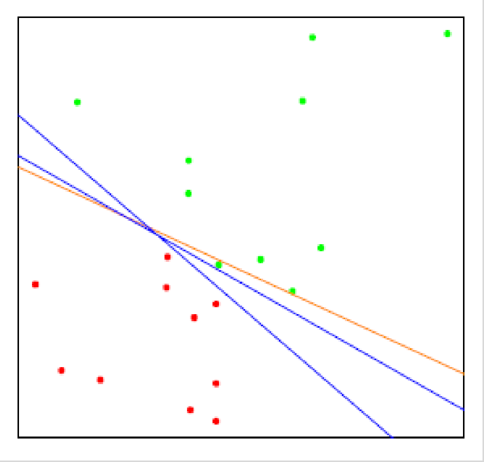


Deep Learning

Using a Convolutional Neural Network

Dr. – Ing. Morris Riedel
Adjunct Associated Professor
School of Engineering and Natural Sciences, University of Iceland
Research Group Leader, Juelich Supercomputing Centre, Germany

LECTURE 6

Other Deep Learning Models & Summary

December 1st, 2017
Ghent, Belgium

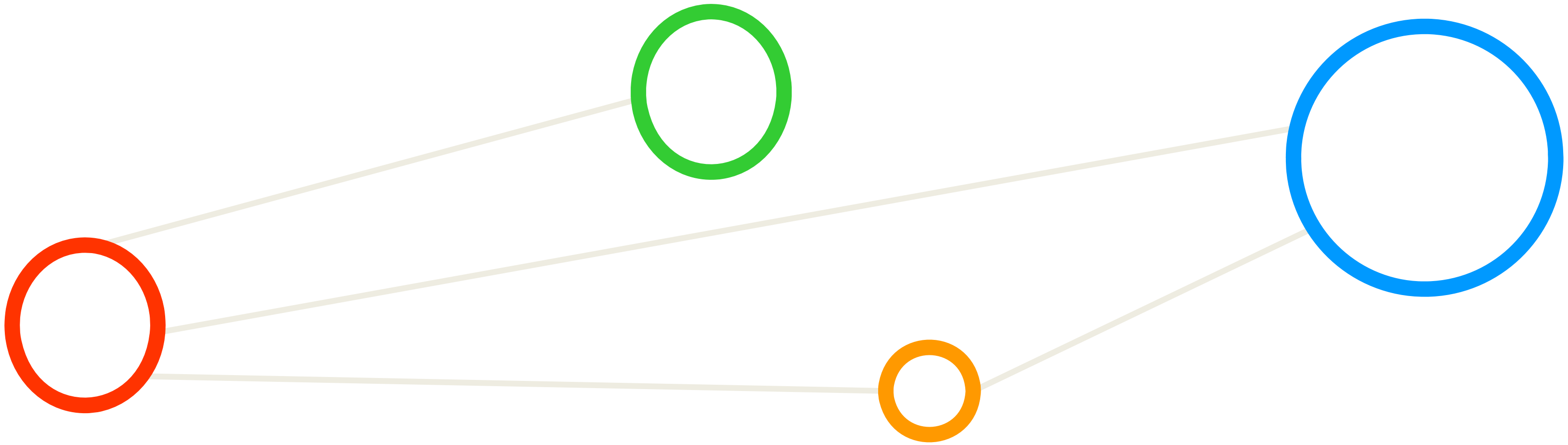
UNIVERSITY OF ICELAND
SCHOOL OF ENGINEERING AND NATURAL SCIENCES
FACULTY OF INDUSTRIAL ENGINEERING,
MECHANICAL ENGINEERING AND COMPUTER SCIENCE

JÜLICH
FORSCHUNGSZENTRUM

Outline of the Course

1. Deep Learning Fundamentals & GPGPUs
2. Convolutional Neural Networks & Tools
3. Convolutional Neural Network Applications
4. Convolutional Neural Network Challenges
5. Transfer Learning Technique
6. Other Deep Learning Models & Summary

