Part III

DL Workflow

Introduction to TensorFlow

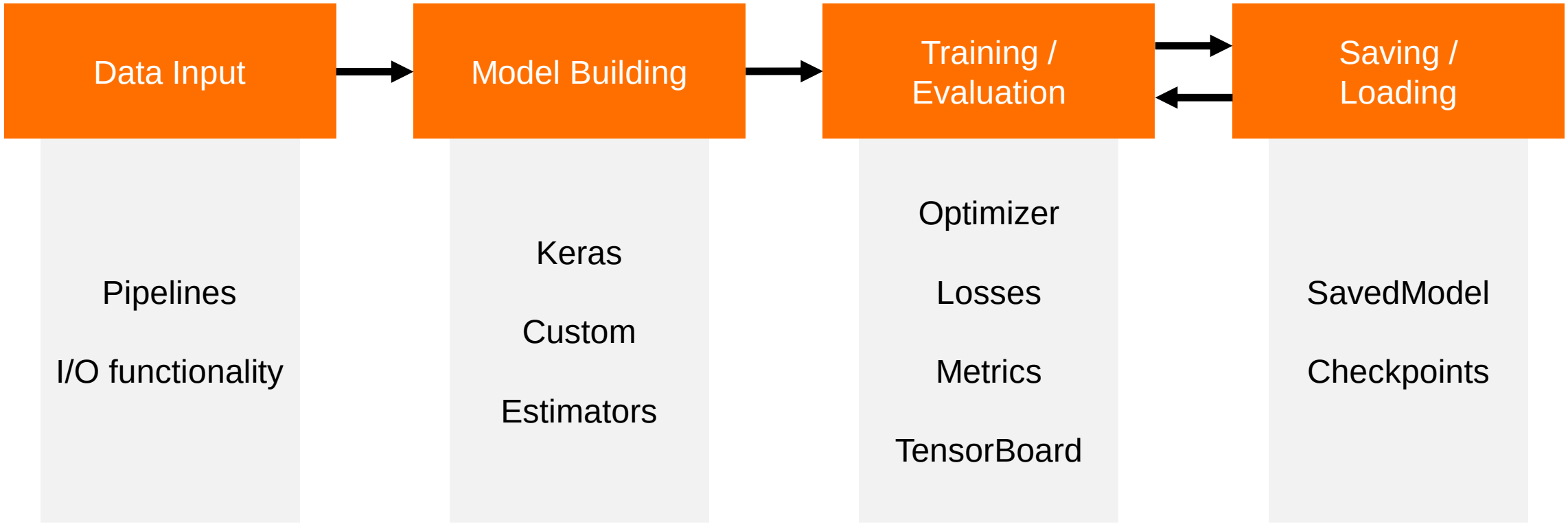
M.Sc. M. Fischer

Prof. Dr.-Ing. B. Yang

Agenda – DL Workflow

- Workflow
- TensorBoard
- Device Placement
- Serialization
- AutoGraph
- Dataset API
- TensorFlow Extensions

Workflow


Data Input
Model Building
Training / Evaluation
Saving / Loading
Pipelines
I/O functionality
Keras
Custom
Estimators
Optimizer
Losses
Metrics
TensorBoard
SavedModel
Checkpoints

