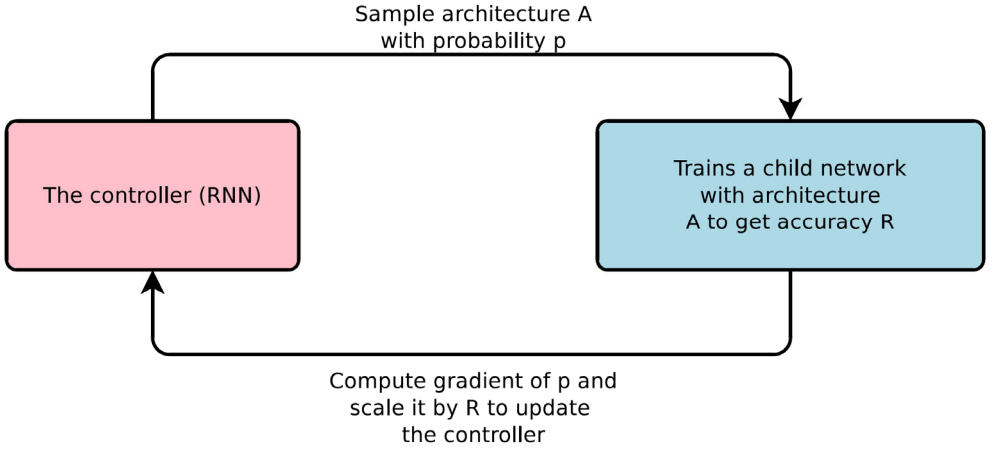
NAS – Using Recurrent Neural Networks (RNNs) – Understanding Controllers

- Recurrent Neural Networks (RNNs) can handle sequence data
- Idea: CNNs are essentially a sequence of layers
- Controller (RNN) generete hyper-parameters as a sequence

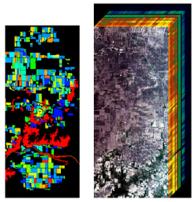


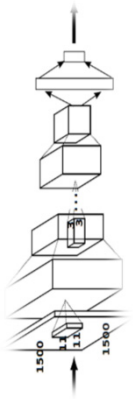
[16] B. Zoph et al., 'Neural Architecture Search with Reinforcement Learning', 2017

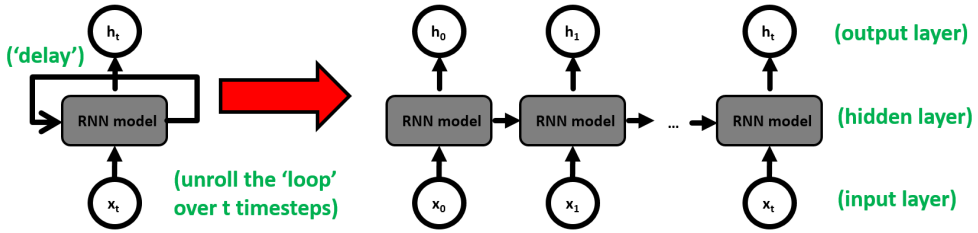




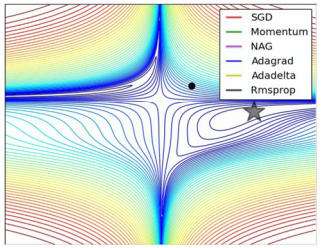
NAS - Using Reinforcement Learning Techniques & Neural Architectures



