Deep Learning
Lecture 13

Recurrent Networks

Prof. Dr. Rainer Herrler
Phone: 09721/ 940-8710
Email: rainer.herrler@fhws.de

Pictures from Wikipedia / Pixabay
Some Pictures generated with Stable Diffusion

5+2

Recurrent Neural Networks

- RNNs are dealing with Sequences of data
- Sequences are collections of elements, where
  - Elements can be repeated
  - Order matters
  - Sequences can be of variable (up to infinite) length
- Examples
  - Audio signals of variable length for captioning / transscription
  - Stock prediction
  - Text e.g for translation

How did we process text in ML SS2022?

Situation with the NN's so far….

- Input had fixed length.
- Order of features was not relevant (almost)
- Output had fixed length

- For some Tasks sequence is essential

Example, where "bag of words" is failing:

"I saw her duck" "Duck! I saw her."

Overview on recursive networks

- RNN - Basic Idea of recurrent networks
- LSTM - Advanced version addressing the problems of RNN
- GRU - Alternative design to LSTM

Examples variable Inputs and outputs

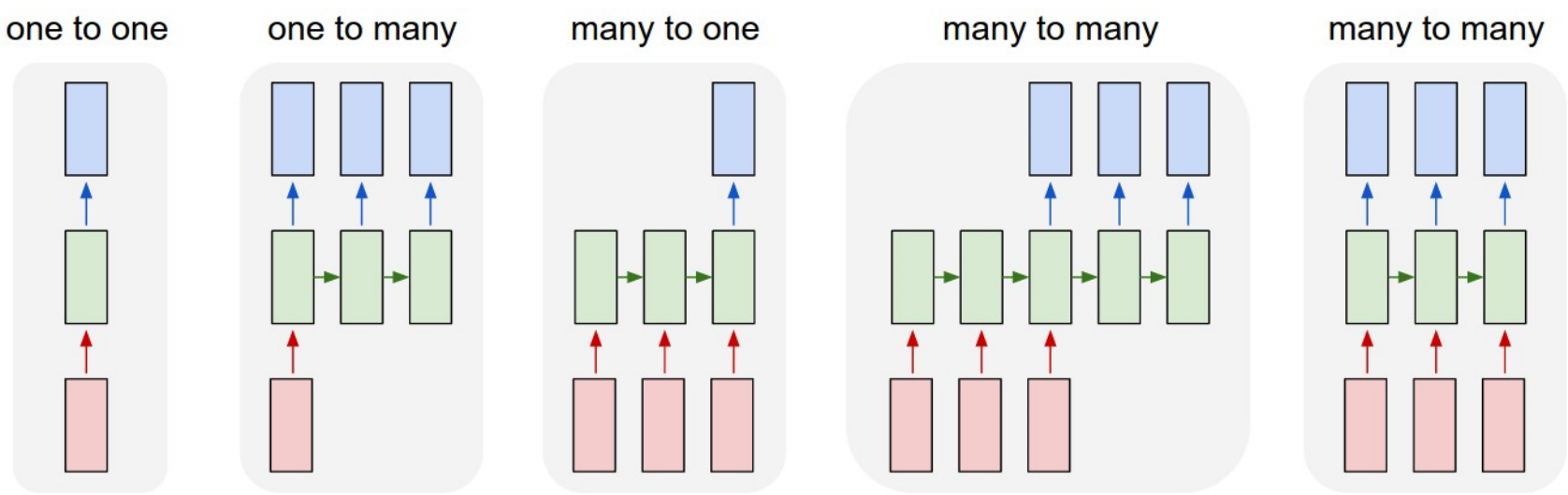


Image Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Text Classification

Image captioning

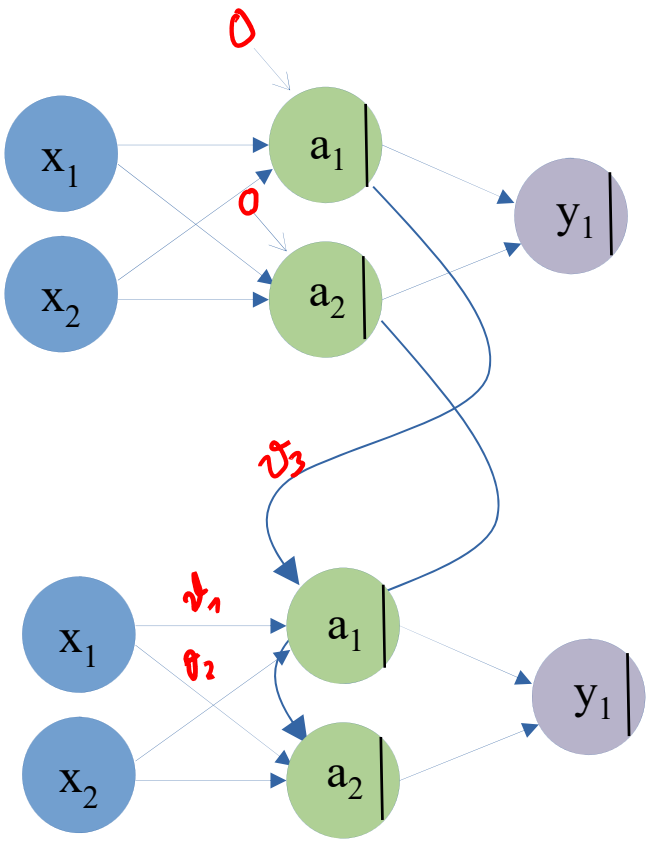
Translation

Stock price prediction

Basic recurrent inference

Time t = 1

Time t = 2

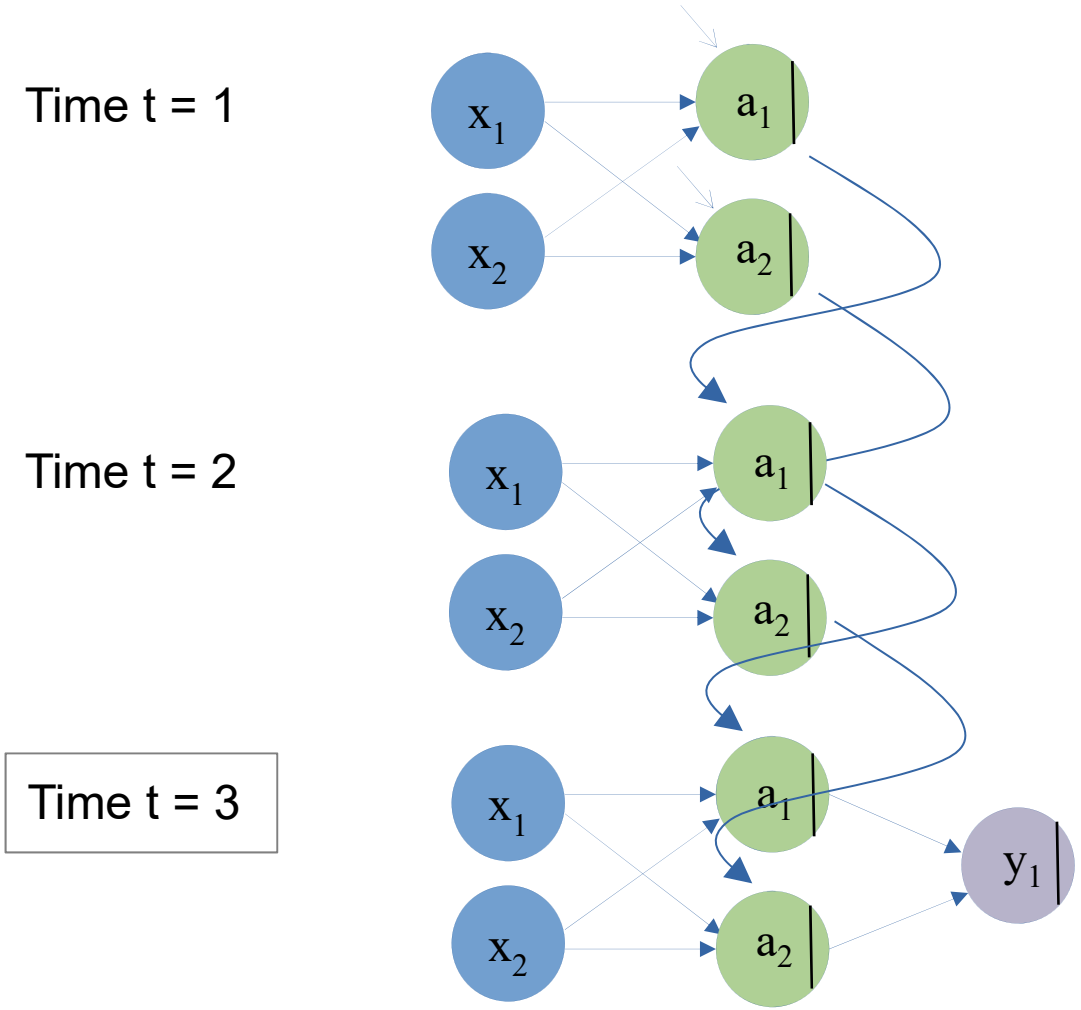


A recurrent layer has Some additional inputs where it gets what he returned as an output the timestep before.

Weights are also given for these recurrent connections!

Initial Value is 0.

Basic recurrent backpropagation



Backpropagation goes Also back in time. This Is similar to very deep networks and leads to the **vanishing gradients problem**

Problems of recurrent neural networks

In theory RNN are powerfully enough to take a long context of samples into account, but in practice its very hard to train these models.

