VL Deep Learning for Natural Language Processing

15. DL in Practice

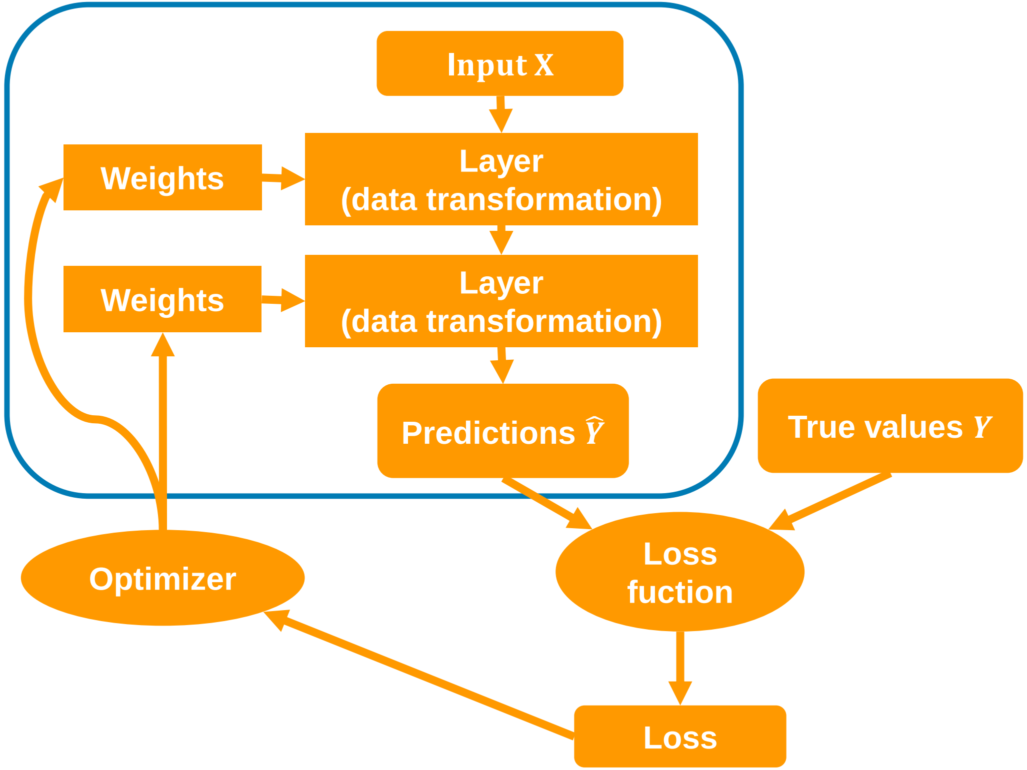
Prof. Dr. Ralf Krestel

AG Information Profiling and Retrieval

Exam Date

- Mo: 18.7.
- Tue: 19.7.
- Wed: 20.7. 10-12
- Thu: 21.7.
- Fri: 22.7.

Anatomy of a Neural Network



Alternative Notation

```
from keras.models import Sequential, Model
from keras import layers
from keras import Input
seq_model = Sequential()
seq_model.add(layers.Dense(32, activation='relu', input_shape=(64,)))
seq_model.add(layers.Dense(32, activation='relu'))
seq_model.add(layers.Dense(10, activation='softmax'))

input_tensor = Input(shape=(64,))
x = layers.Dense(32, activation='relu')(input_tensor)
x = layers.Dense(32, activation='relu')(x)
output_tensor = layers.Dense(10, activation='softmax')(x)
model = Model(input_tensor, output_tensor)
model.summary()
```


```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_2 (Dense)              (None, 32)                1056
_________________________________________________________________
dense_3 (Dense)              (None, 10)                330
=================================================================
Total params: 3,466
Trainable params: 3,466
Non-trainable params: 0
```

Short for:

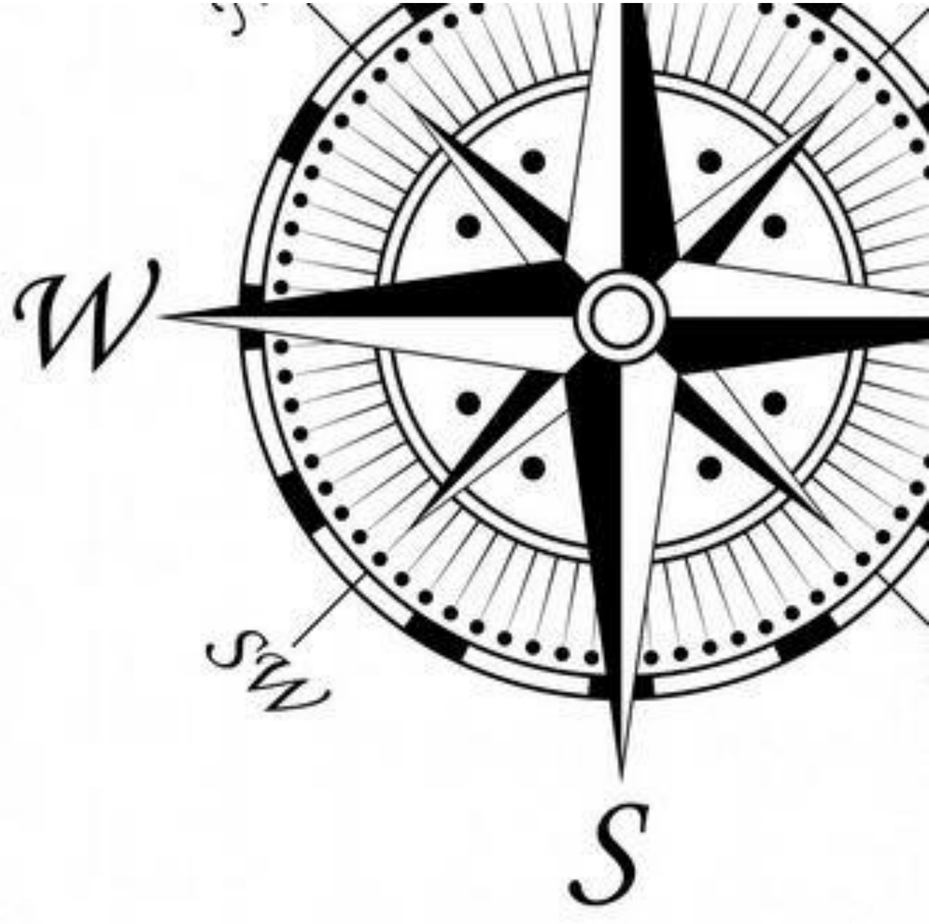
```
dense = layers.Dense(32,
activation='relu')
x = dense(input_tensor)
```

Learning Goals for this Chapter

- Implement non-sequencial models in Keras
  - Multiple in- and outputs
  - Complex architectures
- Monitoring of training process in Keras
  - Callbacks
- Visualization of network parameters
  - TensorBoard
- Optimization of hyperparameters