Outline

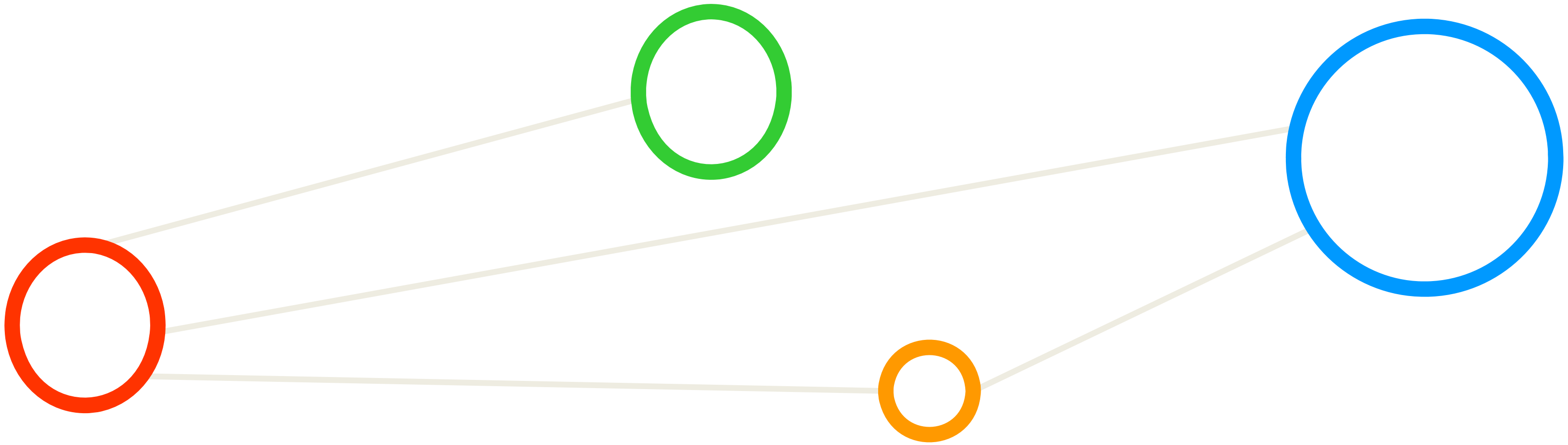


Outline

- Long-Short Term Memory
  - Limitations of Feed Forward Networks
  - Recurrent Neural Network (RNN)
  - LSTM Model & Memory Cells
  - Keras and Tensorflow Tools
  - Application Examples

- Summary
  - Training using Parallel Computing & GPUs
  - Increasing Complexity in Applications
  - Complexity of Parameters needs HPC
  - Group Assignment Discussion
  - Deep Learning & Applications

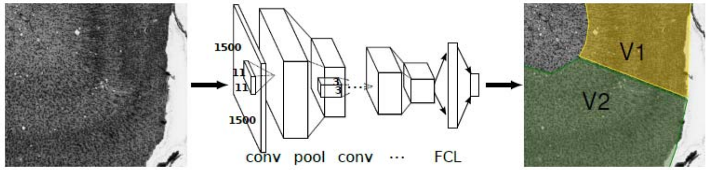
Long-Short Term Memory



Exercises – Group Assignment – Check Status

Deep Learning Architectures

- Deep Neural Network (DNN)
  - ‘Shallow ANN‘ approach with many hidden layers between input/output
- Convolutional Neural Network (CNN, sometimes ConvNet)
  - Connectivity pattern between neurons is like animal visual cortex



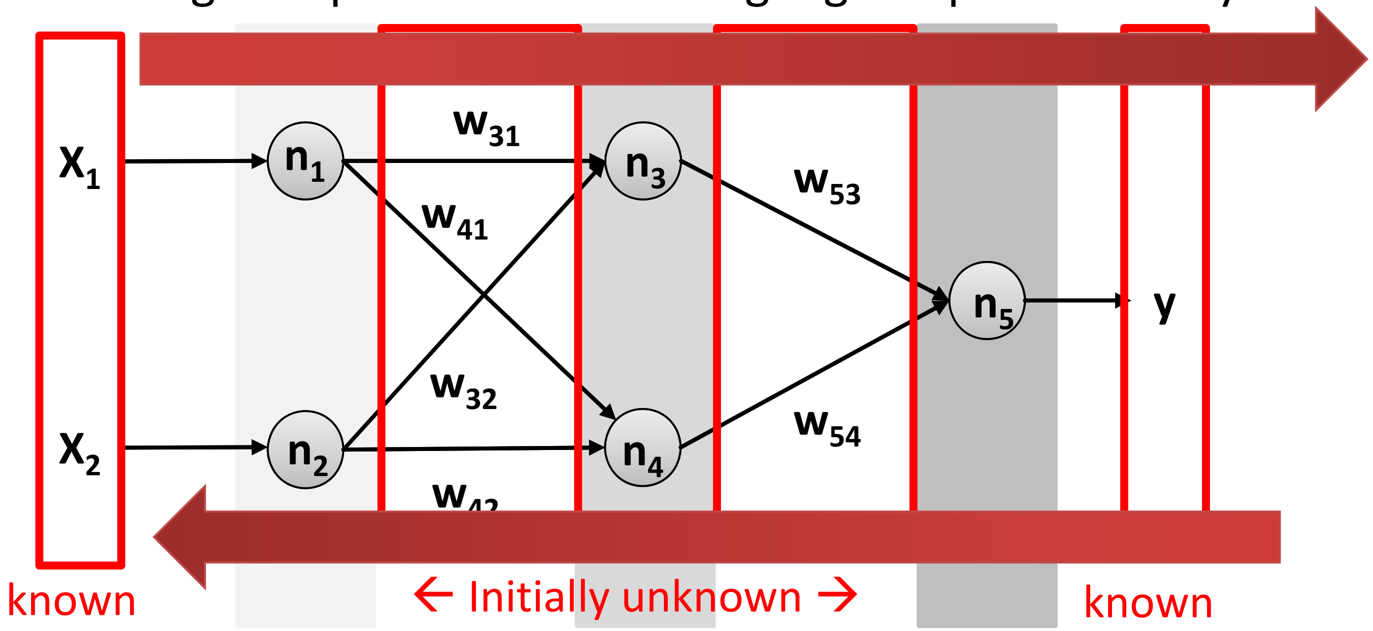
- Deep Belief Network (DBN)
  - Composed of mult iple layers of variables; only connections between layers
- Recurrent Neural Network (RNN) (just short intro in this course)
  - ‘ANN‘ but connections form a directed cycle; state and temporal behaviour

