EVML

TENSORFLOW & KERAS 2
HANDS-ON

JEROEN VEEN

HAN_UNIVERSITY
OF APPLIED SCIENCES

AGENDA

- DL portfolio walkthrough
- Tensorboard, visualizing the training process
- Data augmentation
- Storing and loading models
- Experimental route to optimize architectures
- Fine-tuning neural network hyperparameters

DL PORTFOLIO TEMPLATE

- Walkthrough

CATEGORICAL CROSS-ENTROPY

- Model compilation

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
              metrics=['accuracy'])
```

- Multi-class classification: use categorical crossentropy
- NOT sparse in case of one-hot encoding of the labels:
  e.g.[1,0,0], [0,1,0], [0,0,1]
- Sparse in case labesl are integers, e.g [1], [2], [3]
- See tf.keras.losses.SparseCategoricalCrossentropy | TensorFlow v2.10.0

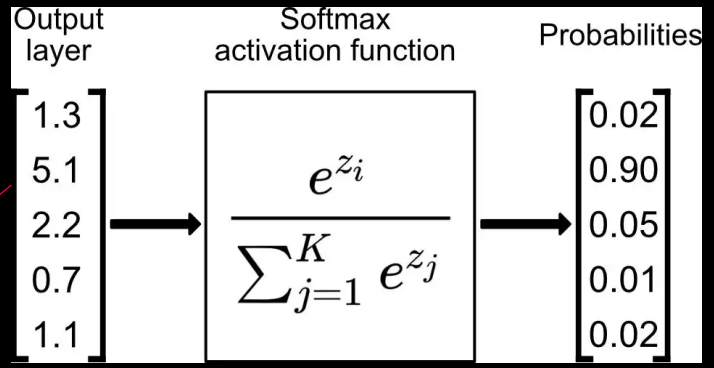
SOFTMAX

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
              metrics=['accuracy'])
```

Is default!

- Softmax returns a confidence score
- Loss can be computed before or after softmax
- Numerical stability

logits



THE HISTORY OBJECT

```
history = model.fit(X_train, y_train, epochs=10, validation_split=0.1, sh
```

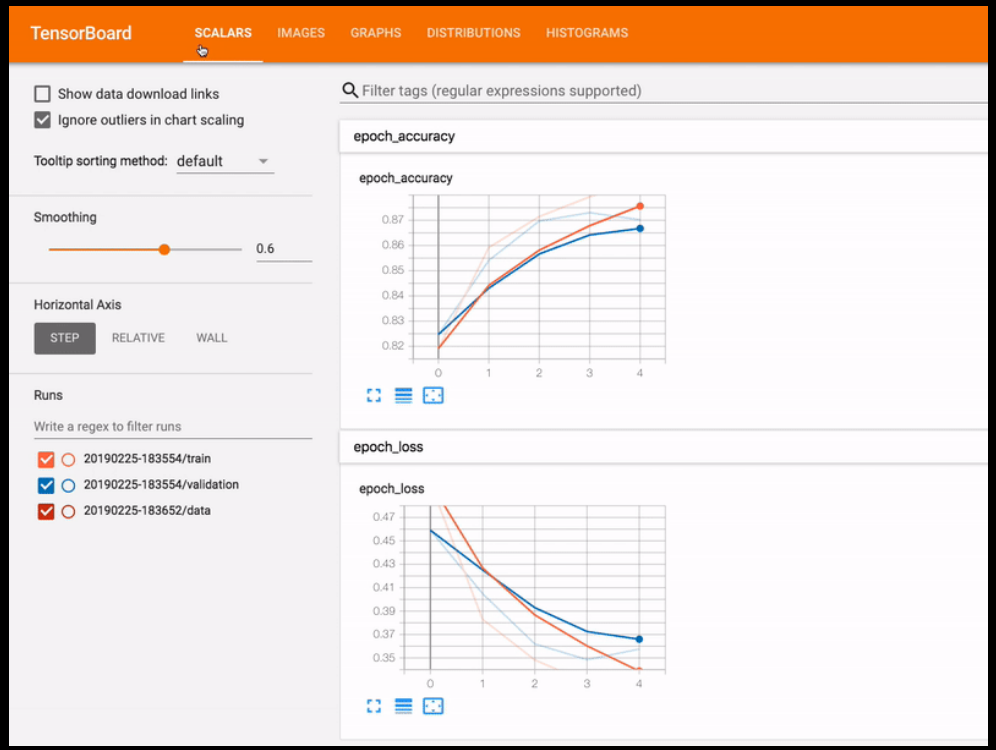
- History.history attribute: a dictionary recording training loss values and metrics values at successive epochs, as well as validation loss values and validation metrics values (if applicable).

