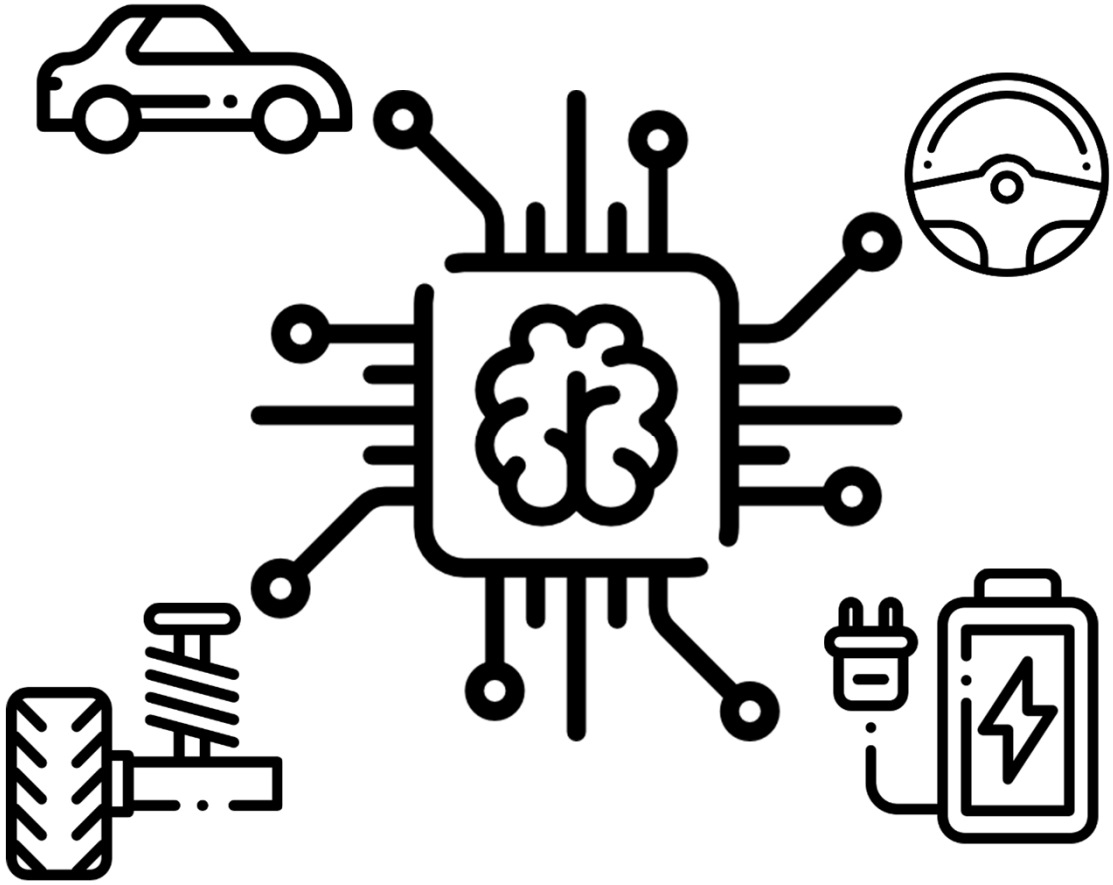
Lehrstuhl für Fahrzeugtechnik
Fakultät für Maschinenwesen
Technische Universität München

Artificial Intelligence in Automotive Technology

Dr. Johannes Betz / Prof. Dr.-Ing. Markus Lienkamp/ Prof. Dr.-Ing. Boris Lohmann

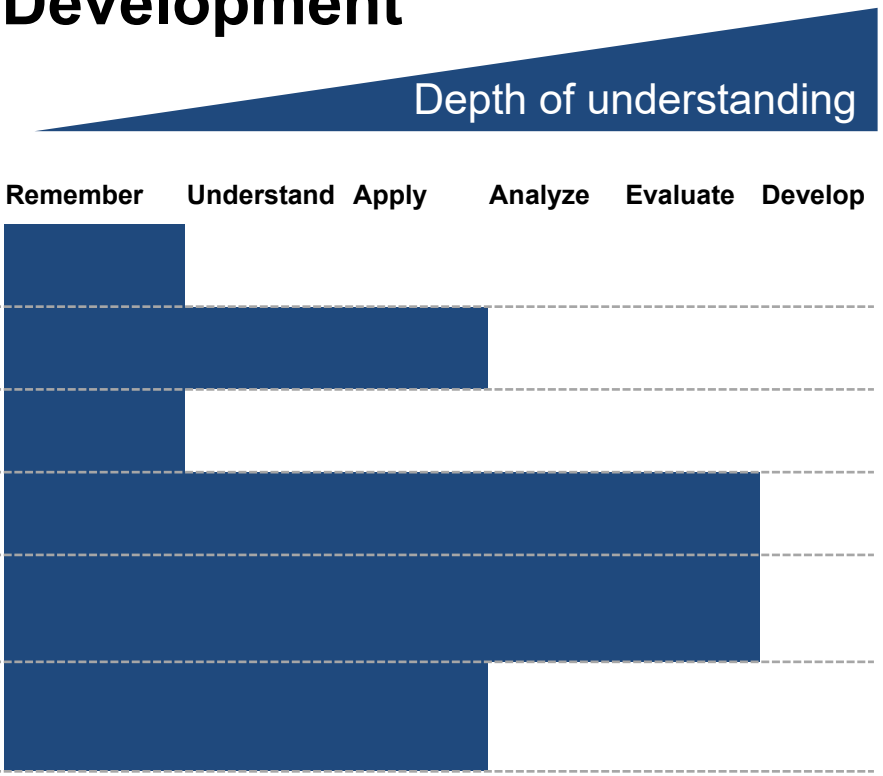


Lecture Overview

Lecture 16:15 – 17:45	Practice 17:45 – 18:30
1 Introduction: Artificial Intelligence 17.10.2019 – Johannes Betz	**Practice 1** 17.10.2019 – Johannes Betz
2 Perception 24.10.2019 – Johannes Betz	**Practice 2** 24.10.2019 – Johannes Betz
3 Supervised Learning: Regression 31.10.2019 – Alexander Wischnewski	**Practice 3** 31.10.2019 – Alexander Wischnewski
4 Supervised Learning: Classification 7.11.2019 – Jan Cedric Mertens	**Practice 4** 7.11.2019 – Jan Cedric Mertens
5 Unsupervised Learning: Clustering 14.11.2019 – Jan Cedric Mertens	**Practice 5** 14.11.2019 – Jan Cedric Mertens
6 Pathfinding: From British Museum to A* 21.11.2019 – Lennart Adenaw	**Practice 6** 21.11.2019 – Lennart Adenaw
7 Introduction: Artificial Neural Networks 28.11.2019 – Lennart Adenaw	**Practice 7** 28.11.2019 – Lennart Adenaw
8 Deep Neural Networks 5.12.2019 – Jean-Michael Georg	**Practice 8** 5.12.2019 – Jean-Michael Georg
9 Convolutional Neural Networks 12.12.2019 – Jean-Michael Georg	**Practice 9** 12.12.2019 – Jean-Michael Georg
10 Recurrent Neural Networks 19.12.2019 – Christian Dengler	**Practice 10** 19.12.2019 – Christian Dengler
11 Reinforcement Learning 09.01.2020 – Christian Dengler	**Practice 11** 09.01.2020 – Christian Dengler
12 AI Development 16.01.2020 – Johannes Betz	**Practice 12** 16.01.2020 – Johannes Betz
13 Guest Lecture: VW Data:Lab 23.01.2020 –	

Objectives for Lecture 12: AI Development

After the lecture you are able to…

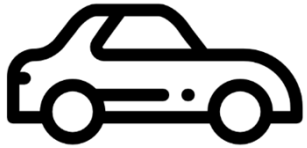


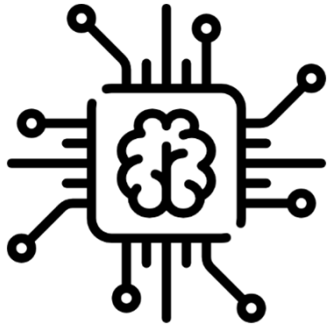
AI Development
Johannes Betz / Prof. Dr. Markus Lienkamp /
Prof. Dr. Boris Lohmann

(Dr. Johannes Betz)

Agenda

1. **Chapter: AI Development Pipeline**
2. Chapter: Transfer Learning
3. Chapter: AI Frameworks
4. Chapter: Data and Labeling
5. Chapter: GPU Computing
6. Chapter: Hyperparameter Tuning
7. Chapter: AI Inference
8. Chapter: Summary





AI Development

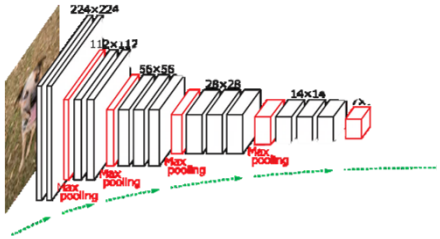
Deep learning

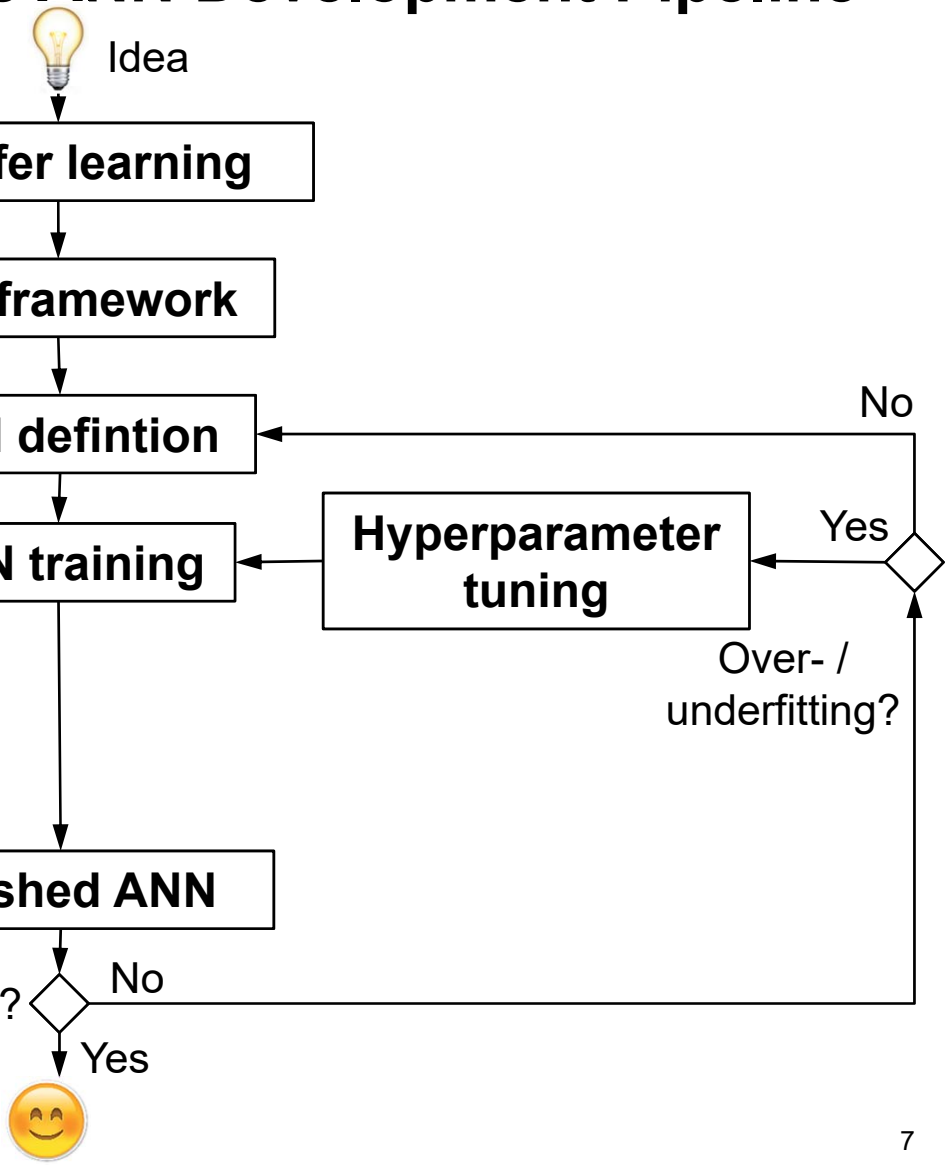
Source: http://p.migdal.pl/2017/04/30/teaching-deep-learning.html

AI Development – General AI Development Pipeline

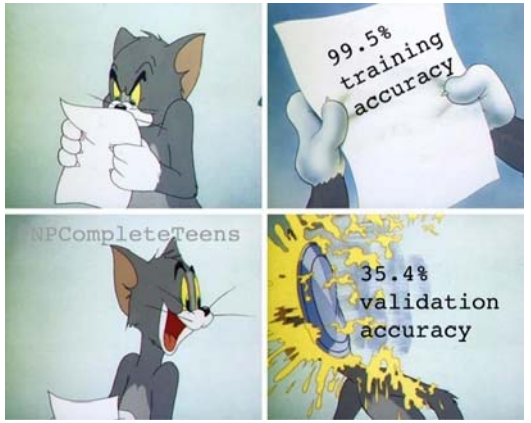
1. What kind of **problem** do I have?
2. What kind of **machine learning method** can I use: Regression, classification or clustering?
3. If I need **deep learning**, what kind of neuronal networks are useful regarding the problem I have?
4. Do I have **enough data** for the problem?
5. Can I **vary the data** I have?
6. Which **programming language** do I have to use?
7. Do I have **small scale GPU power** for first shots?
8. Do I know how to **vary my hyperparameters**?
9. Is my problem too big but scalable so I may use **multiple GPUs**?
10. What hardware do I need for the **inference**?
11. Is my hardware **connectable** with other hardware?

AI Development – Specific ANN Development Pipeline





AI Development – Be aware!



Before you start developing:

1. Think the whole **general pipeline** through first!
2. DL development ≠ DL training ≠ DL inference

General scaling up:

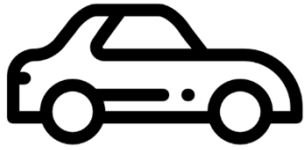
1. Make bigger models: More layers, bigger layers, ...
2. Tackle more data: More data, variation of data, data augmentation, …
3. Reduce research cycle time with fast computing: Parallel computing, GPU usage, …

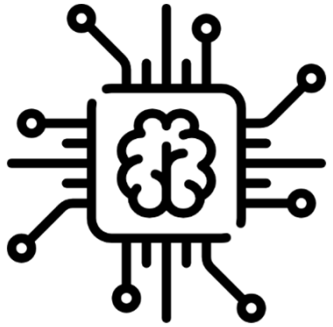
AI Development
Johannes Betz / Prof. Dr. Markus Lienkamp /
Prof. Dr. Boris Lohmann

(Dr. Johannes Betz)

Agenda

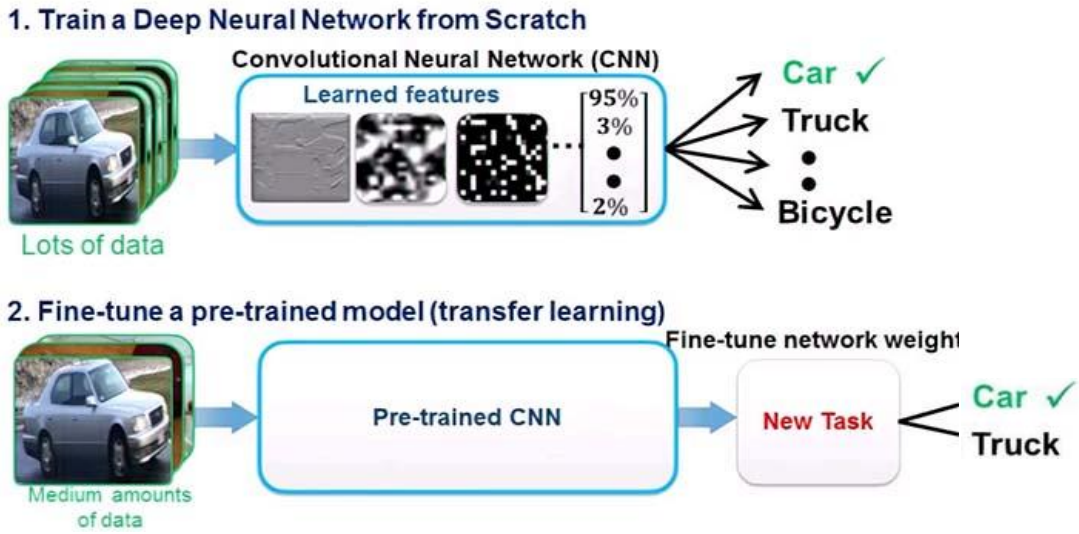
1. Chapter: AI Development Pipeline
2. **Chapter: Transfer Learning**
3. Chapter: AI Frameworks
4. Chapter: Data and Labeling
5. Chapter: GPU Computing
6. Chapter: Hyperparameter Tuning
7. Chapter: AI-Inference
8. Chapter: Summary





Transfer Learning

- When you are tackeling a new problem with an ANN, it might help to look at **existing ANNs** that were built for a similar task
- **Accelerate your work:** People put a lot of effort in developing the architecture and training of their ANN
- Take this ANN and adjust it for your problem

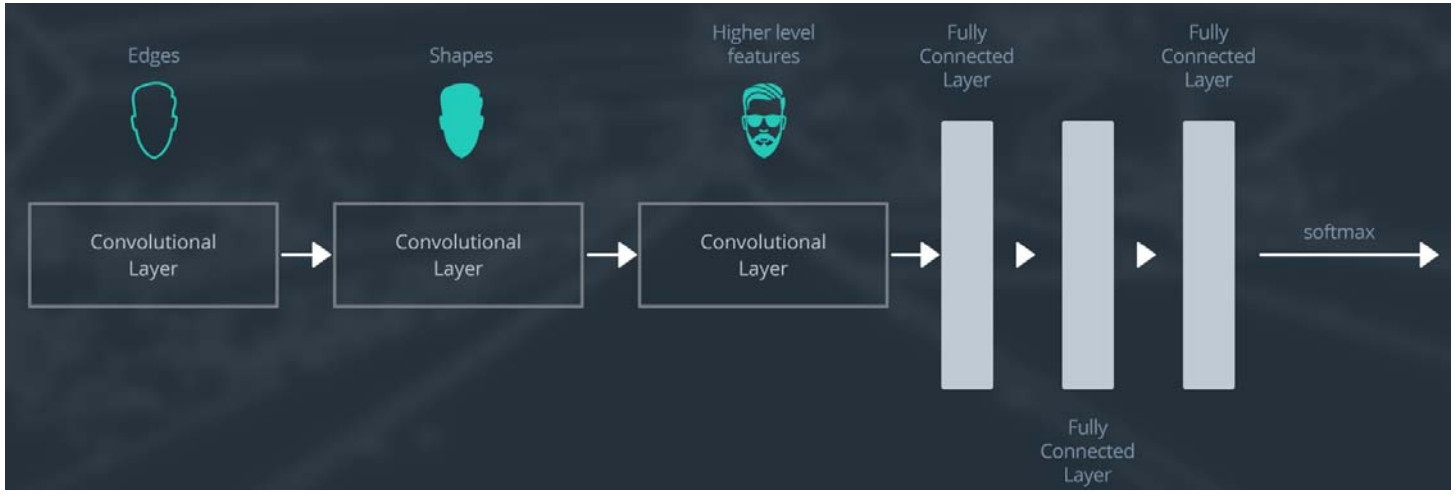


Source: https://de.mathworks.com/videos/training-deep-learning-models-with-transfer-learning-1486670648501.html

Transfer Learning

Examples of well-known neural networks:

- **AlexNet:** CNN for classification
- **VGG:** CNN for classification
- **GoogLeNet:** CNN for classification and detection
- **MobileNet:** Object Detection, object classification, landmark recognition



Example of pretrained CNN

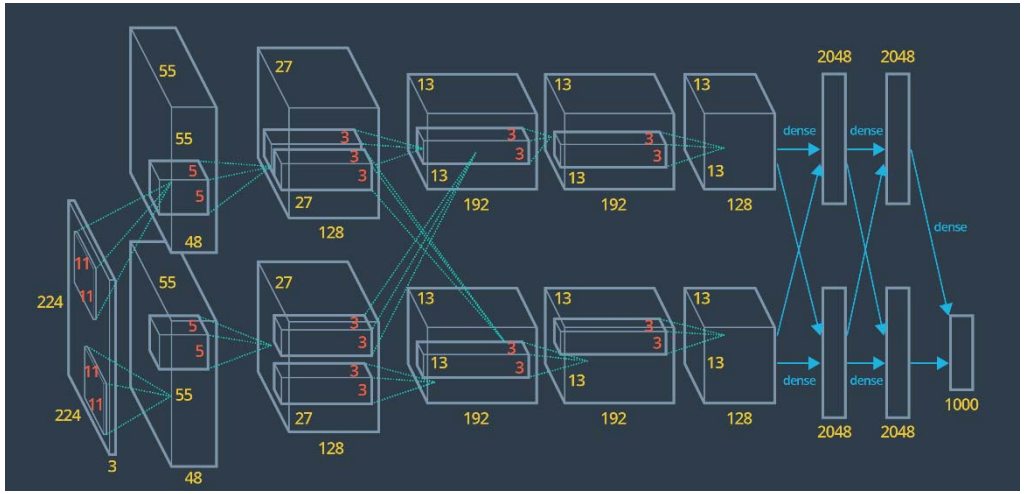
Source: Udacity Self-Driving Car Nanodegree – Part 1, Lesson 15

Additional Slide

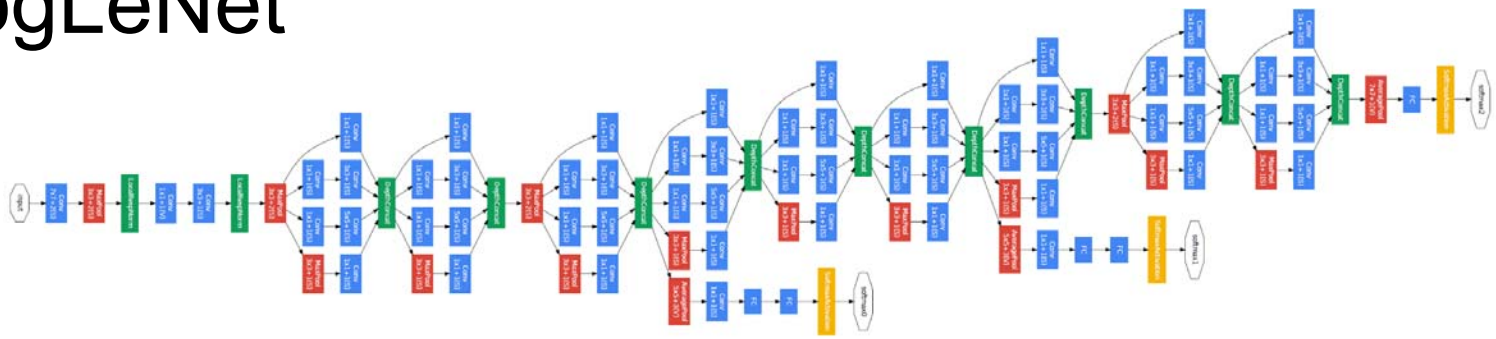
Keep in mind that for a lot of problems you won't need an architecture as complicated and powerful as the neural networks called VGG, Inception or ResNet.

