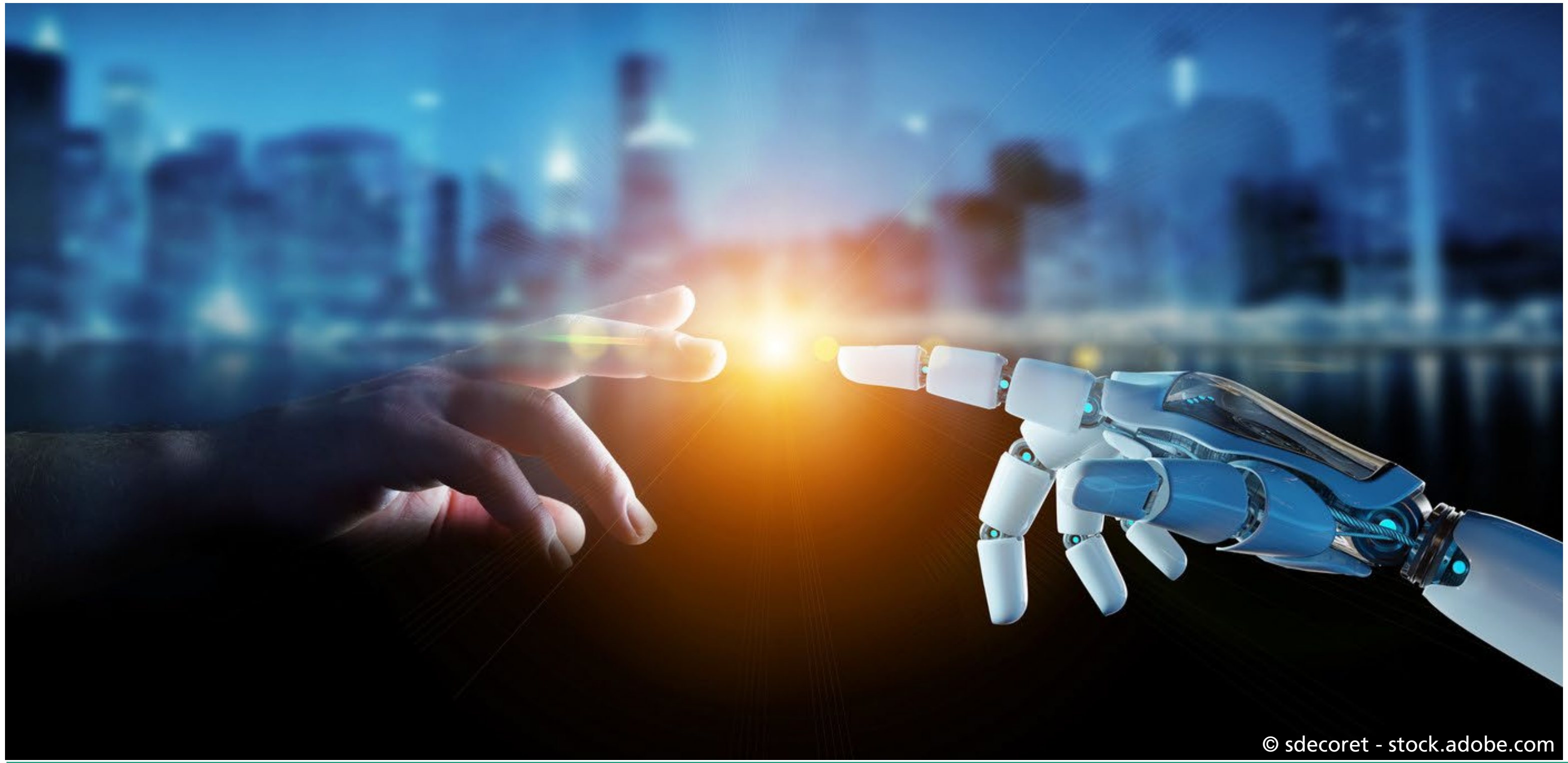
Introduction to TensorFlow

Dr. Gerhard Paaß
Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS)
Sankt Augustin



Nicht zur Veröffentlichung! März 2024

Fraunhofer
BIG DATA AI

Course Overview

1. Intro to Deep Learning	Recent successes, Machine Learning, Deep Learning & types
2. Intro to Tensorflow	Basics of Tensorflow, logistic regression
3. Building Blocks of Deep Learning	Steps in Deep Learning, basic components
4. Unsupervised Learning	Embeddings for meaning representation, Word2Vec, BERT
5. Image Recognition	Analyze Images: CNN, Vision Transformer
6. Generating Text Sequences	Text Sequences: Predict new words, RNN, GPT
7. Sequence-to-Sequence and Dialog Models	Transformer Translator and Dialog models
8. Reinforcement Learning for Control	Games and Robots: Multistep control
9. Generative Models	Generate new images: GAN and Large Language Models

: link to background material, : link to images used in lecture, G. : Terms that may be asked in the exam

Fraunhofer BIG DATA AI

Introduction to TensorFlow

Agenda

1. Training with Tensorflow
2. Jupyter Notebooks
3. Steps to Specify Network

Parallel Processing

- Deep Learning requires high computational effort
  ➔ parallel processing
- Multicore processors
- Graphical Processing Units (GPUs). H100 …
  - ~14592 specialized processors (FP32 CUDA Cores)
  - Use for general computations by CUDA language
  - 80 GB GPU-memory, up to 3958 TeraFlops
  - Memory Bandwidth: ~ 2 TB/s
  - Plugin card to servers
- Cluster of Computers, may have GPUs
  - Slower connection by fast LAN network
  - Usually sublinear speedup
- Computing in the cloud: Amazon cloud

The NVIDIA A100 vs H100, How Do They Compare?