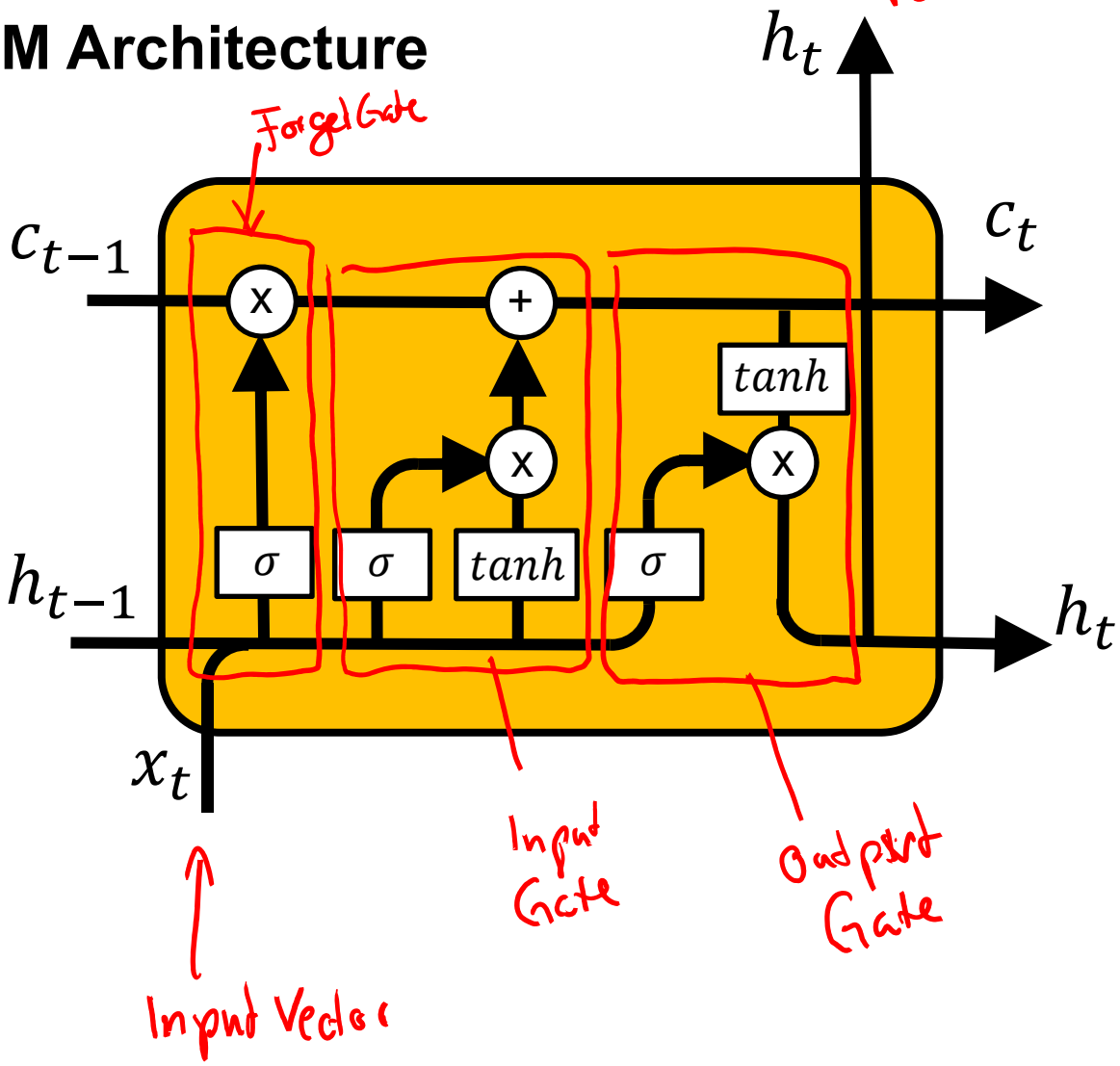
- **Problem 1:** Vanishing gradients, weights cannot be efficiently updated over the long sequence
- **Problem 2:** In a consequence they cannot store a long context efficiently
- **Problem 3:** Slow training process. The training has to be performed sequentially and cannot parallelization is not possible.

The first two problems are addressed by LSTMs, the next development stage of RNN’s

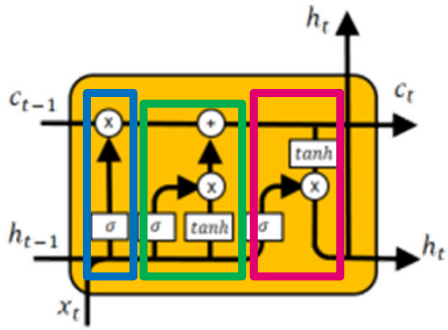
LSTM Long term short term memory

- Inventors Sepp Hochreiter and Jürgen Schmidhuber in 1997
- In german this would translate to “Langes Kurzzeitgedächtnis”
- The network was specifically designed to be able to remember important things longer.
- This allowed to realize deeper recurrent neural networks and allowed applications like language understanding to improve a lot.

The LSTM Architecture



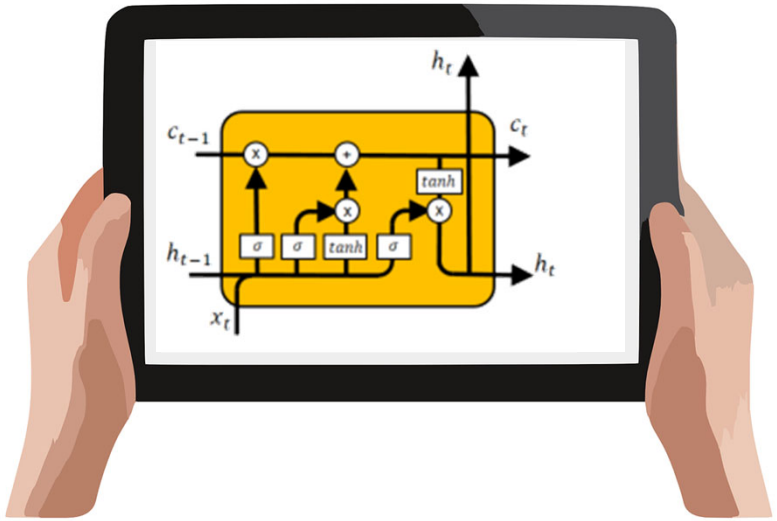
Role of the different components



- States of the LSTM
  - Cell State (upper rail)
    - The cell state is the long term memory of the LSTM Cell.
    - It has influence on current and future output
  - Hidden State (lower rail)
    - The hidden state is the short term memory of the cell.
    - It triggers forgetting, memorizing and output
- Gates of the LSTM
  - Forget Gate (Blue)
    - The forget gate controls if the long term memory should be erased
  - Input Gate (Green)
    - The input gate controls if and how the long term memory should be updated
  - Output Gate (Rot)
    - The output gate controls the current output and next short term memory
- Application in Keras
  - Available as a Layer type
  - Most important parameter: Number of units