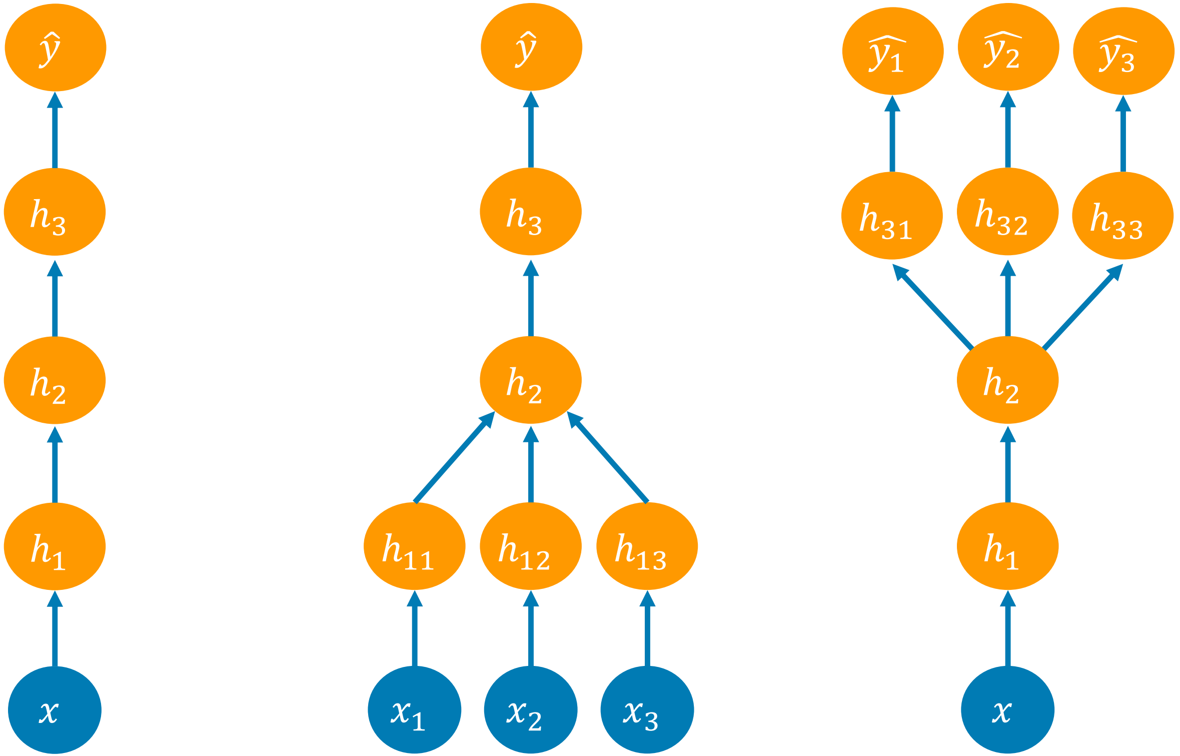
- Relevant chapters:
  - P7
  - S7 (2017) https://www.youtube.com/watch?v=PicxU81owCs

Topics Today

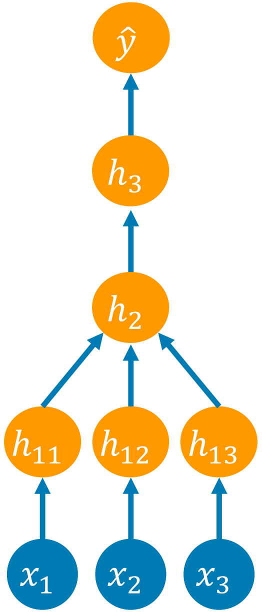


1. **Multi-Input/Output Models**
2. Complex Network Architectures
3. Callbacks and TensorBoard
4. Hyperparameter Optimization

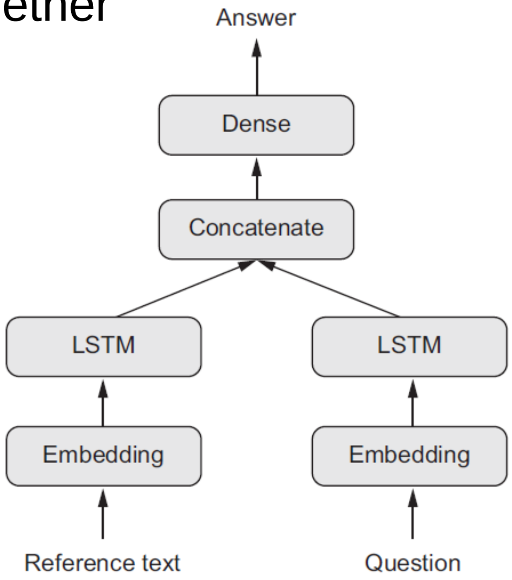
Non-Sequential Network Architectures



Multi-Input Models I



- Multiple different input data types are precessed seperately
- One layer to merge the threads together
  - Summation
  - Concatenation
  - …
- E.g. for question answering
  - Input: news article + question
  - Ouput: answer consisting of one word
    - Softmax



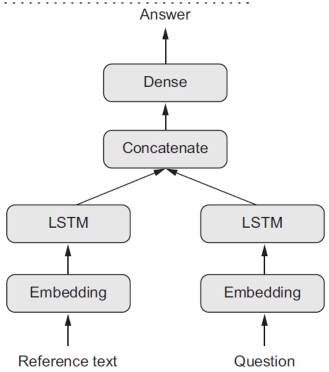
Multi-Input Models II

```
from keras.models import Model
from keras import layers
from keras import Input
text_vocabulary_size = 10000
question_vocabulary_size = 10000
answer_vocabulary_size = 500

text_input = Input(shape=(None,), dtype='int32', name='text')
embedded_text = layers.Embedding(64, text_vocabulary_size)(text_input)
encoded_text = layers.LSTM(32)(embedded_text)

question_input = Input(shape=(None,), dtype='int32', name='question')
embedded_question = layers.Embedding(32, question_vocabulary_size)(question_input)
encoded_question = layers.LSTM(16)(embedded_question)

concatenated = layers.concatenate([encoded_text, encoded_question], axis=-1)
answer = layers.Dense(answer_vocabulary_size, activation='softmax')(concatenated)
model = Model([text_input, question_input], answer)
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['acc'])
```

Sequence of integers of variable lengths

Sequence of embeddings

The sequence becomes a vector



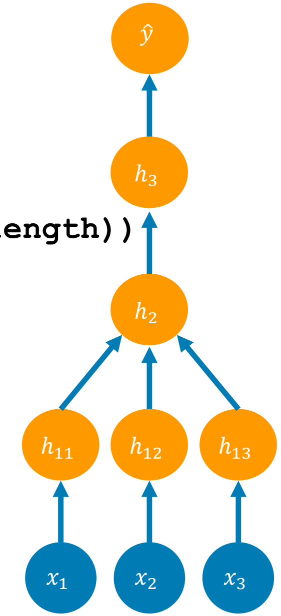
Multi-Input Models III

```
import numpy as np

num_samples = 1000
max_length = 100

text = np.random.randint(1, text_vocabulary_size, size=(num_samples, max_length))
question = np.random.randint(1, question_vocabulary_size, size=(num_samples, max_length))

answers = np.random.randint(0, 1, size=(num_samples, answer_vocabulary_size))

model.fit([text, question], answers, epochs=10, batch_size=128)
```