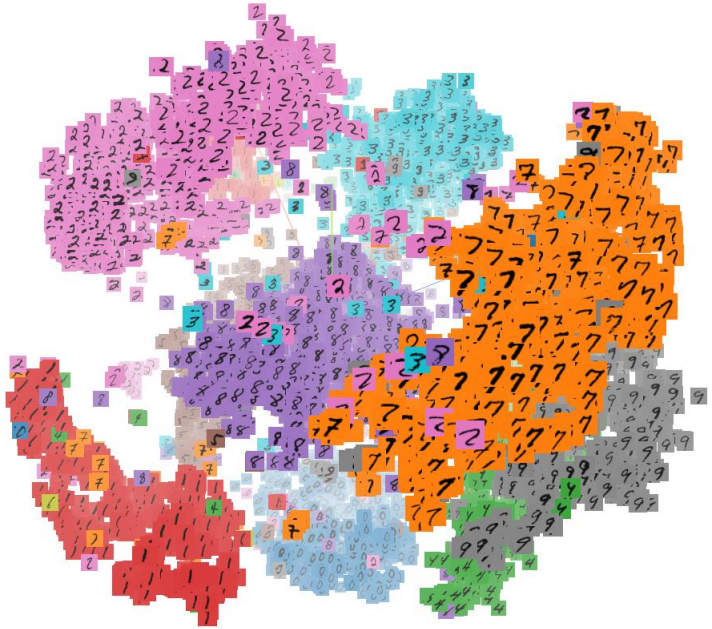
TensorBoard

Tool for visualization and monitoring

- Tracking of scalars

  Losses, Metrics, Variables

- Histograms
- Model architecture / Graph
- Input / Output Images
- Embeddings



MNIST T-SNE via https://projector.tensorflow.org/

TensorBoard

Monitoring the training progress

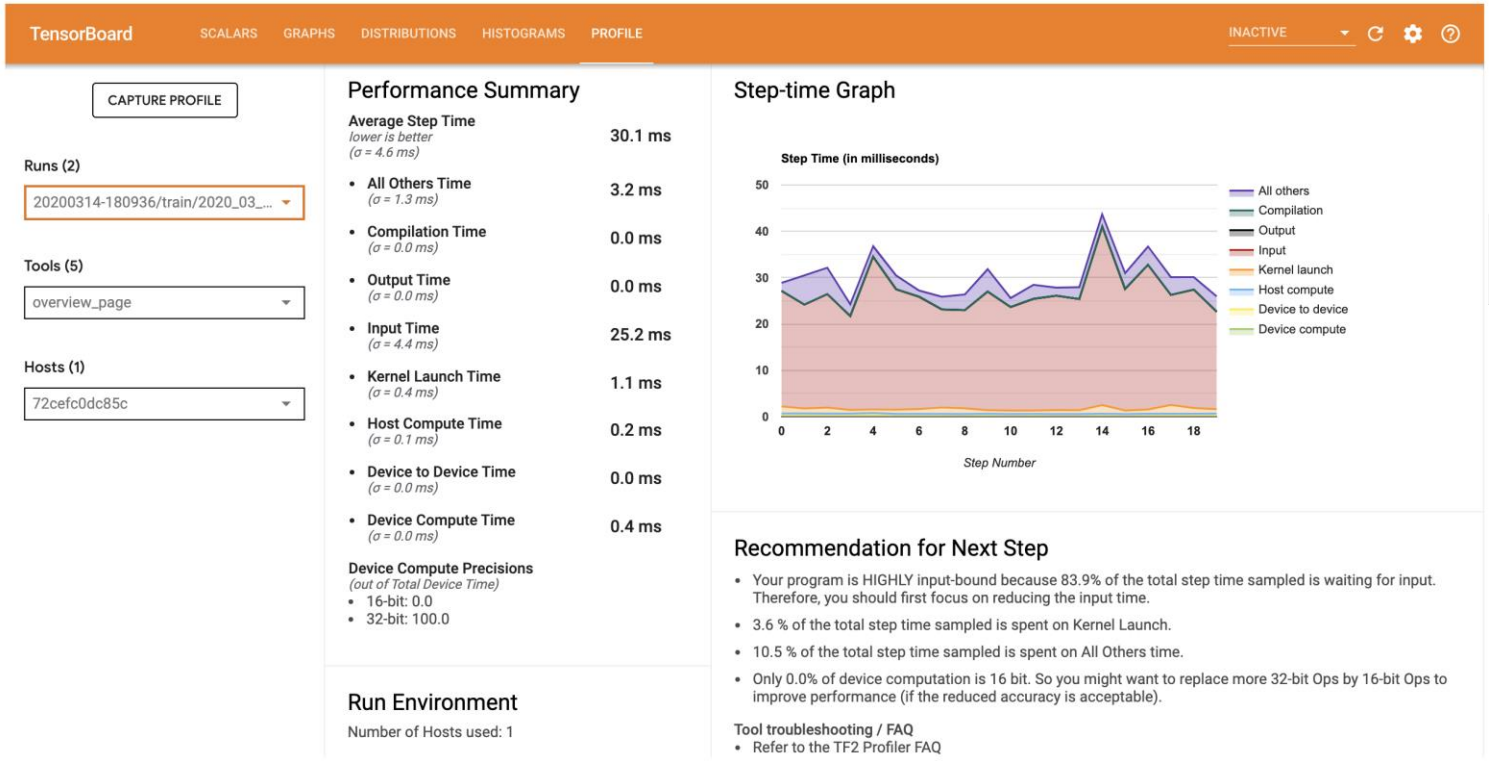
- **Keras**
  - Some logging activated by default
  - TensorBoard callbacks
- **TensorFlow**
  - *tf.summary* API
  - *tf.summary.FileWriter* writes *tf.summary* protocol buffer to event files
  - Most common *tf.summary* buffer