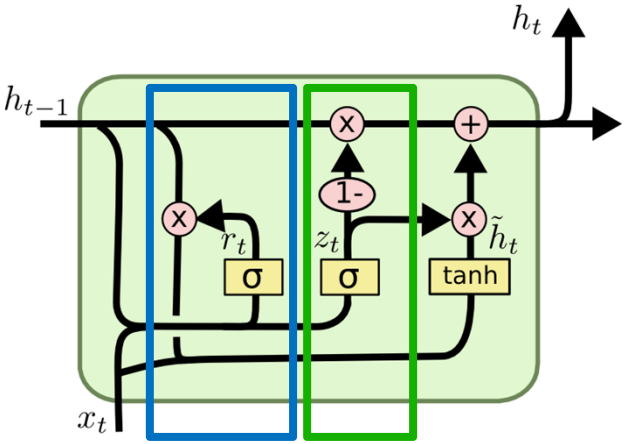
LSTM Example

DL_034_LSTM



GRU

Gated recurrent unit



http://dprogrammer.org/rnn-lstm-gru

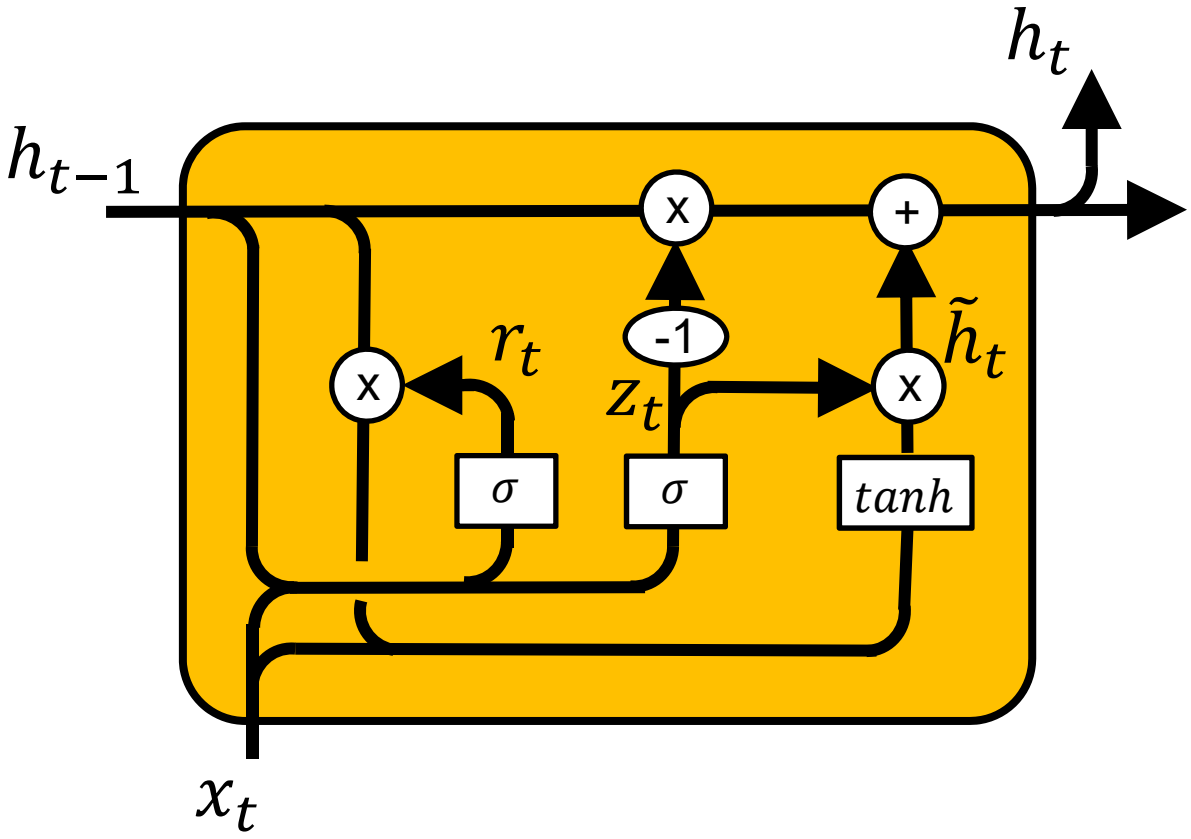
- Comparison to LSTM
  - Similar Gating Mechanisms but fewer parameter
  - No Output Gate
  - Less popular than LSTM but perfom similar on some tasks
- Gates of the GRU
  - Reset Gate (Blue)

    Determines which parts of the long term memory should be kept
  - Update Gate(Green)
    - The input gate controls if and how the long term memory should be updated
- In Keras available as a layer type:

  tf.keras.layer.GRU

GRU

Gated recurrent unit



Summary on recurrent layers

- RNN
  - In theory they are able to reason over past inputs but they are difficult to train
  - Don’t play a big role in practice but are a good introduction to recurrent layers
- LSTM
  - Advanced version addressing the problems of RNN. The architecture uses gates to explicitly model an efficient memorizing process
  - Still they need a lot of time for training and training has to be done sequentially needing a lot of CPU time.
  - Still it is hard for LSTM models to work with a very long context
- GRU
  - Alternative design also using a gating technique
  - Performs similar to LSTMs

Recap Slides

That was the initial plan for this Semester:

- We want to get familiar in building neural networks with state of the art frameworks