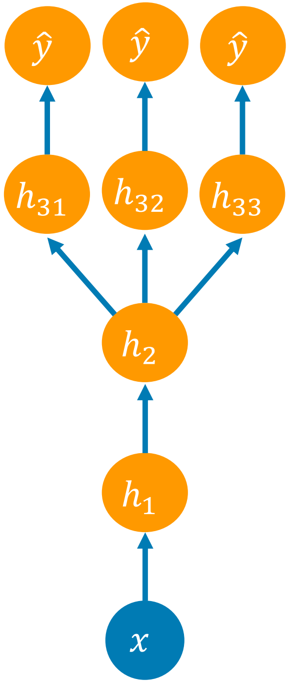
Random data

Alternative:

```
model.fit({'text': text, 'question': question},
          answers, epochs=10, batch_size=128)
```

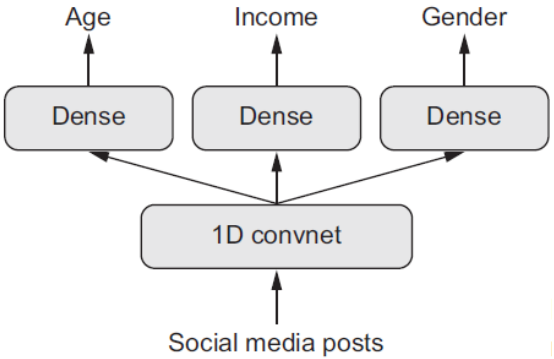


Multi-Output Models I



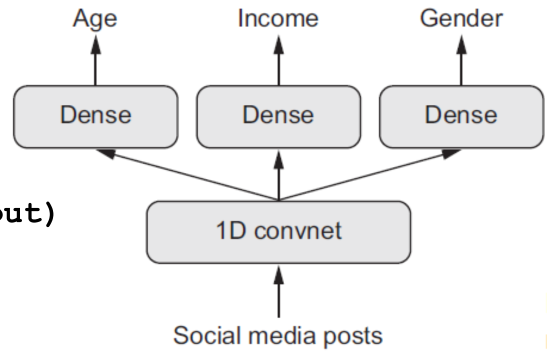
- Models predicting multiple, different target variables (heads)

- E.g. social media profiling
  - Input: a series of posts from one person on a social media platform
  - Output: predict characteristics of this person
    - Age
    - Gender
    - Income



Multi-Output Models II

```
from keras import layers, Input
from keras.models import Model
vocabulary_size = 50000
num_income_groups = 10
posts_input = Input(shape=(None,), dtype='int32', name='posts')
embedded_posts = layers.Embedding(256, vocabulary_size)(posts_input)
x = layers.Conv1D(128, 5, activation='relu')(embedded_posts)
x = layers.MaxPooling1D(5)(x)
x = layers.Conv1D(256, 5, activation='relu')(x)
x = layers.Conv1D(256, 5, activation='relu')(x)
x = layers.MaxPooling1D(5)(x)
x = layers.Conv1D(256, 5, activation='relu')(x)
x = layers.Conv1D(256, 5, activation='relu')(x)
x = layers.GlobalMaxPooling1D()(x)
x = layers.Dense(128, activation='relu')(x)
age_prediction = layers.Dense(1, name='age')(x)
income_prediction = layers.Dense(num_income_groups,activation='softmax',name='income')(x)
gender_prediction = layers.Dense(1, activation='sigmoid', name='gender')(x)
model = Model(posts_input, [age_prediction, income_prediction, gender_prediction])
```



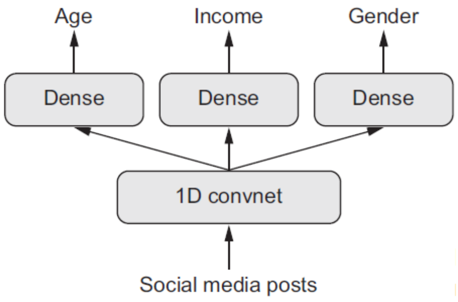
Multi-Output Models III

```
model.compile(optimizer='rmsprop',
        loss=['mse', 'categorical_crossentropy', 'binary_crossentropy'])
model.compile(optimizer='rmsprop',
        loss=['mse', 'categorical_crossentropy', 'binary_crossentropy'],
        loss_weights=[0.25, 1., 10.])
model.fit(posts, [age_targets, income_targets, gender_targets], epochs=10, batch_size=64)
```

Values of the loss functions are summed up to a global loss value

```
model.compile(optimizer='rmsprop',
        loss={'age': 'mse',
              'income': 'categorical_crossentropy',
              'gender': 'binary_crossentropy'})
model.compile(optimizer='rmsprop',
        loss={'age': 'mse','income': 'categorical_crossentropy',
              'gender': 'binary_crossentropy'},
        loss_weights={'age': 0.25,'income': 1., 'gender': 10.})
model.fit(posts, {'age': age_targets,'income': income_targets,'gender': gender_targets},
         epochs=10, batch_size=64)
```