2023-11-30

	H100		A100	
Form Factor	SXM5	x16 PCIe Gen5 2 Slot FHFL 3 NVLINK Bridge	SXM4	x16 PCIe Gen4 2 Slot FHFL 3 NVLink Bridge
Max Power	700W	350W	500W	300W
FP64 TC | FP32 TFLOPS²	67 | 67	51 | 51	19.5 | 19.5	
TF32 TC | FP16 TC TFLOPS²	989 | 1979	756 | 1513	312 | 624	
FP8 TC | INT8 TC TFLOPS/TOPS²	3958 | 3958	3026 | 3026	NA | 1248	
GPU Memory / Speed	80GB HBM3	80GB HBM2e	80GB HBM2e	
Multi-Instance GPU (MIG)	Up to 7		Up to 7	
NVLink Connectivity	Up to 256	2	Up to 8	2

https://docs.nvidia.com/launchpad/ai/h100-mig/latest/h100-mig-gpu.html

G. Parallelization of computati-ons G. graphical processing unit

Fraunhofer BIG DATA AI

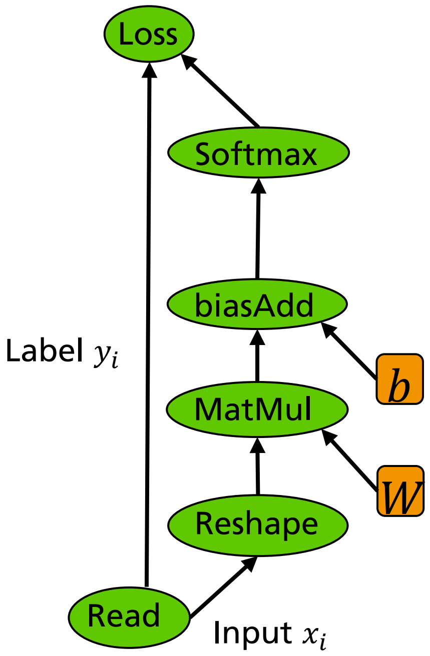
Requirements for Deep Learning Software

- Specify model computations: vectors, matrices, n-dimensional „tensors“
  - Linear algebra: add, multiply
  - Nonlinear functions on tensors: sigmoid, exp, tanh, softmax
  - Compute derivatives for these functions
  - Optimization algorithms: exploit parallel processing
  - Evaluation of performance, apply for prediction
- Large number of available toolkits
  - CNTK: special language, Python, autom. Differentiation. Microsoft
  - PyTorch: Python, autom. Diff., parallel execution. Facebook
  - fast.ai: library on top of Pytorch
  - …

https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software
https://towardsdatascience.com/battle-of-the-deep-learning-frameworks-part-i-cff0e3841750