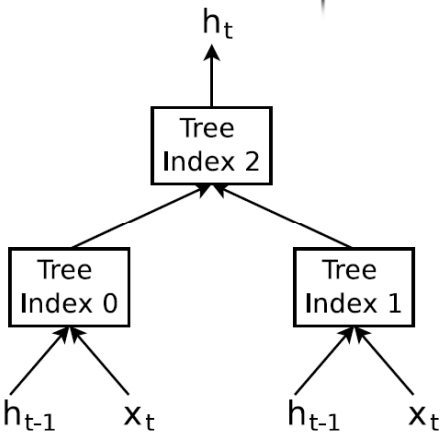


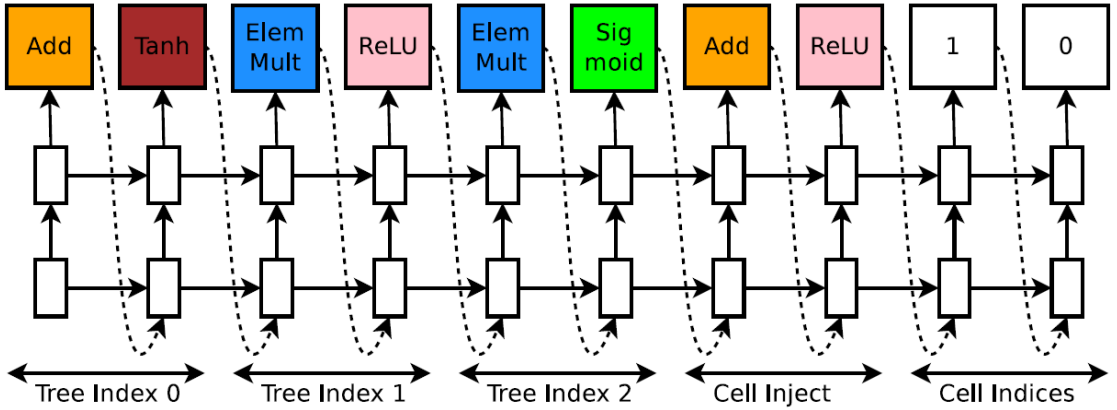


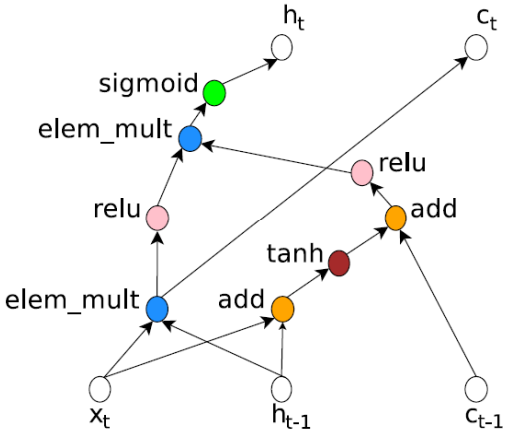
Momentum update, adaptive learning rate



(different optimizers needs to be considered too)

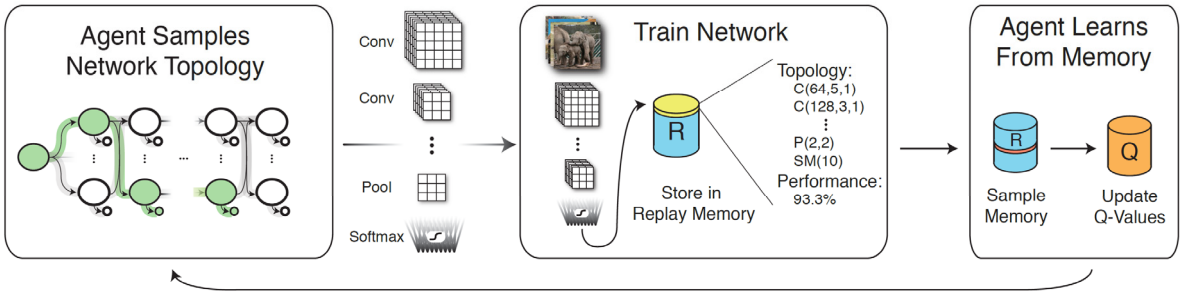






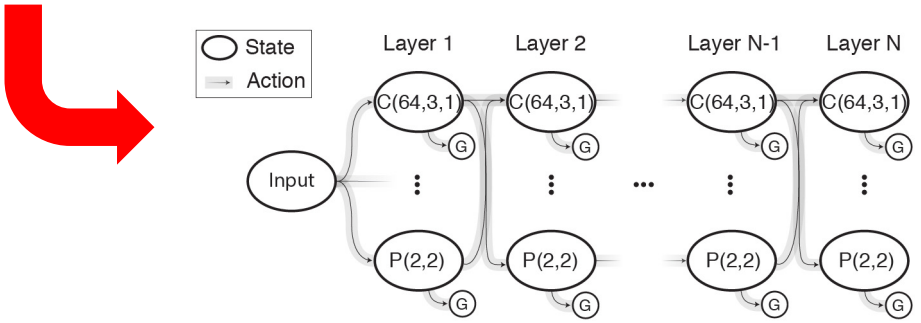
[16] B. Zoph et al., 'Neural Architecture Search with Reinforcement Learning', 2017

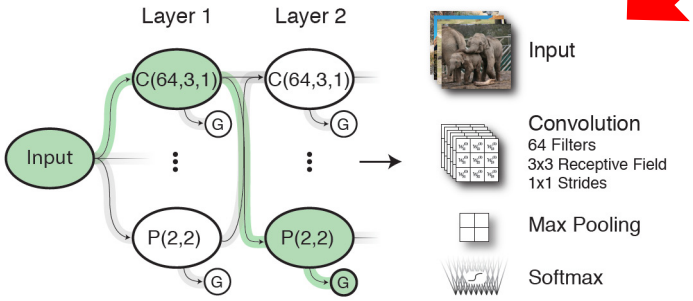
NAS – Using Reinforcement Learning Techniques – Understanding Agents

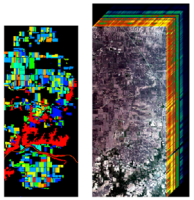


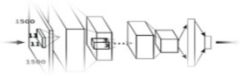
- **Generation of an architecture can be considered as agent's action with space identical with neural architecture search space and rewards are used to guide the process**

(perform distributed training of NAS)





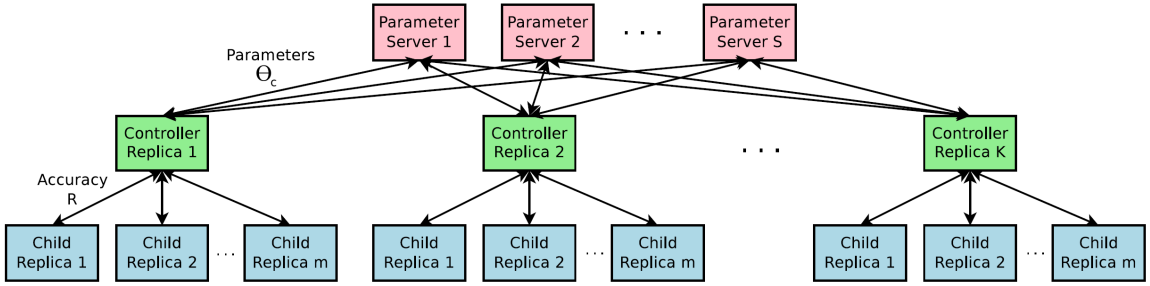




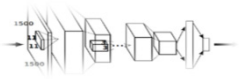
[15] B. Baker et al., 'Designing Neural Network Architectures using Reinforcement Learning', 2017