These architectures were made for the task of classifying thousands of complex classes. A smaller network might be a better fit for a smaller problem, especially if you can comfortably train it on moderate hardware.

AlexNet



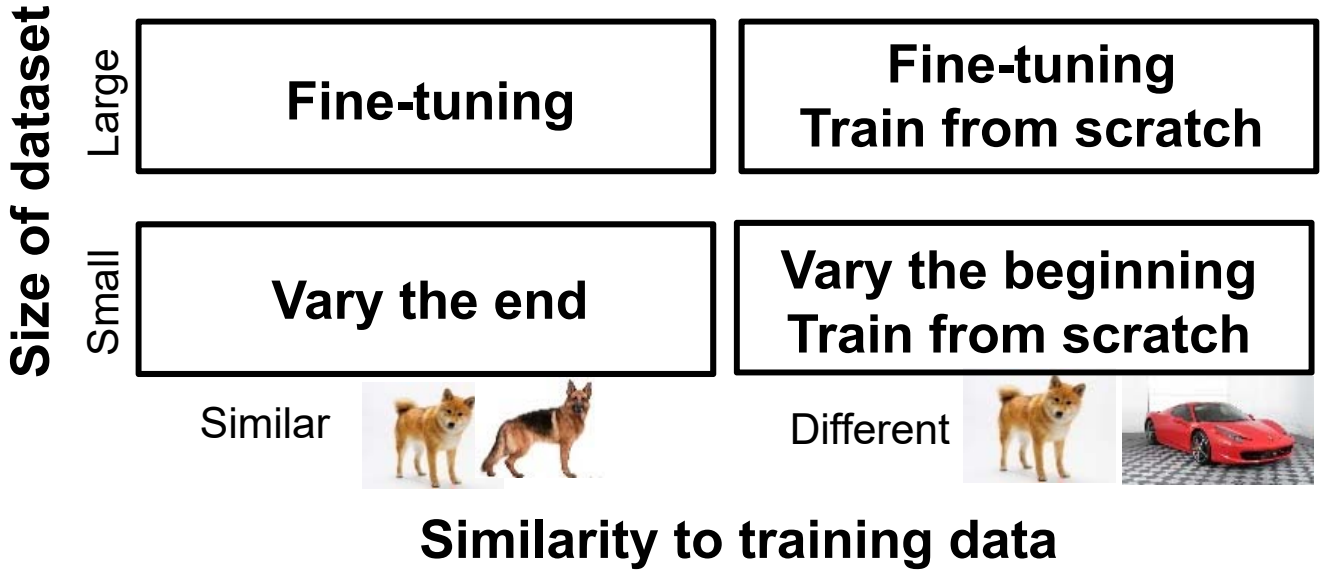
GoogLeNet



Source: Udacity Self-Driving Car Nanodegree – Part 1, Lesson 15 / https://medium.com/coinmonks/paper-review-of-googlenet-inception-v1-winner-of-ilsvlc-2014-image-classification-c2b3565a64e7

Transfer Learning – What Can We Do?

1. **Vary layers:** Vary layers at the end or the beginning of the ANN, the rest of the network remains fixed
2. **Fine-tuning**: Train the entire network end-to-end, start with **pre-trained** weights
3. **Training from scratch:** Train the entire network end-to-end, start with **random** weights

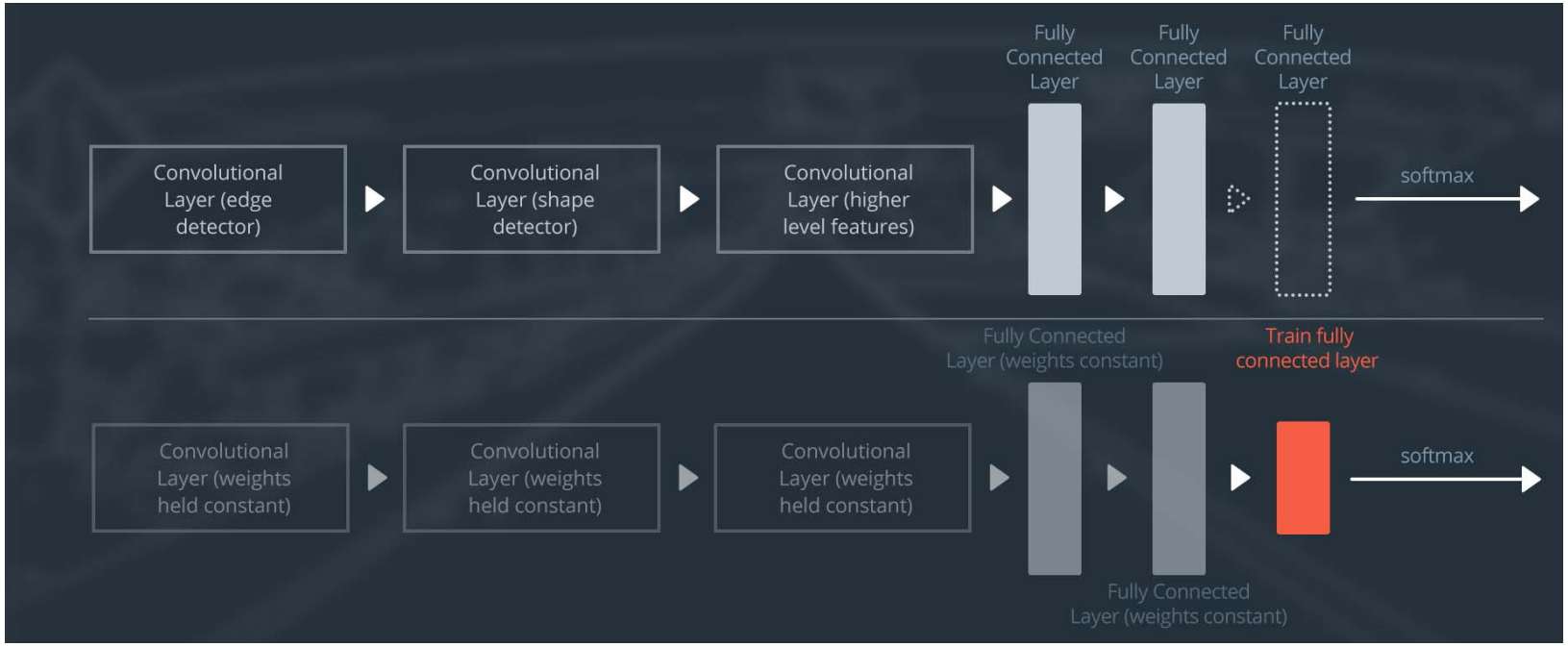


Transfer Learning – Small Dataset And Similar Data

Consider varying the end of the ANN when ...

... the new dataset is small and similar to the original dataset. The higher-level features learned from the original dataset should transfer well to the new dataset.

New ANN | Old ANN



Source: Udacity Self-Driving Car Nanodegree – Part 1, Lesson 15

Additional Slide

If the new dataset is **small and similar** to the original training data:

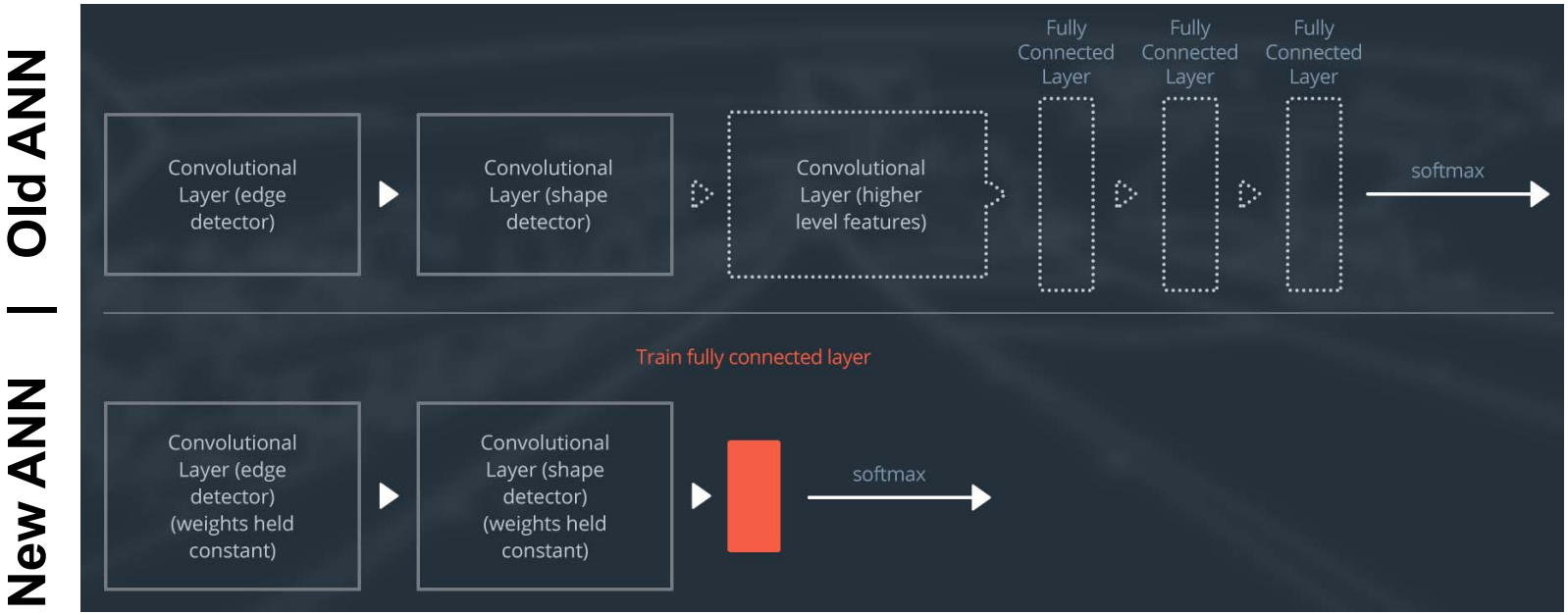
- slice off the end of the neural network,
- add a new fully-connected layer that matches the number of classes in the new dataset,
- randomize the weights of the new fully-connected layer; freeze all the weights from the pre-trained network and
- train the network to update the weights of the new fully-connected layer.

To avoid overfitting on the small dataset, the weights of the original network will be held constant rather than being re-trained.

Since the datasets are similar, images from each dataset will have similar higher level features. Therefore, most or all layers of the pre-trained neural network already contain relevant information about the new dataset and should be kept.

Transfer Learning - Small Dataset And Different Data

Consider varying the beginning of the ANN or train from scratch when ...
…the new dataset is small and different from the original dataset. You could also make the case for training from scratch. In this case, we will only use features from the first few layers of the pre-trained network
→ features from the final layers of the pre-trained network might be **too specific** for the original dataset.



Source: Udacity Self-Driving Car Nanodegree – Part 1, Lesson 15

Additional Slide

If the new dataset is **small and different** from the original training data:

- slice off most of the pre-trained layers near the beginning of the network,
- add to the remaining pre-trained layers a new fully-connected layer that matches the number of classes in the new dataset,
- randomize the weights of the new fully-connected layer; freeze all the weights from the pre-trained network and
- train the network to update the weights of the new fully-connected layer.

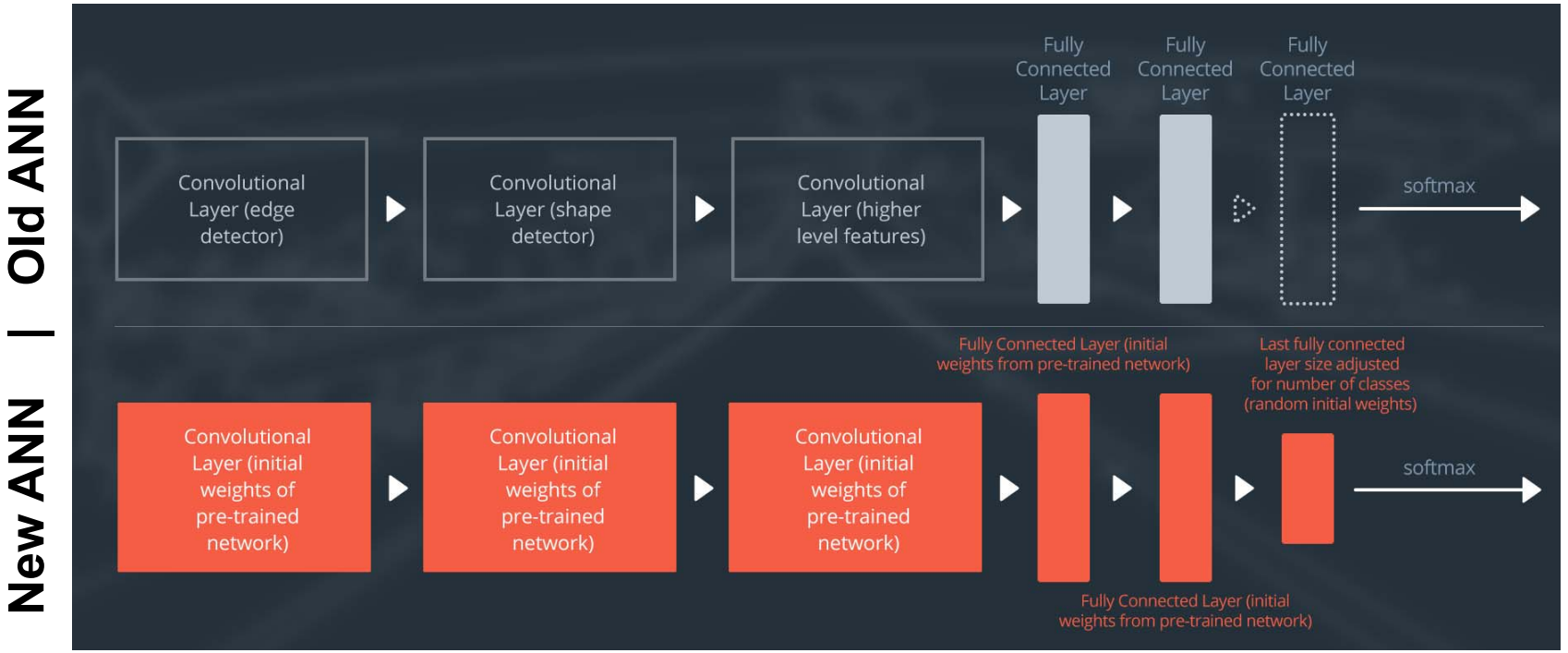
Because the dataset is small, overfitting is still a concern. The weights of the original neural network will be held constant to combat overfitting, like in the first case.

But the original training set and the new dataset do not share similar higher level features. In this case, the new network will only use the layers containing lower level features.

Transfer Learning - Large Data Set Ans Similar Data

Consider fine-tuning when ...

... the new dataset is large and similar to the original dataset. Altering the original weights should be safe because the network is unlikely to overfit the new, large dataset.



Source: Udacity Self-Driving Car Nanodegree – Part 1, Lesson 15

Additional Slide

If the new data set is **large and similar** to the original training data:

- remove the last fully-connected layer and replace with a layer matching the number of classes in the new dataset,
- randomly initialize the weights of the new fully-connected layer,
- initialize the rest of the weights using the pre-trained weights and
- re-train the entire neural network.

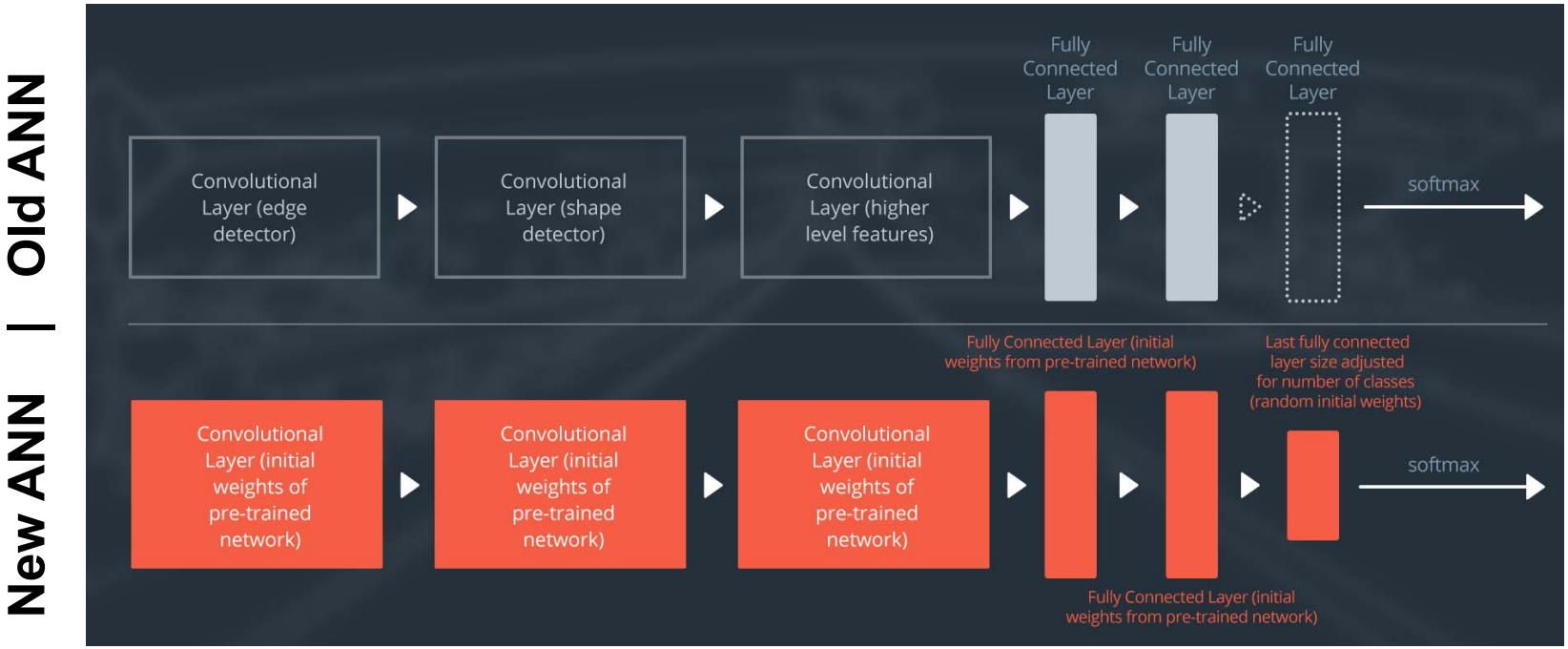
Overfitting is not as much of a concern when training on large datasets and you can therefore re-train all weights of the neural network.

Since the original training set and the new dataset share same higher level features, the entire neural network is used as well.

Transfer Learning – Large Data Set & Different Data

Consider training from scratch or fine-tuning when ...

... the dataset is large and very different from the original dataset. In this case, we have enough data to confidently train from scratch. However, even in this case, it might be beneficial to initialize the entire network with pre-trained weights and fine-tune it on the new dataset.



Source: Udacity Self-Driving Car Nanodegree – Part 1, Lesson 15

Additional Slide

If the new dataset is **large and different** from the original training data:

- remove the last fully-connected layer and replace with a layer matching the number of classes in the new dataset and
- retrain the network from scratch with randomly initialized weights,
- alternatively, you could just use the same strategy as the "large and similar" data case.

Even though the dataset is different from training data, initializing the weights of the pre-trained network might make training faster. This case is therefore exactly the same as the case with a large and similar dataset.

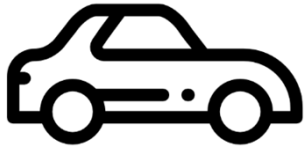
If using the pre-trained network as a starting point does not produce a successful model, another option is to randomly initialize the convolutional neural network weights and train the network from scratch.

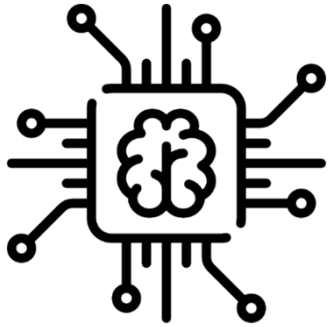
AI Development
Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann

(Dr. Johannes Betz)

Agenda

1. Chapter: AI Development Pipeline
2. Chapter: Transfer Learning
3. **Chapter: AI Frameworks**
4. Chapter: Data and Labeling
5. Chapter: GPU Computing
6. Chapter: Hyperparameter Tuning
7. Chapter: AI-Inference
8. Chapter: Summary





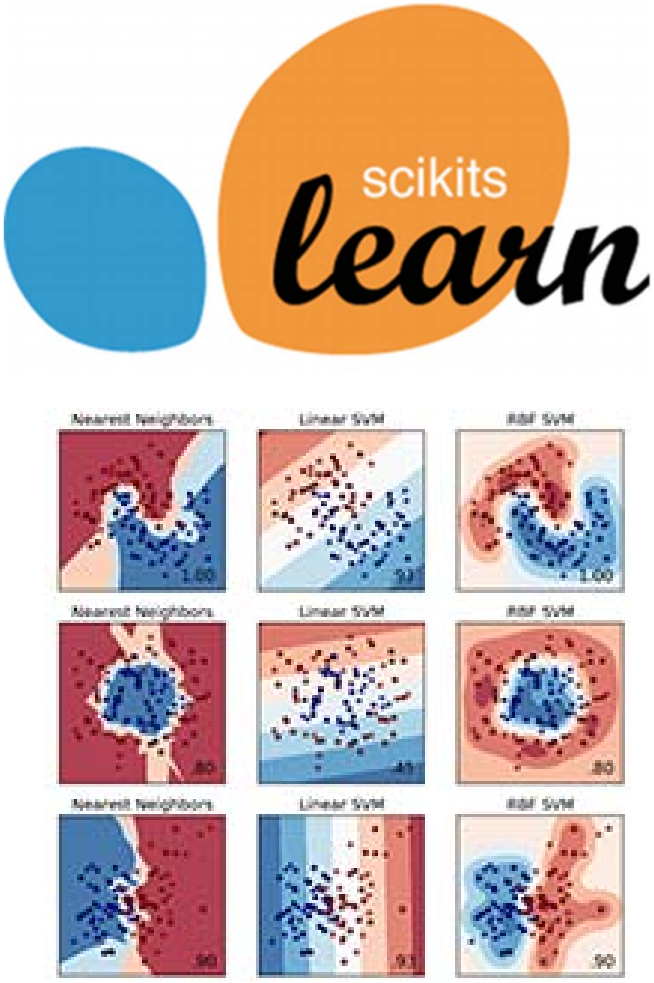
AI Frameworks – What Is That ?