- Keys depend on the loss function chosen for the optimizer, e.g. dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])

- Keys: “val_loss”, “val_mean_squared_error”, “loss”, “mean_squared_error”

TENSORBOARD FOR VISUALIZATION

- TensorFlow's visualization toolkit



https://www.tensorflow.org/tensorboard

TENSORBOARD

- Visualize learning curves during training
- Compare learning curves between multiple runs
- TensorBoard server
- https://www.youtube.com/watch?v=2U6Jl7oqRkM

- More elaborate video
  https://www.youtube.com/watch?v=eBbEDRsCmv4

TENSORBOARD GET STARTED

- https://www.tensorflow.org/tensorboard/get_started

```
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)

model.fit(x=x_train,
          y=y_train,
          epochs=5,
          validation_data=(x_test, y_test),
          callbacks=[tensorboard_callback])
```

- Run tensor board server from command line

```
%tensorboard --logdir logs/fit
```

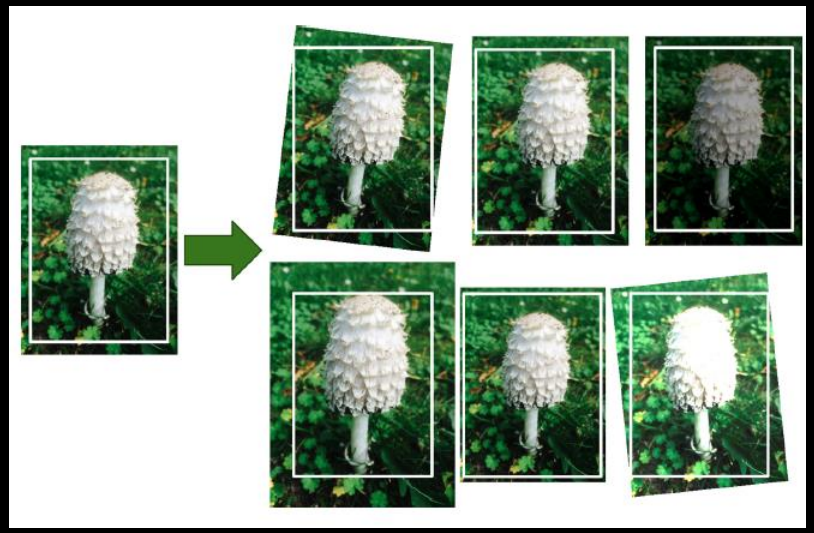
EXERCISE: TENSORBOARD

- Change your MNIST fashion exercise code, so that tensorboard is used
  See e.g. Géron 10.2, (page 326?)