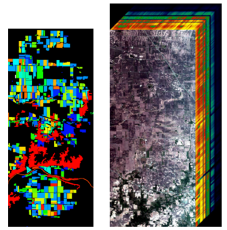
NAS - Using Reinforcement Learning Techniques – Distributed Training

- Distributed training for Neural Architecture Search can use a set of S parameter servers
- Parameter servers store and send parameters to K controller replicas
- Each controller replica then samples m architectures and run the multiple child models in parallel
- Accuracy of each child model is recorded to compute the gradients with respect to parameters
- In turn sent back to the parameter servers



(each controller replica
then samples m architectures)



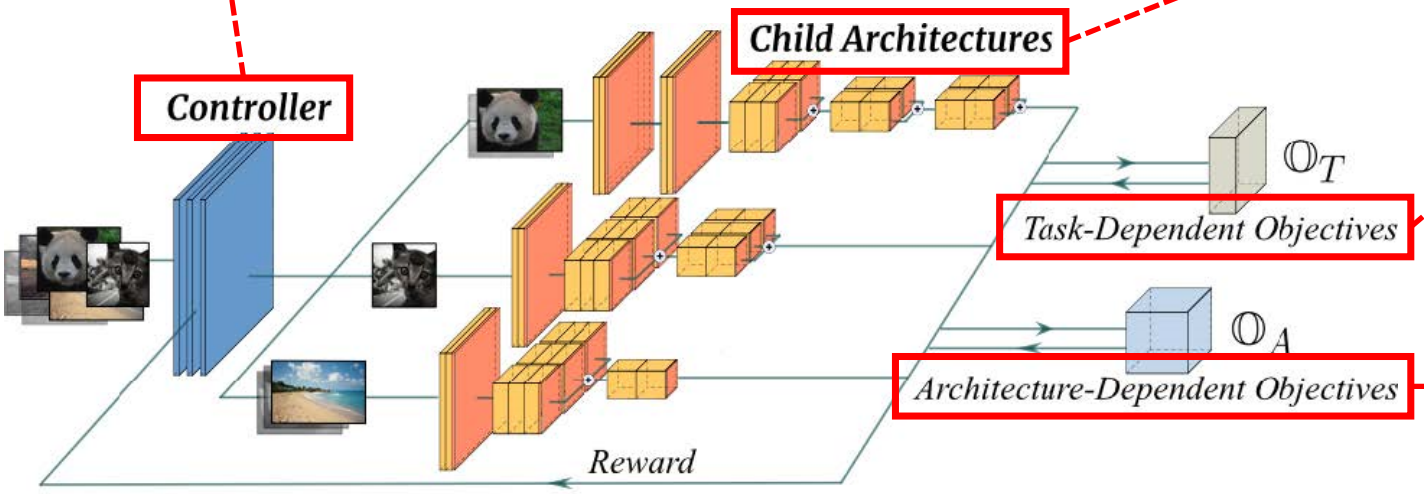


[16] B. Zoph et al., 'Neural Architecture Search with Reinforcement Learning', 2017