- AI software can be developed from scratch, but it is tediously and complex
- A lot of people put time and effort in the development of computer programs, application programming interfaces (APIs), libraries and software frameworks, which make the development much easier

  → **USE THEM !!!!!**
- A **software framework** is an abstraction in which software providing generic functionality can be selectively changed by additional user-written code
  - A software framework provides a standard way to build and deploy applications
  - Better then normal libraries: Inversion of control, extensibility, non modifiable framework code

AI Frameworks – Scikit-Learn Library

- Free software framework
- Machine learning library
- **Language:** Python, C, C++
- **Operating system:** Linux, macOS, Windows
- **Includes:** Clustering, regression, clustering with algorithms like SVM, nearest neighbors, Gaussian process, decision trees
- **Pros:** Everything you need, good documentation, powerful, GPU boost
- **Cons:** Not for hardcore statistics, limited in parameters



Source: https://scikit-learn.org/stable/

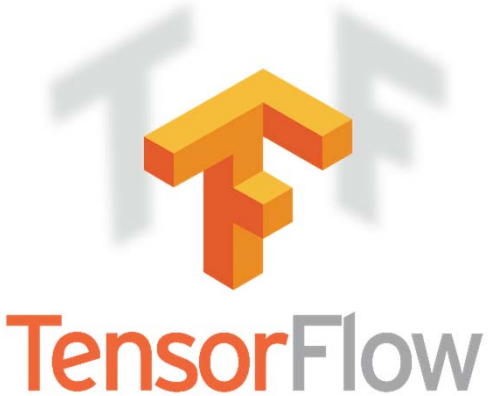
AI Frameworks – Matlab

- Commercial Software (free for **students**)
- Machine and deep learning toolbox
- **Language:** Matlab, Simulink
- **Operating system:** Linux, macOS, Windows
- **Includes:**
  - Machine learning
  - Deep learing
- **Pros:** Easy to use, good documentation, GPU boost
- **Cons:** Closed environment, performance

Source: https://www.cbcity.de/portfolio/matlab-simulink

AI Frameworks – Tensorflow

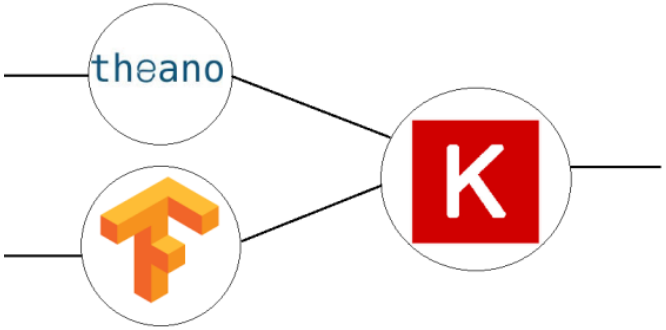
- Free software framework
- Deep learning software framework
- **Language:** Python, C++
- **Operating System:** Linux, macOS, Windows
- **Includes:** CNN, RNN → voice and image recognition
- **Pros:** High performance, multiple GPU, connects research and production, true portability, tensorboard for visualization, good documentation
- **Cons:** Hard to learn in comparison to other frameworks



Source: https://www.tensorflow.org/

AI Frameworks – Keras

- Free software framework
- Neural network library
- **Language:** Python
- **Operating System:** Linux, macOS, Windows
- **Includes:** Neural network interface for CNN and RNN – Speach recognition, image classification
- **Pros:** Makes Tensorflow and Theano easier to use, keras in tf2.0 integrated, easy prototyping, fully configurable models, GPU support
- **Cons:** It might be too high-level and not always easy to customize



Source: https://keras.io/

AI Frameworks – Caffe and Caffe 2

- Free software framework
- Deep learning software framework
- **Language:** C, C++, Python, MATLAB
- **Operating System:** Linux, macOS, Windows
- **Includes:** CNN
- **Pros:** Pre-trained networks available, fast and scalable, GPU usage
- **Cons:** Only few input formats and one output format, no exact layer definition like Tensorflow

Caffe

Berkeley
UNIVERSITY OF CALIFORNIA

Caffe2

facebook

Source: https://caffe2.ai/

AI Frameworks – mxnet

- Free software framework
- Deep learning software framework
- **Language:** Python, C++, Javascript, R, ….
- **Operating System:** Linux, macOS, Windows
- **Includes:** CNN, RNN, LSTM
- **Pros:** Fast and flexible, GPU support, can run on any device (productivity), highly scalable, different languages
- **Cons:** Small community

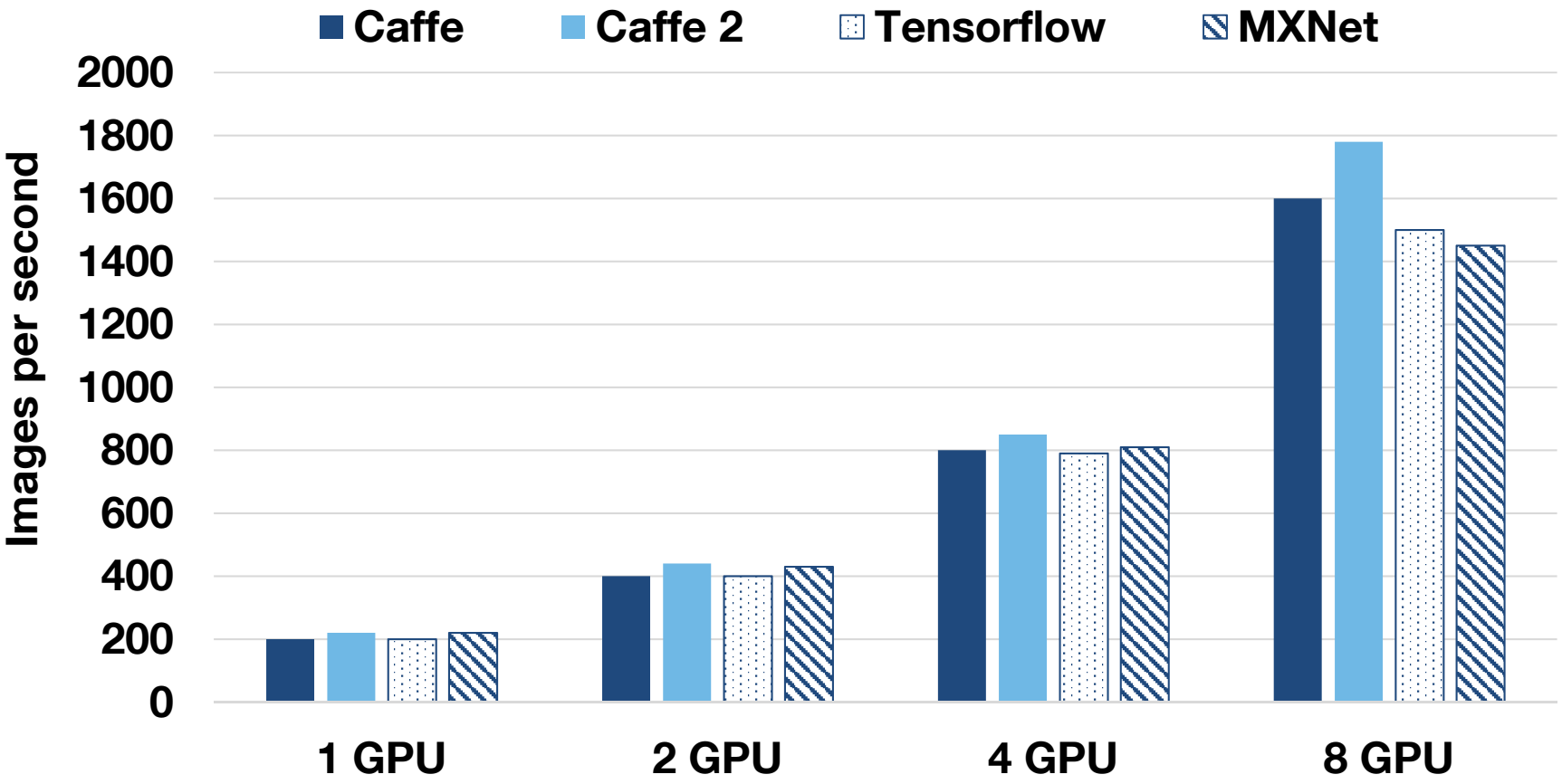
Source: https://mxnet.apache.org/

Additional Slide

	Languages	Tutorials and training materials	CNN modeling capability	RNN modeling capability	Architecture: easy-to-use and modular front end	Speed	Multiple GPU support	Keras compatible
Theano	Python, C++	++	++	++	+	++	+	+
Tensor-Flow	Python	+++	+++	++	+++	++	++	+
Torch	Lua, Python (new)	+	+++	++	++	+++	++	
Caffe	C++	+	++		+	+	+	
MXNet	R, Python, Julia, Scala	++	++	+	++	++	+++	
Neon	Python	+	++	+	+	++	+	
CNTK	C++	+	+	+++	+	++	+	

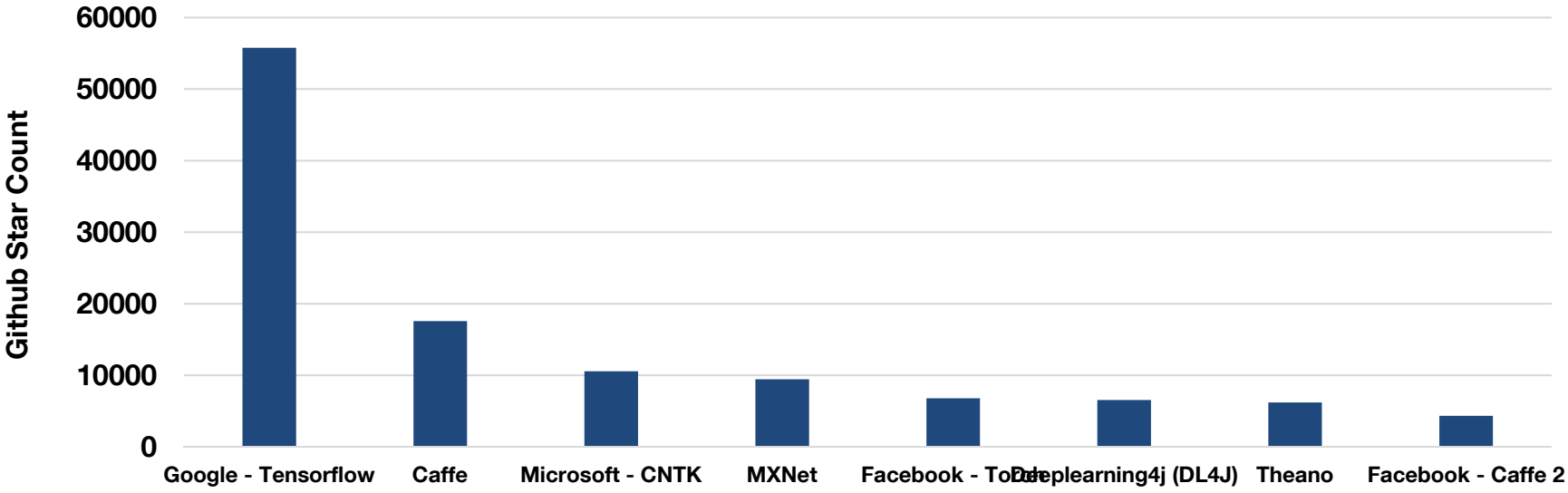
Quelle: https://www.datasciencecentral.com/profiles/blogs/open-source-deep-learning-frameworks-and-visual-analytics

AI Frameworks - Comparison



RESNET-50 FP32 Performance

AI Frameworks - Comparison



new contributors from 2017-02-11 to 2017-04-12			new forks from 2017-02-11 to 2017-04-12	
#1:	131	tensorflow/tensorflow	4192	tensorflow/tensorflow
#2:	63	fchollet/keras	991	fchollet/keras
#3:	51	pytorch/pytorch	810	BVLC/caffe
#4:	49	dmlc/mxnet	517	deeplearning4j/deeplearning4j
#5:	18	Theano/Theano	414	dmlc/mxnet
#6:	11	BVLC/caffe	307	pytorch/pytorch
#7:	11	Microsoft/CNTK	244	Microsoft/CNTK
#8:	9	tflearn/tflearn	211	tflearn/tflearn
#9:	9	pfnet/chainer	134	torch/torch7
#10:	8	torch/torch7	131	Theano/Theano
#11:	5	deeplearning4j/deeplearning4j	116	baidu/paddle
#12:	4	NVIDIA/DIGITS	88	NVIDIA/DIGITS
#13:	3	baidu/paddle	55	pfnet/chainer

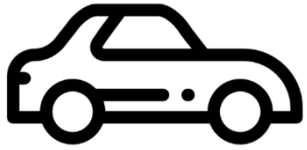
Source: https://www.cio.com/article/3193689/artificial-intelligence/which-deep-learning-network-is-best-for-you.html / http://p.migdal.pl/2017/04/30/teaching-deep-learning.html

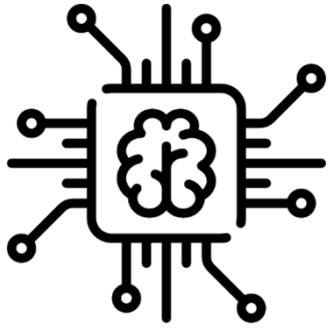
AI Development
Johannes Betz / Prof. Dr. Markus Lienkamp /
Prof. Dr. Boris Lohmann

(Dr. Johannes Betz)

Agenda

1. Chapter: AI-Development Pipeline
2. Chapter: Transfer Learning
3. Chapter: AI-Frameworks
4. **Chapter: Data and Labeling**
5. Chapter: GPU Computing
6. Chapter: Hyperparameter Tuning
7. Chapter: AI-Inference
8. Chapter: Summary

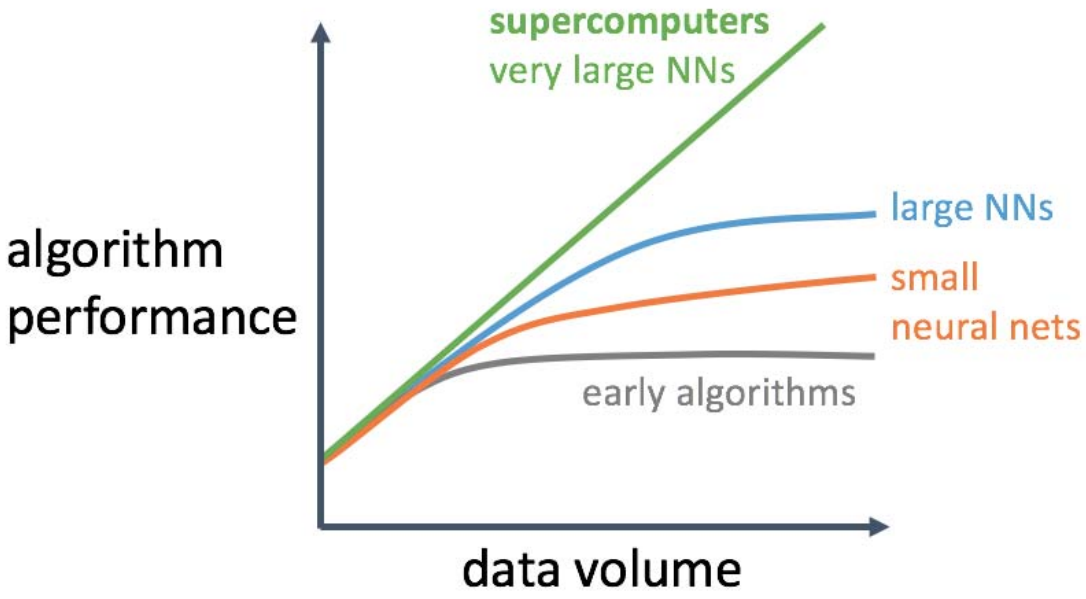




Data

Data is the crucial part for machine learning:

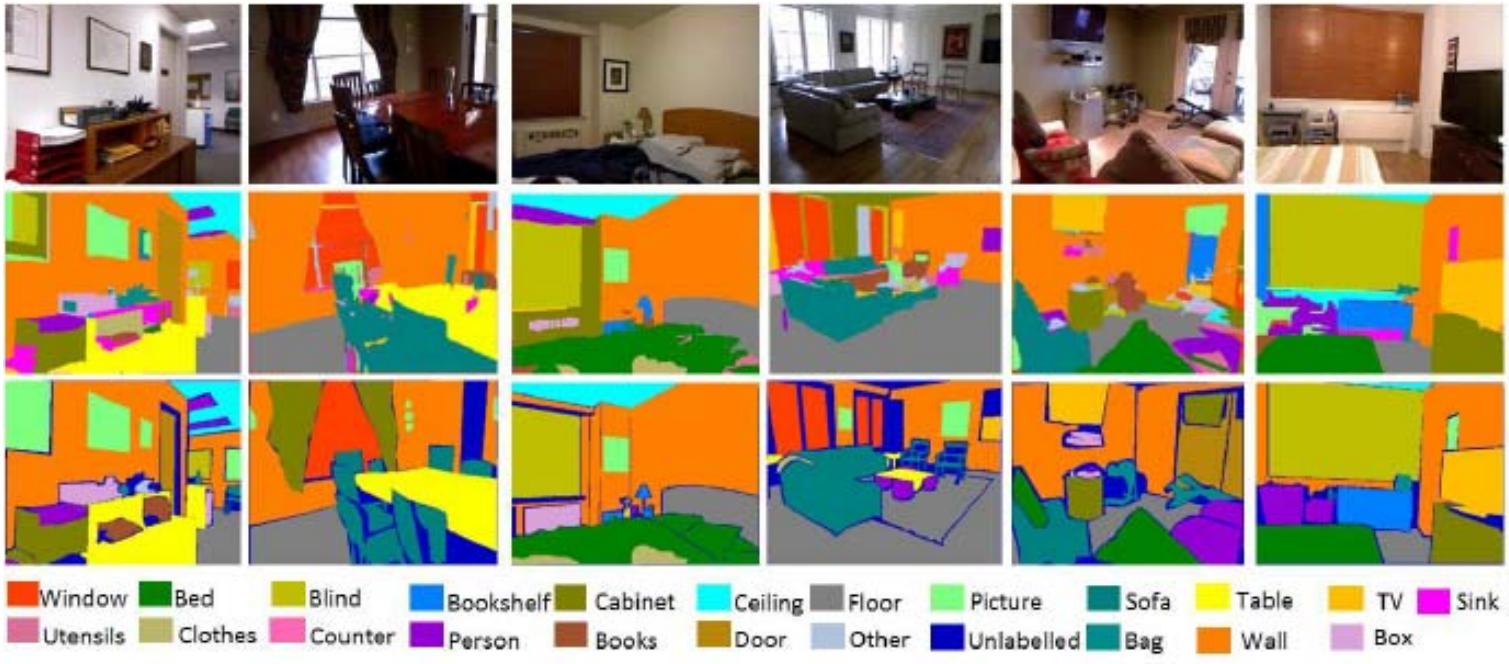
- We need **data** for training our algorithms
- We need **labeled data** for training our algorithms
- We need **more and different data** for not overfitting our training
- We need **even more data** for good regression, clustering, classification



Source: http://visualdl.paddlepaddle.org/docs/develop/documentation/fluid/en/design/memory/memory_optimization.html

Data – Labeled Datesets

- We need data that is labeled for supervised learning, (clustering: unsupervised)
- Label = What does the data include?
- The more specific, the better the regression, classification, clustering



Source: https://www.researchgate.net/publication/274780609_Geometry_Driven_Semantic_Labeling_of_Indoor_Scenes/figures?lo=1

Data – Labeled Datesets

1. Search for datasets online, other people have done a lot in this area
   - **Cityscapes:** Pixel based label of streets
   - **Kitti Dataset:** Pixeld based label of over 5000 pictures
   - **Berkeley Deep Drive:** Labeled pictures of streets, GPS locations, IMU data, …

2. Create your own dataset
   - Aggregate your data
   - Label the data with the information you want to be detected later
   - Takes a lot of time

Source: https://www.cityscapes-dataset.com/examples/

Additional Slide

Berkeley DeepDrive - Explore 100,000 HD video sequences of over 1,100-hour driving experience across many different times in the day, weather conditions, and driving scenarios. Our video sequences also include GPS locations, IMU data, and timestamps.

Udacity - Udacity driving datasets released for Udacity Challenges. Contains ROSBAG training data (~ 80 GB).

Comma.ai - 7 and a quarter hours of largely highway driving. Consists of 10 videos clips of variable size recorded at 20 Hz with a camera mounted on the windshield of an Acura ILX 2016. In parallel to the videos, also recorded some measurements such as car's speed, acceleration, steering angle, GPS coordinates, gyroscope angles. These measurements are transformed into a uniform 100 Hz time base.

KITTI Vision Benchmark Suite - 6 hours of traffic scenarios at 10-100 Hz using a variety of sensor modalities such as high-resolution color and grayscale stereo cameras, a Velodyne 3D laser scanner and a high-precision GPS/IMU inertial navigation system.

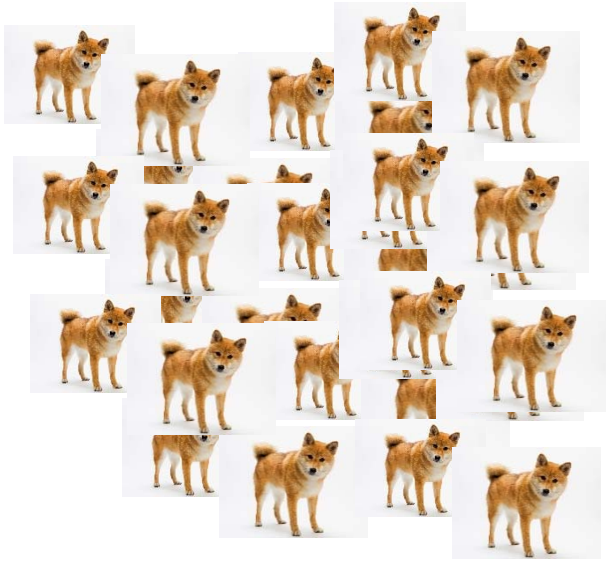
University of Michigan North Campus Long-Term Vision and LIDAR Dataset - consists of omnidirectional imagery, 3D lidar, planar lidar, GPS, and proprioceptive sensors for odometry collected using a Segway robot. pedestrian, cyclist, lanemarking are integrated

Cityscape Dataset - focuses on semantic understanding of urban street scenes. Large-scale dataset that contains a diverse set of stereo video sequences recorded in street scenes from 50 different cities, with high quality pixel-level annotations of 5 000 frames in addition to a larger set of 20 000 weakly annotated frames. The dataset is thus an order of magnitude larger than similar previous attempts. Details on annotated classes and examples of our annotations are available.

MIT AGE Lab - a small sample of the 1,000+ hours of multi-sensor driving datasets collected at AgeLab.

Data – Check Your Dataset!

Is there a bias in your dataset? If yes, it will impact the training!