Alternative

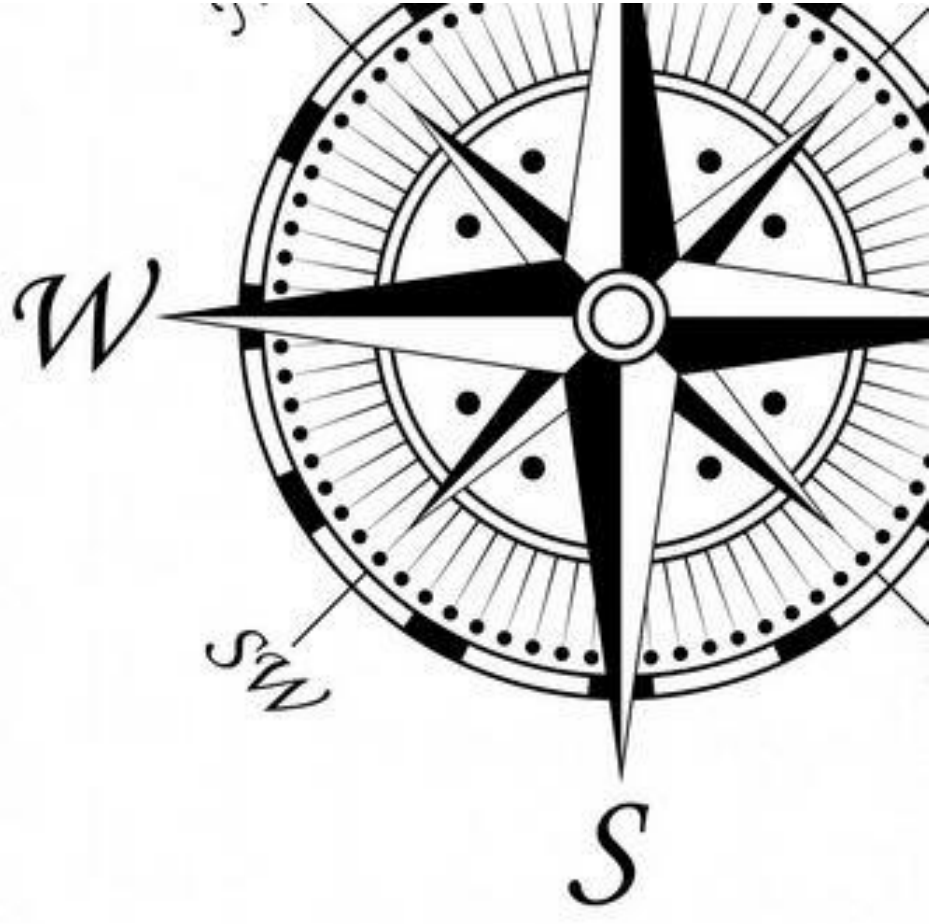


Multi-Input/-Outpt Models

- What are useful tasks for multi-input models?
- What are useful tasks for multi-output models?

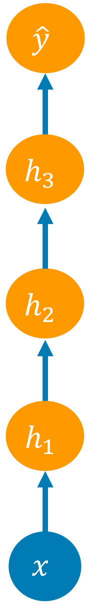
Start 5 4 3 2 1 End

Topics Today

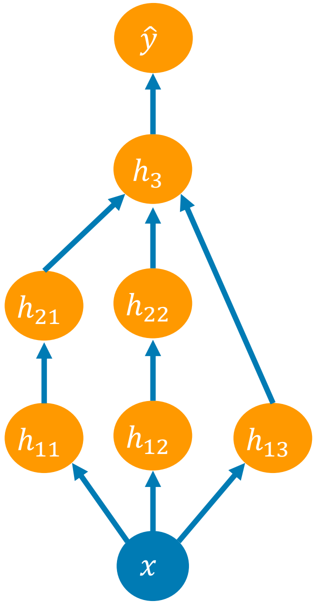


1. Multi-Input/Output Models
2. **Complex Network Architectures**
3. Callbacks and TensorBoard
4. Hyperparameter Optimization

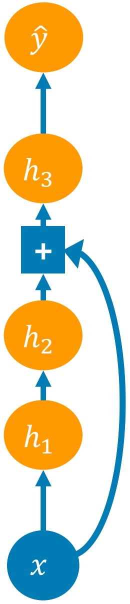
Directed Acyclic Graphs as Network Architectures



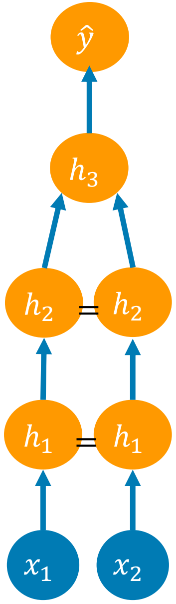
Inception Modules



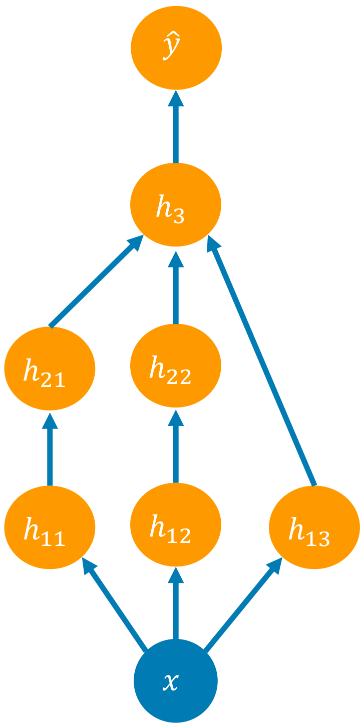
Residual Connections



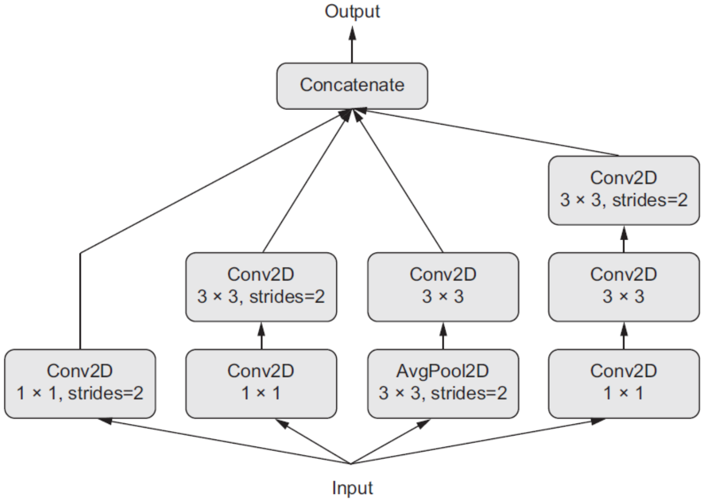
Siamese Nets



Inception Module I



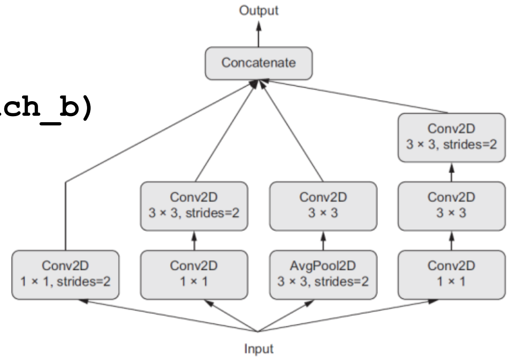
- Is used in convolutional neural networks
- Developed by Google in 2013
- A network consists of many small modules
  - Each module is a small network



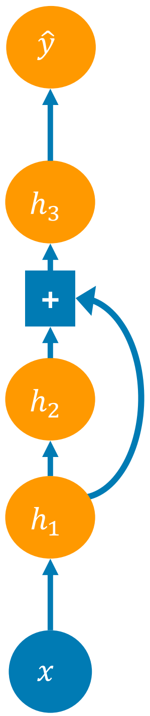
- General architecture
  - 3-4 branches
  - 1x1-convolutions
  - 3x3-convolutions
  - Partly results are concatenated