Instance-Aware NAS Approach – Using Multiple Neural Networks

- Often a Recurrent Neural Network (RNN) technique that performs the agent steps

- Derived specific architectures that perform good for specific dataset samples

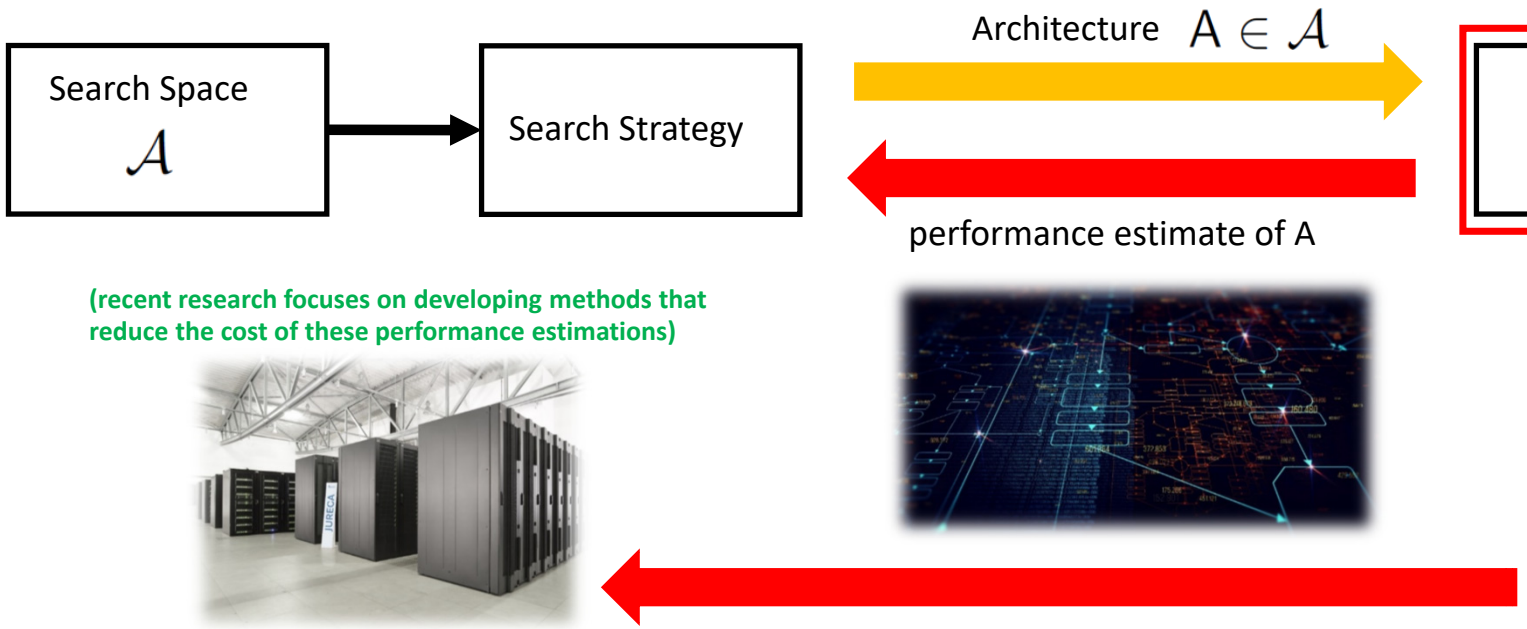


- E.g. what is the accuracy or error rate we obtain as metric to guide the search for specific architectures for specific dataset samples

- E.g. what is the latency of the network for a given dataset sample to guide the search for specific architectures that offer better latency by keeping accuracy(!)

[19] A.C. Cheng et al., 'InstaNAS: Instance-aware Neural Architecture Search', 2018

NAS - Understanding the Performance Estimation Strategy



(recent research focuses on developing methods that reduce the cost of these performance estimations)

Speed-up method	How are speed-ups achieved?
Lower fidelity estimates	Training time reduced by training for fewer epochs, on subset of data, downscaled models, downscaled data, ...
Learning Curve Extrapolation	Training time reduced as performance can be extrapolated after just a few epochs of training.
Weight Inheritance/ Network Morphisms	Instead of training models from scratch, they are warm-started by inheriting weights of, e.g., a parent model.
One-Shot Models/ Weight Sharing	Only the one-shot model needs to be trained; its weights are then shared across different architectures that are just subgraphs of the one-shot model.

[14] T. Elsken et al., Neural Architecture Search: A Survey, Journal of Machine Learning Research, 2019

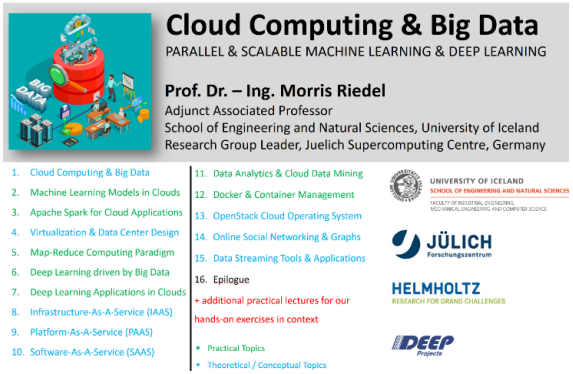
- **Objective of neural architecture search is typically to find architectures that achieve high predictive performance on unseen data**
- **Performance Estimation refers to the process of estimating this performance and the usefulness of the architecture that has been 'found/explored'**
- **Simplest option: perform a standard training and validation of the architecture on data → unfortunately computationally expensive (even with HPC!)**
- **Simplest option thus limits the number of neural network architectures that can be explored or apply a number of speed-up methods**

NAS – Summary and Outlook as new Vibrant Research Field

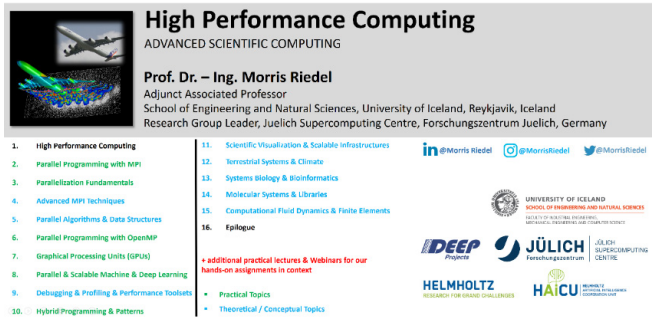
- High Performance Computing & Machine Learning more intertwined today
  - GPUs can significantly speed-up the training of machine and deep learning models
- Recent Deep Learning models have tremendous success in many application areas
  - Pro: Manual feature engineering processes is often automated using automated feature learning
  - Contra: Employed neural network architectures are still often developed manually by human experts
  - Lessons learned: Manual time-consuming and error-prone process shifted to architecture engineering
- Automated Neural Architecture Search
  - Need since there is a growing number of fine-tuned architectures with a high number of hyper-parameters
  - Approaches differ in (a) search space, (b) search strategy, and (c) performance estimation strategy
  - Reinforcement Learning for NAS is just one of the possible search strategies, but a promising technique
  - Overlaps with meta-learning and hyper-parameter optimization approaches and subfield of AutoML

- **The Juelich Supercomputing Centre (JSC) has started to create an environment for NAS and will support this research as part of the Helmholtz AI initiative – open to all researchers interested!**

Trainings at Juelich Supercomputing Centre & Helmholtz AI – NAS/ML/DL/HPC

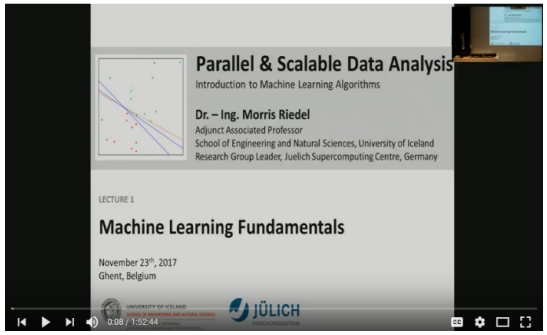


[21] M. Riedel, 'Cloud Computing & Big Data – Parallel & Scalable Machine Learning & Deep Learning', 2018