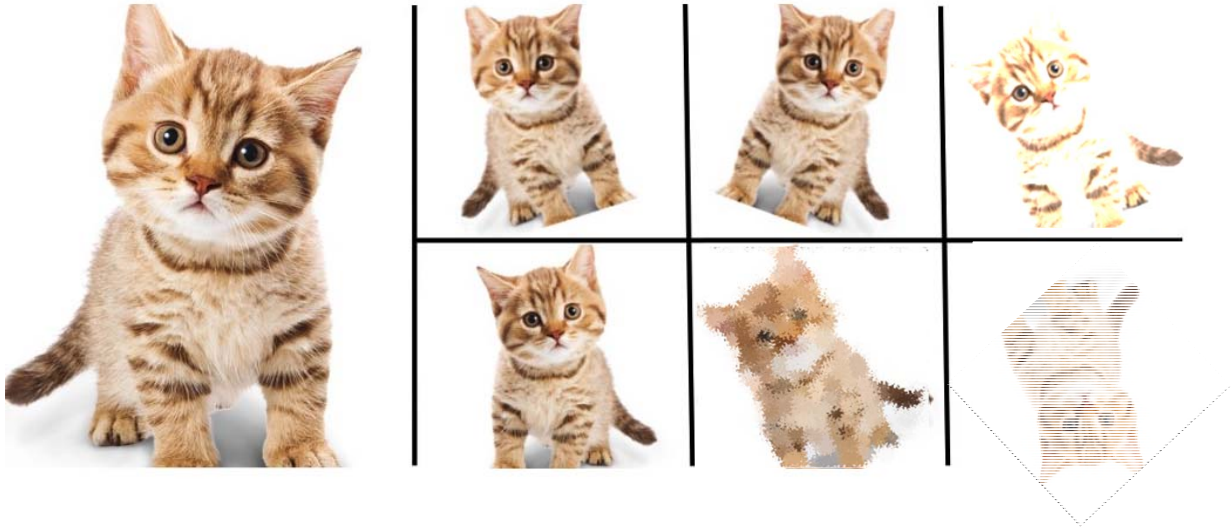
1000 dogs

1 cat

Source: https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

Data – More Data With Data Augmentation

How to use deep learning when you have limited data?



Enlarge your available dataset with **data augmentation**

Source: https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

Data – More Data With Data Augmentation

1. **Flip** images

2. **Rotate** images

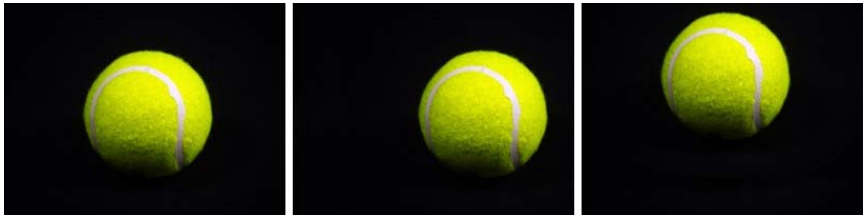
3. **Scale** images outward or inward

Source: https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

Data – More Data With Data Augmentation

4. **Crop** images

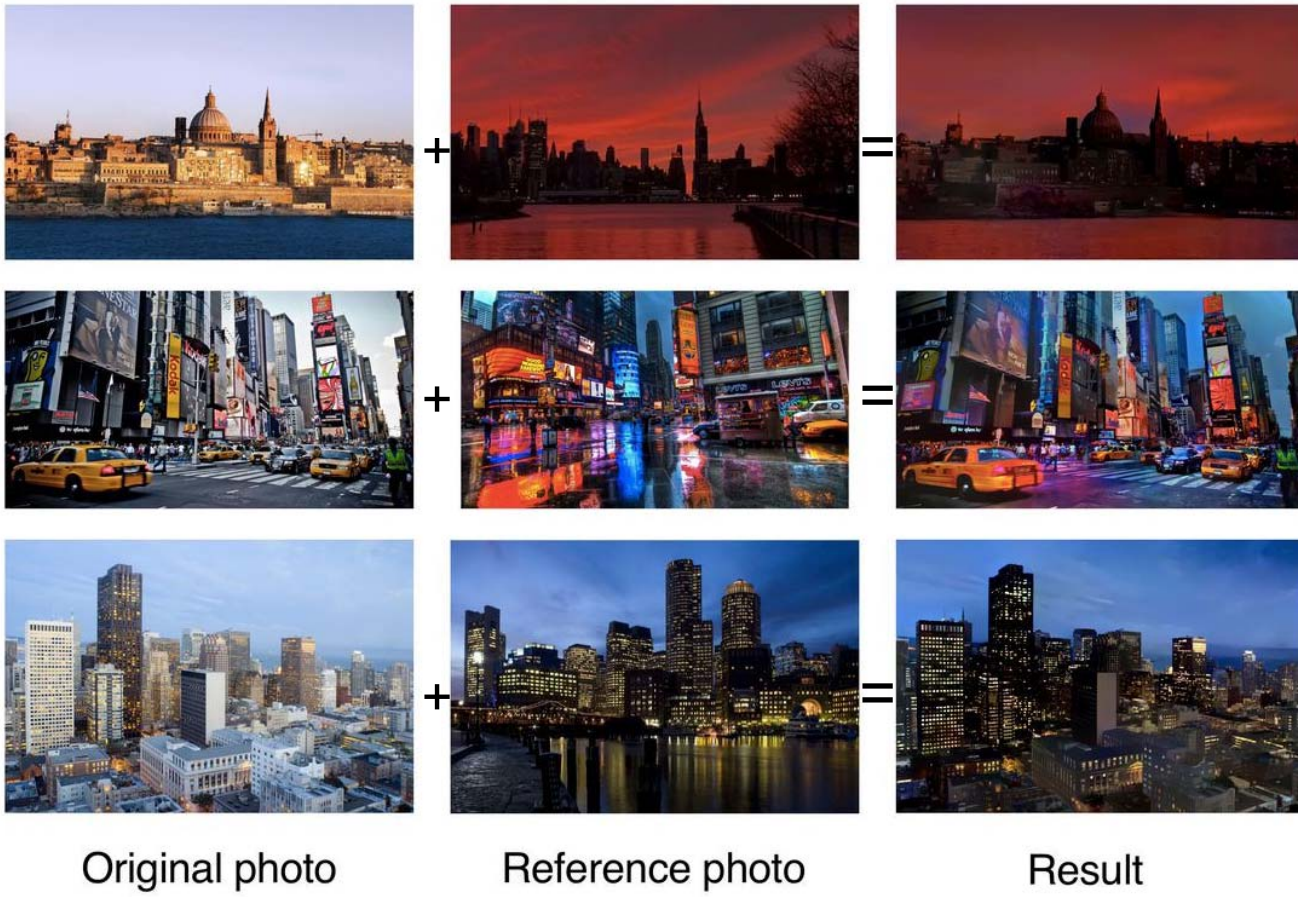
5. **Translation** of objects in x,y-position



6. Add **Gaussian Noise**

Source: https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

Data – More Data With Data Augmentation



7. **Deep photo style transfer:** Transform an image from one domain to an image of another domain using deep learing

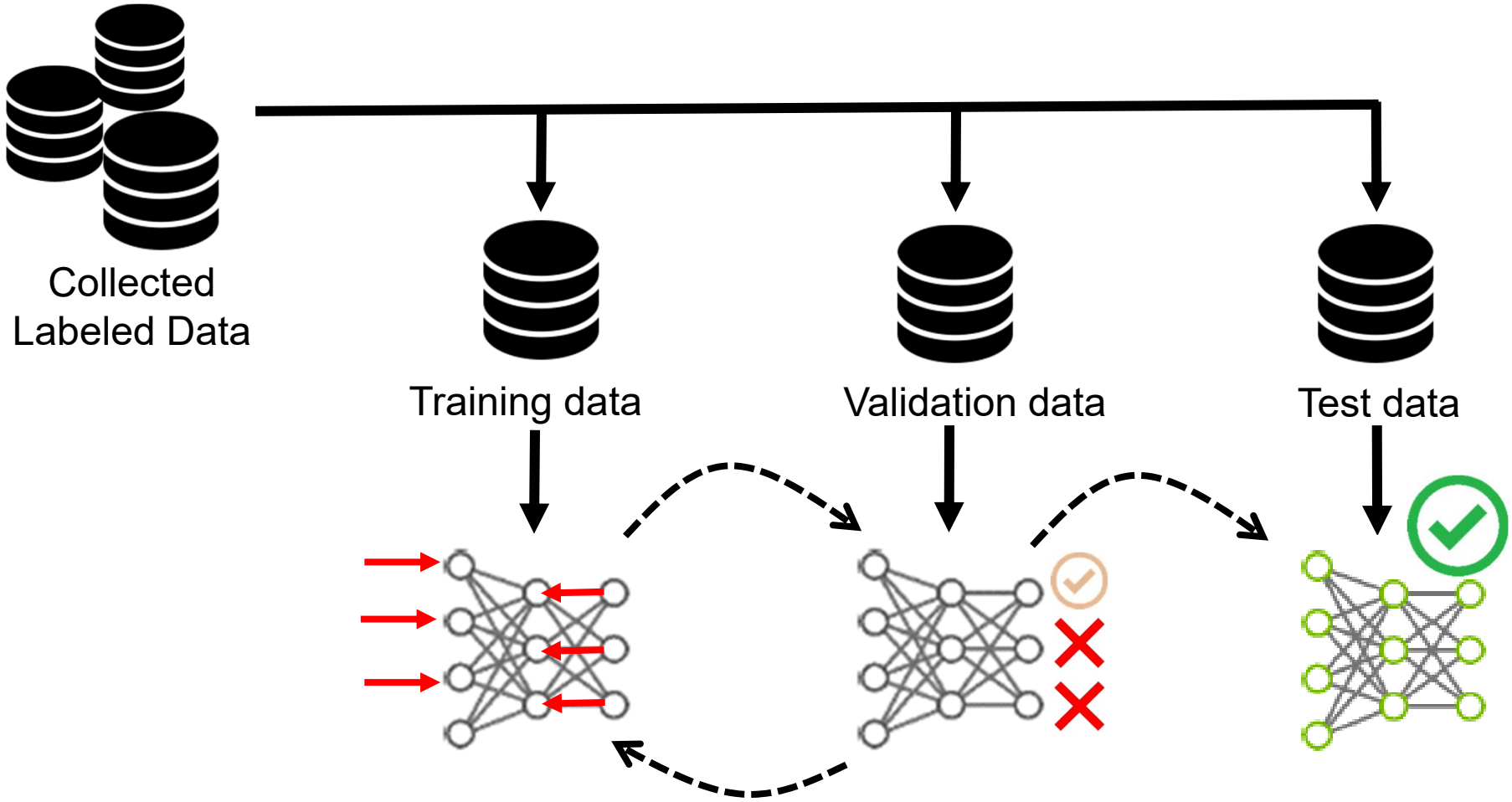
Source: https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

Data – Improvements With Data Augmentation

	Top-1 Accuracy	Top-5 Accuracy
Baseline	48.13 ± 0.42%	64.50 ± 0.65%
Flipping	49.73 ± 1.13%	67.36 ± 1.38%
Rotating	50.80 ± 0.63%	69.41 ± 0.48%
Cropping	**61.95 ± 1.01%**	**79.10 ± 0.80%**
Color Jittering	**49.57 ± 0.53%**	67.18 ± 0.42%
Edge Enhancement	49.29 ± 1.16%	66.49 ± 0.84%
Fancy PCA	49.41 ± 0.84%	**67.54 ± 1.01%**

Source: https://arxiv.org/pdf/1708.06020.pdf

Data – Provide the Data of Your Dataset



Split your dataset into training (60%), validation (20%) and test (20%) sets

Additional Slide

Difference between training, validation and test dataset

Normally, you need two types of datasets to perform supervised learning:

1. You have the input data in one dataset (your "gold standard") together with the correct/expected output. This dataset is usually duly prepared either by humans or by collecting some data in semi-automated way. But it is important that you have the expected output for every data row here, because you need labels for supervised learning.
2. The data you are going to apply your model to. In many cases, this is the data where you are interested in the output of your model and thus you don't have any "expected" outputs yet.

You do the following while performing machine learning :

1. Training phase: You present data from your "gold standard" and train your model by pairing the input with the expected output.

2. Validation/test phase: In order to estimate how well your model has been trained (which depends upon the size of your data, the value you would like to predict, input, etc.) and to estimate model properties (mean error for numeric predictors, classification errors for classifiers, recall and precision for IR models, etc.)

3. Application phase: Now you apply your freshly developed model to the real-world data to results. Since you normally don't have any reference values for this type of data (otherwise, why would you need your model?), you can only speculate about the quality of your model output by using the results of the validation phase.

The validation phase is often split into two parts:

1. In the first part, you just look at your models and select the best performing approach regarding the validation data (= validation)
2. Then estimate the accuracy of the chosen approach (= test).

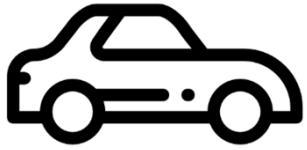
Training set -> to fit the parameters, e.g., weights
Validation set -> to tune the hyperparameters, e.g., architecture
Test set -> to assess the performance on unseen data, e.g., generalization and predictive power

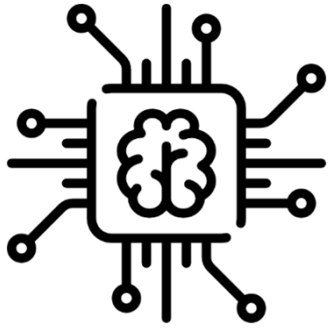
AI Development
Johannes Betz / Prof. Dr. Markus Lienkamp /
Prof. Dr. Boris Lohmann

(Dr. Johannes Betz)

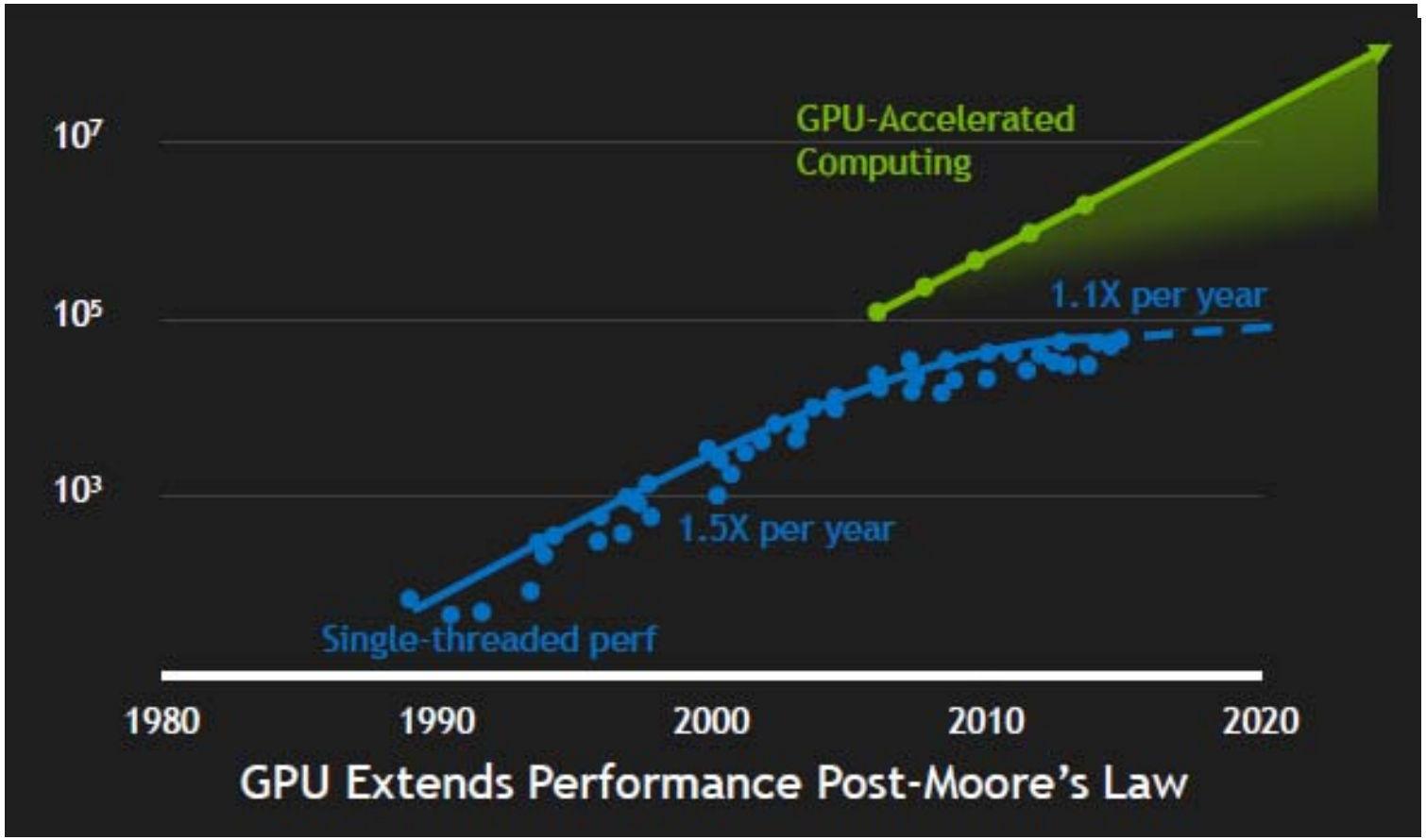
Agenda

1. Chapter: AI Development Pipeline
2. Chapter: Transfer Learning
3. Chapter: AI Frameworks
4. Chapter: Data and Labeling
5. **Chapter: GPU Computing**
6. Chapter: Hyperparameter Tuning
7. Chapter: AI Inference
8. Chapter: Summary





GPU Computing – What is a GPU?



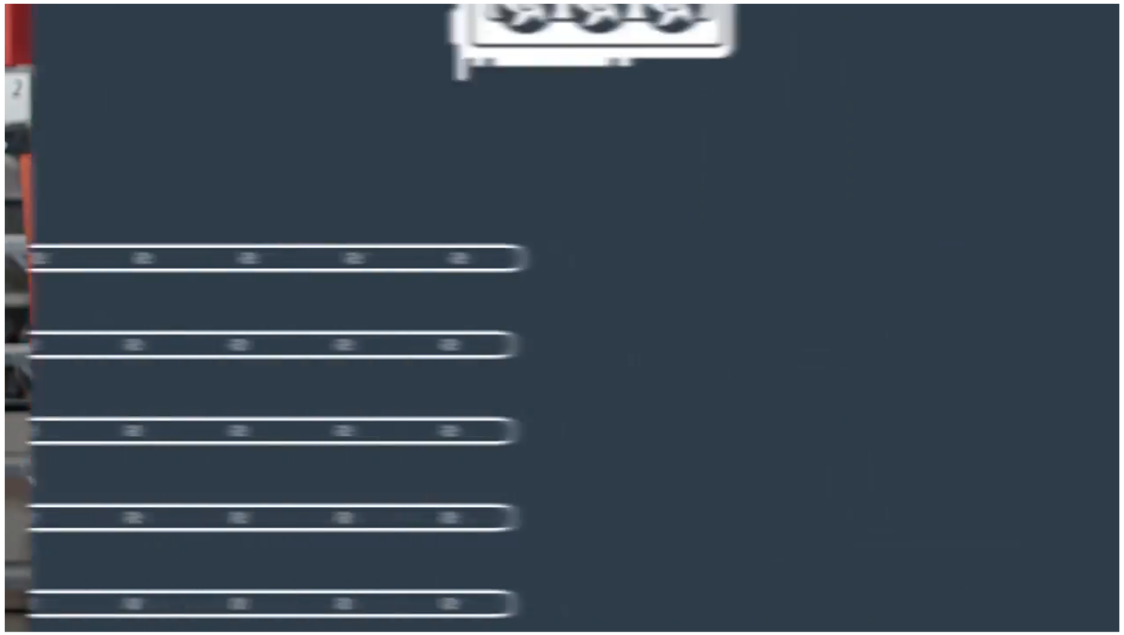
Source: https://www.fedscoop.com/era-ai-computing/

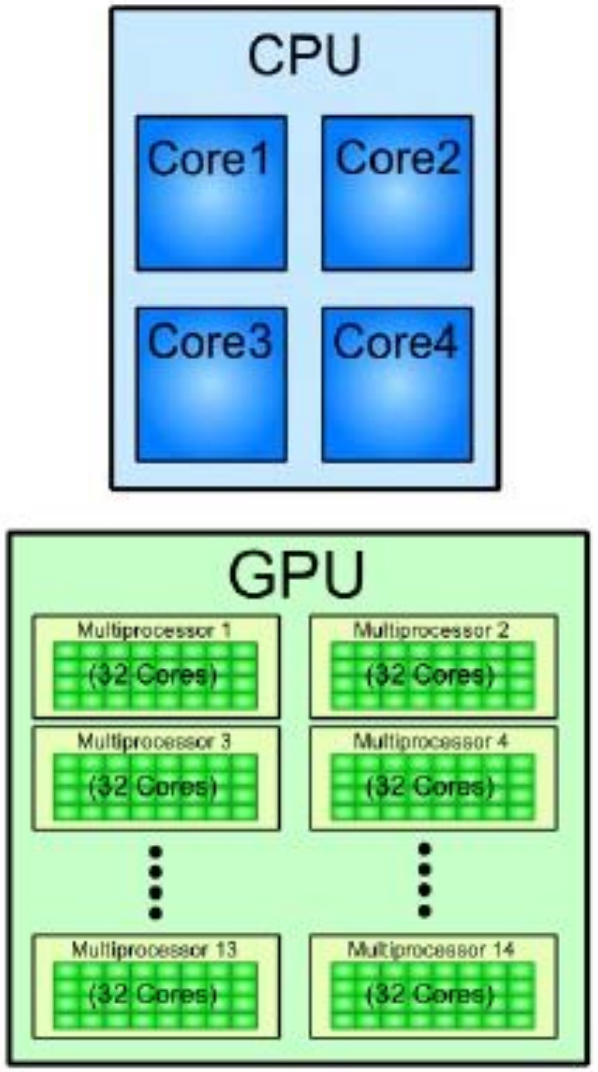
GPU Computing – What is a GPU?