Tensorflow

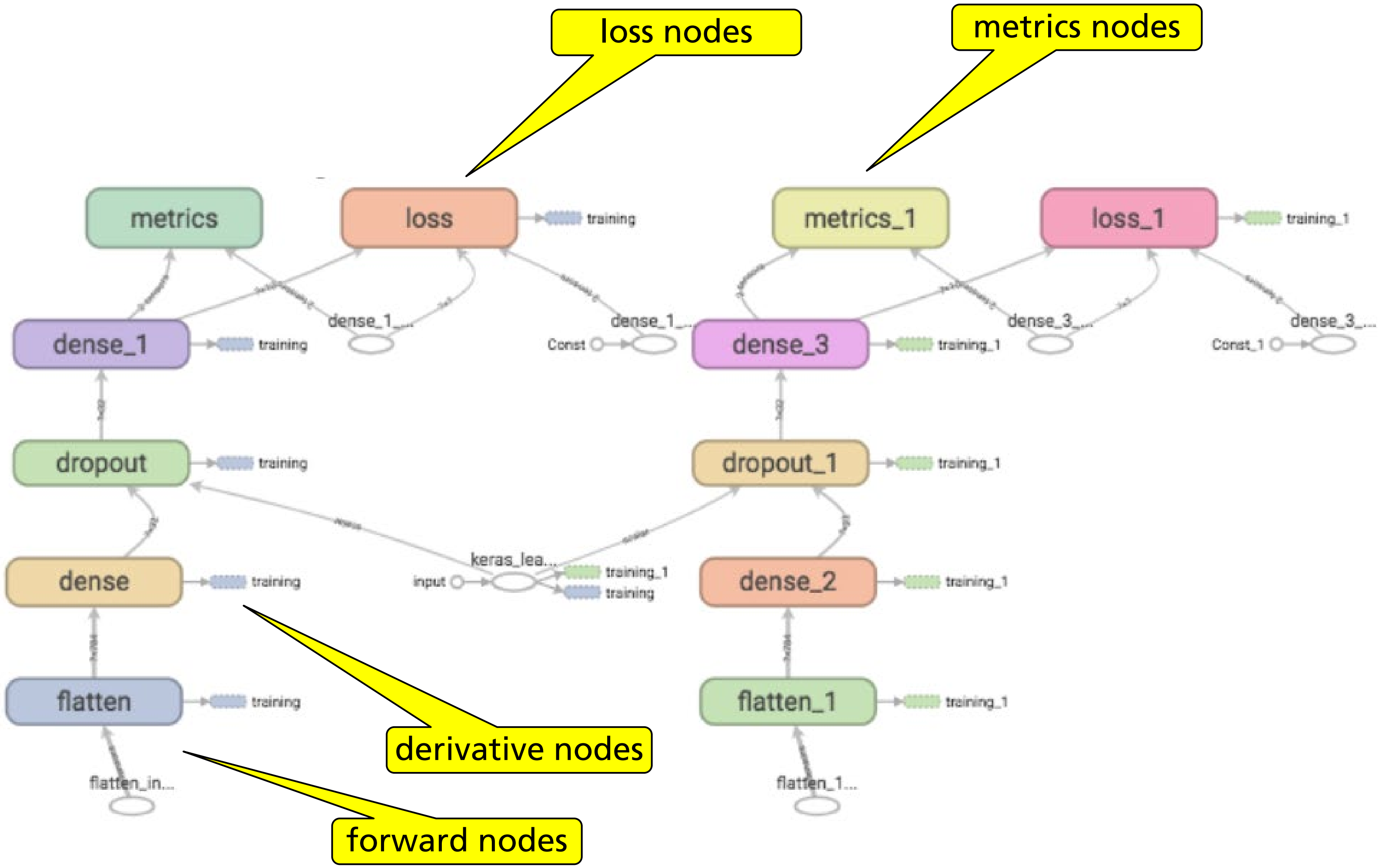
- Python open source library for numerical computation
  - Represents calculation steps as a flow graph
    - Nodes in the graph are mathematical operations or read/write operations
    - Edges in the graph are multidimensional data arrays (tensors)
- Data Flow Graph
  - May be used to compute derivatives automatically (previously major source of error)
- Why Tensorflow?
  - Released by Google in Nov. 2015
  - Python is currently the most polular language of data analysis
  - Large community of contributors
  - 77728 repositories on GitHub mentioning TensorFlow in title



G. Tensorflow G. Flowgraph

Tensorflow Execution

- Data Flow Graph
  - Nodes may be assigned to computational devices
    - different processors of a machine
    - Graphical Processing Units
    - Compute Clusters
- Execute asynchronously and in parallel
  - once all the tensors on their incoming edges becomes available.



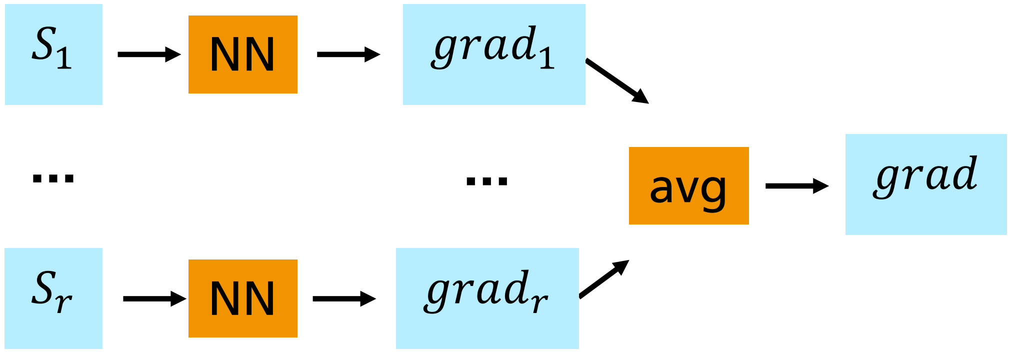
How to Parallelize Neural Network Training

- Assume we have a neural network
  - m layer
  - n elements (x_i, x_i) in the training set