    summary.scalar, summary.histogramm

TensorBoard

Performance Profiler

- Shows time spent at each op on CPU & GPU
- Optimizing feeding and training routine



Notebook - TensorBoard

TensorBoard

- Visualization of model, metrics and further variables

→ **see intro_nb_tensorboard.ipynb**

Device Placement

Hardware Device Placement

- Many TensorFlow ops can (and should) be run on the GPU
- Useful functionality
  - Show available physical devices

```
gpus = tf.config.experimental.list_physical_devices('GPU')
```

  - Show actual device placement

```
tf.debugging.set_log_device_placement(True)
```

  - Allow different device, if placement is not possible

```
tf.config.set_soft_device_placement(True)
```

  - Perform training step on GPU

```
with tf.device('/gpu:*'):
    with tf.GradientTape() as tape:
        ...
```

Distributed Training

- *tf.distribute.Strategy* is an API to distribute operations across GPUs and TPUs.
- Compatible with Keras, Estimators and custom training loops
- TF 2.0 provides several easy to use distribution strategies
  - *MirroredStrategy, TPUStrategy, CentralStorageStrategy*
  - More strategies are being added

Distributed Training

- Useful strategy: *MirroredStrategy*
  - Synchronous distributed training - variables are kept in sync
  - One replica per GPU - each variable is mirrored
  - Batch is divided equally among the multiple replicas
- Model is defined within distribution scope - requires minimal code changes

```
mirrored_strategy = tf.distribute.MirroredStrategy()
with mirrored_strategy.scope():
  model = tf.keras.Sequential([tf.keras.layers.Dense(1, input_shape=(1,))])  # define your model
  model.compile(loss='mse', optimizer='sgd')
model.fit(dataset, epochs=n_epochs)
```