- A GPU is a **graphical processing unit**
- Specialized electronic circuit design
- Rapidly manipulate and alter memory to **accelerate the creation of images in a frame buffer**
- By 2012, GPUs had evolved into highly **parallel multi-core systems** allowing very efficient manipulation of large blocks of data.
- This design is **more effective** than general-purpose central processing units (CPUs) for algorithms in situations where processing large blocks of data is done in parallel, such as:
  - Push-relabel maximum flow algorithm
  - Fast sort algorithms of large lists
  - Two-dimensional fast wavelet transform
  - Molecular dynamics simulations
  - Artificial neural networks

Source: https://www.vrnerds.de/nvidia-pascal-ab-27-mai/ ; https://www.notebookcheck.net/Like-AMD-Nvidia-promises-to-ramp-up-GPU-production.282024.0.html

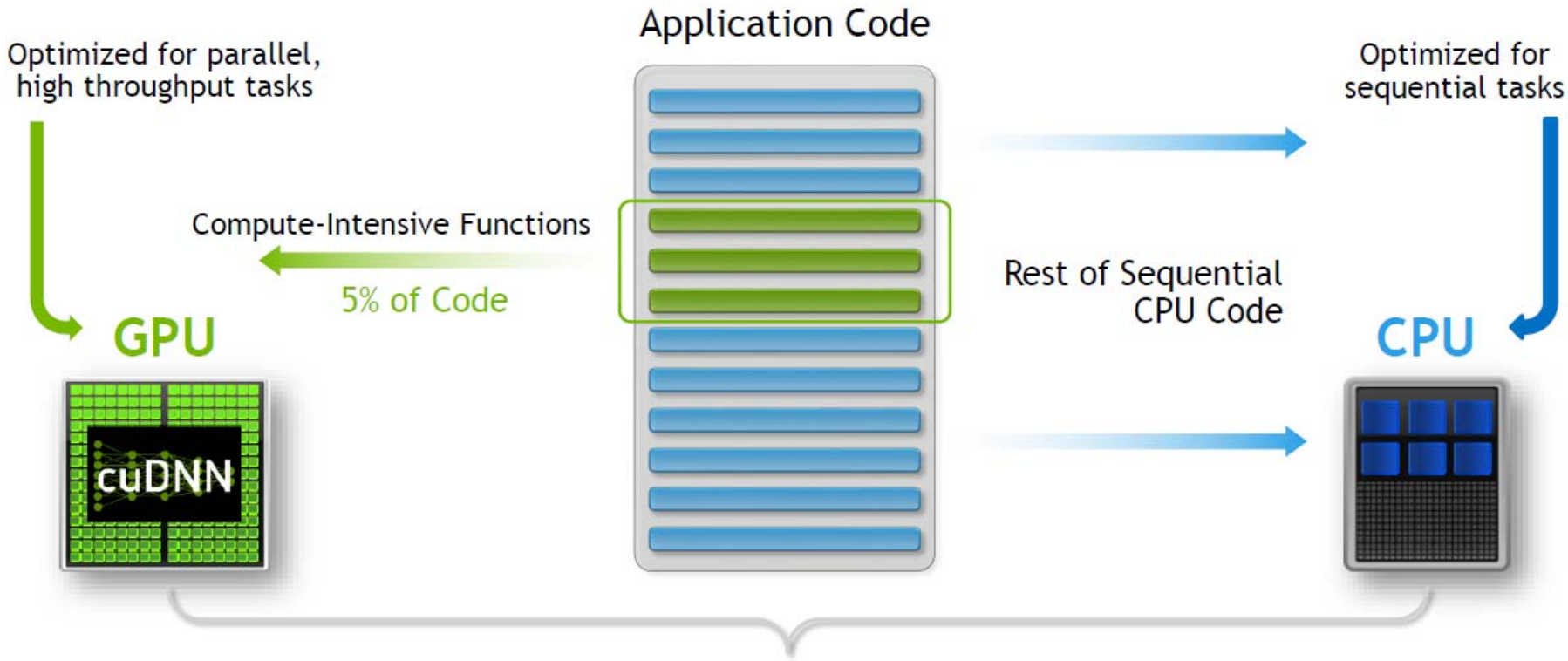
GPU Computing – What is a GPU?





Source: Udacity Self-Driving Car Nanodegree – Part 1, Lesson 15

Additional Slide



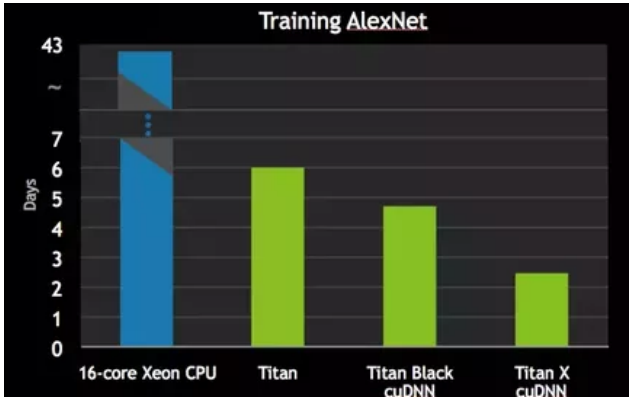
GTX 1080 Ti

GPU engine specs
CUDA Cores
3584
Graphics clock (MHz)
1480
Processor clock (MHz)
1582

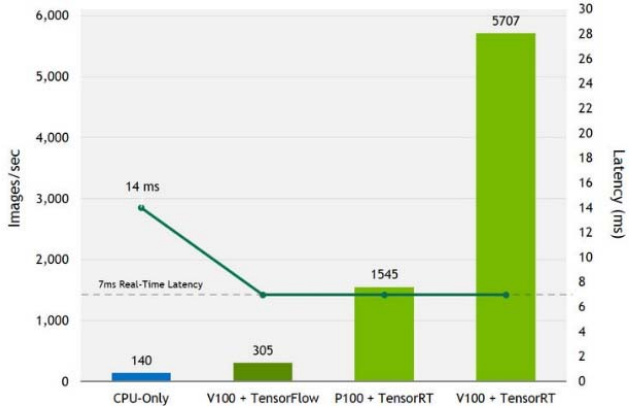
Memory specs
Standard memory config
11 GB GDDR5X
Memory interface width
352-bit
Memory bandwidth (GB/sec)
11 Gbps

Source: Nvidia Deep Learning Workshop @LRZ

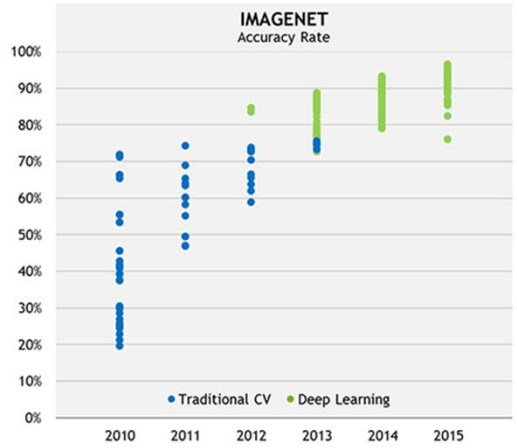
GPU Computing – Why a GPU for Deep Learning?



1. Faster training



2. Faster inference



3. Better results

Source: https://www.quora.com/Would-the-success-of-deep-learning-have-been-possible-without-GPUs-and-CUDA/ https://www.tomshardware.co.uk/nvidia-ai-technology-improves-efficiency,news-56853.html

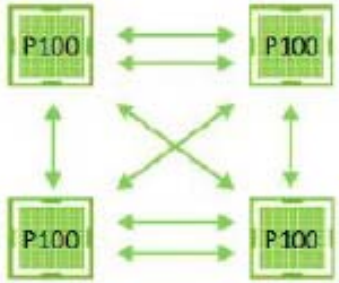
GPU Computing – Nvidia GPU for Deep Learning

- Use the NVIDIA hard- and software for your deep learning development
- NVIDIA is one of the leading GPU manufacteres
- NVIDIA is one of the leading deep learning developers building a complete environment

Consumer GPUs
~800 €

HPC GPUs
~16.000 €

GPU cluster (8x)
~120.000 €



NVLINK
>5x-12x PCI

CUDA API
Parallel computing

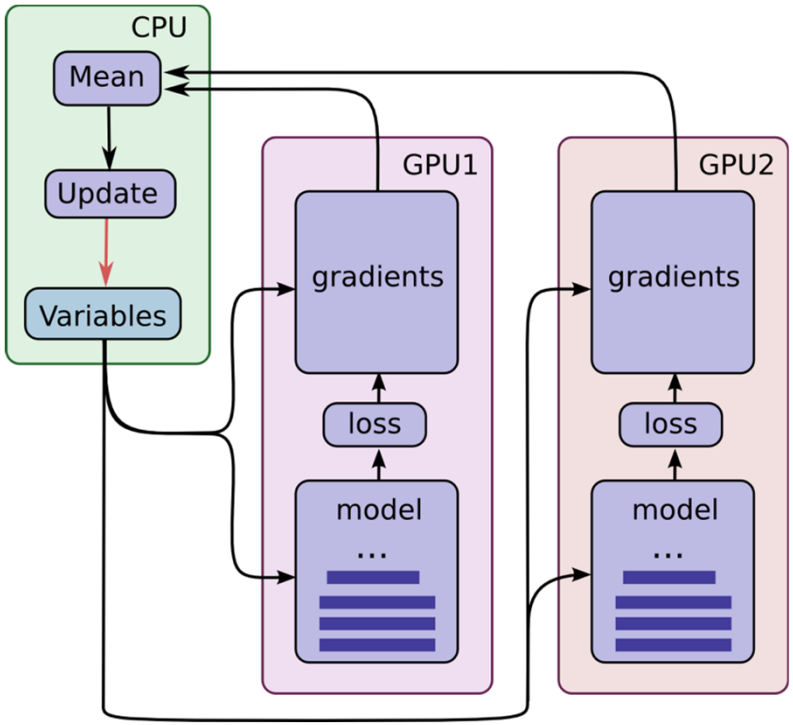
cuDNN
DL library

Digits
DL training

TensorRT
Faster inference

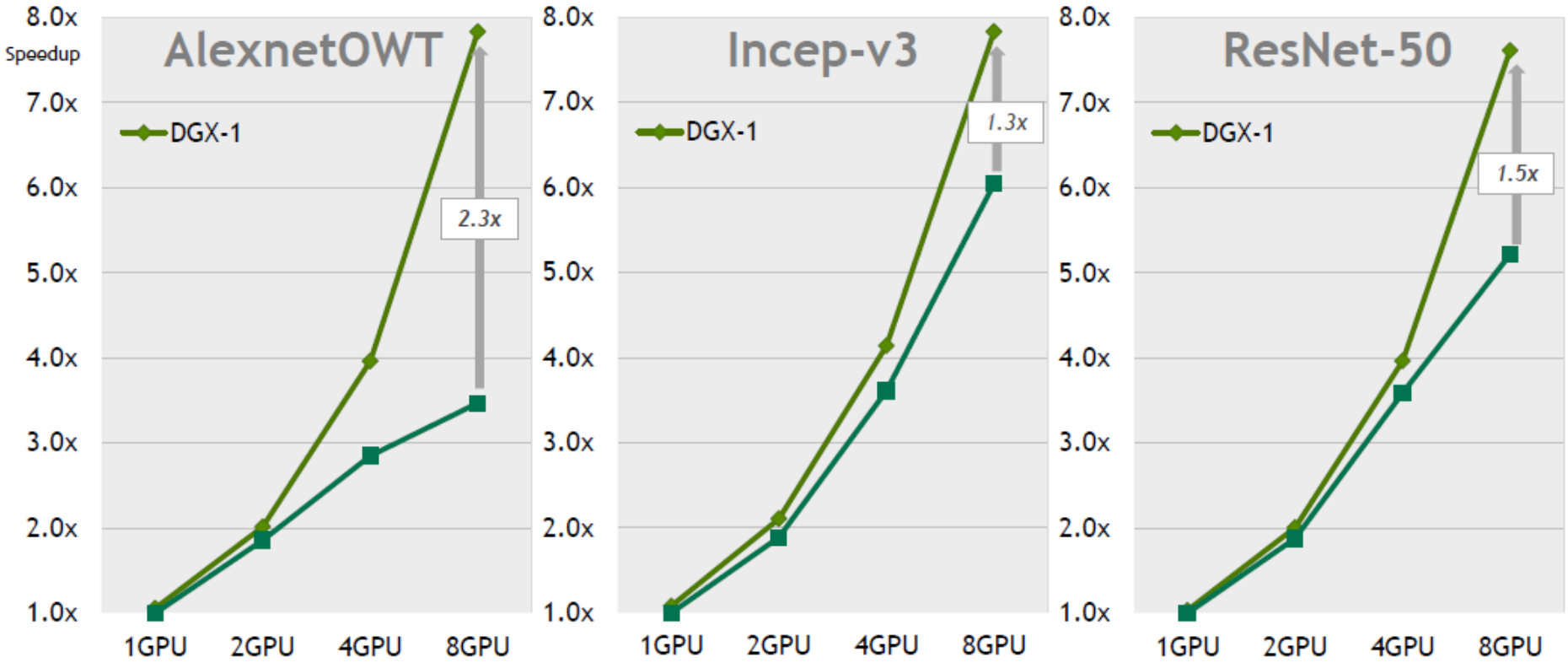
GPU Computing – Multiple GPU

- What is better than training faster on a single GPU? → Training on **multiple GPUs!**
- You can either use a normal PC with a motherboard that fits multiple GPUs (2x, 3x, 4x) or use a special GPU cluster (NVIDIA DGX-1)
- Multi-GPU can be used in different ways:
  - Use a single GPU for training of a specific hyperparameter set and do the same on the others
  - Use multiple GPUs for simultanous calculation → takes time to adjust the model



Source: https://blog.rescale.com/deep-learning-with-multiple-gpus-on-rescale-tensorflow/

GPU Computing – Multiple GPU Training

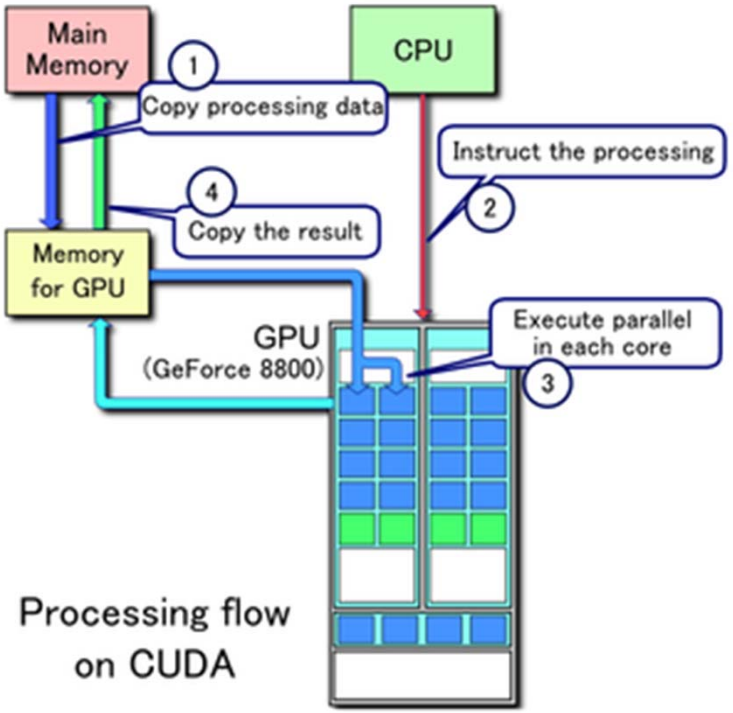


Multiple GPU: DGX vs. normal motherboard

Source: https://blog.slavv.com/picking-a-gpu-for-deep-learning-3d4795c273b9

GPU Computing – Cuda

- **CUDA** is a parallel computing platform and application programming interface (API) model created by Nvidia.
- It allows to use a CUDA-enabled GPU for general purpose processing
- The CUDA platform is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements for the execution of computational kernels
- Programming languages in C, C++
- Full support for integer and bitwise operations including integer texture lookups
- Unified memory



GPU Computing – Cuda

Standard C Code

```
void saxpy_serial(int n,
                  float a,
                  float *x,
                  float *y)
{
  for (int i = 0; i < n; ++i)
    y[i] = a*x[i] + y[i];
}

// Perform SAXPY on 1M elements
saxpy_serial(4096*256, 2.0, x, y);
```

Parallel C Code

```
__global__
void saxpy_parallel(int n,
                    float a,
                    float *x,
                    float *y)
{
  int i = blockIdx.x*blockDim.x +
          threadIdx.x;
  if (i < n) y[i] = a*x[i] + y[i];
}

// Perform SAXPY on 1M elements
saxpy_parallel<<<4096, 256>>>(n,2.0,x,y);
```

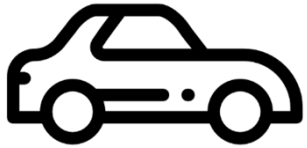
Source: https://www.youtube.com/watch?v=nRSxp5ZKwhQ/
https://intranet.birmingham.ac.uk/it/teams/infrastructure/research/bear/documents/public/CUDA-2013-07-31/GPU-overview.pdf

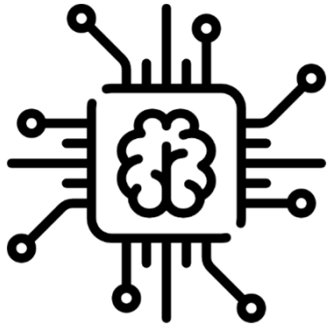
AI Development
Johannes Betz / Prof. Dr. Markus Lienkamp /
Prof. Dr. Boris Lohmann

(Dr. Johannes Betz)

Agenda

1. Chapter: AI Development Pipeline
2. Chapter: Transfer Learning
3. Chapter: AI Frameworks
4. Chapter: Data and Labeling
5. Chapter: GPU Computing
6. **Chapter: Hyperparameter Tuning**
7. Chapter: AI Inference
8. Chapter: Summary





Hyperparameter Tuning – What is That?

- When training ANNs, we focus on **not overfitting** the training loss and to get a **good evaluation** at the end
- To achieve that, we can vary different parameters for the training of an ANN → these are the **hyperparameters**
- The tuning of hyperparameters is the **"magic"** behind a good ANN and changes the ANN from a black box to a system we can understand
- Hyperparameter tuning is nothing official, but rather like a "**best practice**" or "**tipps and tricks**" collection

Important: Hyperparameter tuning differs from problem to problem!

Hyperparameter Tuning – What is That?

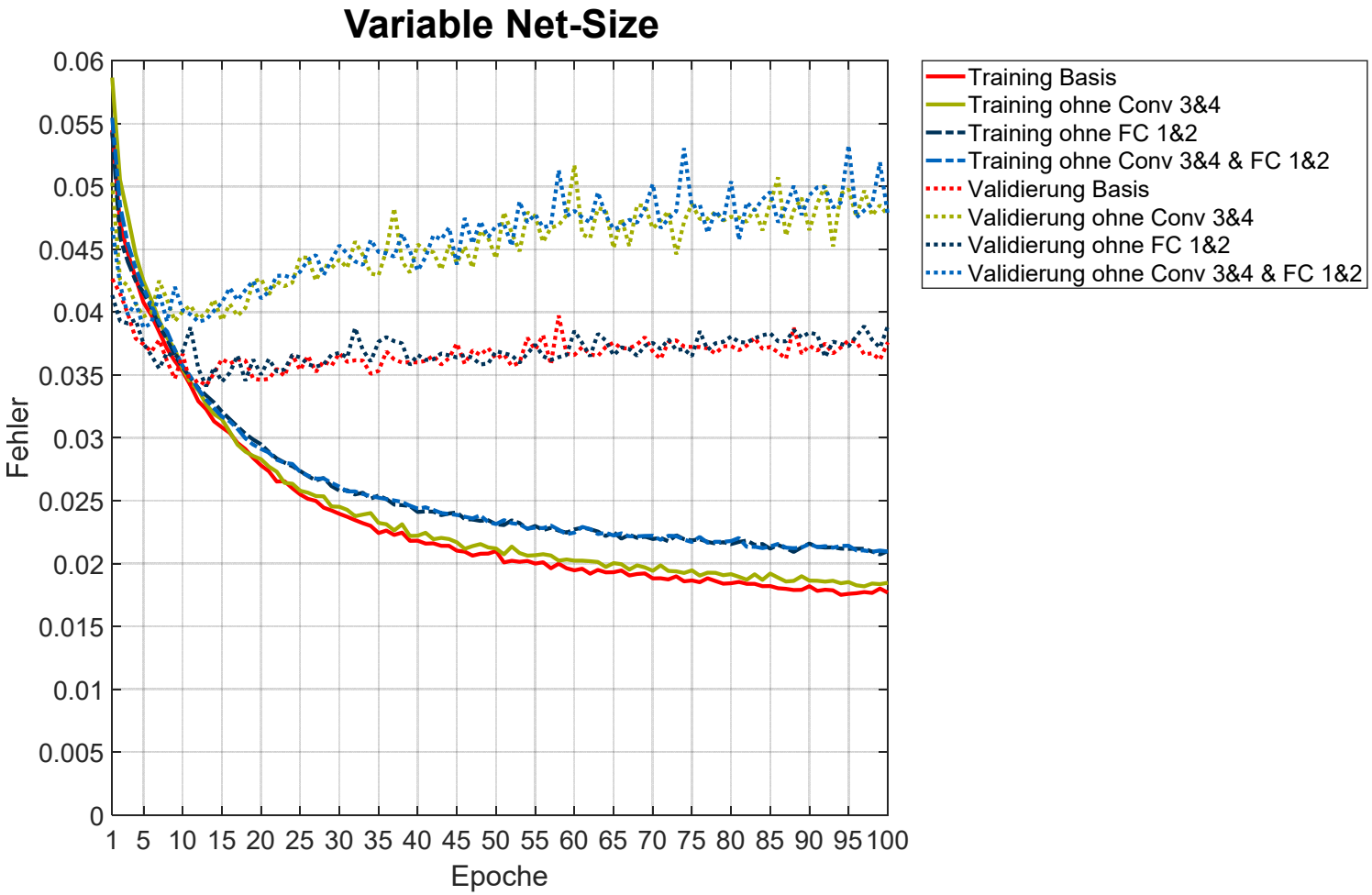
Things you can vary in your ANN:

- Number of hidden layers
- Number of fully-connected layers
- Number of neurons in one layer
- Number of training epochs
- Weight initialization
- Learning rate
- Batch size
- Activation function
- Dropout rate
- …