- **Deep Learning architectures can be classified into Deep Neural Networks, Convolutional Neural Networks, Deep Belief Networks, and Recurrent Neural Networks all with unique characteristica**
- **Deep Learning needs ‘big data‘ to work well & for high accuracy – works not well on sparse data**

Exercises – How to Encode a Sequence in ANN?

Limitations of Feed Forward Artificial Neural Networks

- Selected application examples
  - Predicting next word in a sentence requires 'history' of previous words
  - Translating european in chinese language requires 'history' of context

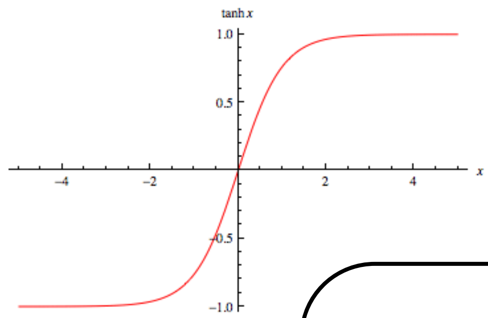


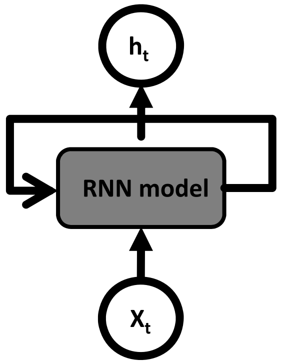
- **Traditional feed forward artificial neural networks show limits when a certain 'history' is required**
- **Each Backpropagation forward/backward pass starts a new pass independently from pass before**
- **The 'history' in the data is often a specific type of 'sequence' that required another approach**

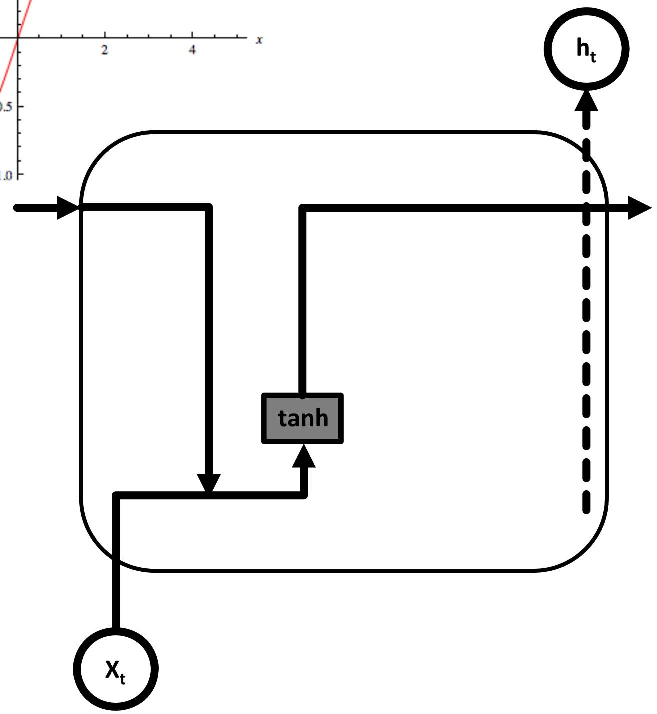
Recurrent Neural Network (RNN)

- **A Recurrent Neural Network (RNN) consists of cyclic connections that enable the neural network to better model sequence data compared to a traditional feed forward artificial neural network (ANN)**
- **RNNs consists of ‘loops‘ (i.e. cyclic connections) that allow for information to persist while training**
- **The repeating RNN model structure is very simple whereby each has only a single layer (e.g. tanh)**