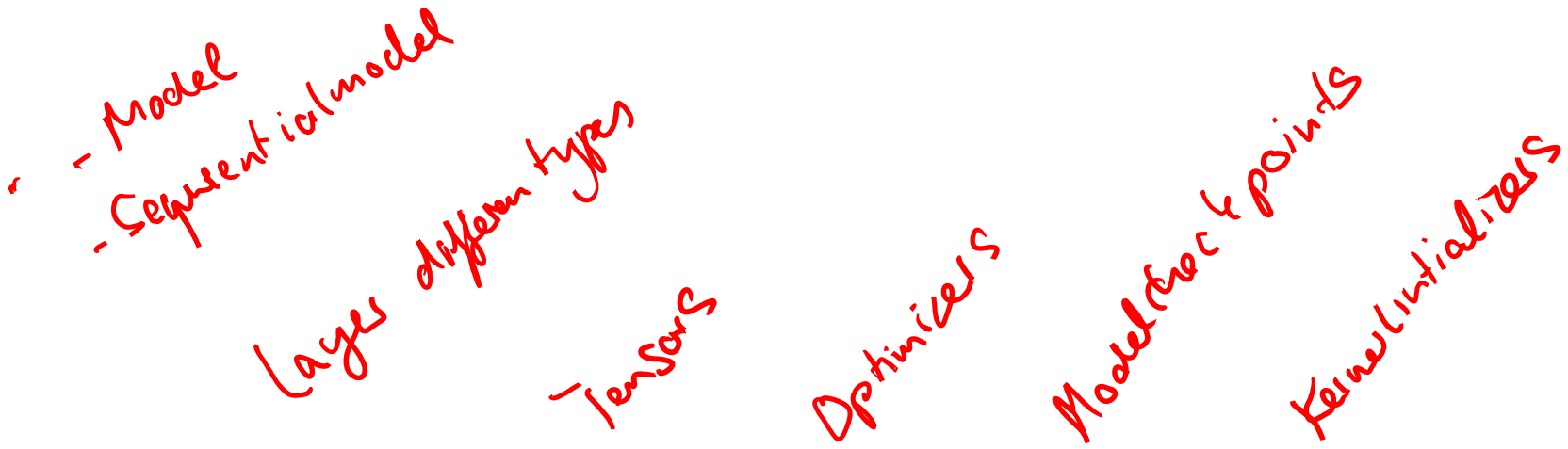
- We learn how to create different network structures and when they are useful:
  - Recurrent Networks
  - Autoencoder Networks
  - Convolutional Neuronal networks
  - …
- We learn how to find and use existing trained models
- We learn how to fine tune existing models for own tasks
- We learn how to train models with GPU's or TPU's in the cloud.
- …

State of the art frameworks!

Framework Questions:

- How are Tensorflow and Keras related ?

- What important classes do we need to create and train networks ?



Cloud Training. What were your experiences?

- Just Colab

In which context did we mention the following AI researchers

- Yann Le Cun ConvNets
- Geoffrey Hinton Backpropagation
- François Chollet Keras Framework
- Fei Fei Li Imagenet
- Sepp Hochreiter LSTM

Loss and Score

- What is the difference ?

- What are they needed for ?

Gradient Descent operates on the Loss!
Score is for validation

- Mention Examples:

Loss: Log Loss, SquareLoss/MSE, HingeLoss

Score: Accuracy, F1 Score, Recall, Precision (Classification) | R2, MSE (Regression)

- Describe the input for each:

Score & Loss → predicted values | ground truth

Input for Loss: Batch

Input for the Score: Training Data, Test Data

Our Tool Environment

- What is anaconda?
  - Software Package Manager
  - Environment Management
  - Dependency Management
  - Python related
- Two ways to install python packages
  - pip install
  - conda install
- What is juypter notebook
  - IDE

To which Objects could this code belong

Vector or Matrix

```
def forward(self, input):
    return 1/(1 + np.exp(-input))
```

Sigmoid Layer

```
def forward(self,input):
    return np.dot(input, self.weights)+self.biases
```

Dense Layer
Fully connected Layer

```
def loss(self, y, pred):
    assert pred.shape == y.shape
    vec = -np.log(pred)*y-(np.log(1-pred)*(1-y))
    return np.mean(vec);
```

Log Loss Object

Model creation

- What different ways to create Keras models did we use?

Sequential Interface adding Layer by Layer with the add Function

Sequential Interface with Constructor

Model with the first Layer | Last Lay. Layers have to be connected Manually

- What is done during model compilation?

Weights initialization

Computational graph is created

→ Therefore we need to specify
- Optimizer
- Loss
- Kernel Initialization

Model training

- What is done in a training loop ?

Loop over all epochs

Loop over all batches → Forward & Backward prop

- How is it usually initiated in Keras?

fit

- What are important parameters for the loop?

\

Model evaluation

- How did we evaluate models in Scikit learn?

  score

- How to evaluate models in Keras

  evaluate

```
Model.compile(
    optimizer="rmsprop",
    loss=None,
    metrics=None,
    loss_weights=None,
    weighted_metrics=None,
    run_eagerly=None,
    steps_per_execution=None,
    jit_compile=None,
    pss_evaluation_shards=0,
    **kwargs
)
```

```
Model.fit(
    x=None,
    y=None,
    batch_size=None,
    epochs=1,
    verbose="auto",
    callbacks=None,
    validation_split=0.0,
    validation_data=None,
    shuffle=True,
    class_weight=None,
    sample_weight=None,
    initial_epoch=0,
    steps_per_epoch=None,
    validation_steps=None,
    validation_batch_size=None,
    validation_freq=1,
    max_queue_size=10,
    workers=1,
    use_multiprocessing=False,
)
```

```
Model.evaluate(
    x=None,
    y=None,
    batch_size=None,
    verbose="auto",
    sample_weight=None,
    steps=None,
    callbacks=None,
    max_queue_size=10,
    workers=1,
    use_multiprocessing=False,
    return_dict=False,
    **kwargs
)
```

python Copy code

```
loss, accuracy = model.evaluate(test_data, test_labels)
```