Serialization

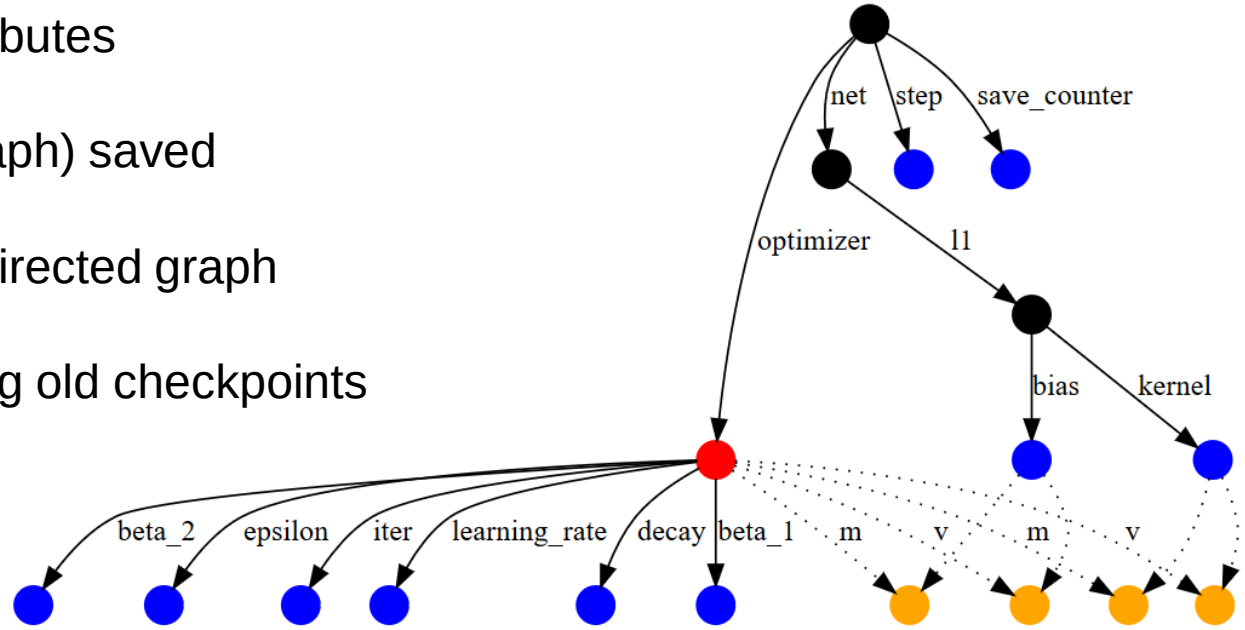
API for saving of (trained) architectures

- Common use cases
  - Saving training progress
  - Sharing models
  - Evaluation (in parallel)
- Three available options for serialization
  - TensorFlow Checkpoints
  - TensorFlow SavedModel
  - Keras.Model saving

Serialization

TensorFlow Checkpoint

- Object that track variables assigned to their attributes
- No description of defined computations (e.g. graph) saved
- TensorFlow matches variables by traversing a directed graph
- CheckpointManager handles saving and keeping old checkpoints



Checkpoint graph, source: tensorflow.org

TensorFlow SavedModel

- SavedModel contains a complete TensorFlow program

Serialization

Keras.Model saving

- Sequential and Functional API allow saving of a single file
  - Models provide data structure that represents a DAG of layers.
  - Can be safely serialized and deserialized
- Keras relies on its own model file, consisting of
  - Model‘s architecture
  - Model‘s weight values
  - Model‘s training config
  - Optimizer and its state
- Keras supports architecture-only and weights-only saving

```
# Save the model
model.save('path_to_my_model.h5')

# Recreate the exact same model purely from the file
new_model = keras.models.load_model('path_to_my_model.h5')
```

Data Management

Dataset API - Motivation

- Feeding training data is one of the most crucial parts
- *tf.data* API
  - provides simple, reusable, efficient pieces
  - enables to build complex input pipelines
- Main building block: *tf.data.Dataset*
  - Represents abstract sequence of elements
  - Each element consist of one or more Tensor objects
- Whole dataset does not have to be loaded into RAM