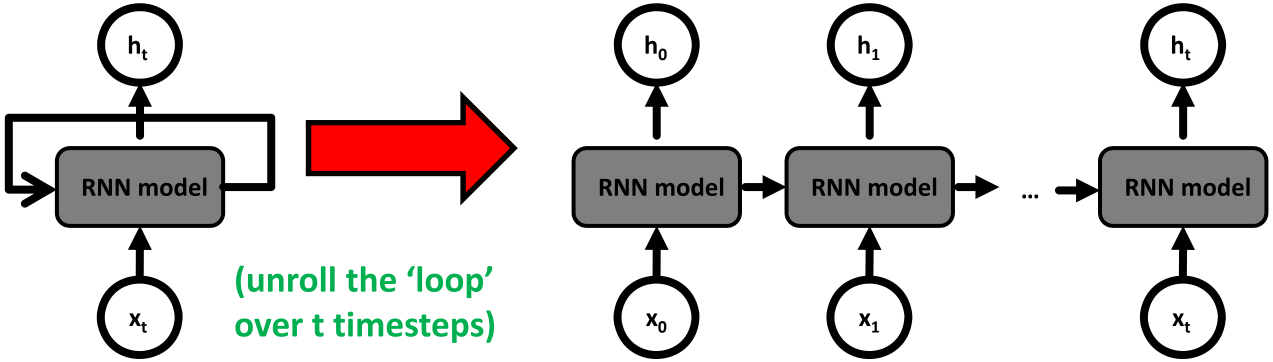
- Selected applications
  - Sequence labeling
  - Sequence prediction tasks
  - E.g. handwriting recognition
  - E.g. language modeling
- Loops / cyclic connections
  - Enable to pass information from one step to the next iteration
  - Remember ‘short-term‘ data dependencies







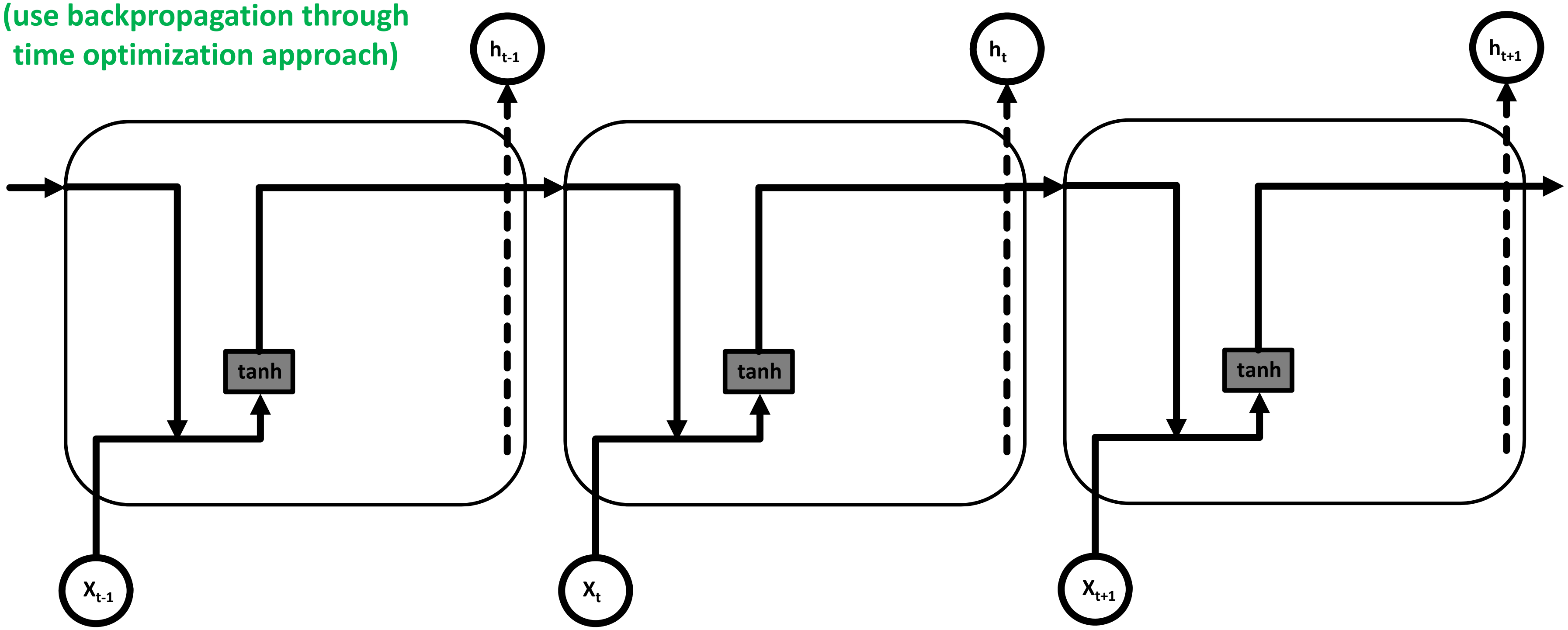
Unrolled RNN



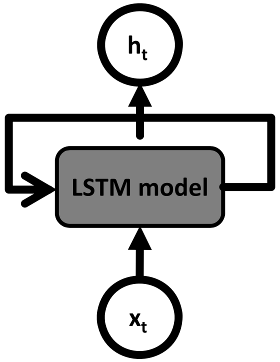
(unroll the 'loop' over t timesteps)