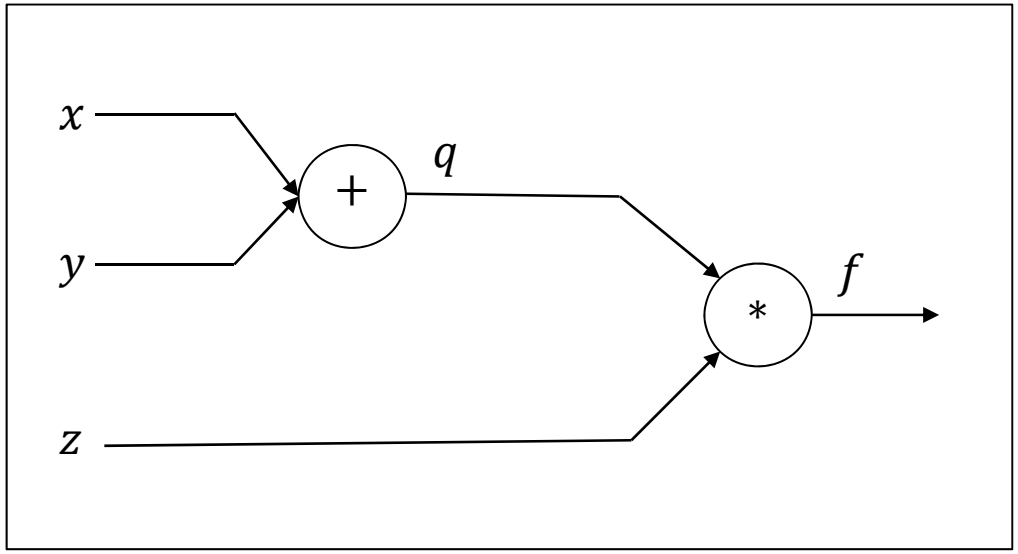
Model inference

- How can we do inference in Keras ?

predict

Backpropagation

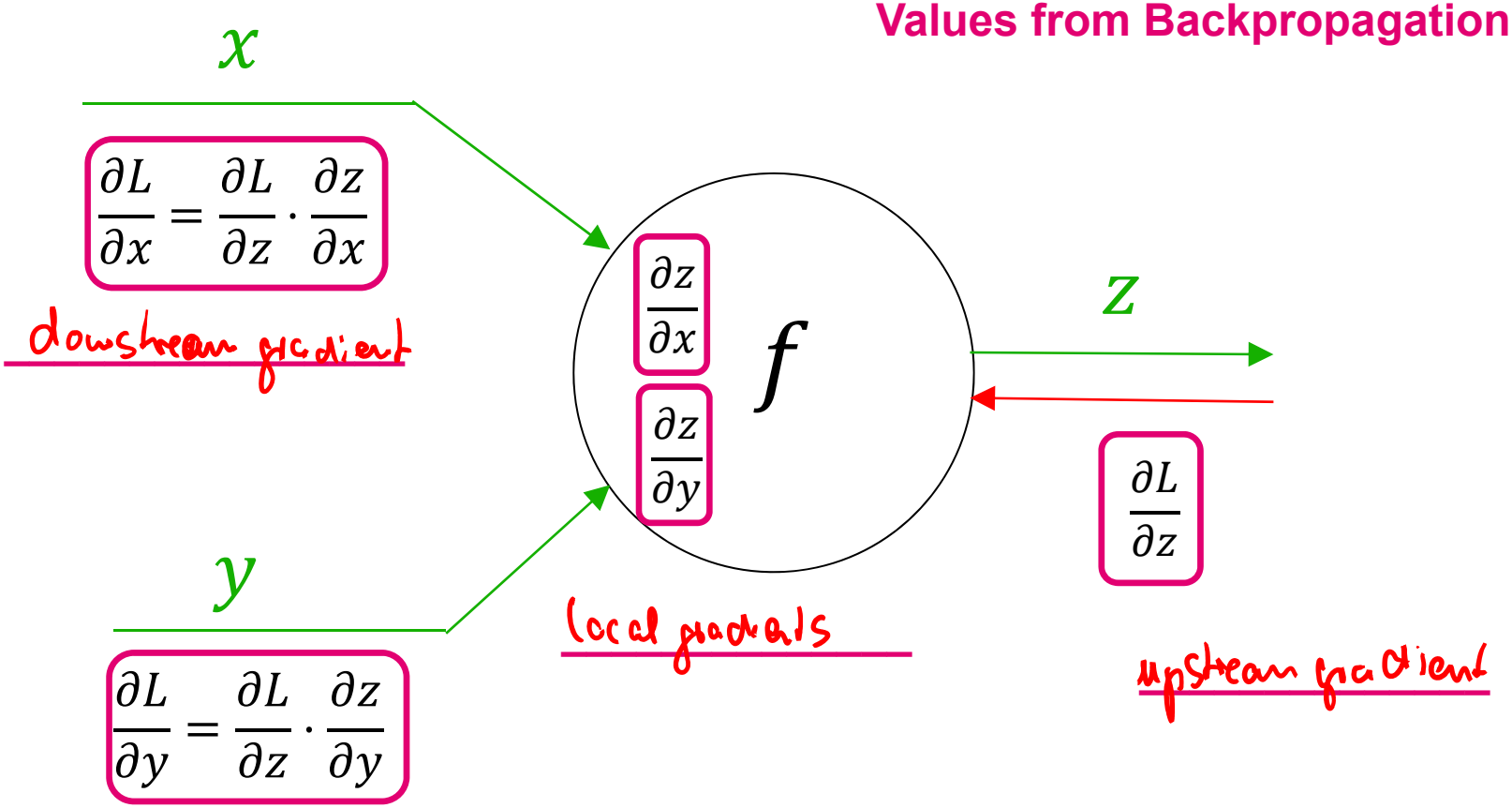
Computational Graph



How do we call this graphical structure formula ?