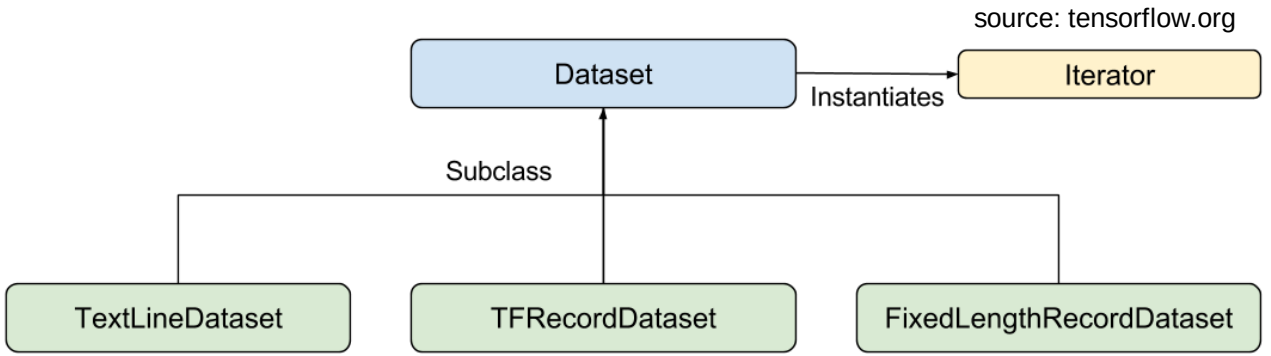
  → Dynamic loading of required files

Data Management

Dataset construction

- From source construction
  - Data in memory
    - *tf.data.Dataset.from_tensors()*
    - *tf.data.Dataset.from_tensor_slices()*
    - *tf.data.Dataset.range()*
  - Data from a file
    - E.g. *tf.data.TFRecordDataset*

      own data format, built on Googles‘ *tf.Example* protocol buffer messages



Dataset API

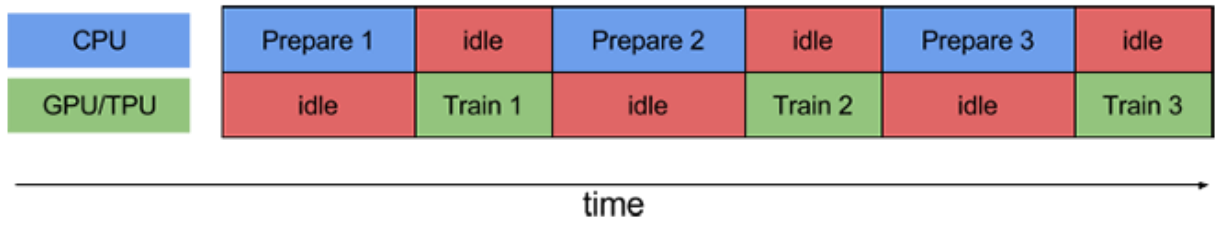
Data transformations

- Constructs a new Dataset from an existing Dataset
- Manipulation of each data element on the fly
- Transformation can vary across each sample
- Commonly used functionality available
  - shuffle, repeat, zip, batch, prefatch, cache
  - Mapping functions: map, flat_map, interleave, filter

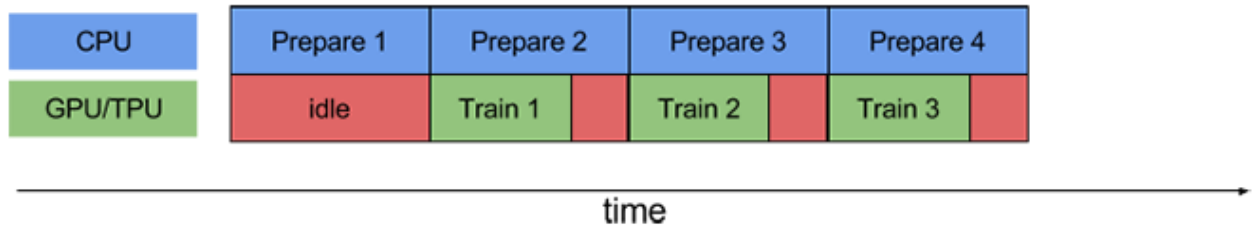
Dataset API

Pipelining

- **Goal:** Achieve maximum GPU utilization
- Essential for any ML/DL framework
- No pipelining – high idle time