- A RNN can be viewed as multiple copies of the same network, each passing a message to a successor – this gets clear when 'unrolling the RNN loop'

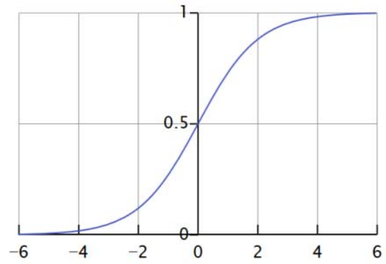
(use backpropagation through time optimization approach)



Long Short Term Memory (LSTM) Model

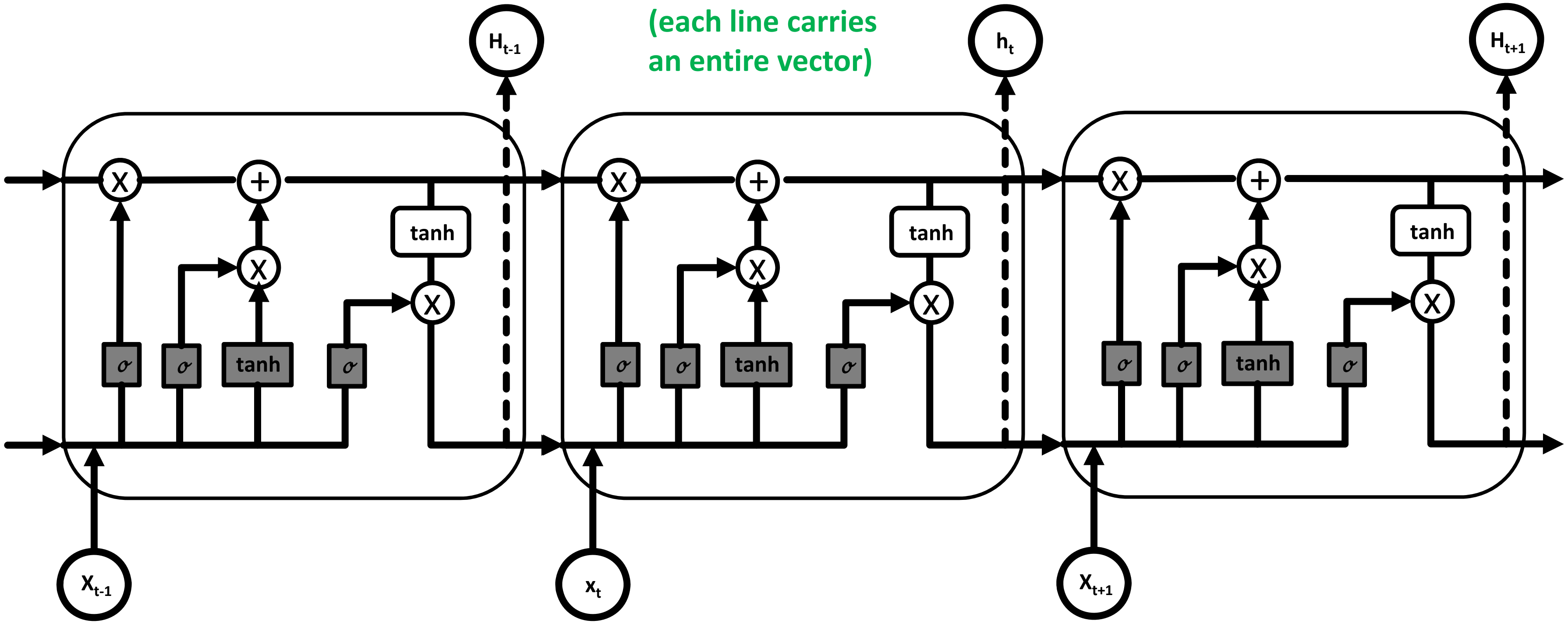


- Long Short Term Memory (LSTM) networks are a special kind of Recurrent Neural Networks (RNNs)
- LSTMs learn long-term dependencies in data by remembering information for long periods of time
- The LSTM chain structure consists of four neural network layers interacting in a specific way