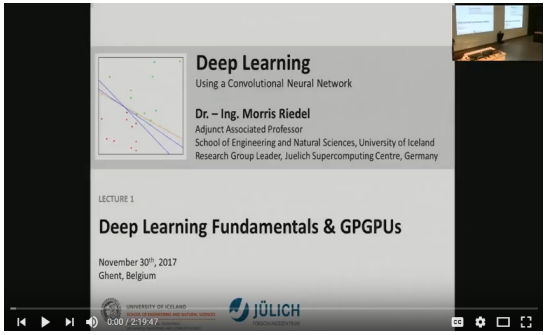


[22] M. Riedel, 'High Performance Computing – Advanced Scientific Computing', 2019

[23] M. Riedel et al., 'DEEP-EST Tutorial: Introduction to Deep Learning', 2018



[24] M. Riedel, 'Introduction to Machine Learning Algorithms', Invited YouTube Lecture, six lectures, University of Ghent, 2017

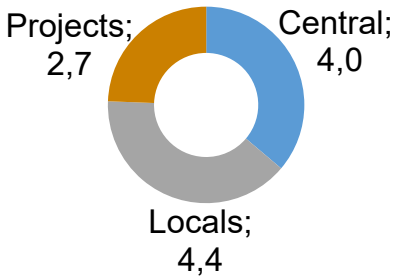


[25] M. Riedel, 'Deep Learning - Using a Convolutional Neural Network', Invited YouTube Lecture, six lectures, University of Ghent, 2017

HELMHOLTZ AI | ARTIFICIAL INTELLIGENCE COOPERATION UNIT

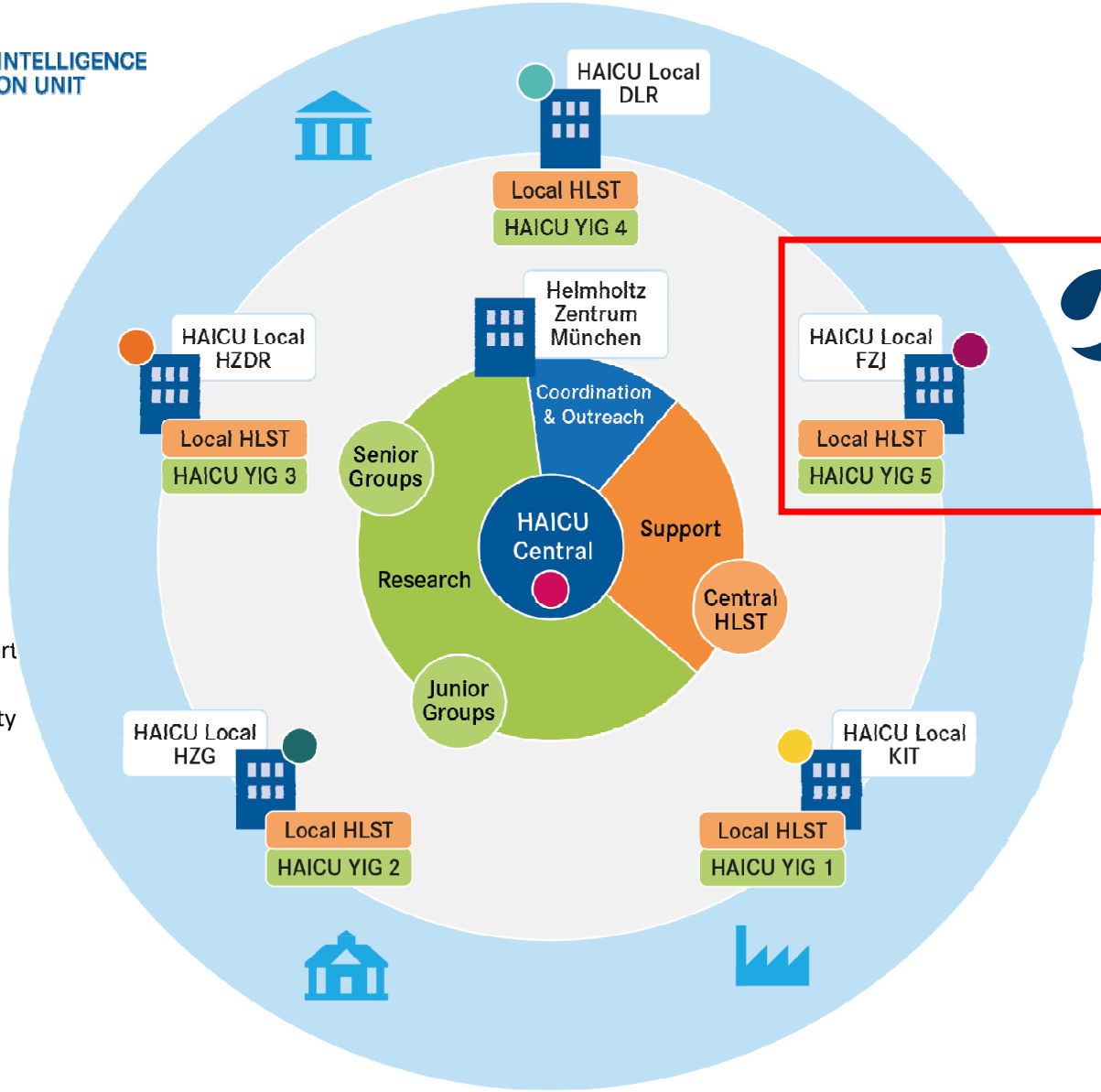
HELMHOLTZAI | ARTIFICIAL INTELLIGENCE COOPERATION UNIT

Overall Helmholtz AI funding (M€/year)