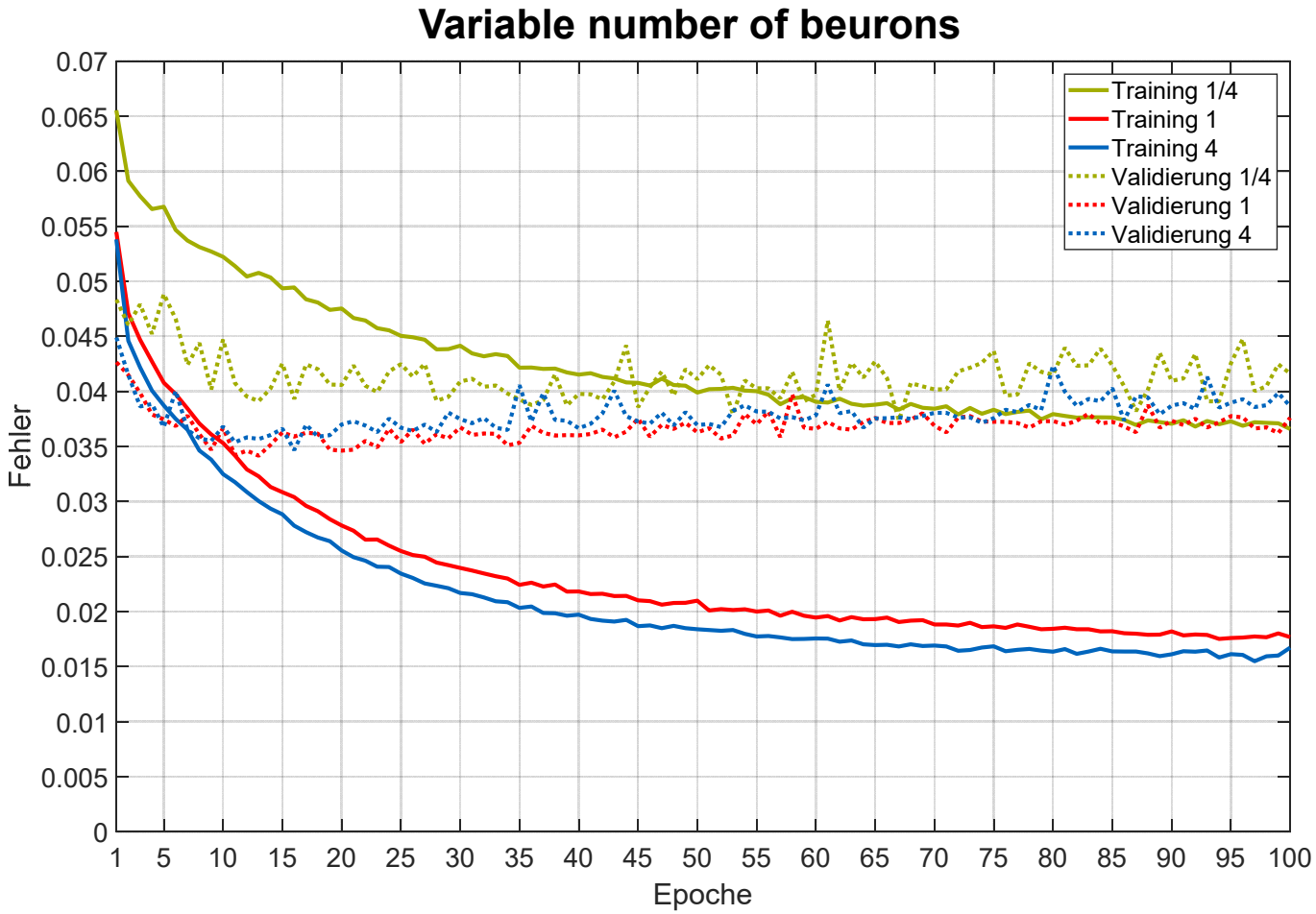
→ **Usage of optimization algorithms for hyperparameter tuning**

Source: http://colinraffel.com/wiki/neural_network_hyperparameters

Hyperparameter Tuning – Size of Net

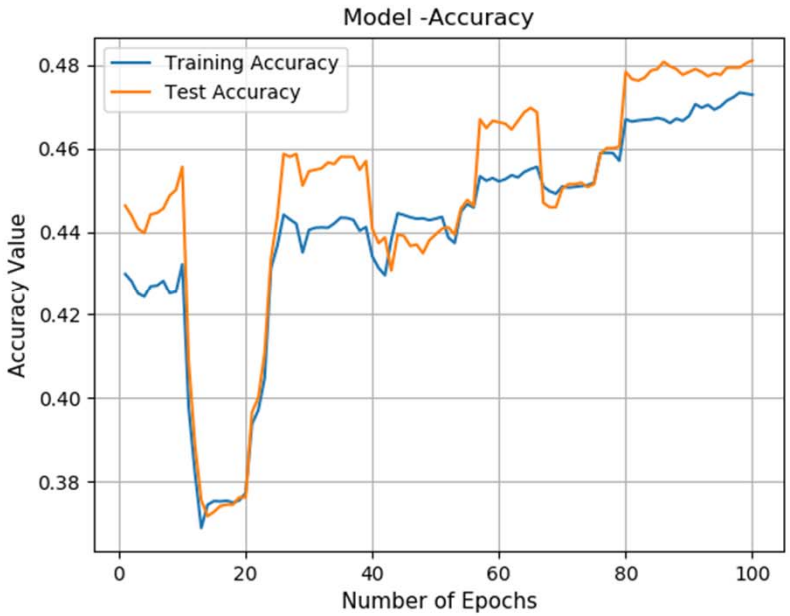


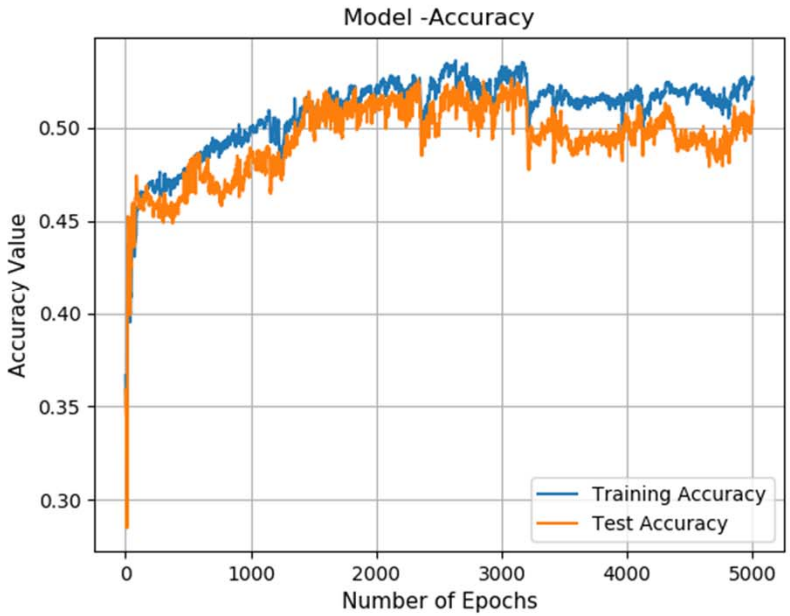
Hyperparameter Tuning – Size of Net



Hyperparameter Tuning – Number of Epochs

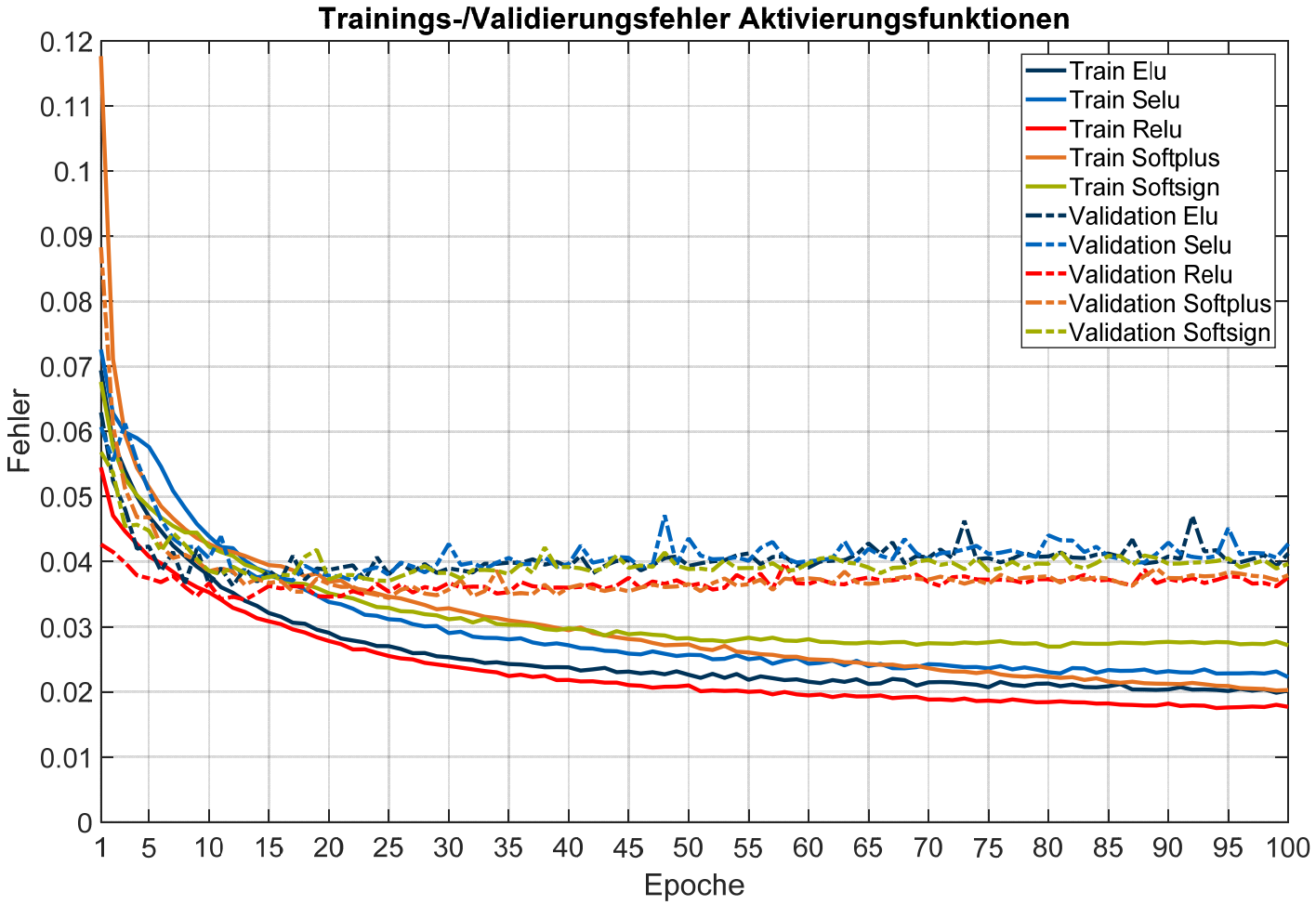
Same ANN structure, same input data, different Epochs



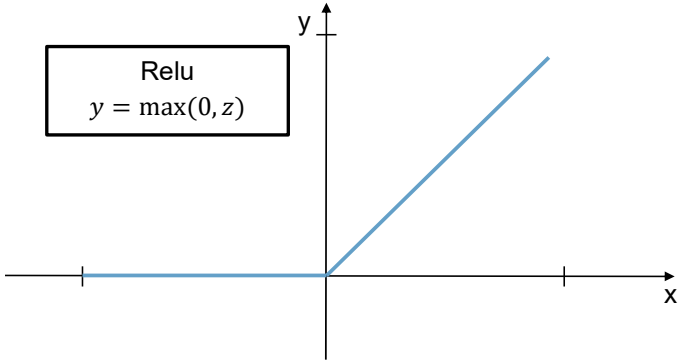


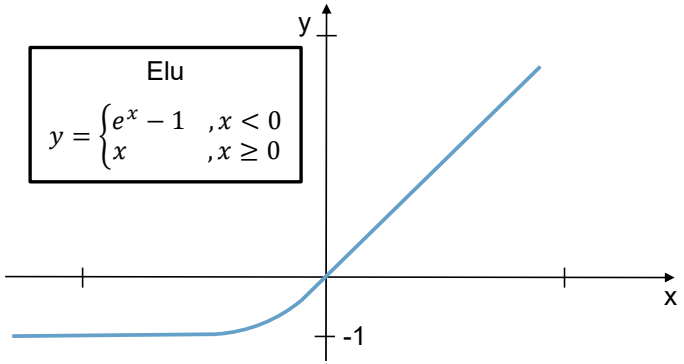
- A minimum number of epochs must be given for reaching a saturation of the loss
- You can insert abort criterions if the loss doesn‘t change over a long time
- The bigger the net/the more layers the net contains, the more epochs you need
- The time for calculating a single epoch depends on net size and the number of input data

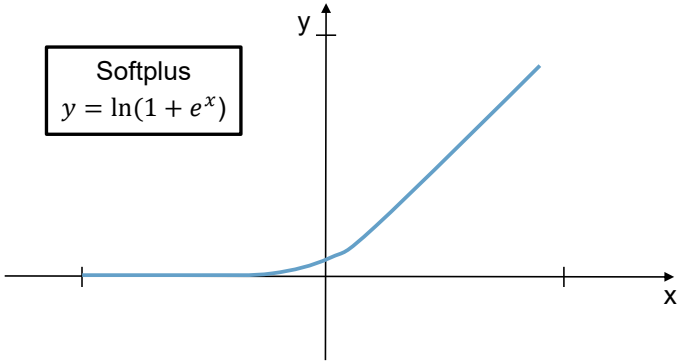
Hyperparameter Tuning – Activation Function