Inception Module II

```
from keras import layers

branch_a = layers.Conv2D(128, 1, activation='relu', strides=2)(x)

branch_b = layers.Conv2D(128, 1, activation='relu')(x)
branch_b = layers.Conv2D(128, 3, activation='relu', strides=2)(branch_b)

branch_c = layers.AveragePooling2D(3, strides=2)(x)
branch_c = layers.Conv2D(128, 3, activation='relu')(branch_c)

branch_d = layers.Conv2D(128, 1, activation='relu')(x)
branch_d = layers.Conv2D(128, 3, activation='relu')(branch_d)
branch_d = layers.Conv2D(128, 3, activation='relu', strides=2)(branch_d)

output = layers.concatenate([branch_a, branch_b, branch_c, branch_d], axis=-1)
```

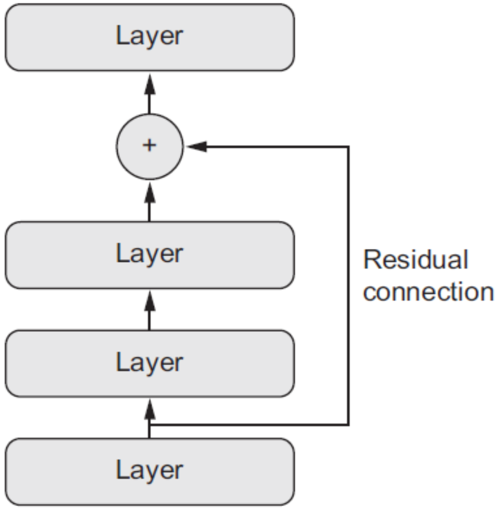


Residual Connections I



- Introduced by Microsoft 2015
- Tackles two problems of large deep learning models
  - Vanishing Gradients
    - Deep networks are untrainable
  - Representation bottlenecks
    - Example: Frequence filter for audio processing

- All models with more than 10 layers should use residual connections!

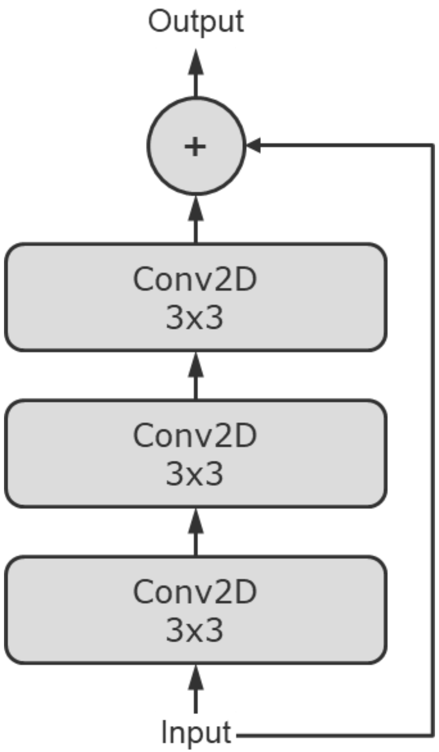


Residual Connections II

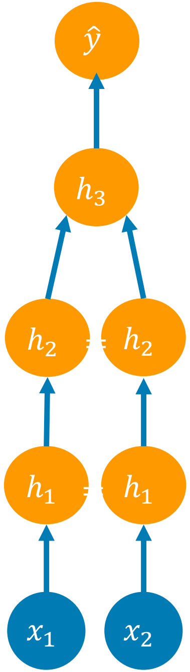
```
from keras import layers
x = ...
y = layers.Conv2D(128, 3, activation='relu', padding='same')(x)
y = layers.Conv2D(128, 3, activation='relu', padding='same')(y)
y = layers.Conv2D(128, 3, activation='relu', padding='same')(y)
y = layers.add([y, x])
```

If feature-map size differ:

```
from keras import layers
x = ...
y = layers.Conv2D(128, 3, activation='relu', padding='same')(x)
y = layers.Conv2D(128, 3, activation='relu', padding='same')(y)
y = layers.MaxPooling2D(2, strides=2)(y)
residual = layers.Conv2D(128, 1, strides=2, padding='same')(x)
y = layers.add([y, residual])
```



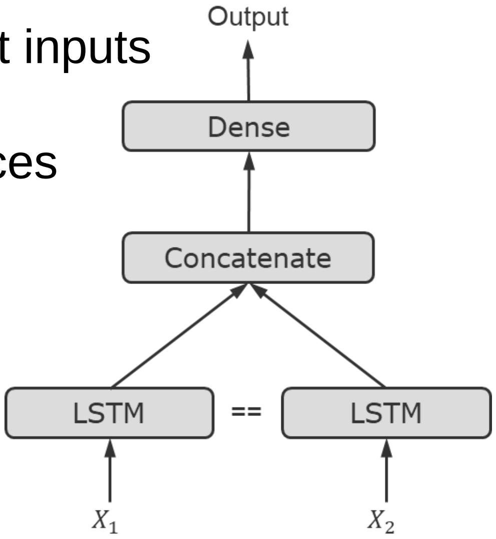
Siamese Nets I



- Reuse of layers
  - Sharing of learnt weights
  - One layer can be used in multiple branches

- Representation is simultaneously learnt for different inputs

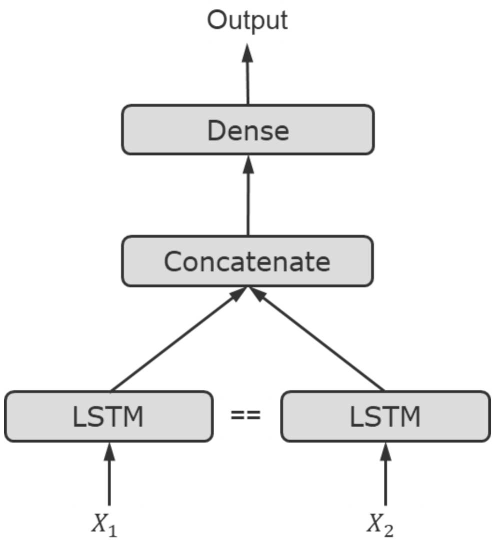
- E.g. compute the semantic similarity of two sentences
  - Input: sentence X_1 and X_2
    - Similarity is symetric
  - Output: $[0,1]$ with 0=no relation and
    1=semantically identical
  - Siamese LSTM