Next time prepare Max

Values from forward propagation
Values from Backpropagation



How to choose?

Trial and Error

- The batchsize ?

  ~~Powers of two~~ are very common

  Unstable Learning process → increasing Batchsize

  Out of memory, too long iterations → decrease the Batchsize

- The Number of Epochs ?

- The last layer activation function ?

How can we initialize weights before training ?

- Theoretical Methods
  - random gaussian (small ones)
  - Xavier → sigmoid / tan
  - He → RELU
- How to do it in Keras ?

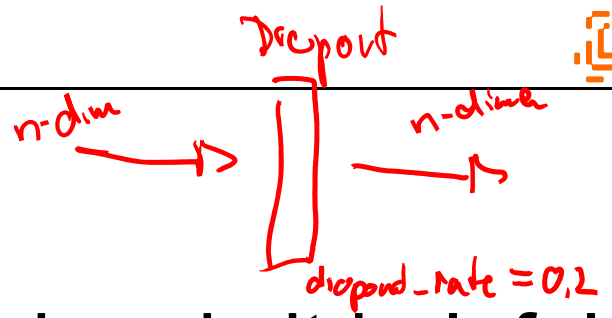
```
tf.keras.layers.Dense(
    units,
    activation=None,
    use_bias=True,
    kernel_initializer="glorot_uniform",
    bias_initializer="zeros",
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs
)
```

identical to xavier

https://towardsdatascience.com/weight-initialization-for-neural-networks-does-it-matter-e2fd99b3e91f

thws

Dropout



- What is dropout and when is it helpful ?

Regulation method prevents overfitting
Removes values from the "input"
Realized as a Layer Type

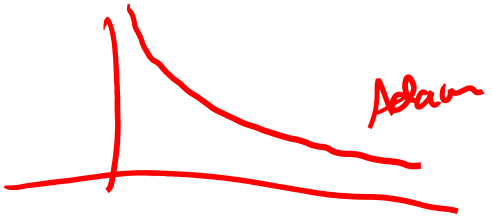
- What different operation modes are there?

Mode 1: Training → switched on to create an incentive to make a more robust model
Mode 2: Prediction → switched off to make the best possible predictions

– Not useful if there is no overfitting
– Can lead to a situation where your test accuracy is better than training accuracy

Optimizer

What can be your benefits if you use a different optimizer than SGD ?



Optimizer names: Adam, Rmsprop, Momentum

Benefits:
- Faster convergence
- (More stable optimization process)
- Easier to optimize (finding a good learning rate)

How does the optimizer achieve that effect ?

- We take "momentum" information from the last batch to subsequent batches

→ dampening Oszillations

even small Batchsizes are not oszillating that much

When is one hot encoding necessary ?

Multiclass classification.

Why?

Convnets – Guess the output shape

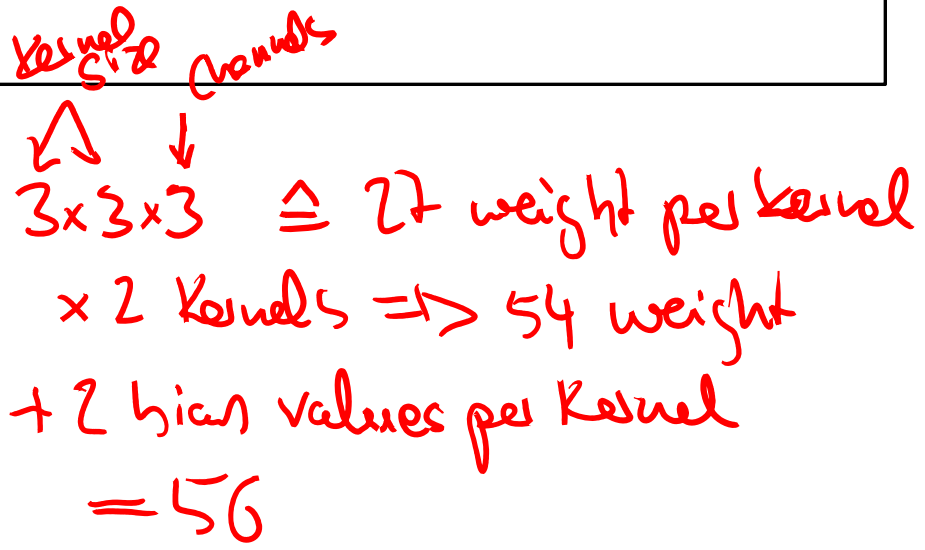
```
input_shape = (4, 28, 28, 3)
x = tf.random.normal(input_shape)
y = tf.keras.layers.Conv2D(
     2, 3,
    activation='relu',
    strides=2,
    input_shape=input_shape[1:],
    padding="same"
)(x)
print(y.shape)
```

Convnets – What is the output

```
x= tf.constant([[1., 2., 3.],
                [4., 5., 6.],
                [7., 8., 9.]])
x = tf.reshape(x, [1, 3, 3, 1])
max_pool_2d = tf.keras.layers.MaxPooling2D(pool_size=(2, 2),
    strides=(1, 1), padding='valid')
max_pool_2d(x)
```