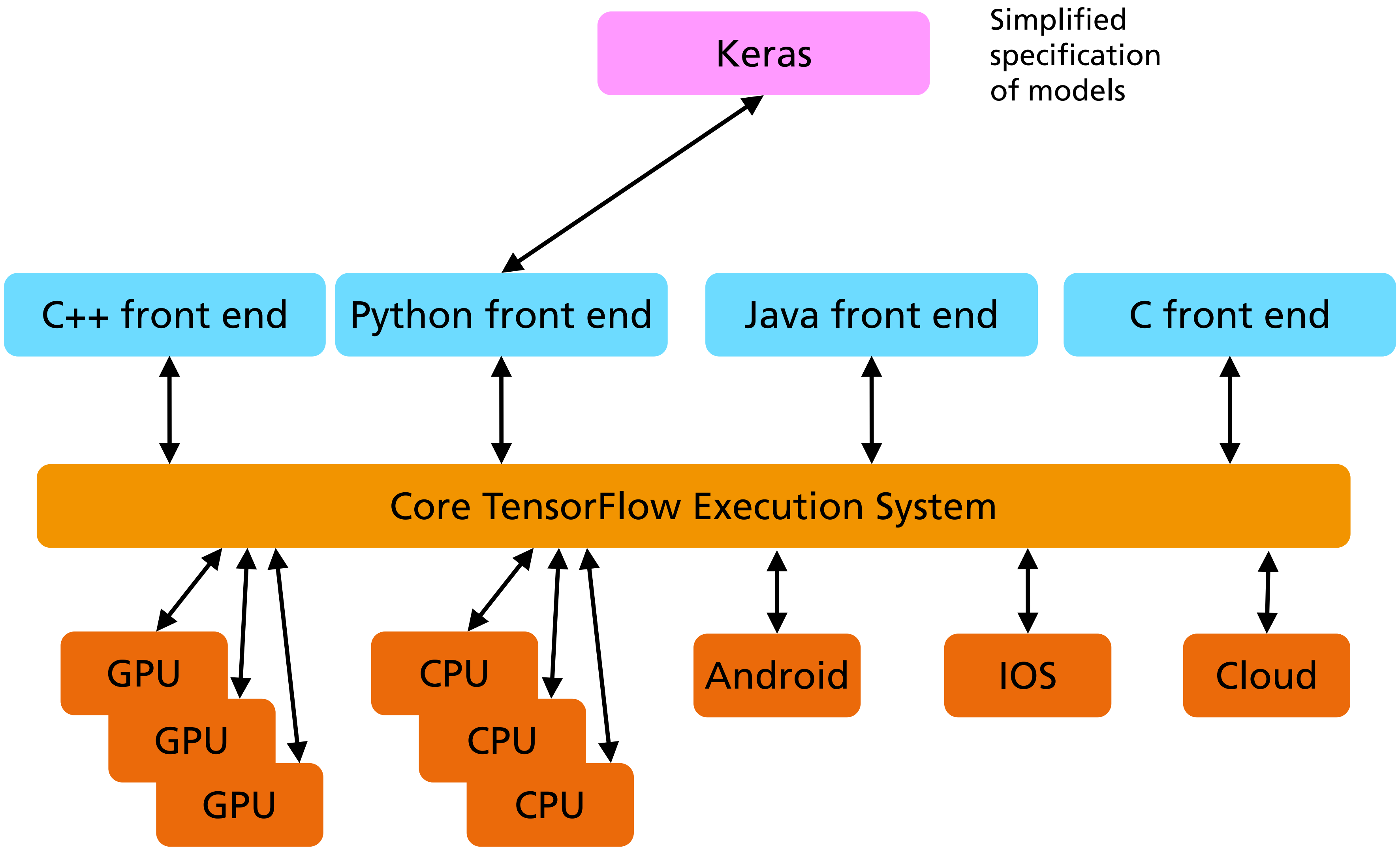
- Parallelization by data:
  - Split the training data into subsets $S_1, \ldots, S_r$
    distribute the S_j to different processors
    train them with the same model code

- Parallelization by operators:
  - In addition distribute the different layers to different processors
  - Establish a pipeline between processors

- Usually this is perfomed automatically by Tensorflow / Pytorch or specialized tools