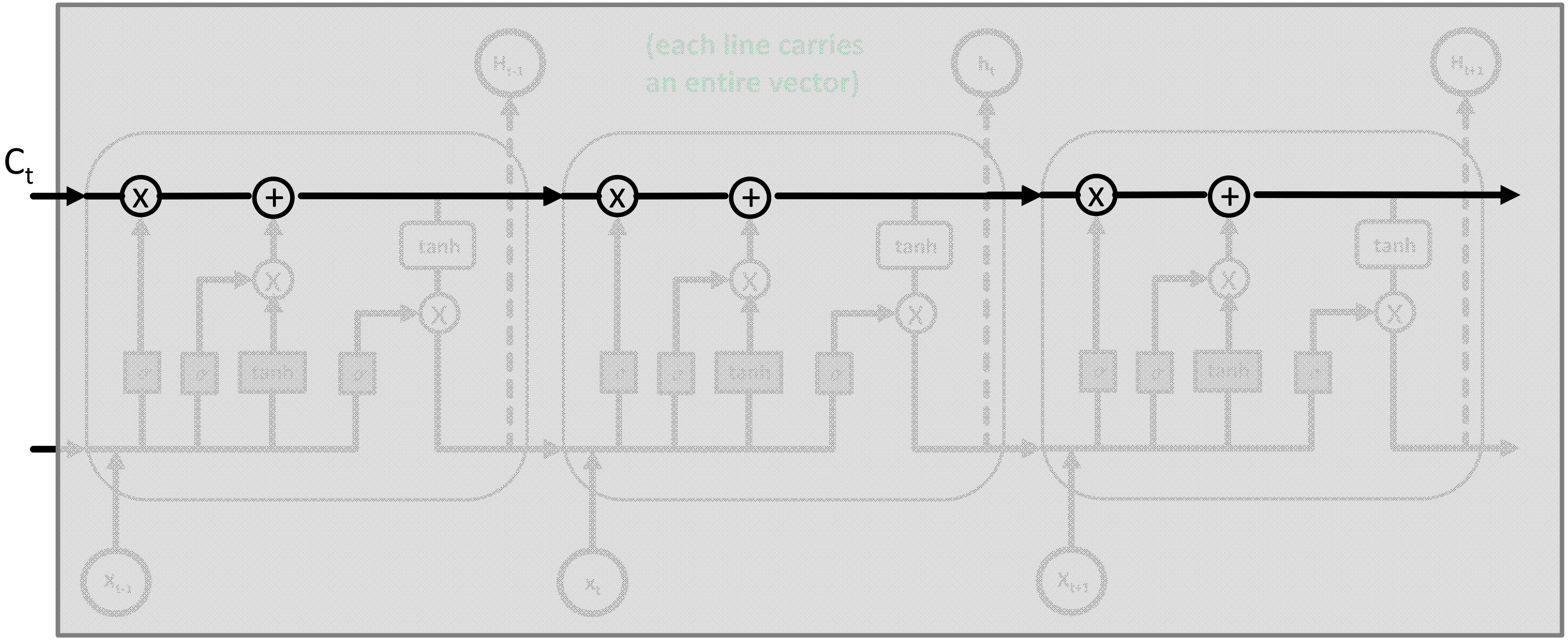
(uses sigmoid σ)

(each line carries an entire vector)