Siamese Nets II

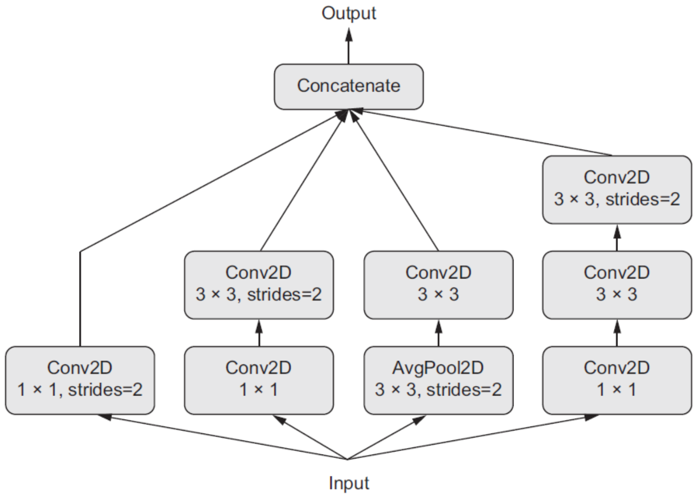
```
from keras import layers
from keras import Input
from keras.models import Model
lstm = layers.LSTM(32)
left_input = Input(shape=(None, 128))
left_output = lstm(left_input)
right_input = Input(shape=(None, 128))
right_output = lstm(right_input)
merged = layers.concatenate([left_output, right_output], axis=-1)
predictions = layers.Dense(1, activation='sigmoid')(merged)
model = Model([left_input, right_input], predictions)
model.fit([left_data, right_data], targets)
```

Layer is instantiated once



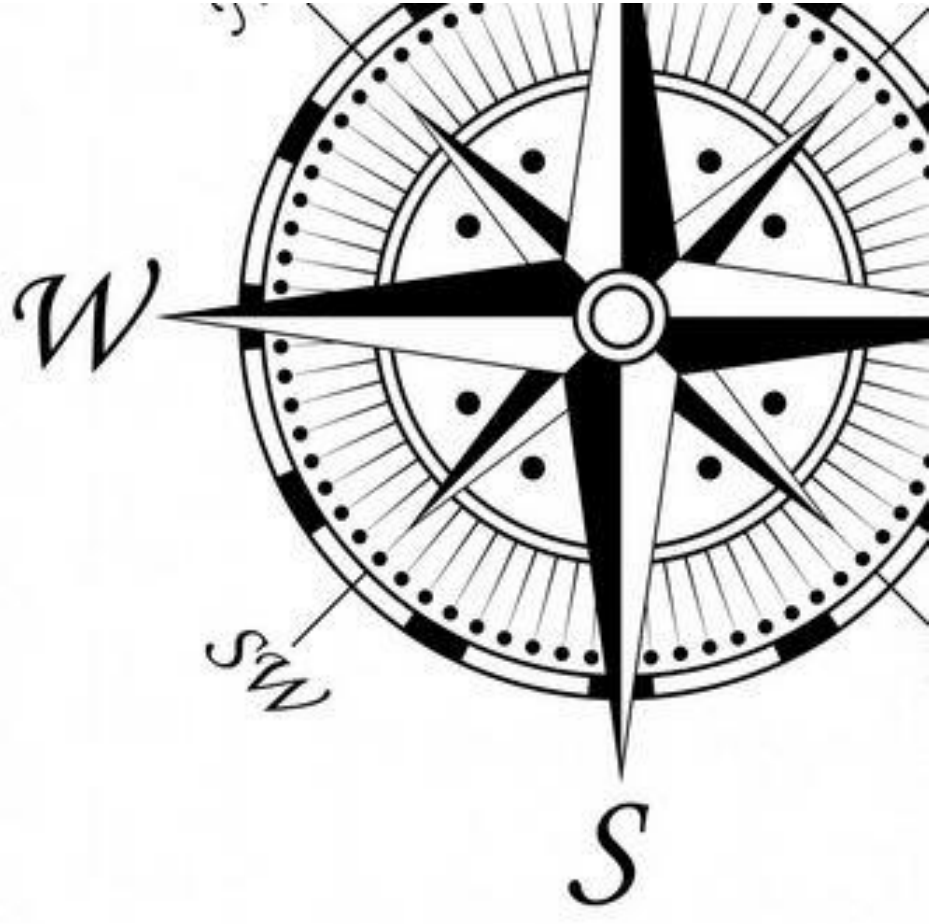
Complex Network Architectures

- What is the use of 1x1-convolutions, e.g. used in one of the moduls of the V3 Network below?



Start 5 4 3 2 1 End

Topics Today



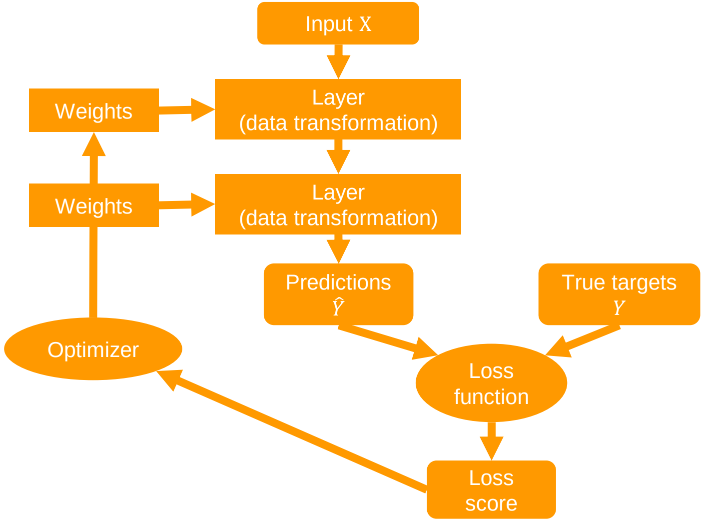
1. Multi-Input/Output Models
2. Complex Network Architectures
3. **Callbacks and TensorBoard**
4. Hyperparameter Optimization

Monitoring the Training Phase

- Up to now:
  1. Train a model for a long time.
  2. Evaluate metrics to figure out for how many epochs *n* the validation error goes down.
  3. Train the model again for *n* epochs.

- Better:
  1. Train the model only once and stop when the validation error stops dropping.

- Solution:
  - Pass a **callback**-object to `model.fit()` which gets called at certain points in time during training.
  - Interrupt training, store model, load weights, …



Callbacks

- Callbacks can be used to…
  - **Store** an **intermediate state** (weights) of a model during training.
  - **Stop the training** at predifined events (**early stopping**), e.g. validation error is not decreasing.
  - **Dynamically adapt parameters** of models during training (e.g. the learning rate of the optimizer).
  - **Store training and validation metrics,** during training, e.g. to visualize learning progress.
- Some build-in callbacks:

```
keras.callbacks.ModelCheckpoint
keras.callbacks.EarlyStopping
keras.callbacks.LearningRateScheduler
keras.callbacks.ReduceLROnPlateau
keras.callbacks.CSVLogger
```

Callbacks Examples

```
import keras
callbacks_list = [
        keras.callbacks.EarlyStopping(
                monitor='acc',
                patience=1),
        keras.callbacks.ModelCheckpoint(
                filepath='my_model.h5',
                monitor='val_loss',
                save_best_only=True),
        keras.callbacks.ReduceLROnPlateau(
                monitor='val_loss`,
                factor=0.1,
                patience=10)
]

model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])

model.fit(x, y, epochs=10, batch_size=32, callbacks=callbacks_list,
        validation_data=(x_val, y_val))
```