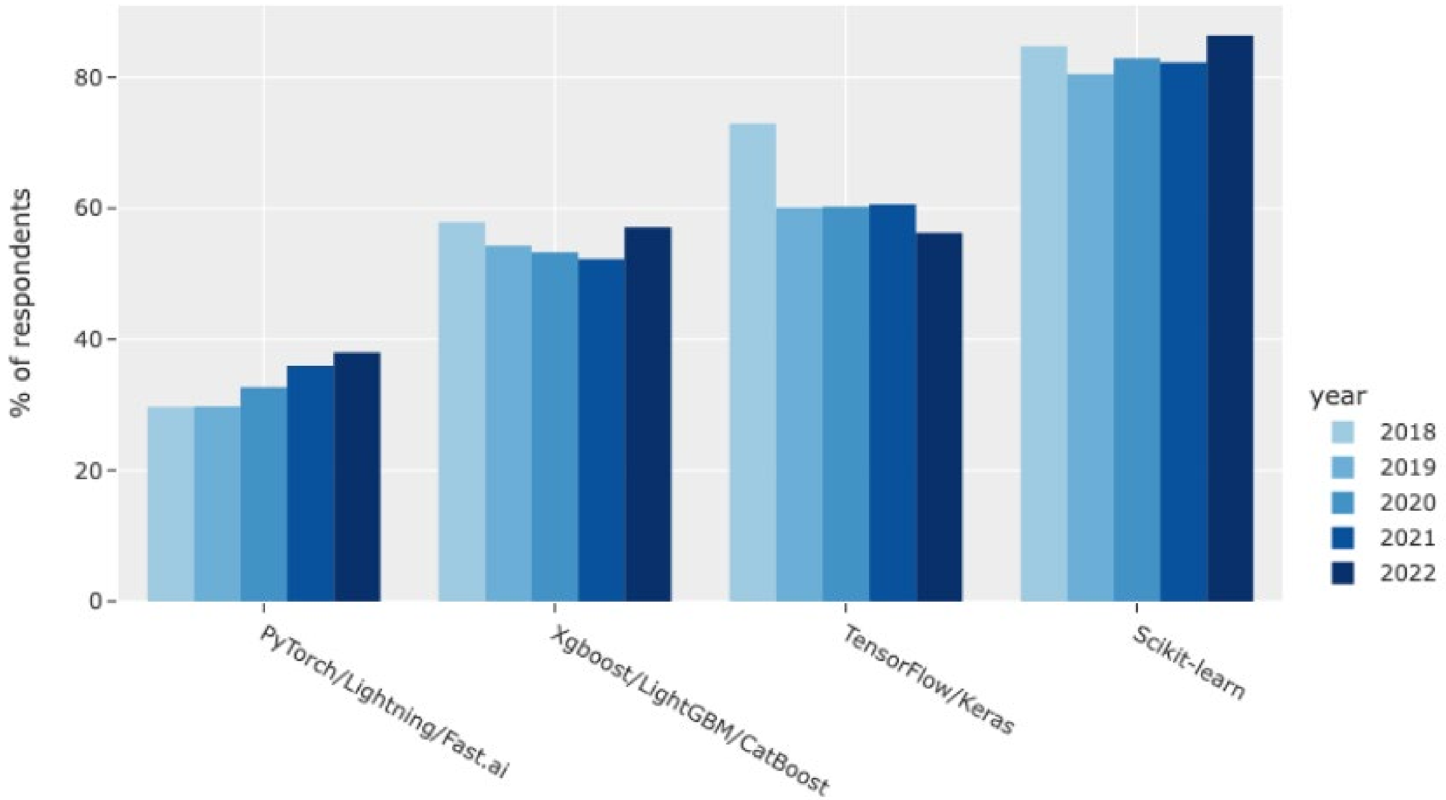


Tensorflow Architecture



ML Frameworks

ML Framework usage by data scientists



https://www.kaggle.com/kaggle-survey-2022

? 02-a

Introduction to TensorFlow

Agenda

1. Training with Tensorflow
2. Keras
3. Jupyter Notebooks
4. Steps to Specify Network

Keras

- Environment on top of Tensorflow: generates Tensorflow commands
- Usually each Command defines a layer
- Example: **Dense** layer: implements $f(Ax + b)$
- Dense generates a function

```
dense_fct = Dense(units=10,activation='softmax’)
```

- Computing $softmax(Ax + b)$, may be applied to a numeric input tensor

```
hid= dense_fct(x)
```

Name of output tensor

```
hid = Dense(units=10,activation='softmax’)(x)
```

Name of layer function

hyperparameters

Name of input tensor

How to specify a network

- Simple network: sequence of layers

```
model=Sequential()
model.add(Dense(32,input_shape(16,),activation='tanh')
model.add(Dense(10, activation='softmax')
```

- alternative

```
model=Sequential(
        Dense(32,input_shape(16,),activation='tanh')
        Dense(10, activation='softmax')
}
```

- model is a function and may be applied to numeric input tensor

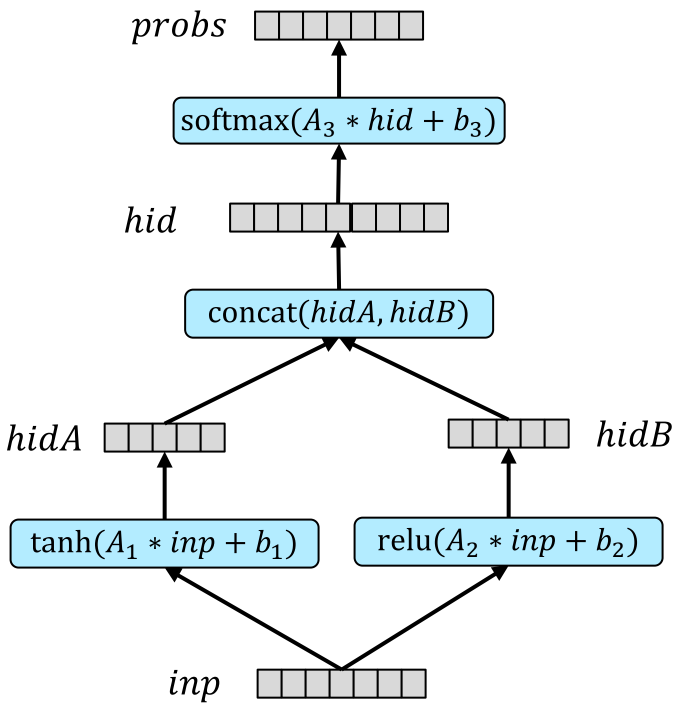
```
out= model(inp)
```


Fraunhofer BIG DATA AI

How to specify a network

- Network with **parallel** paths:
  need to specify input and output tensors

```
inp = keras.Input(shape=(None,28,28)
hidA=Dense(100, activation='tanh')(inp)
hidB=Dense(100, activation='relu')(inp)
hid = layers.concatenate(hidA,hidB)
probs = Dense(10,activation='softmax')(hid)
```