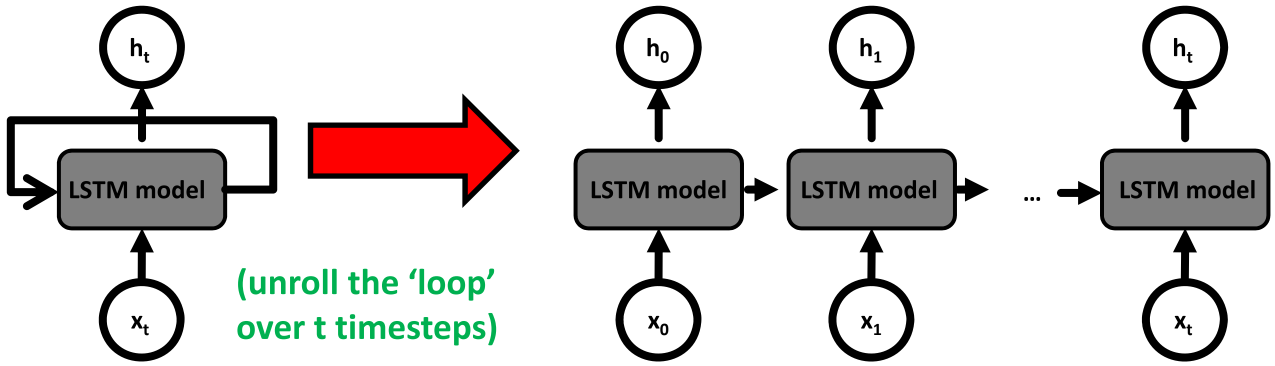


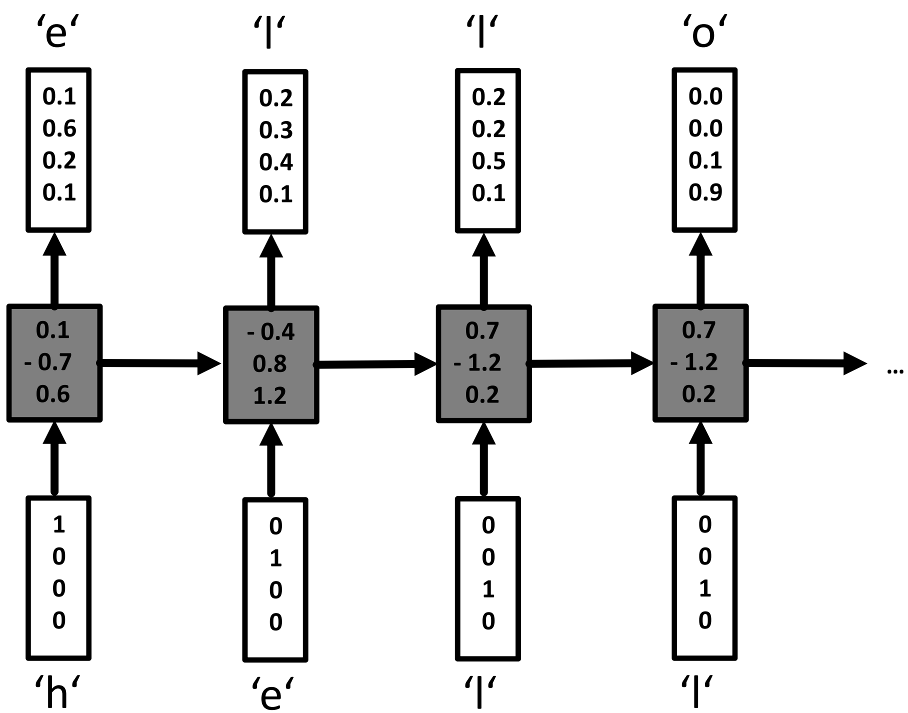
LSTM Model – Memory Cell & Cell State

- LSTM introduce a ‘memory cell‘ structure into the underlying basic RNN architecture using four key elements: an input gate, a neuron with self-current connection, a forget gate, and an output gate
- The data in the LSTM memory cell flows straight down the chain with some linear interactions (x,+)
- The cell state C_t can be different at each of the LSTM model steps & modified with gate structures
- Linear interactions of the cell state are pointwise multiplication (x) and pointwise addition (+)
- In order to protect and control the cell state C_t three different types of gates exist in the structure



LSTM Application Example – Predict Next Character





High-level Tools – Keras

- **Keras is a high-level deep learning library implemented in Python that works on top of existing other rather low-level deep learning frameworks like Tensorflow, CNTK, or Theano**
- **The key idea behind the Keras tool is to enable faster experimentation with deep networks**
- **Created deep learning models run seamlessly on CPU and GPU via low-level frameworks**

```
keras.layers.LSTM(  units,
                    activation='tanh',
                    recurrent_activation='hard_sigmoid',
                    use_bias=True,
                    kernel_initializer='glorot_uniform',
                    recurrent_initializer='orthogonal',
                    bias_initializer='zeros',
                    unit_forget_bias=True,
                    kernel_regularizer=None,
                    recurrent_regularizer=None,
                    bias_regularizer=None,
                    activity_regularizer=None,
                    kernel_constraint=None,
                    recurrent_constraint=None,
                    bias_constraint=None,
                    dropout=0.0, ...)
```

- **Tool Keras supports the LSTM model via keras.layers.LSTM() that offers a wide variety of configuration options**

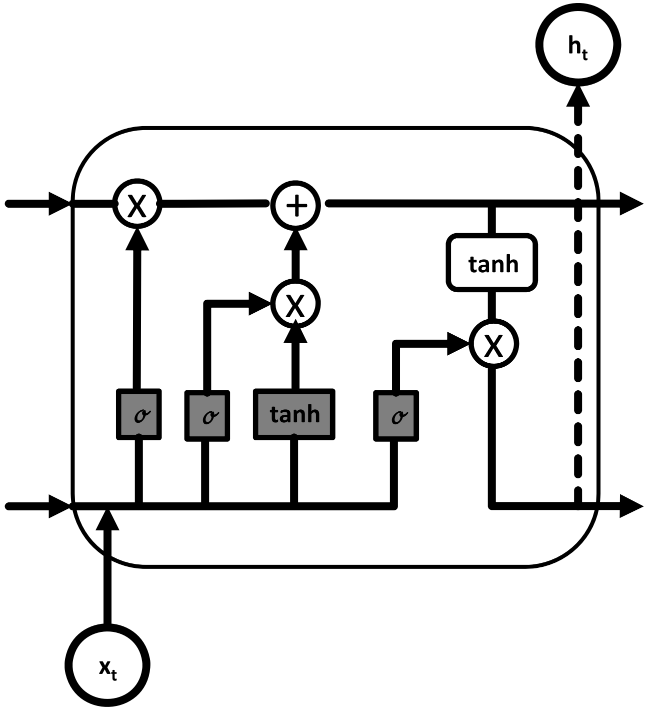
K Keras *[1] Keras Python Deep Learning Library*