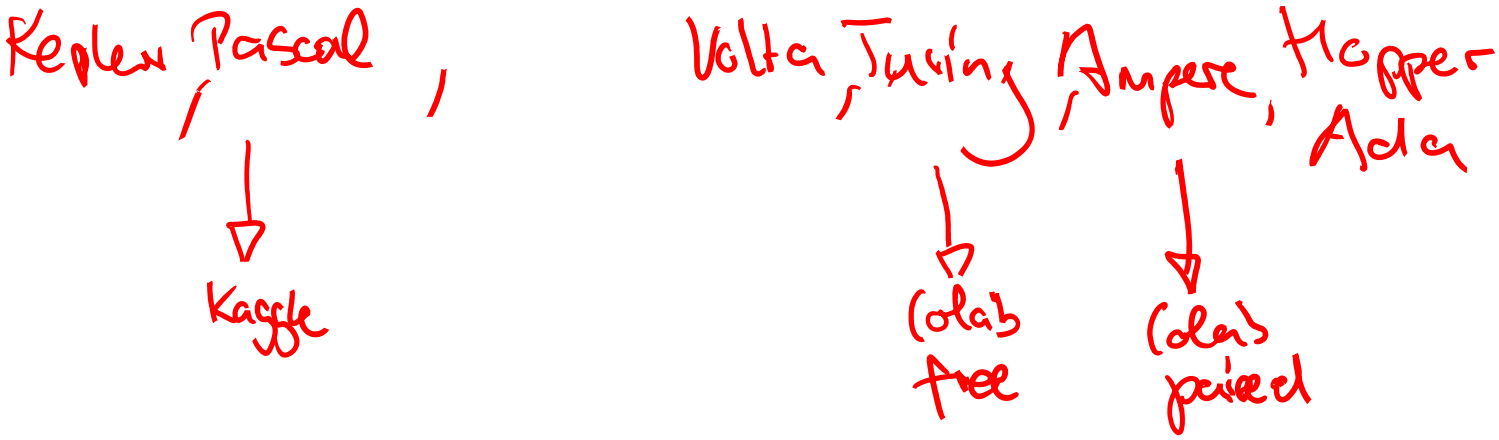
```
net = tf.keras.Sequential()
net.add(tf.keras.layers.Conv2D(
     2, 3,
    activation='relu',
    use_bias=True,
    strides=2,
    input_shape=(28, 28, 3),
    padding="same"
))
net.summary()
```

How many trainable parameter has the model



What are those names? How to sort?

- Ada, Kepler, Hopper,Pascal, Volta, Turing, Ampere, ~~Ada~~



Batch Normalization

- What is Batch normalization

  Normalizing Data inside the network based on the current batch.

- What is the effect of batch normalization

  - faster convergence
  - regularization

- When not to use batch normalization ?

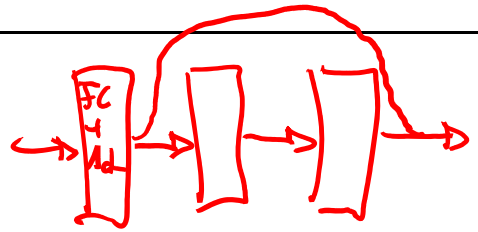
  do not apply with small batchsizes

Again => Normalization parameters are just varying during training

At inference time they are fixed based on the data seen in the last training iteration

thws

Resnet



- What was introduced in the Resnet Architecture ?

  Skip Connections

- How did the Trick introduced in resnets influence network architectures ?

What is done by the following code ?

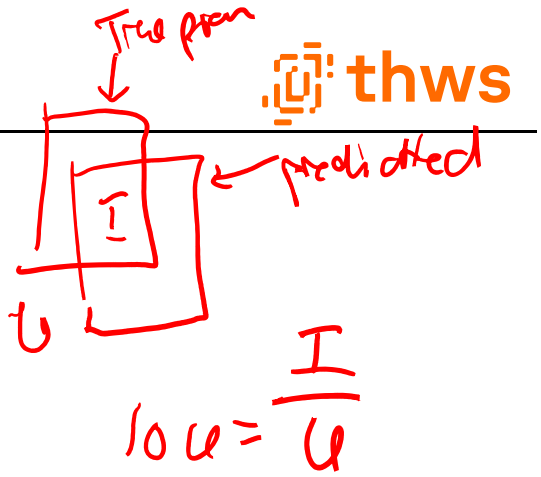
```
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.applications.inception_v3 import preprocess_input, decode_predictions
from tensorflow.keras.applications.inception_v3 import InceptionV3
```

```
image = load_img("oldtimer.jpg", target_size=(299, 299))
image = img_to_array(image)
image = image.reshape((1, 299, 299, 3))
image = preprocess_input(image)
```

```
model = InceptionV3()
pred = model.predict(image)
```

```
decode_predictions(pred)
```

```
model.summary()
```



Metrics for object detection

- What does IoU stand for ?
- What is the range of IoU ?
- What other type of losses are typically evaluated?
  - Bounding Box regression error
  - Classification error

Layer types

- What is the opposite Layer to Flatten

  Reshape-layer

- What is the opposite layer to Conv2D

  Conv2DTranspose

- What is the opposite layer to MaxPool2D

  UpSampling2D

How to prepare well

- Read the jupyter notebooks and try to understand what they are doing
- Make own experiments with the code. Try to reformulate things and inspect data between the steps.
- Look at slides and recap slides, if things are unclear search for help in the linked articles.