- May specify arbitrary Directed Acyclic Graphs:
  Networks without cyclic connections

? 02-b

Fraunhofer BIG DATA AI

Introduction to TensorFlow

Agenda

1. Training with Tensorflow
2. Jupyter Notebooks
3. Steps to Specify Network

Fraunhofer
BIG DATA AI

Jupyter Notebook

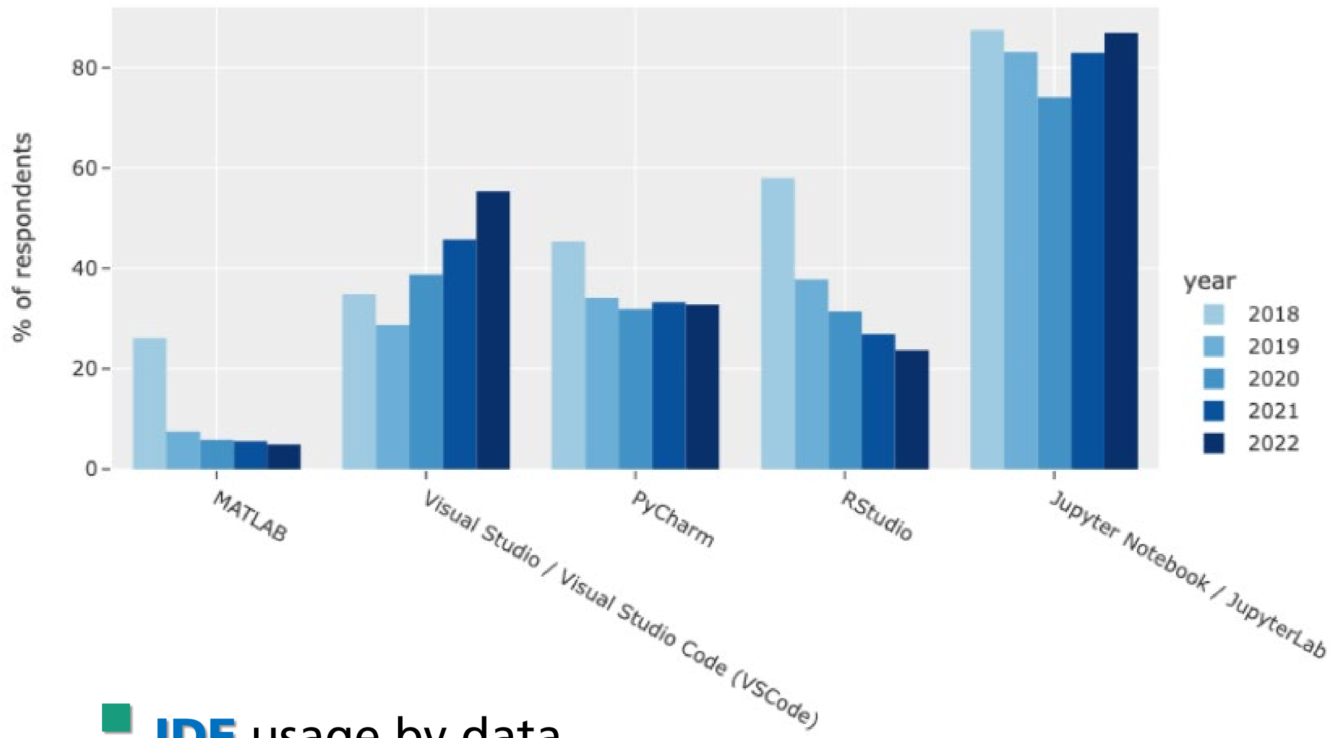
- interactive computing environment
- documents include: - Narrative text – Equations - Images - Live code - Interactive widgets - Plots - Video.
- Three components:
- Notebook web application:
  interactive authoring notebook documents and running code