- **Helmholtz AI Central**
  - 5 research groups for applied fundamental research
  - Knowledge transfer through High Level Support Team (HLST)
  - Reach a critical mass and international visibility
- **Helmholtz AI Locals at 5 Helmholtz Centers**
  - Helmholtz AI Young Investigator Groups + HLSTs
  - Domain specific research and translation of expertise into the domain
- **Collaborative Projects of different size**

[26] Helmholtz AI Webpage

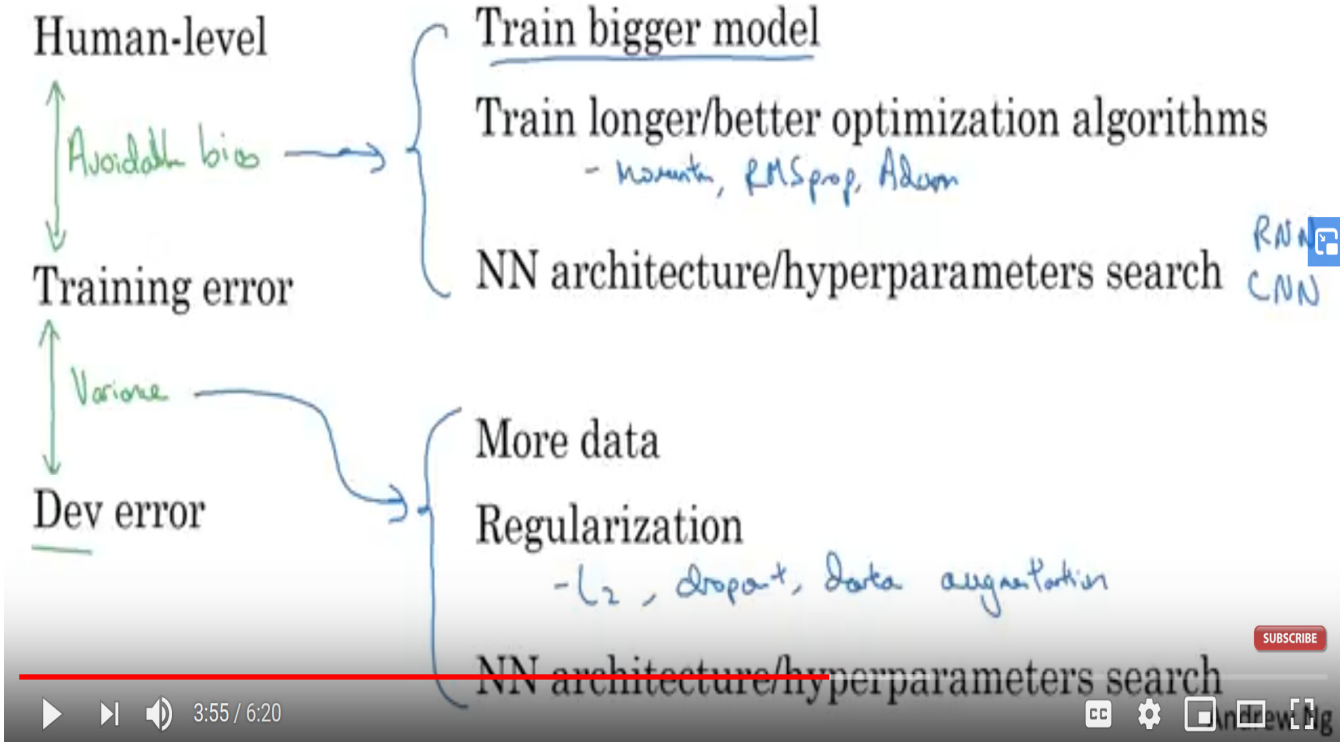


JÜLICH Forschungszentrum

Industry Partners
University
Academic Partners

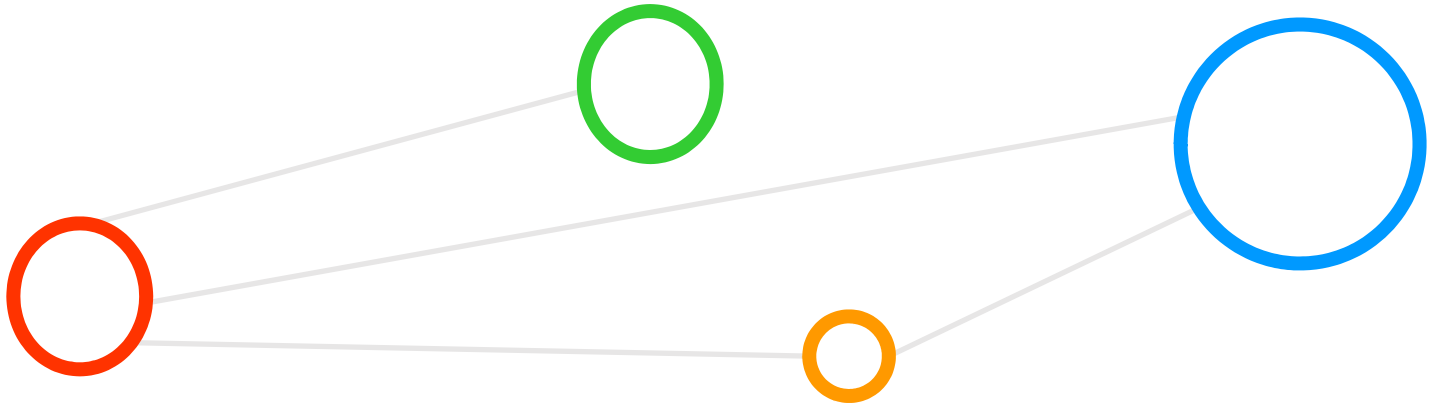
Key Technologies
Matter
Energy
Aeronautics, Space, and Transport
Health
Earth & Environment