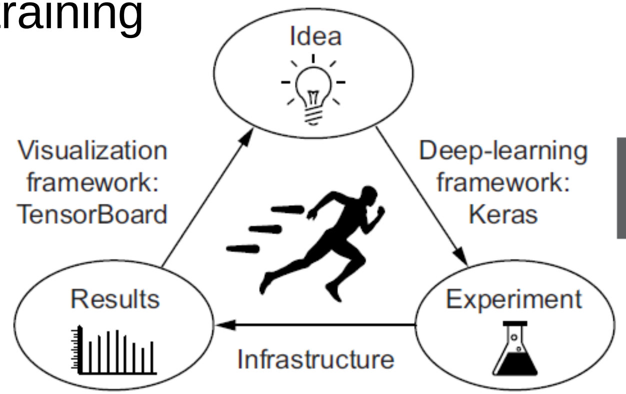
Stops training as soon as the validation accuracy has not dropped for two straight epochs

Stores the weights after each epoch

Learning rate is divided by 10 as soon as the validation loss has not improved for 10 straight epochs

TensorBoard

- Visualization of parts/aspects of a model during training
  - Monitoring of metrics
  - Model architecture
  - Distributions of activations and gradients
  - Embeddings in 3D



- **TensorBoard**…
  - Is a browser-based, graphical user environment
  - Runs by default at http://localhost:6006
  - Should have been automatically installed together with TensorFlow

Text Classification Model

```
import keras
from keras import layers
from keras.datasets import imdb
from keras.preprocessing import sequence
max_features = 2000
max_len = 500
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
x_train = sequence.pad_sequences(x_train, maxlen=max_len)
x_test = sequence.pad_sequences(x_test, maxlen=max_len)
model = keras.models.Sequential()
model.add(layers.Embedding(max_features, 128, input_length=max_len, name='embed'))
model.add(layers.Conv1D(32, 7, activation='relu'))
model.add(layers.MaxPooling1D(5))
model.add(layers.Conv1D(32, 7, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(1))
model.summary()
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
```

Callback für TensorBoard

- Create a log directory

```
$ mkdir my_log_dir
```

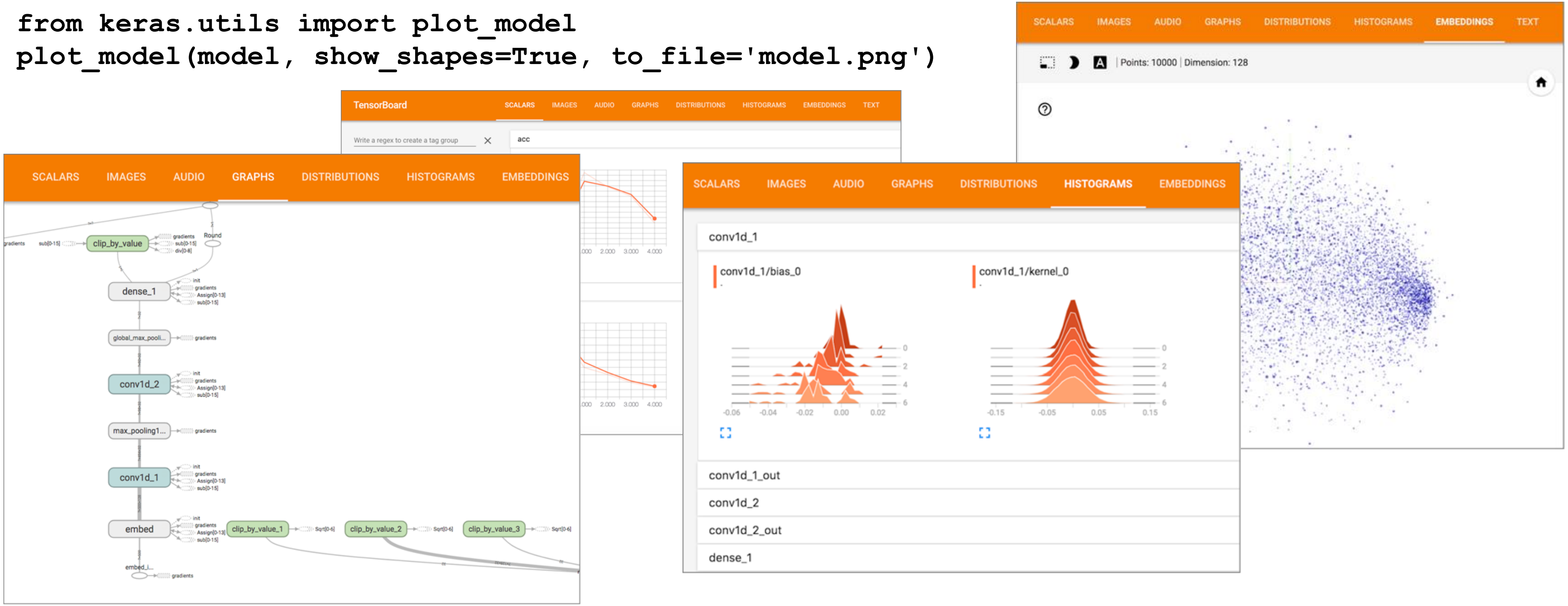
```
callbacks = [
      keras.callbacks.TensorBoard(
        log_dir='my_log_dir',
        histogram_freq=1,
        embeddings_freq=1,
      )
]
history = model.fit(x_train, y_train,
        epochs=20,
        batch_size=128,
        validation_split=0.2,
        callbacks=callbacks)
```