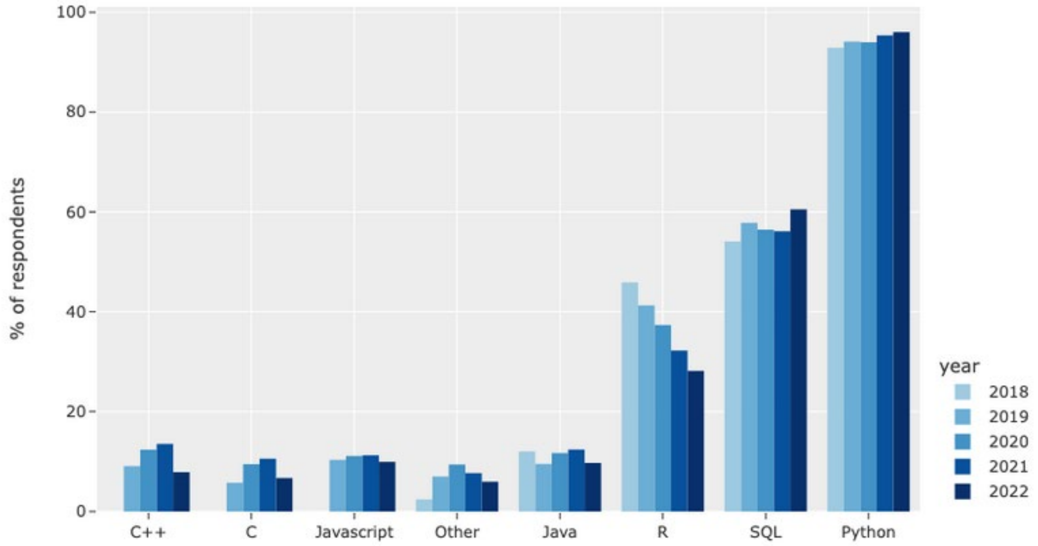
Kernels: Separate process started by the notebook web application runs users' code.

- Notebook documents: contain content visible in the web application:
  - narrative text, equations, images,
  - inputs and outputs of the computations, rich media representations of objects.
- Installation instructions:
  https://www.tensorflow.org/install/
  http://jupyter.org/install.html