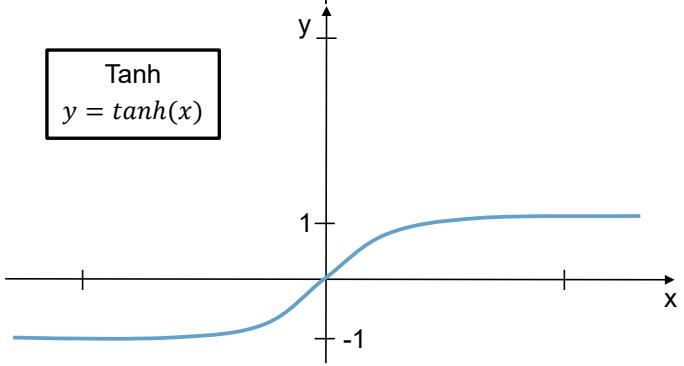


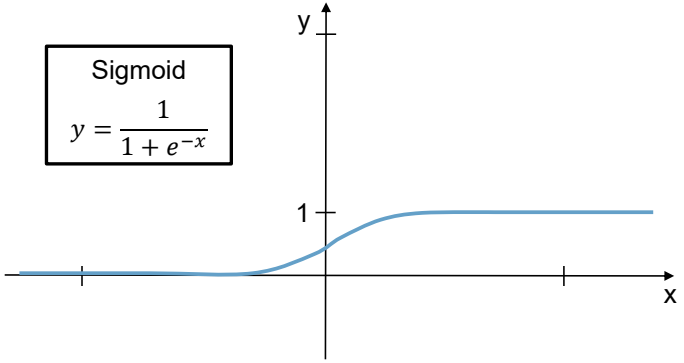
Additional Slide

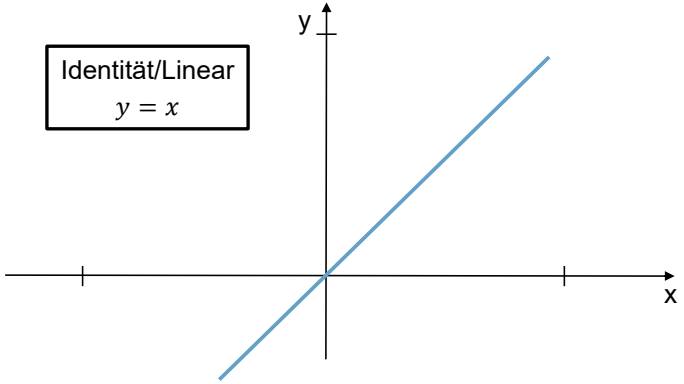




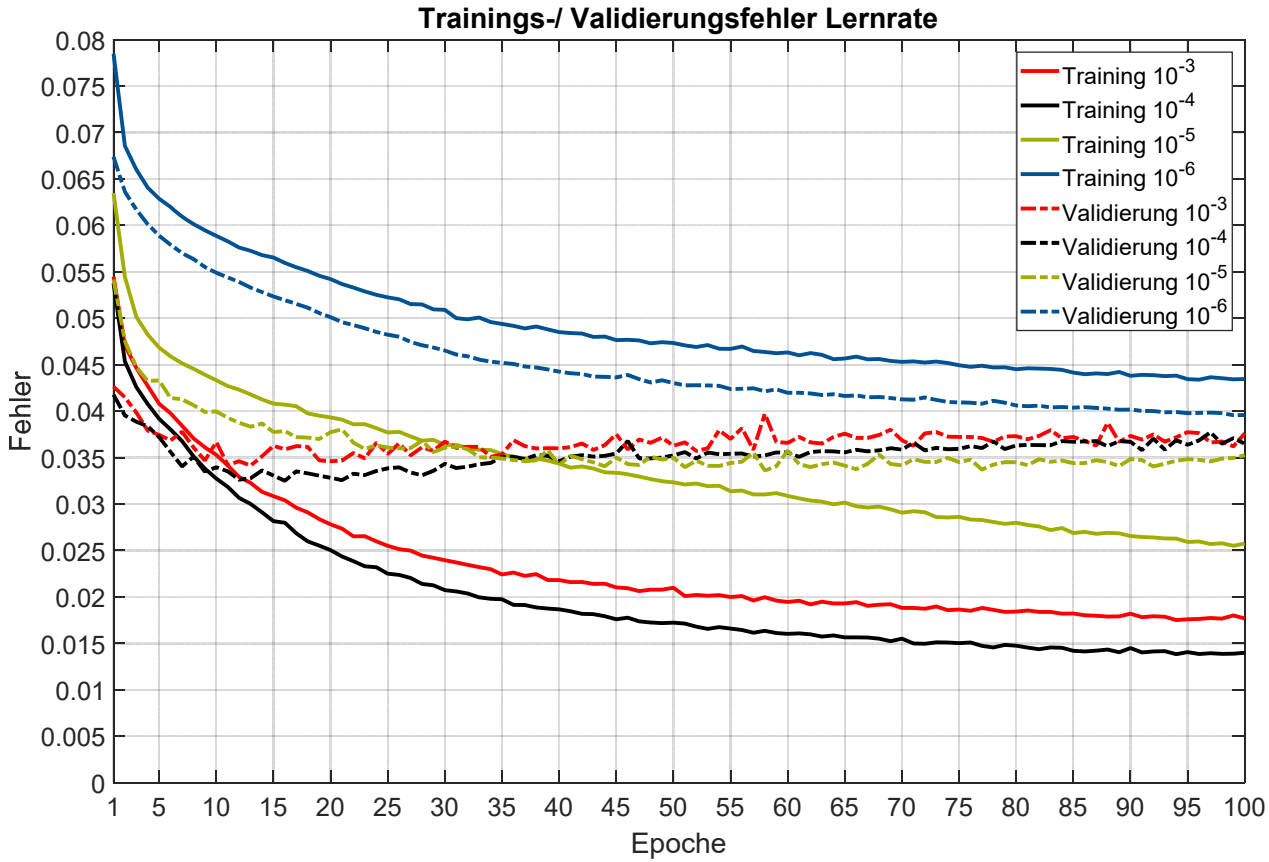




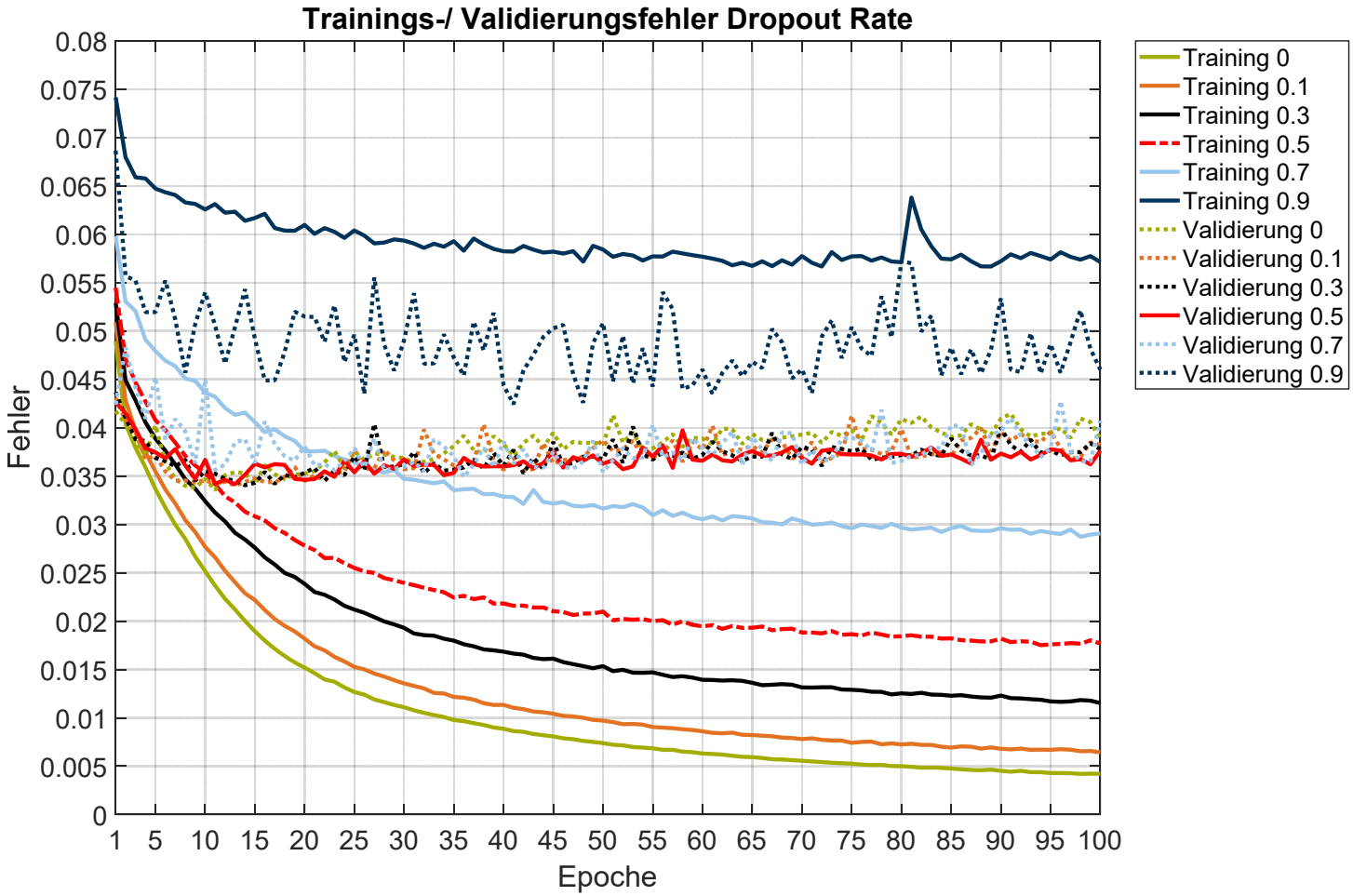




Hyperparameter Tuning – Learning rate

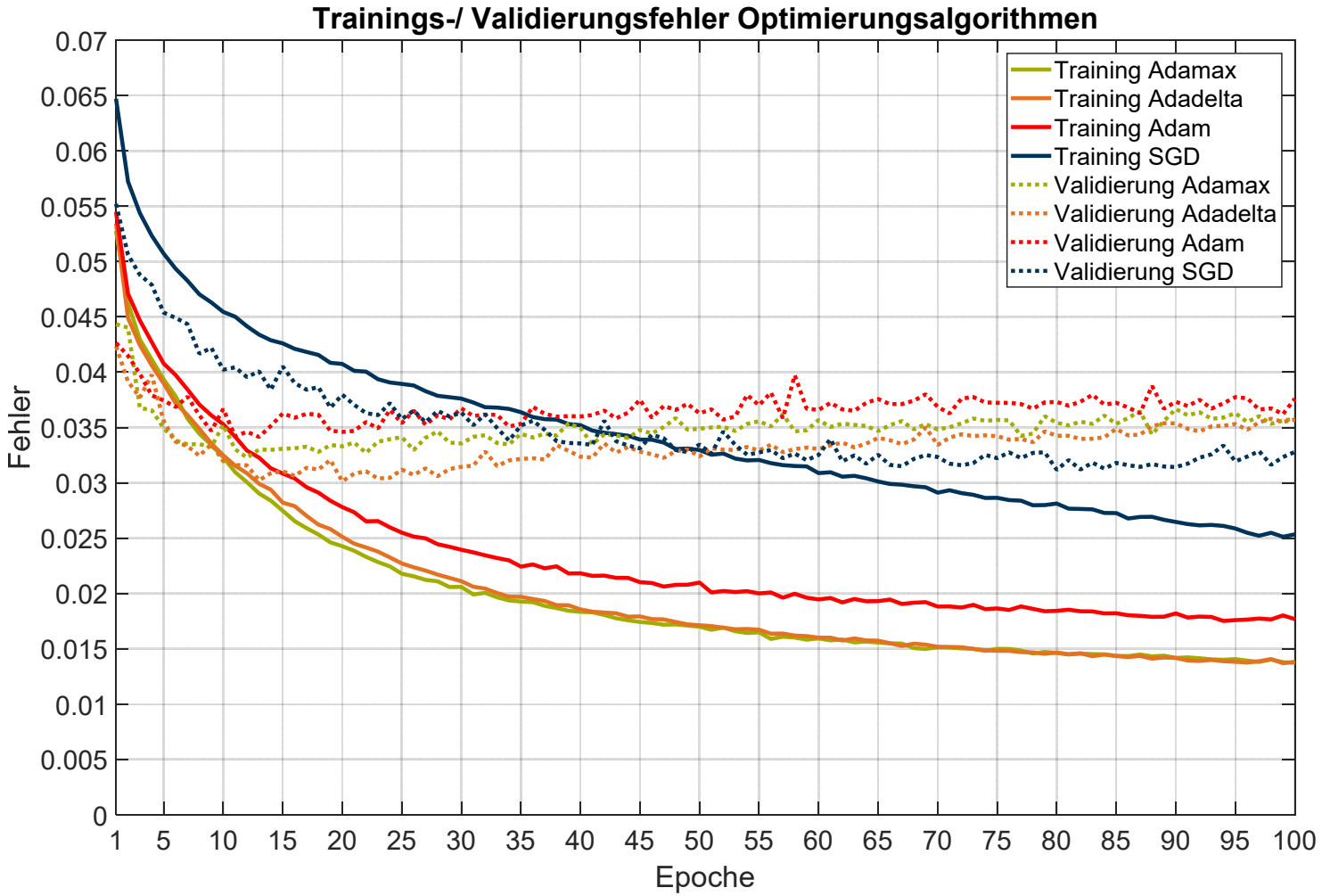


Hyperparameter Tuning – Dropout



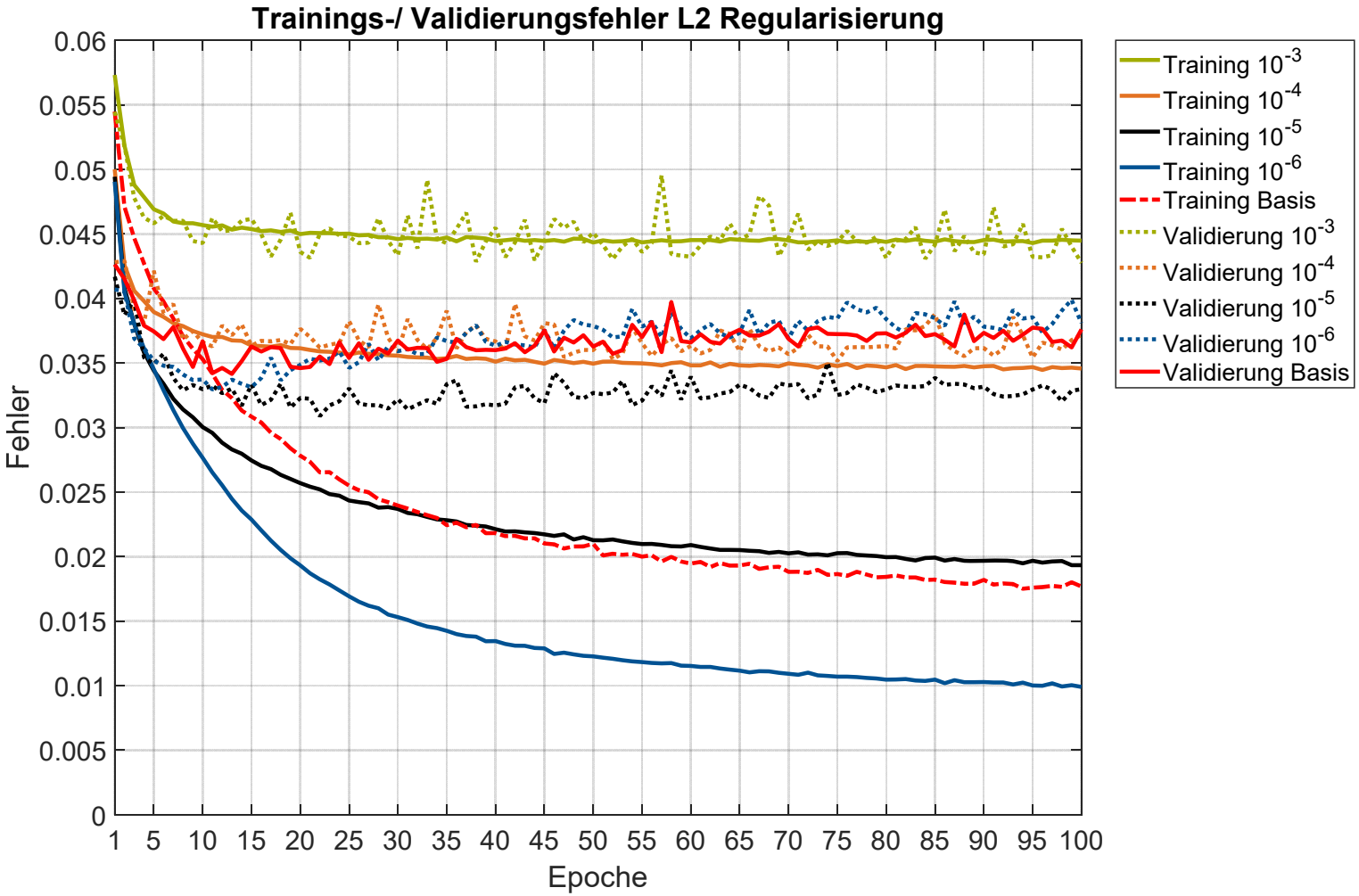
Additional Slide

Hyperparameter Tuning – Optimization Function



Additional Slide

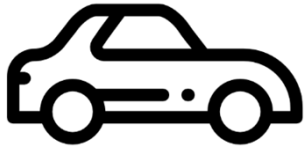
Hyperparameter Tuning – Regularization

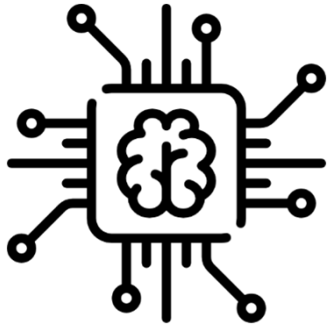


AI Development
Johannes Betz / Prof. Dr. Markus Lienkamp /
Prof. Dr. Boris Lohmann

(Dr. Johannes Betz)

Agenda





AI Inference – What is Inference?

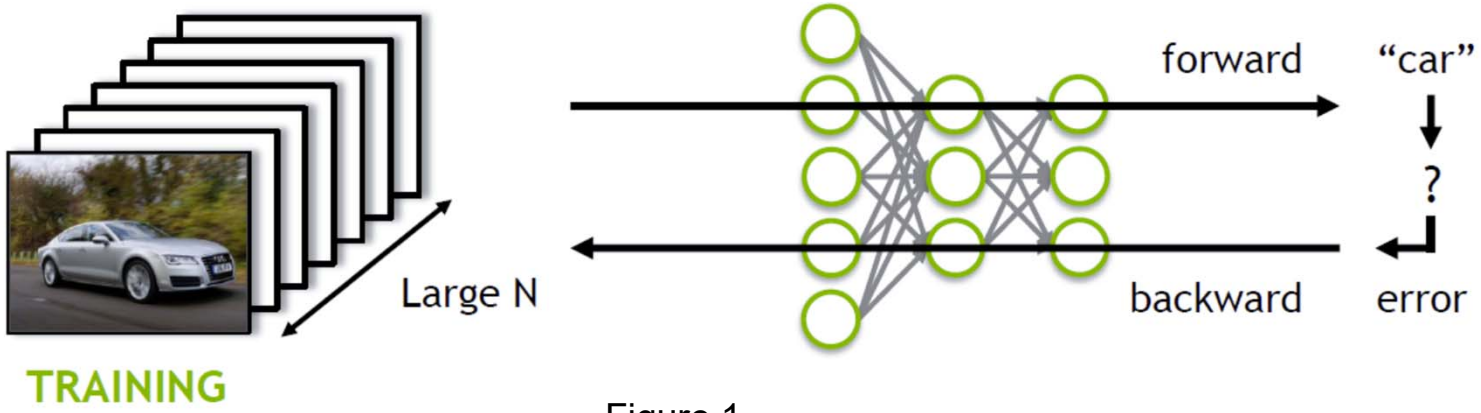


Figure 1

Inference: The actual application and usage of your ANN

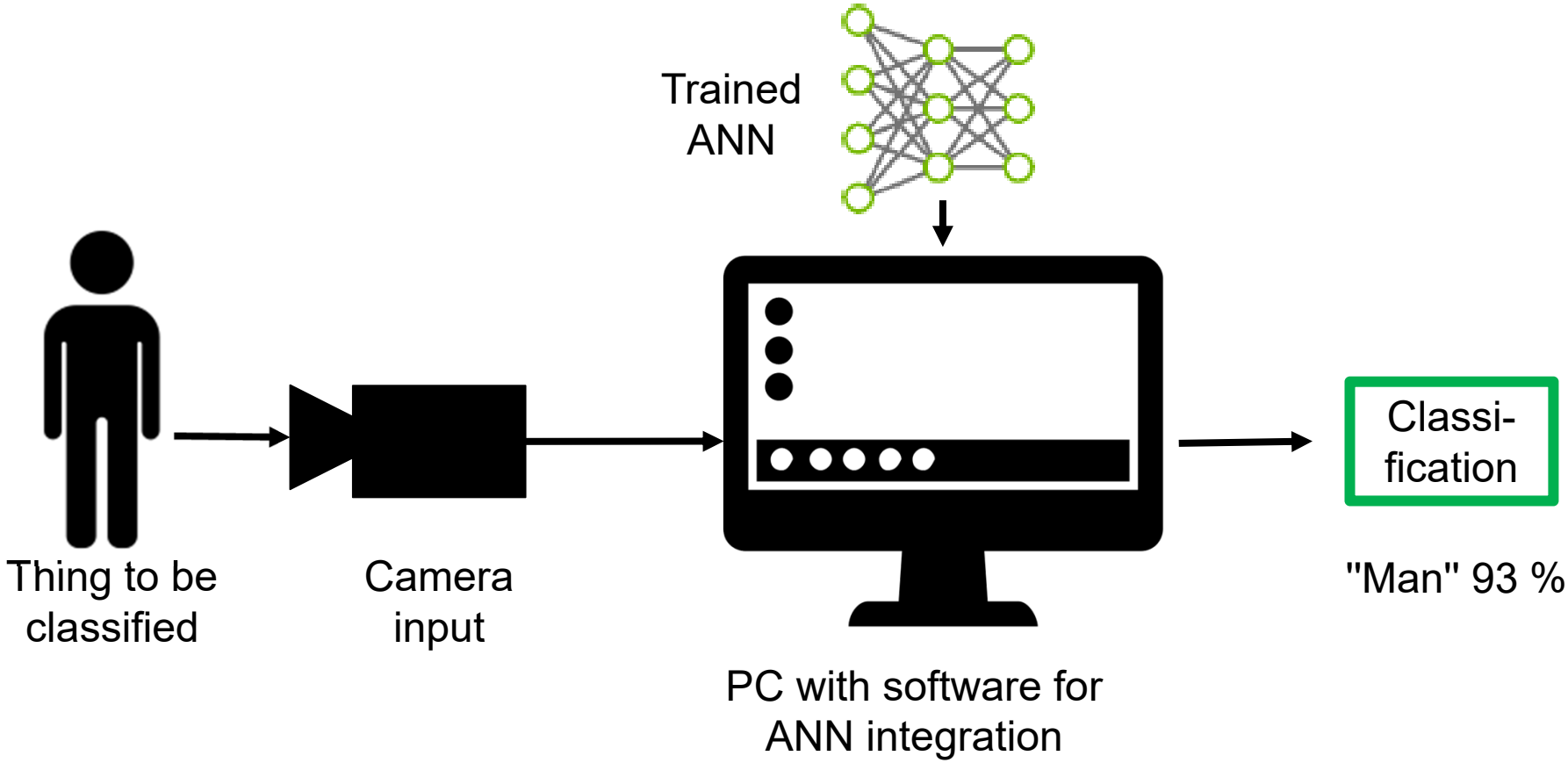
Source: Nvidia Deep Learning Workshop @LRZ

Additional Slide

Both DNN training and inference start out with the same *forward propagation* calculation but model training goes further. As Figure 1 on the previous slide illustrates, the results from the forward propagation are compared against the (known) correct answer to compute an error value after *forward propagation*. A *backward propagation* phase propagates the error back through the network layers and updates their weights using gradient descent in order to improve the network's performance at the task it is trying to learn. It is common to batch hundreds of training inputs (e.g., images in an image classification network or spectrograms for speech recognition) and operate on them simultaneously during DNN training in order to prevent overfitting and, more importantly, amortize loading weights from GPU memory across many inputs, increasing computational efficiency.

For inference, performance goals are different. To minimize the network's end-to-end response time, inference typically batches a smaller number of inputs than training, as services relying on inference to work (e.g.,, a cloud-based image-processing pipeline) are required to be as responsive as possible so users do not have to wait several seconds while the system is accumulating images for a large batch. In general, we might say that the per-image workload for training is higher than for inference, and while high *throughput* is the only thing that counts during training, *latency* becomes important for inference as well.

AI Inference – What is Inference?



Source: https://github.com/dusty-nv/jetson-inference;
https://www.google.de/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwj7oZOz7LncAhUGmrQKHThBDIMQjRx6BAgBEAU&url=https%3A%2F%2Fwww.onlinewebfonts.com%2Ficon%2F395988&psig=AOvVaw35HUnkc45PR1hxkOdoZQ42&ust=1532594180959688

AI Inference – Inference Hardware: Nvidia Drive

- Linux Ubuntu based „Computer“ (SoC = system on a chip)
- Automotive grade!

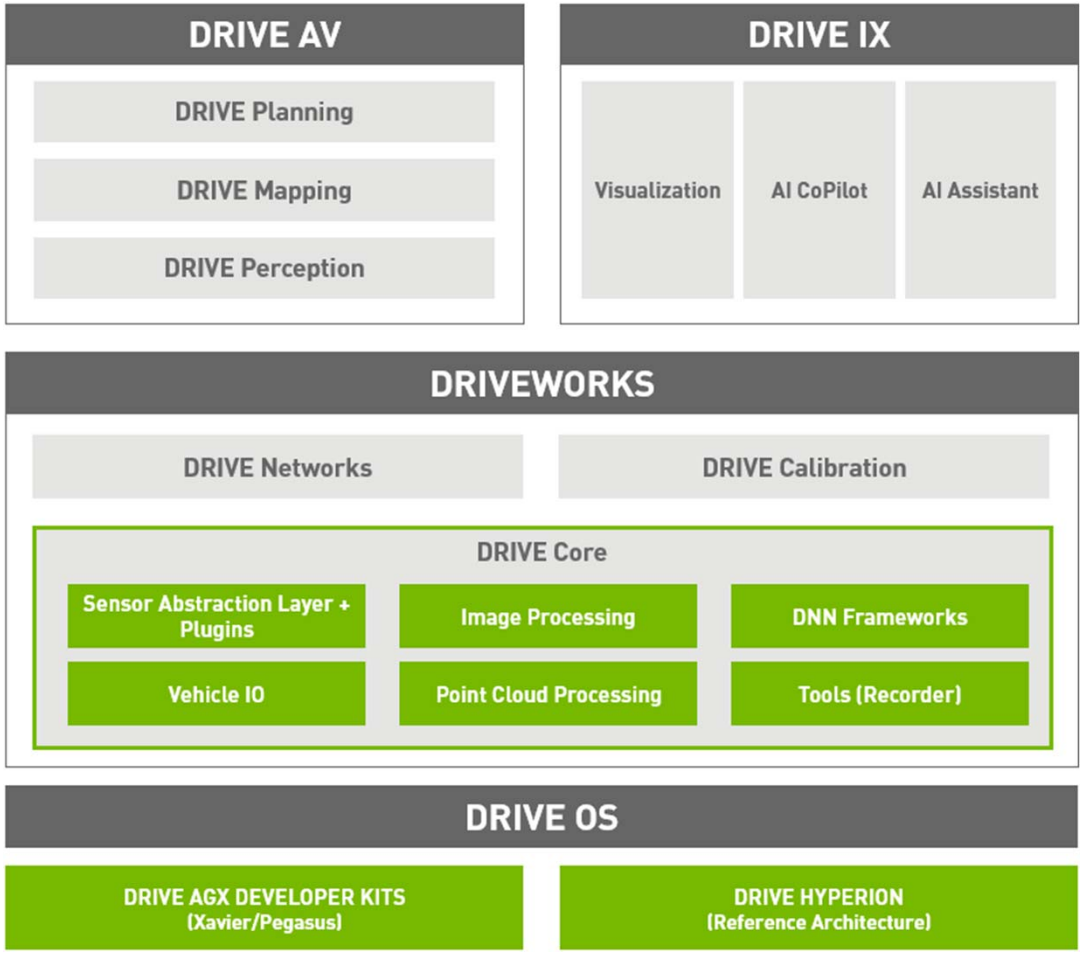
2x Tegra architecture („Xavier“)

- 8x Carmel ARM64
- 1x 512-core Volta GPU
- 16GB LPDDR4, 128-bit interface
- Peripherie: USB, HDMI, SATA, UART, SPI, I2C, GPIO,CAN, LIN,…

+ Software package „Driveworks SDK“: Cuda, CudNN, TensorRT + autonomous driving functions API

Additional Slide

Software SDK for inference of AI software for autonomous driving



Nvidia Drive Core

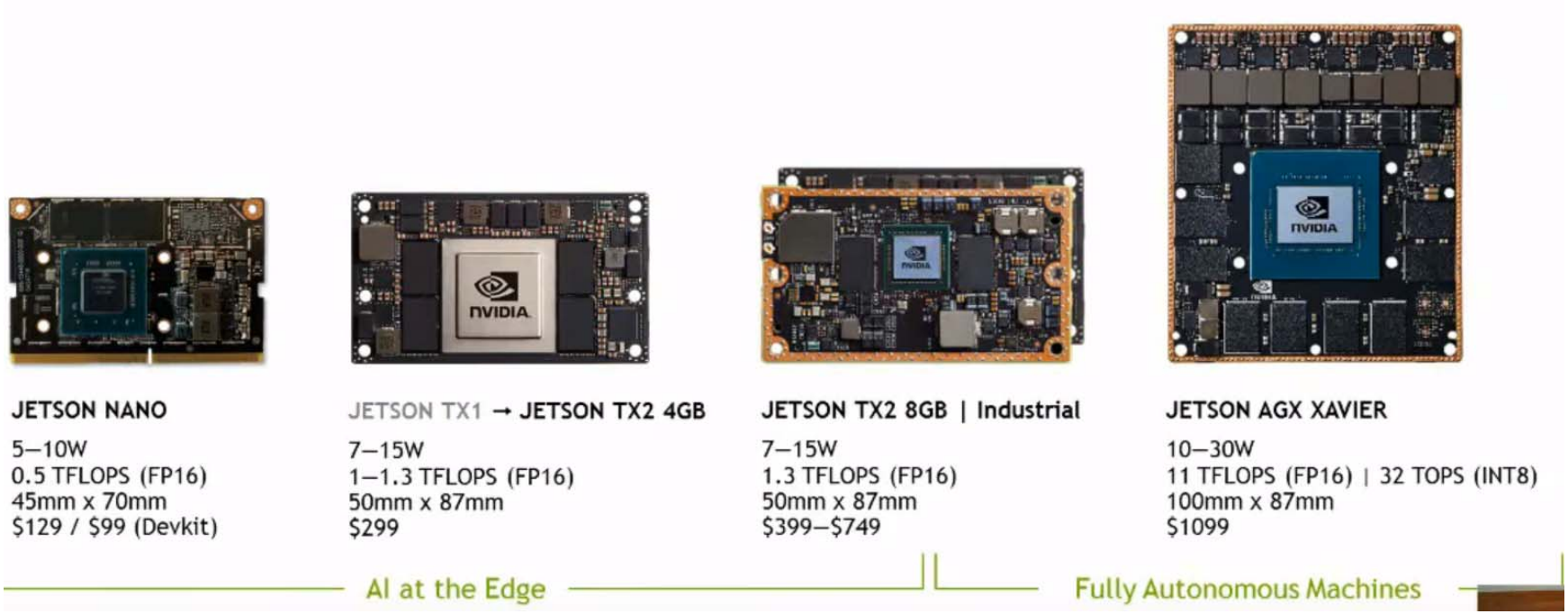
Source: https://developer.nvidia.com/drive/drive-core