[Video] Improving Model Performance in Neural Networks



[20] YouTube Video, Neural Networks - Improving your model performance

Lecture Bibliography



Lecture Bibliography (1)

- [1] G. Ciaburro, 'Keras Reinforcement Learning Projects', book, Packt Publishing, Online: https://www.packtpub.com/eu/big-data-and-business-intelligence/keras-reinforcement-learning-projects
- [2] OpenAI Gym Toolkit, Online: https://gym.openai.com/
- [3] An Introduction to Statistical Learning with Applications in R, Online: http://www-bcf.usc.edu/~gareth/ISL/index.html
- [4] YouTube Video, 'Machine Learning : Model Selection & Cross Validation', Online: http://www.youtube.com/watch?v=hihuMBCuSlU
- [5] G. Cavallaro, M. Riedel, M. Richerzhagen, J. A. Benediktsson and A. Plaza, "On Understanding Big Data Impacts in Remotely Sensed Image Classification Using Support Vector Machine Methods," *in the IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 10, pp. 4634-4646, Oct. 2015, Online: https://www.researchgate.net/publication/282524415_On_Understanding_Big_Data_Impacts_in_Remotely_Sensed_Image_Classification_Using_Support_Vector_Machine_Methods
- [6] SVM and Parameters, Online: https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_15.png
- [7] Wikipedia on 'particle Swarm Optimization', Online: https://en.wikipedia.org/wiki/Particle_swarm_optimization
- [8] M.Goetz, M. Riedel et al.,'HPDBSCAN – Highly Parallel DBSCAN', Proceedings of MLHPC Workshop at Supercomputing 2015, Online: https://www.researchgate.net/publication/301463871_HPDBSCAN_highly_parallel_DBSCAN
- [9] M. Goetz, G. Cavallaro, T. Geraud, M. Book, M. Riedel, 'Parallel Computation of Component Trees on Distributed Memory Machines', Journal of Transactions on Parallel and Distributed Systems, 2018, Online: https://www.researchgate.net/publication/325212343_Parallel_Computation_of_Component_Trees_on_Distributed_Memory_Machines

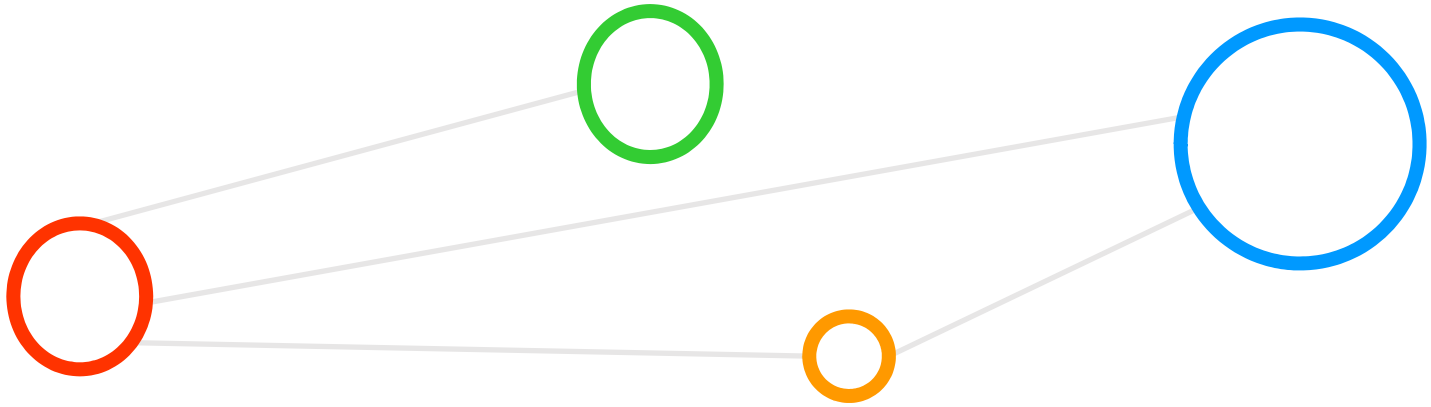
Lecture Bibliography (2)