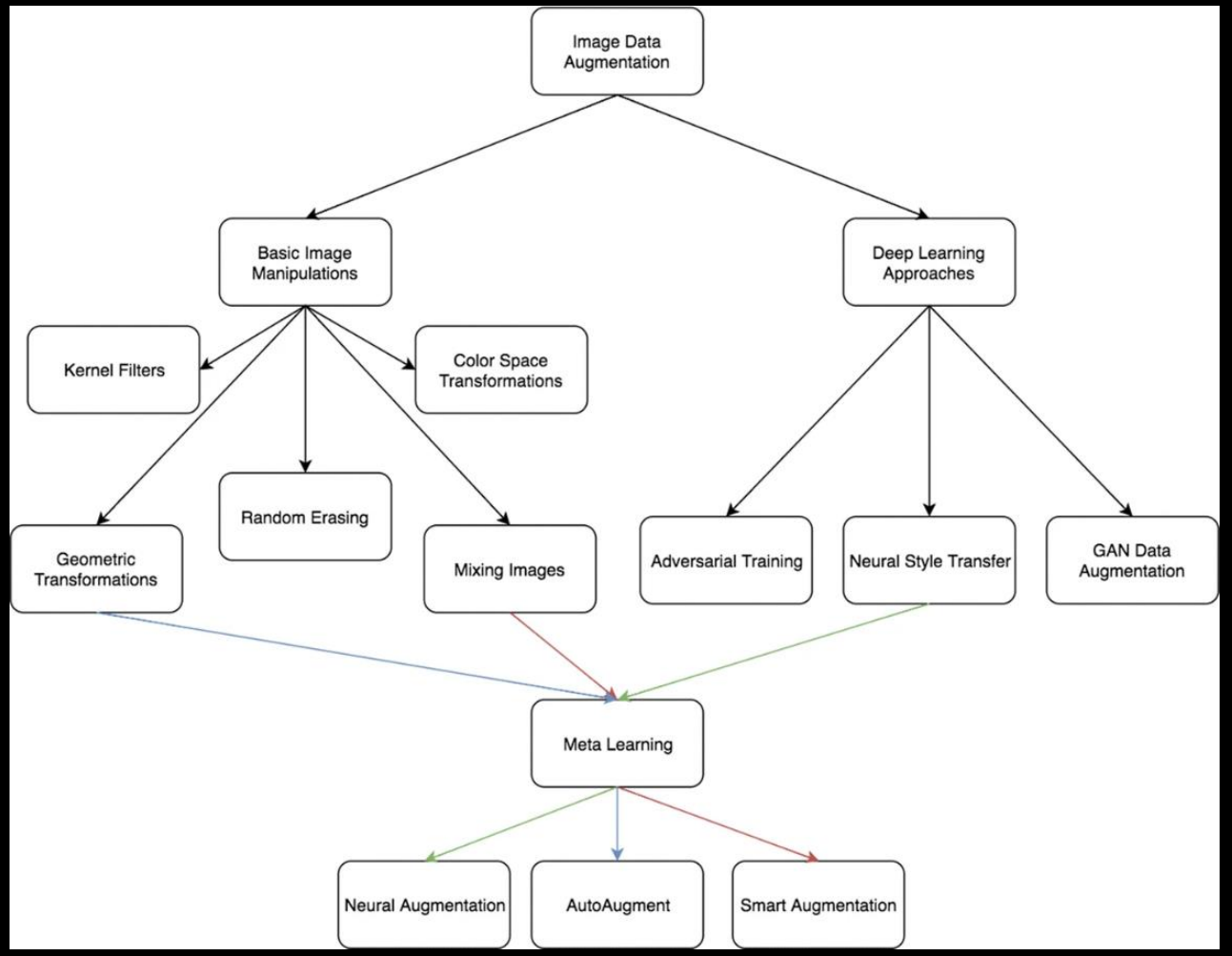
- Examine the TensorFlow Graph, see also
  https://www.tensorflow.org/tensorboard/graphs

DATA AUGMENTATION

- Artificially growing the training set
- Randomly shifting the training images by various offsets, flipping them horizontally, and changing the lighting conditions.



APPROACHES TO AUGMENTATION



Source: M.Khoshgoftaar, C. S. (2019). A survey on Image Data Augmentation

EXERCISE

Try out Keras ImageDataGenerator method, see e.g.

- https://keras.io/api/preprocessing/image/
- https://www.pyimagesearch.com/2019/07/08/keras-imagedatagenerator-and-data-augmentation/

MORE ON IMAGEDATAGENERATOR

```python
# Using image data generator and directory iterator
# See: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image
data_gen = keras.preprocessing.image.ImageDataGenerator(rescale=1./255, \
        preprocessing_function=None, validation_split=0.2) #, rotation_range=20)

images, labels = next(data_gen.flow_from_directory(data_path))

train_ds = keras.preprocessing.image.DirectoryIterator(data_path, data_gen, \
        target_size=image_shape[0:2], batch_size=batch_size, subset="training")
```

```python
tf.keras.preprocessing.image.ImageDataGenerator(
    featurewise_center=False,
    samplewise_center=False,
    featurewise_std_normalization=False,
    samplewise_std_normalization=False,
    zca_whitening=False,
    zca_epsilon=1e-06,
    rotation_range=0,
    width_shift_range=0.0,
    height_shift_range=0.0,
    brightness_range=None,
    shear_range=0.0,
    zoom_range=0.0,
    channel_shift_range=0.0,
    fill_mode="nearest",
    cval=0.0,
    horizontal_flip=False,
    vertical_flip=False,
    rescale=None,
    preprocessing_function=None,
    data_format=None,
    validation_split=0.0,
    dtype=None,
)
```