AI Inference – Inference Hardware: Nvidia Drive AGX

Nvidia Drive AGX Hardware + Driveworks software SDK

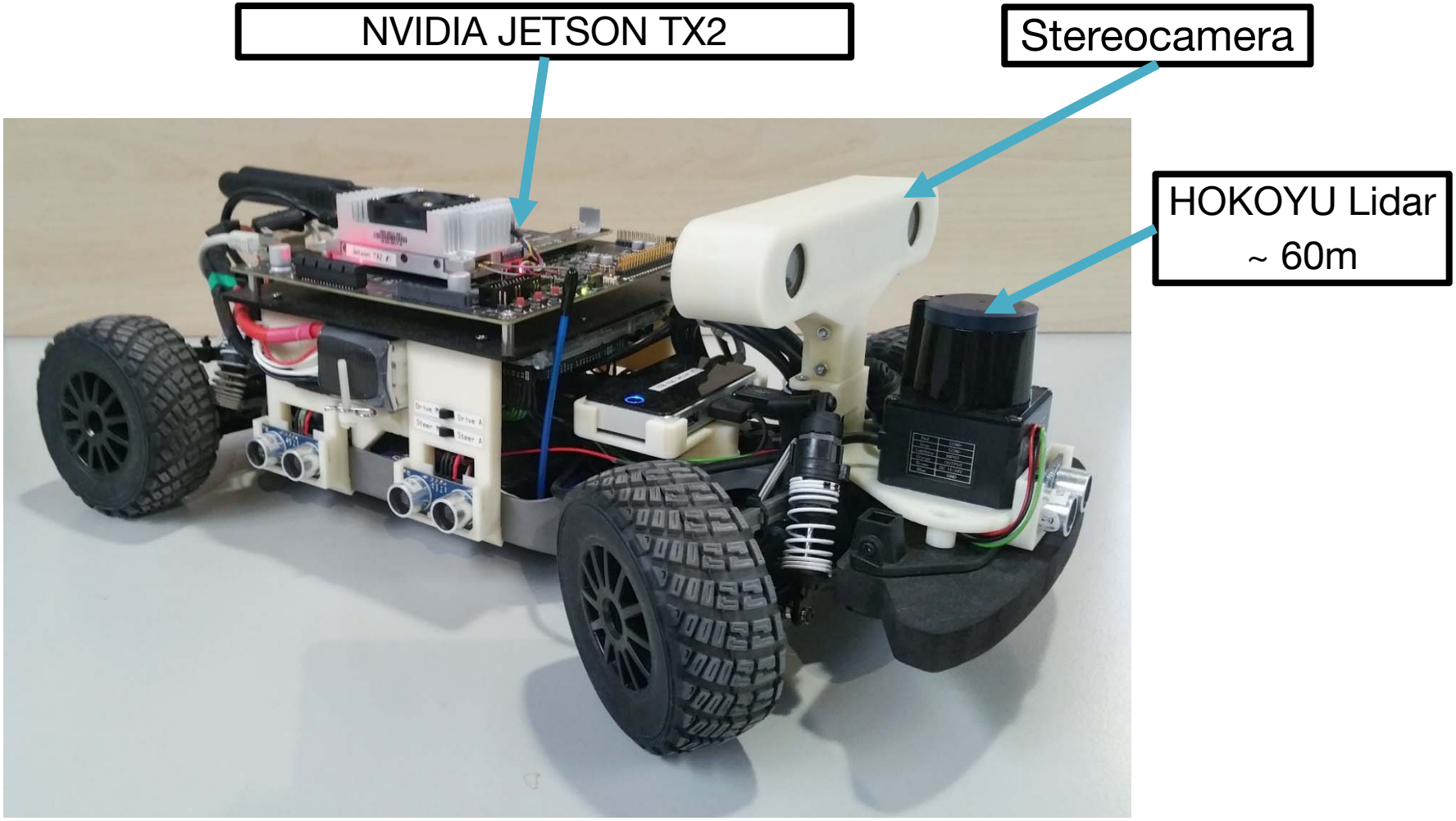
Source: https://www.youtube.com/watch?v=KS_4xjXNTxg&t=20s

AI Inference –Hardware: Nvidia Jetson Family



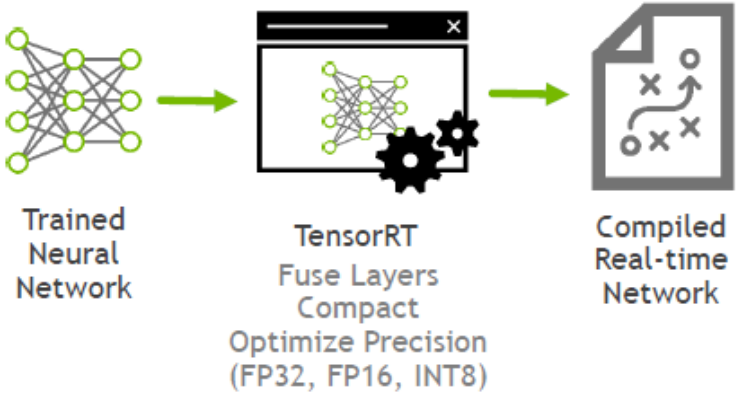
- Linux Ubuntu based "Mini-Computer"
- Different sizes and packages
- Different GPU or CPU architectures
- Peripherie: USB, HDMI, SATA, UART, SPI, I2C, GPIO, ...
- Software Package „Jetpack“; Cuda, CudNN, TensorRT, …

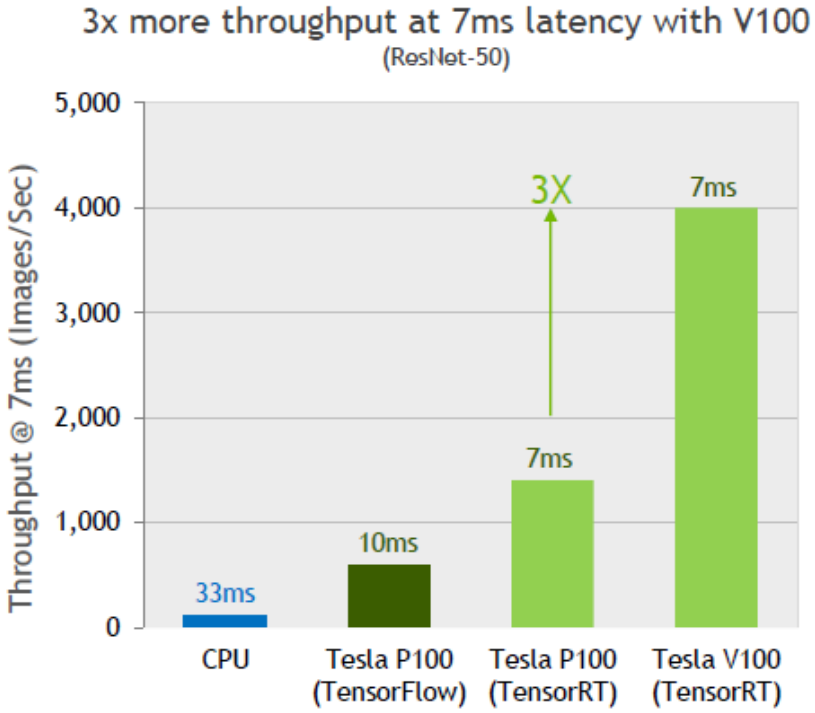
AI Inference – Inference Hardware: Nvidia Jetson



FTM autonomous RC car

AI Inference – Fasten the Inference

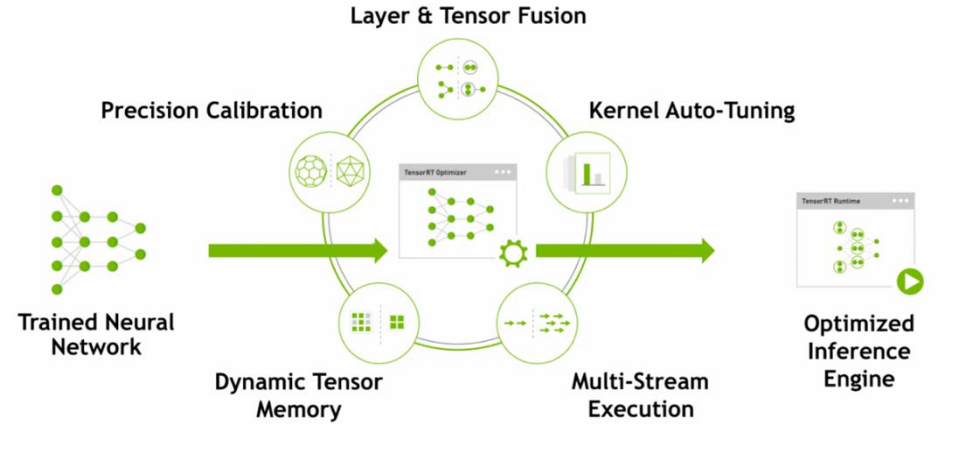




TensorRT can help to otimize the trained ANN

Additional Slide

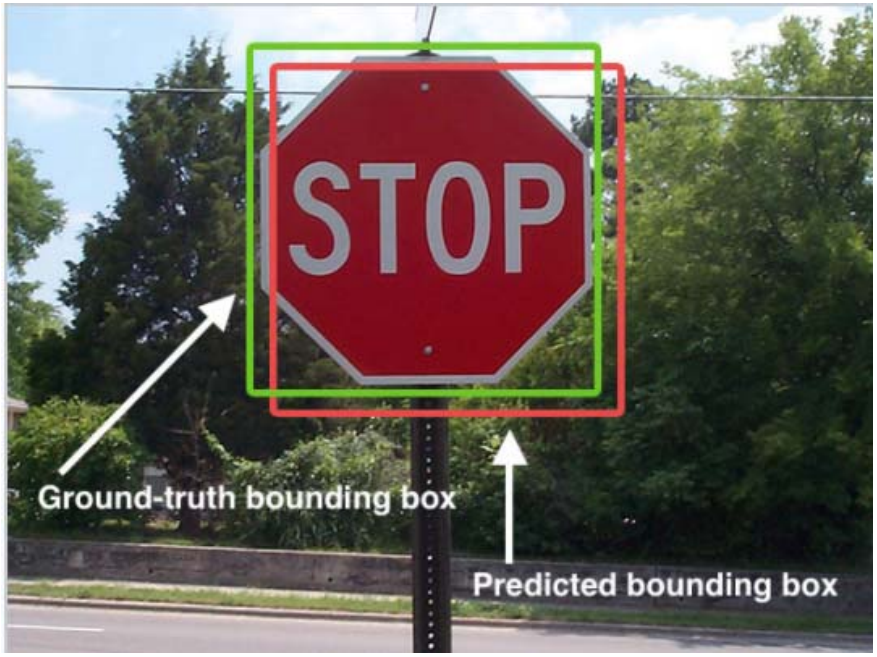
NVIDIA TensorRT™ is an SDK for high-performance deep learning inference. It inclues a deep learning inference optimizer and runtime that delivers low latency and high-throughput for deep learning inference applications.
TensorRT is built on CUDA, NVIDIA's parallel programming model, and enables you to optimize inference for all deep learning frameworks leveraging libraries, development tools and technologies in CUDA-X for artificial intelligence, autonomous machines, high-performance computing, and graphics.
TensorRT provides INT8 and FP16 optimizations for production deployments of deep learning inference applications such as video streaming, speech recognition, recommendation and natural language processing. Reduced precision inference significantly reduces application latency, which is a requirement for many real-time services, auto and embedded applications.

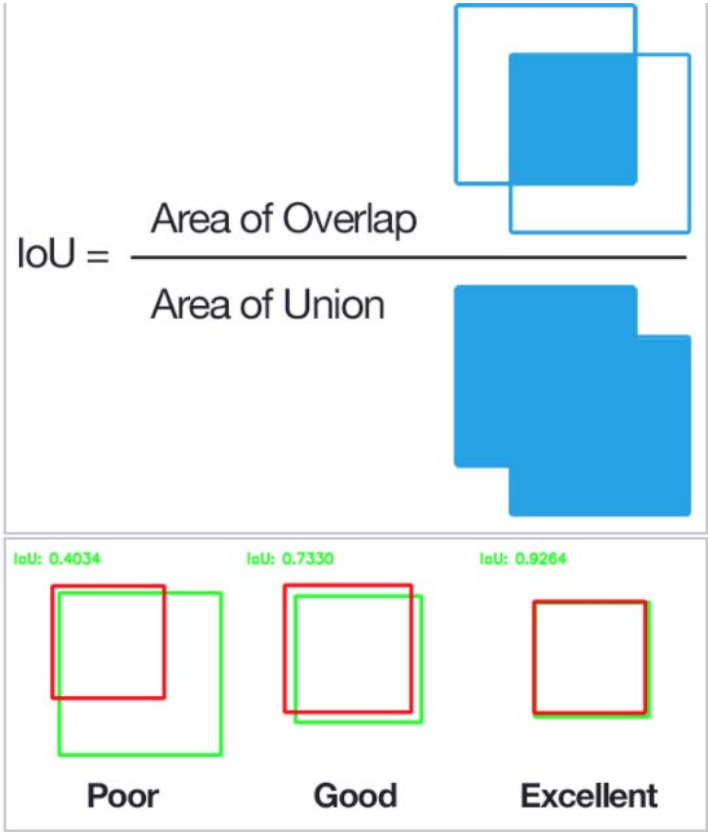


You can import trained models from every deep learning framework into TensorRT. After applying optimizations, TensorRT selects platform specific kernels to maximize performance on Tesla GPUs in the data center, Jetson embedded platforms, and NVIDIA DRIVE autonomous driving platforms.
To use AI models in data center production, the TensorRT Inference Server is a containerized microservice that maximizes GPU utilization and runs multiple models from different frameworks concurrently on a node. It leverages Docker and Kubernetes to integrate seamlessly into DevOps architectures.
With TensorRT developers can focus on creating novel AI-powered applications rather than performance tuning for inference deployment.

Source: https://developer.nvidia.com/tensorrt

AI Inference – Rate Your Inference

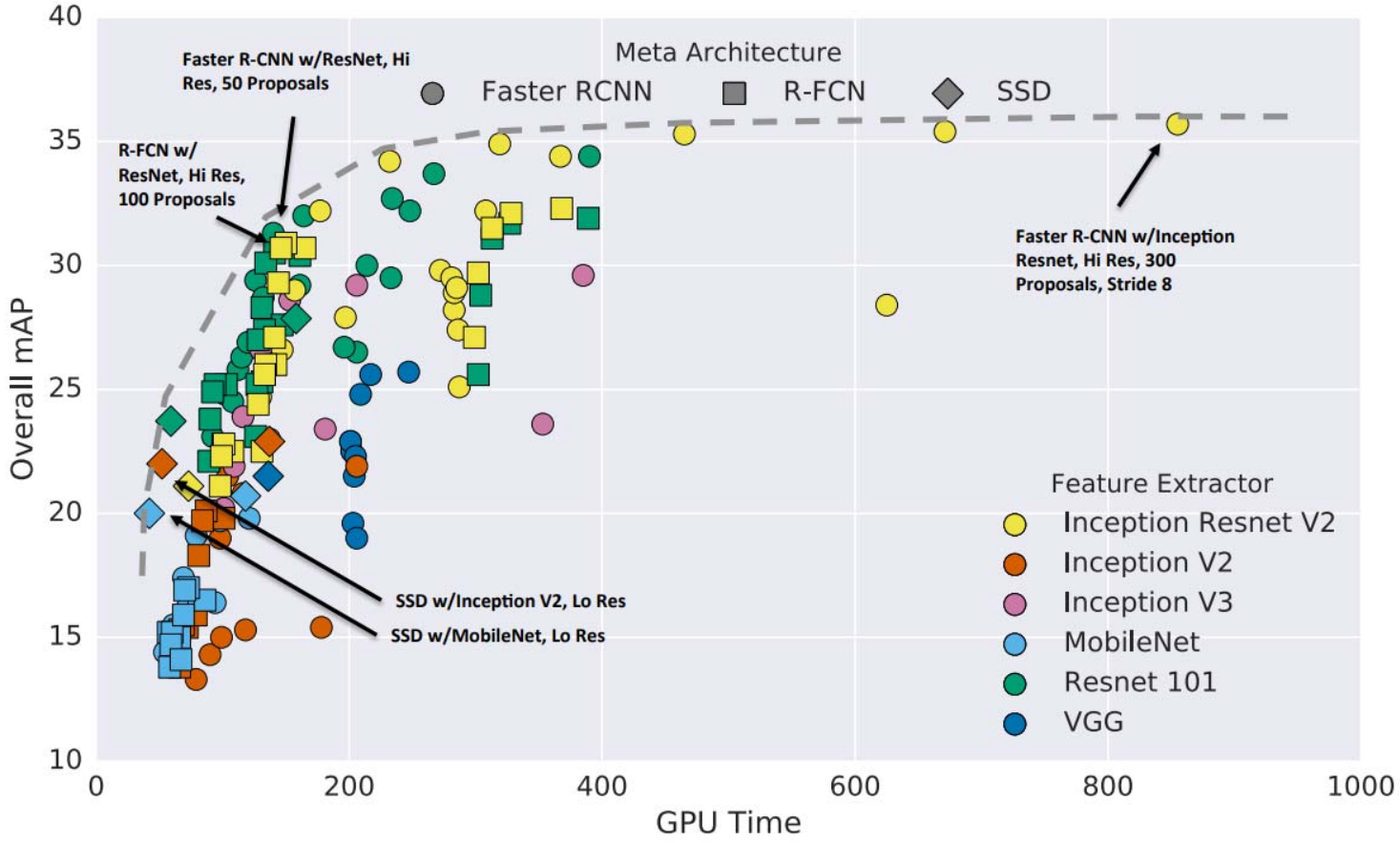




Intersection over Union (IoU)

Source: https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab

AI Inference – Rate Your Inference



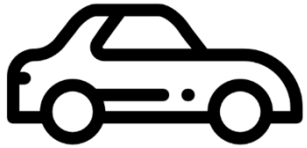
GPU load / images per second / GPU calculation time / mean average precision (mAP)

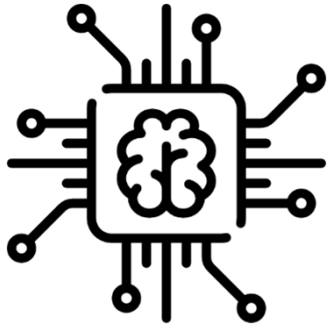
Source: https://arxiv.org/pdf/1611.10012.pdf

AI Development
Johannes Betz / Prof. Dr. Markus Lienkamp /
Prof. Dr. Boris Lohmann

(Dr. Johannes Betz)

Agenda





Summary

What did we learn today:

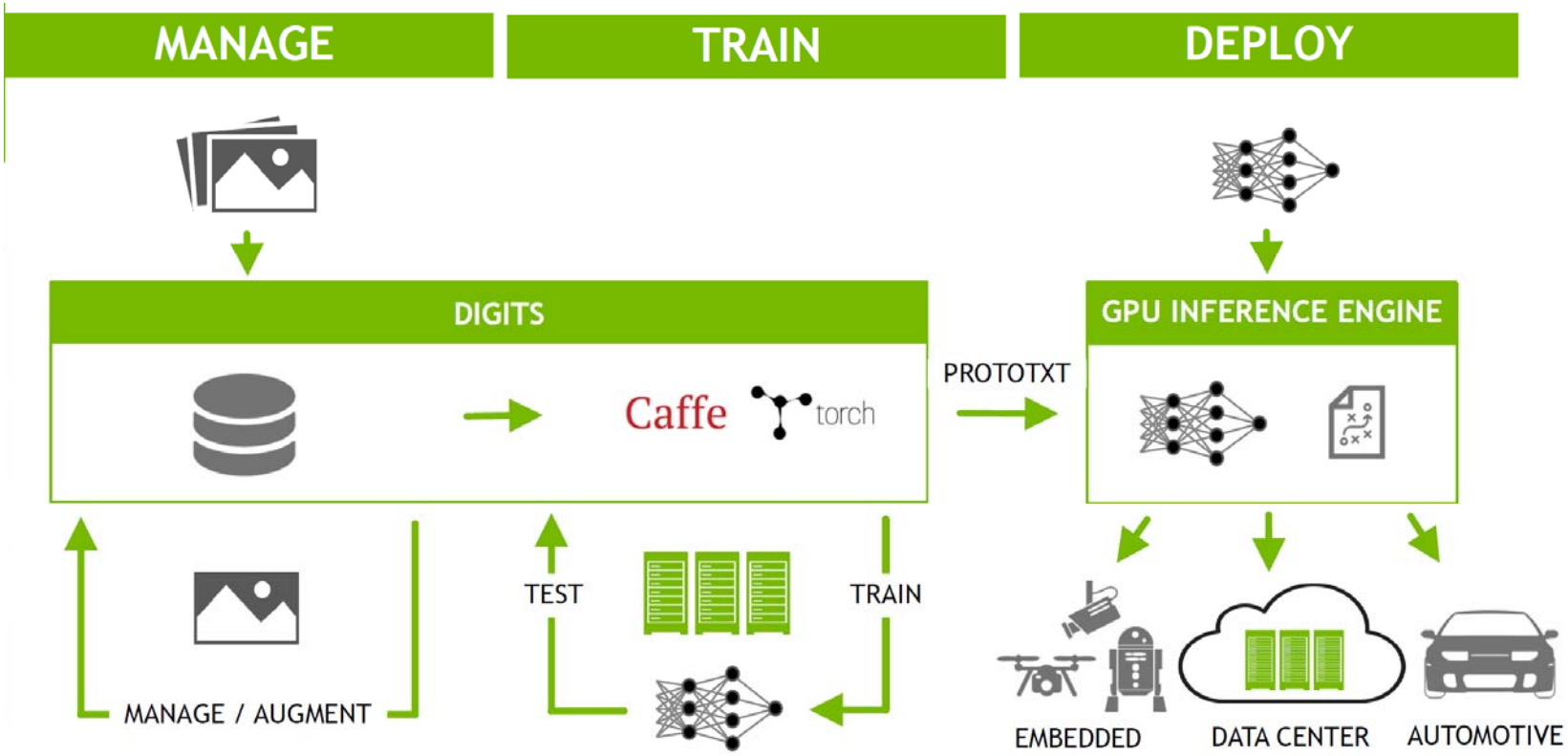
- Development of deep learning algorithm is all about the problem: **Same same but different.**
- Take advantage of **pre-trained networks** and use them for your problem. In addition, you can used published ANN architectures for your problem by using **transfer learning.** You will save a lot time.
- There are a lot of **frameworks** for setting up your ANN in code. Check first what do you need (e.g., multiple GPUs, inference hardware) and decide what you want to use.
- Be aware of the data you need. Search for **labeled datasets** first before you acquire your own data.
- **Data is the crucial part** in deep learning:
  - The better and specific your data, the better your results will be
  - The more data you have, the better your results will be.

Summary

What did we learn today:

- **GPUs** are your number one tool for fast training and fast inference
- Scale your model for training on multiple GPUs
- ANN development is all about **hyperparameter tuning.** Check the crucial parameters first and try to understand your ANN.
- When your ANN is ready, it is time for the real live experience called **inference**. You can test your ANN model in your application and evaluate it afterwards.
- Hardware like the **Nvidia Jetson (embedded)** and **Drive (automotive)** enable GPU accelerated inference for your development and offer software SDKs for fast development.

Summary


MANAGE
TRAIN
DEPLOY
DIGITS
Caffe
torch
PROTOTXT
GPU INFERENCE ENGINE
TEST
TRAIN
MANAGE / AUGMENT
EMBEDDED
DATA CENTER
AUTOMOTIVE

Guest Lecturer Dr. Christopher Prohm – 23.02.2020

What is the talk about?

- Insights on the differences between academia and industry when applying deep learning
- Insights from the VW:Datalab Deep Learning approach
- Deep Learning on real vehicles: Motorsport example
- Virtual sensors
- Technical tricks for building robust real-world deep learning applications for automotive usage

+ Discussion for the exam