- [10] J. Lange, G. Cavallaro, M. Goetz, E. Erlingsson, M. Riedel, 'The Influence of Sampling Methods on Pixel-Wise Hyperspectral Image Classification with 3D Convolutional Neural Networks', Proceedings of the IGARSS 2018 Conference, Online: https://www.researchgate.net/publication/328991957_The_Influence_of_Sampling_Methods_on_Pixel-Wise_Hyperspectral_Image_Classification_with_3D_Convolutional_Neural_Networks
- [11] Haut, J.M., Gallardo, J.A., Paoletti, M.E., Cavallaro, G., Plaza, J., Plaza, A., Riedel, M.: Cloud Deep Networks for Hyperspectral Image Analysis, IEEE Transactions on Geoscience and Remote Sensing, PP(99):1-17, 2019, Online: https://www.researchgate.net/publication/335181248_Cloud_Deep_Networks_for_Hyperspectral_Image_Analysis
- [12] Apache Spark, Online: https://spark.apache.org/
- [13] Neural Architecture Search, Morris Riedel Webpage, Online: http://www.morrisriedel.de/neural-architecture-search
- [14] Elsken, T., Metzen, J.H., Hutter, F., Neural Architecture Search: A Survey, Journal of Machine Learning Research, 2019, Online: https://arxiv.org/pdf/1808.05377.pdf
- [15] Baker, B., Gupta, O., Naik, N., Raskar, R., 'Designing Neural Network Architectures using Reinforcement Learning', Online: https://arxiv.org/abs/1611.02167
- [16] Zoph, B., Le, Q.V., 'Neural Architecture Search with Reinforcement Learning', 2017, Online: https://arxiv.org/pdf/1611.01578.pdf
- [17] An Introduction to Statistical Learning with Applications in R, Online: http://www-bcf.usc.edu/~gareth/ISL/index.html
- [18] OpenAI Gym Toolkit, Online: https://gym.openai.com/
- [19] Cheng, A.C, Lin, C.H., Juan, D.C., InstaNAS: Instance-aware Neural Architecture Search, Online: https://arxiv.org/abs/1811.10201

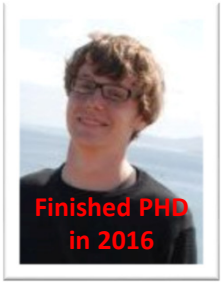
Lecture Bibliography (3)

- [20] YouTube Video, 'Neural Network - Improving your model performance', Online: https://www.youtube.com/watch?v=PxC4B41Ey-I
- [21] M. Riedel, 'Cloud Computing & Big Data – Parallel & Scalable Machine Learning & Deep Learning', 2018, Online: http://www.morrisriedel.de/cloud-computing-and-big-data-course-fall-2018
- [22] M. Riedel, 'High Performance Computing – Advanced Scientific Computing', 2017, Online: http://www.morrisriedel.de/hpc-course-fall-2019
- [23] M. Riedel et al., 'DEEP-EST Tutorial: Introduction to Deep Learning', Online: http://www.morrisriedel.de/deep-est-tutorial-deep-learning
- [24] Morris Riedel, 'Introduction to Machine Learning Algorithms', Invited YouTube Lecture, six lectures & exercises, UoGhent, 2017, Online: https://www.youtube.com/watch?v=KgiuUZ3WeP8&list=PLrmNhuZo9sgbcWtMGN0i6G9HEvh08JG0J
- [25] Morris Riedel, 'Deep Learning - Using a Convolutional Neural Network', Invited YouTube Lecture, six lectures & exercises, UoGhent, 2017, Online: https://www.youtube.com/watch?v=gOL1_YIosYk&list=PLrmNhuZo9sgZUdaZ-f6OHK2yFW1kTS2qF
- [26] Helmholtz AI, Online: https://www.haicu.de/

Acknowledgements



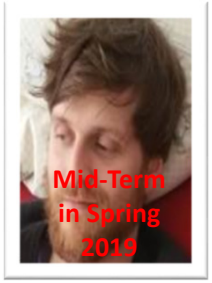
Acknowledgements – High Productivity Data Processing Research Group



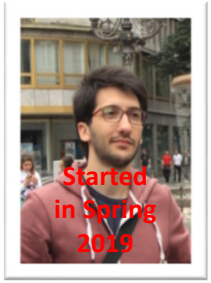
PD Dr.
G. Cavallaro

Senior PhD
Student A.S. Memon

Senior PhD
Student M.S. Memon



PhD Student
E. Erlingsson

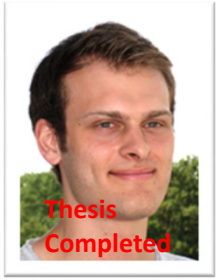


PhD Student
S. Bakarat

PhD Student
R. Sedona

Dr. M. Goetz
(now KIT)

MSc M.
Richerzhagen
(now other division)



MSc
P. Glock
(now INM-1)

MSc
C. Bodenstein
(now
Soccerwatch.tv)

MSc Student
G.S. Guðmundsson
(Landsverkjun)

This research group has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 763558 (DEEP-EST EU Project)


Big Data
Science
Research
science
simulations
methods
Computational
E.g
big
Centre
Computing
climate
nodes
centers
storage
processing
technologies
computing
analysis
computational
research
HPC
Juelich
Scientific
European
TB
Enable
Modelling
Environment
scientific
important
Infrastructure
ENES
EUDAT
increasing
brain
databases
Supercomputing
images
performance
datasets
data-intensive
Service
Concrete
Matters
Landscape
Services
network
management
Climate
modelling
structures
access
Hardware
Technologies
Earth
directory
Structure
General
project
distributed
large-scale
cores
current
hundreds
compute
system
Resources
stored
DLCL
JSC
Project
MapReduce
create
Processing
Health
day
Energy
systems
Simulation
Germany
NASA
Provide
Understanding
DLCLs
System
AR5
Level
use
acquisition
e.g
Work
often
Summary
Large
Using
Methods
IPCC
Common
physical
Fusion
construction
Approaches
expertise
disciplines
results
Key
basis
files
different
PB
community
High
Test
DKRZ
Use
etc
via
GB
analyze
external
levels
Nutshell
network
Policy-based
concepts
funding
fusion
within
measurement
services
new
forms
cross-disciplinary
resources
year
like
Cross-Disciplinary
EU
device
Analysis
Computer
SMAQ
international
engineering
References
model
formats
codes