SAVING AND RESTORING A MODEL

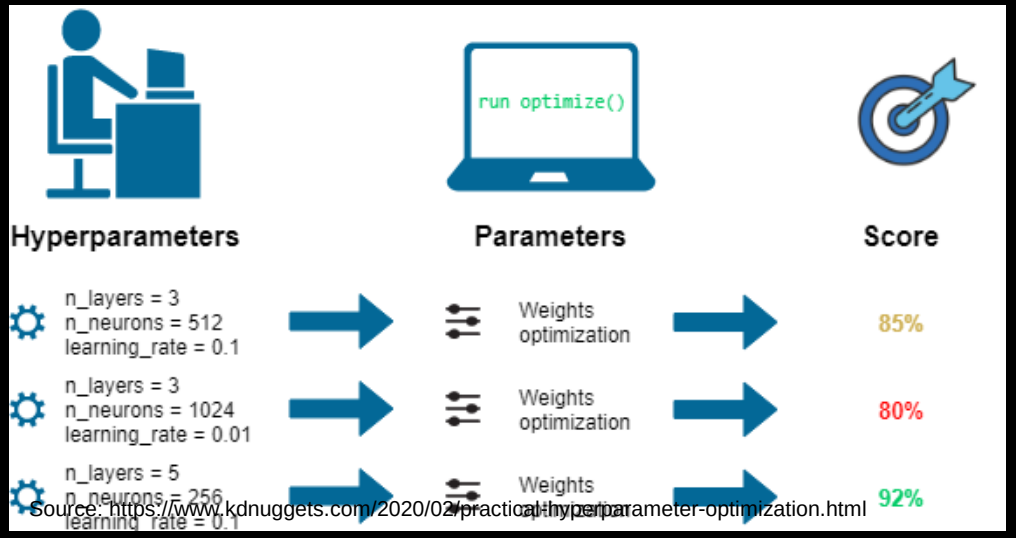
- See Géron page 314

- Watch https://www.youtube.com/watch?v=qFJeN9V1ZsI&t=3152s until about 1:01:00

```
model = keras.layers.Sequential([...]) # or keras.Model([...])
model.compile([...])
model.fit([...])
model.save("my_keras_model.h5")
```

FINE-TUNING HYPERPARAMETERS

- Explore the hyperparameter space either on a grid or at random
- Keras tuner, a.o.
- Cloud hyperparameter tuning service (e.g. google AI)



Source: https://www.kdnuggets.com/2020/02/practical-hyperparameter-optimization.html

SOME HEURISTICS FOR MODEL TUNING

- Small batch sizes can cause instability
- Beware that the effects of different hyperparameters are data dependent
- Training error should steadily decrease, steeply at first, and should eventually plateau as training converges.
  - If the training has not converged, try running it for longer.
  - If the training error decreases too slowly, increasing the learning rate may help it decrease faster.
- Sometimes the exact opposite may happen if the learning rate is too high.
  - If the training error varies wildly, try decreasing the learning rate.
  - Lower learning rate plus larger number of steps or larger batch size is often a good combination

NUMBER OF HIDDEN LAYERS

- Start with just one or two hidden layers
- Deep networks have higher parameter efficiency
- Ramp up the number of hidden layers until you start overfitting the training set
- Reuse parts of a pretrained state-of-the-art network that performs a similar task

NUMBER OF NEURONS

- Nr of neurons in input and output layers is determined by the task
- Hidden layers: pyramidal
- Try increasing the number of neurons gradually until the network starts overfitting. (or use regularization)

EXERCISE: FINETUNE USING SKLEARN

- Explore tuning of parameters in the MNIST fashion exercise
- See Géron page 320-322
- Note that the exploration may last many hours, depending on the hardware, the size of the dataset, the complexity of the model.
- If your machine is too slow, you could start working with Colab