- With pipelining – low idle time

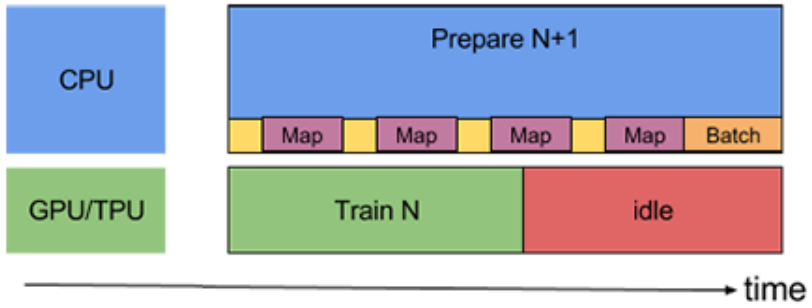


Source: https://www.tensorflow.org/

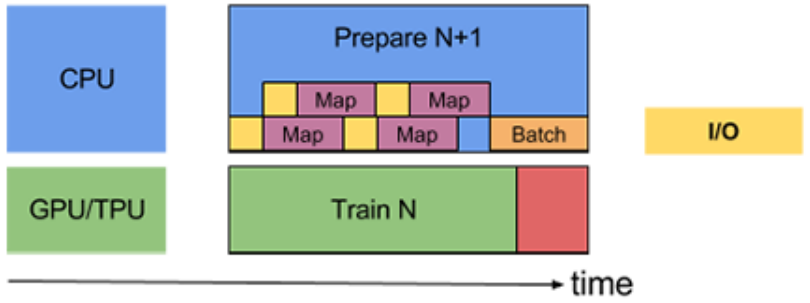
Dataset API

Pipelining

- Making use of advanced tf.data API
- Sequential data transformations



- Parallel data transformations – use multi-core CPU architecture



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

Dataset API

Pre-packaged Datasets

- Keras and TensorFlow both offer public datasets

Keras Datasets

- Provides a few popular datasets (CIFAR10, MNIST, IMDB Movie reviews, …)
- Returns NumPy arrays

TensorFlow Datasets

- Underlying is the *tensorflow_datasets* package
- Several datasets covered: audio, image, structured, text, translate, video
- Returns *tf.data.Dataset* and additional info

AutoGraph

Basic Idea: Eager operations & optimized TF computation graph

- AutoGraph allows conversion of eager and most Python statements into graph code
- Graph optimization can be applied
- Common Python control statements are supported
- Tensor expressions and Python control flow can be mixed
- Function is “traced“ on conversion

AutoGraph

Python decorators

- Python functions are objects
- Concept to wrap functions
- Decorator can be applied to multiple functions

***tf.function* decorator**

- Triggers **AutoGraph** conversion
- *tf.function* decorator is transparent
- **Any** functions called within an annotated function will also run in graph mode

```
def my_decorator(func):
    def wrapper():
        print("Something is happening before the function is called.")
        func()
        print("Something is happening after the function is called.")
    return wrapper

@my_decorator
def say_whee():
    print("Whee!")


>>> say_whee()
Something is happening before the function is called.
Whee!
Something is happening after the function is called.
```

see: https://realpython.com/primer-on-python-decorators/

Notebook - Workflow

Explore the workflow

- Keras model serialization and loading
- TF datasets and input pipelines
- AutoGraph conversion of TF and Python code

→ **see intro_nb_workflow.ipynb**

Note: Information that can only be found within this notebook will not be relevant to the exam

Not just Deep Learning

- Implementations of popular ML algorithms for TensorFlow are available
  - K-means clustering
  - Boosted Trees
  - Kernel methods (Support Vector Machines)
  - Gaussian Mixture Model
  - Linear/logistic regression

- **But**, main focus is on DNNs
  - Functionality may be scattered / outdated
  - Custom code adjustments may be needed.

  → for non DL / Machine Learning (ML) algorithms

  scikit-learn is an excellent Python library

TensorFlow Hub

- Repository and library for reusable machine learning
- Provides pre-trained models

  text embeddings, image classification models, and more

Arbitrary Neural Artistic Stylization

TensorFlow Extensions

A fast growing ecosystem

- TF Hub
- TF Serving
- TF Lite
- TF.js
- TF Addons
- TF Transforms
- TF Probability
- TF Agents
- TF Graphics

Head over to www.tensorflow.org/resources/libraries-extensions for more

Beyond the Introduction

- Knowledge of fundamentals, API and workflow of TensorFlow
- Experiment with the notebooks / programming assignments
- Find more on https://www.tensorflow.org/tutorials