- Start TensorBoard

```
$ tensorboard --logdir=my_log_dir
```

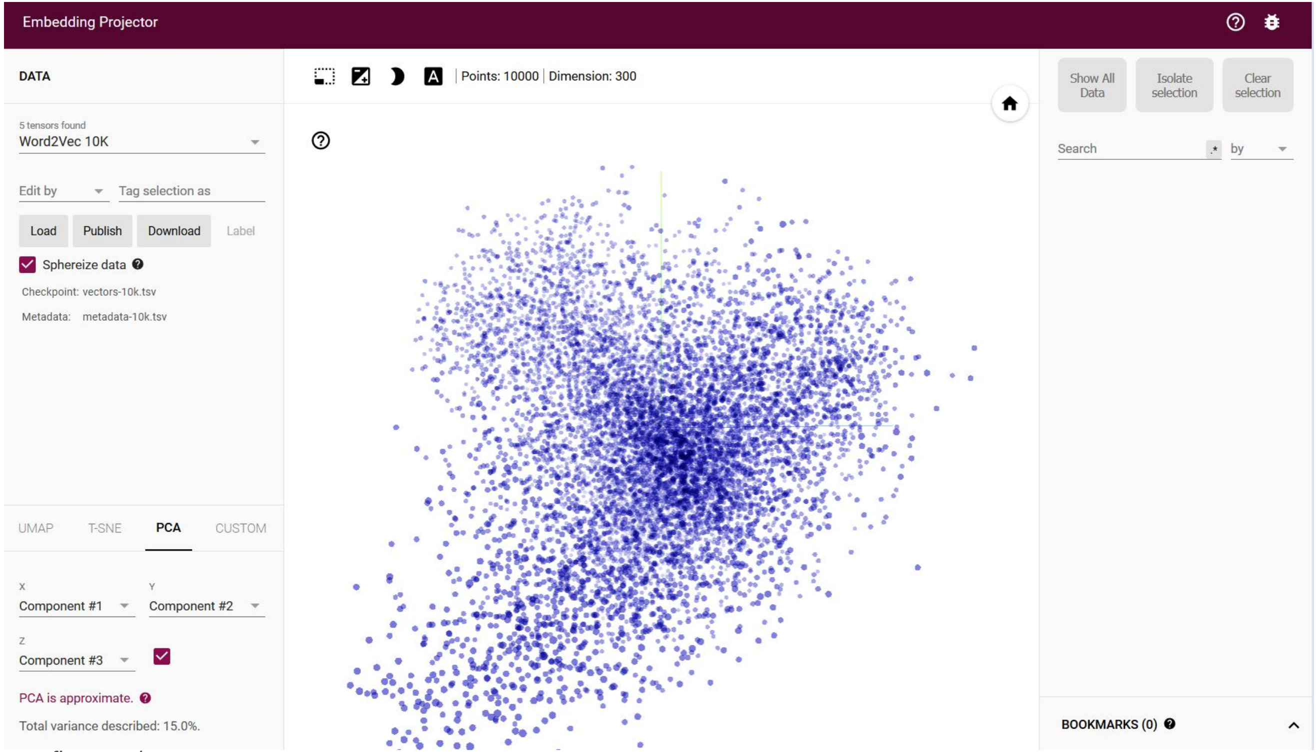
TensorBoard Example

```
from keras.utils import plot_model
plot_model(model, show_shapes=True, to_file='model.png')
```



https://colab.research.google.com/github/tensorflow/tensorboard/blob/master/docs/tensorboard_in_notebooks.ipynb

Tensorflow Projector



https://projector.tensorflow.org/

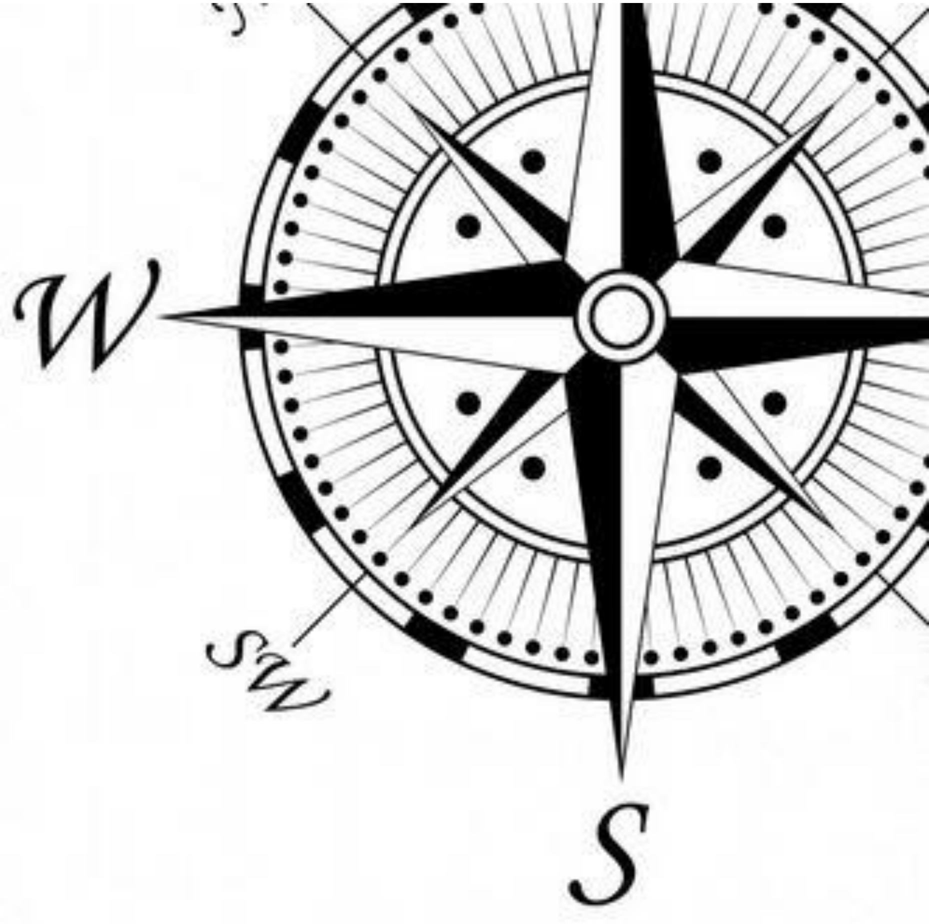
TensorBoard in Colab

- Run the example in https://colab.research.google.com/github/tensorflow/tensorboard/blob/master/docs/get_started.ipynb

Start 5 4 3 2 1 End

Topics Today

1. Multi-Input/Output Models
2. Complex Network Architectures
3. Callbacks and TensorBoard
4. **Hyperparameter Optimization**



Optimization of Hyperparameters

- Parameters of a model
  - **Weights** (will be learnt)
  - **Architecture-related** (are set)
    - How many layers?
    - How many units per layer?
    - Which activation function?
    - Normalization?
    - Dropout?
    - …?

Hyperparameter

- Goal: optimize hyperparameters automatically
  - E.g. Bayesian optimization, genetic algorithms, random search, …

Optimization Process

1. (Automatic) selection of hyperparameters
2. Assemling of the respective model
3. Training of the model on training data
4. Validation of the model on validation data
5. (Automatic) update of hyperparameters
6. Continue with 2.
7. If satisfied: Test model on test data

- One iteration is extremely expensive

- Problem not solvable with gradient descent
  - The hyperparameter space consists of discrete decisions
    - Not continuous, **non-differentiable**

Optimization

- Optimization tools
  - Hyperopt (https://github.com/hyperopt/hyperopt)
  - Hyperas (https://github.com/maxpumperla/hyperas) integrates Hyperopt in Keras

- Meta-machine-learning
  - Training the optimal hyperparameter values is done on validation data
  - Overfitting!

- General approaches are needed!
  - Recent research field
  - Cf. Manual feature engineering vs. deep learning
  - Best practice for now: manual hyperparameter tuning/optimization

Learning Goals for this Chapter

- Implement non-sequencial models in Keras
  - Multiple in- and outputs
  - Complex architectures
- Monitoring of training process in Keras
  - Callbacks
- Visualization of network parameters
  - TensorBoard
- Optimization of hyperparameters

- Relevant chapters:
  - P7
  - S7 (2017) https://www.youtube.com/watch?v=PicxU81owCs