Jupyter Notebook

- **Programmming skills of** data scientists
- according to Kaggle survey 2022 🔗



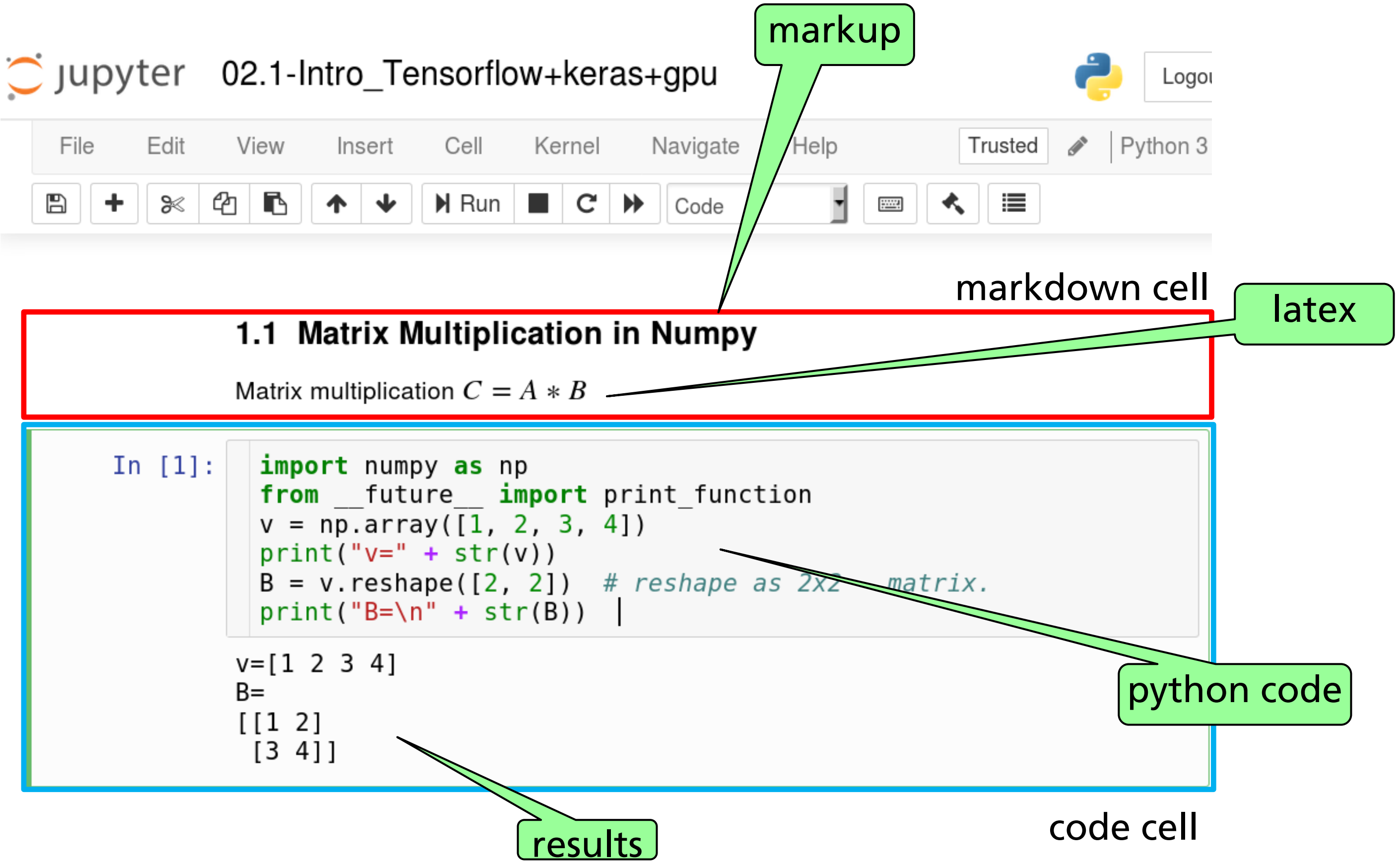


- **IDE** usage by data scientists
- according to Kaggle survey 2022 🔗

https://www.kaggle.com/kaggle-survey-2022

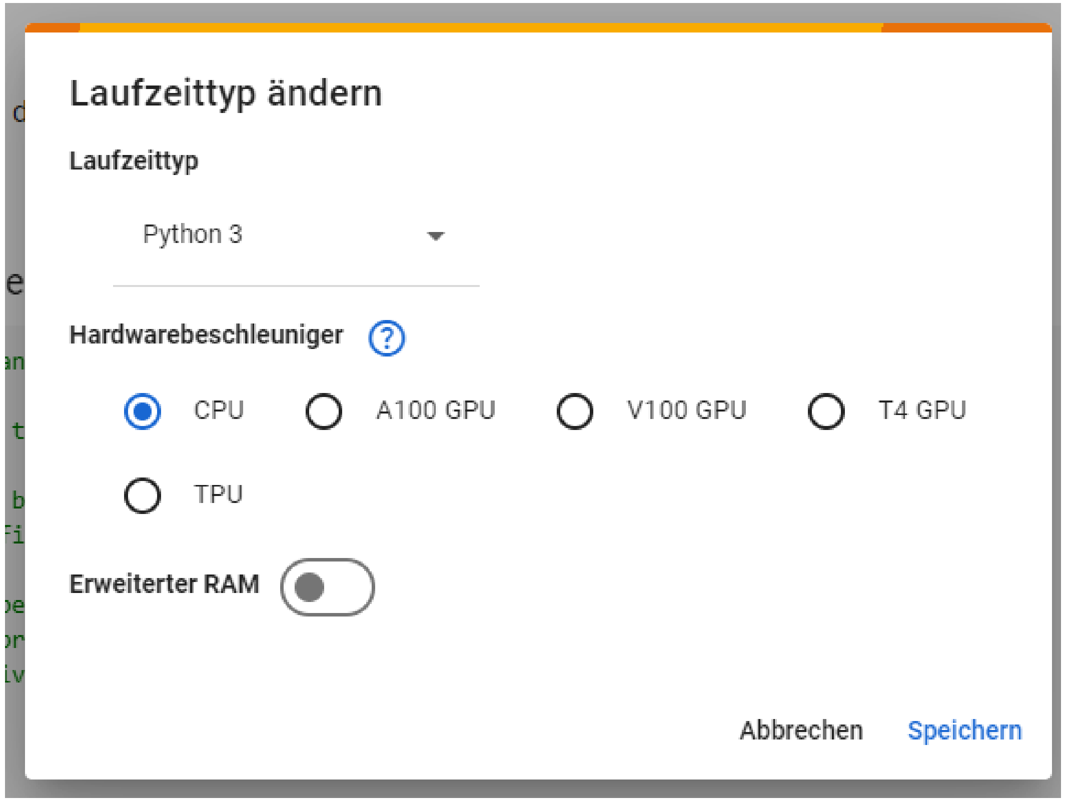
Notebook Web Application

- Notebook consist of a linear sequence of cells.
  - Markdown cells contain narrative text and equations
  - Code cells contain code in a programming language



Google Colab

- Notebook environment for Python hosted in the cloud
  - Installation of Libraries on the fly
  - Very similar to Jupyter Notebook
  - Interactive selection of runtime environment
  - Loading data
    - Load / store from local computer
    - Load / store from Google drive
    - Load / store from Google cloud
    - Load from GitHub and Web
- Start with free version
- Purchase more compute time and better GPUs



Nicht zur Veröffentlichung! März 2024

Fraunhofer BIG DATA AI

Introduction to TensorFlow

Agenda

1. Training with Tensorflow
2. Jupyter Notebooks
3. Steps to Specify Network

Fraunhofer
BIG DATA AI

Model training and Application

Steps for Model Training

1. Read training & test data
2. Preprocess training & test data
3. Define model
4. Estimate model by optimization on training data, save trained model
5. Validate Model on test data

Steps for Model Application

1. Read application data
2. Preprocess application data
3. Read trained model
4. Apply model to application data

G. Model training steps G. Model application steps

Keras

Keras Steps

1. Define **input** tensors using the `Input` function
   `InputTensor = Input(Inputarray)`
2. Define **operators** / Layers with specific functions, e.g.
   `Outputtensor=Layertype(hyperparams)(Inputtensors)`
3. Define the **model** using the `Model` function
   `model = Model(inputTensors, outputTensors)`
4. Define loss, optimizer, and evaluation metric using `compile`
   `model.compile(loss=…, optimizer=…, metrics=…)`
5. Start **training** with the `fit` function
   `history = model.fit(trainData, valData, hyperparam)`