Low-level Tools – Theano

- Theano is a low-level deep learning library implemented in Python with a focus on defining, optimizing, and evaluating mathematical expressions & multi-dimensional arrays
- The Theano tool supports the use of GPUs and CPUs via expressions in NumPy syntax
- Theano work with the high-level deep learning tool Keras in order to create models fast
- LSTM models are created using mathematical equations but there is no direct class for it

```
...
import numpy
import theano
from theano import config
import theano.tensor as tensor
...
def lstm_layer(tparams, state_below,
   options, prefix='lstm', mask=None):
...
i = tensor.nnet.sigmoid(_slice(preact, 0,
                        options['dim_proj']))
f = tensor.nnet.sigmoid(_slice(preact, 1,
                        options['dim_proj']))
o = tensor.nnet.sigmoid(_slice(preact, 2,
                        options['dim_proj']))
c = tensor.tanh(_slice(preact, 3,
                        options['dim_proj']))
```

theano



[2] Theano Deep Learning Framework   *[3] LSTM Networks for Sentiment Analysis*

Low-Level Tools – Tensorflow

- Tensorflow is an open source library for deep learning models using a flow graph approach
- Tensorflow nodes model mathematical operations and graph edges between the nodes are so-called tensors (also known as multi-dimensional arrays)
- The Tensorflow tool supports the use of CPUs and GPUs (much more faster than CPUs)
- Tensorflow work with the high-level deep learning tool Keras in order to create models fast
- LSTM models are created using tensors & graphs and there are LSTM package contributions

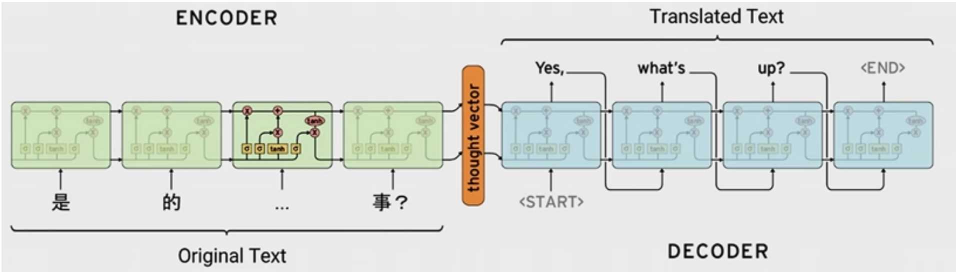
[4] Tensorflow Deep Learning Framework

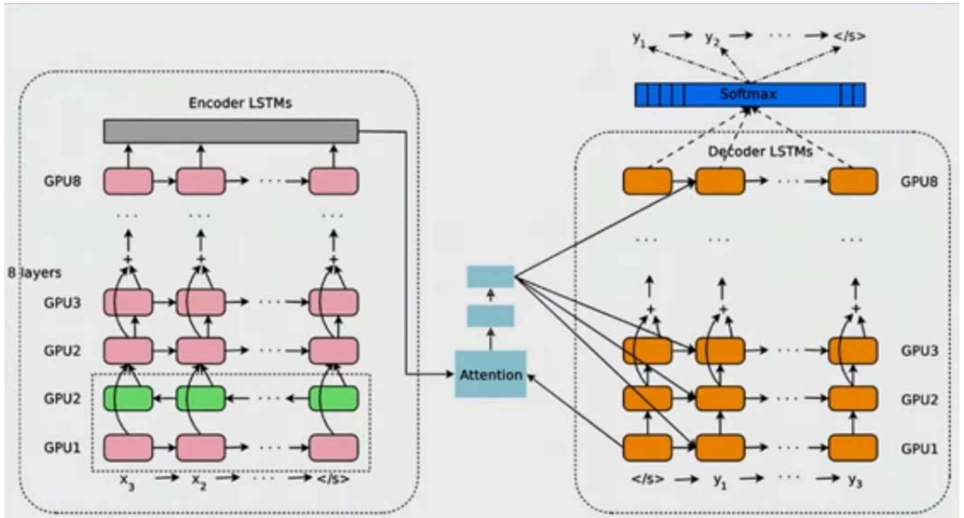
```
...
lstm = rnn_cell.BasicLSTMCell(lstm_size, state_is_tuple=False)
...
stacked_lstm = rnn_cell.MultiRNNCell([lstm] * number_of_layers,
    state_is_tuple=False)
...
initial_state = state = stacked_lstm.zero_state(batch_size, tf.float32)

for i in range(num_steps):
    # The value of state is updated
    # after processing each batch of words.
    output, state = stacked_lstm(words[:, i], state)

    # The rest of the code.
    # ...

final_state = s
```

- The class BasicLSTMCell() offers a simple LSTM Cell implementation in Tensorflow

Tensorflow – LSTM Google Translate Example & GPUs

- Use of 2 LSTM networks in a stacked manner
  - Called 'sequence-2-sequence' model
  - Encoder network
  - Decoder network
  - Needs context of sentence (memory) for translation

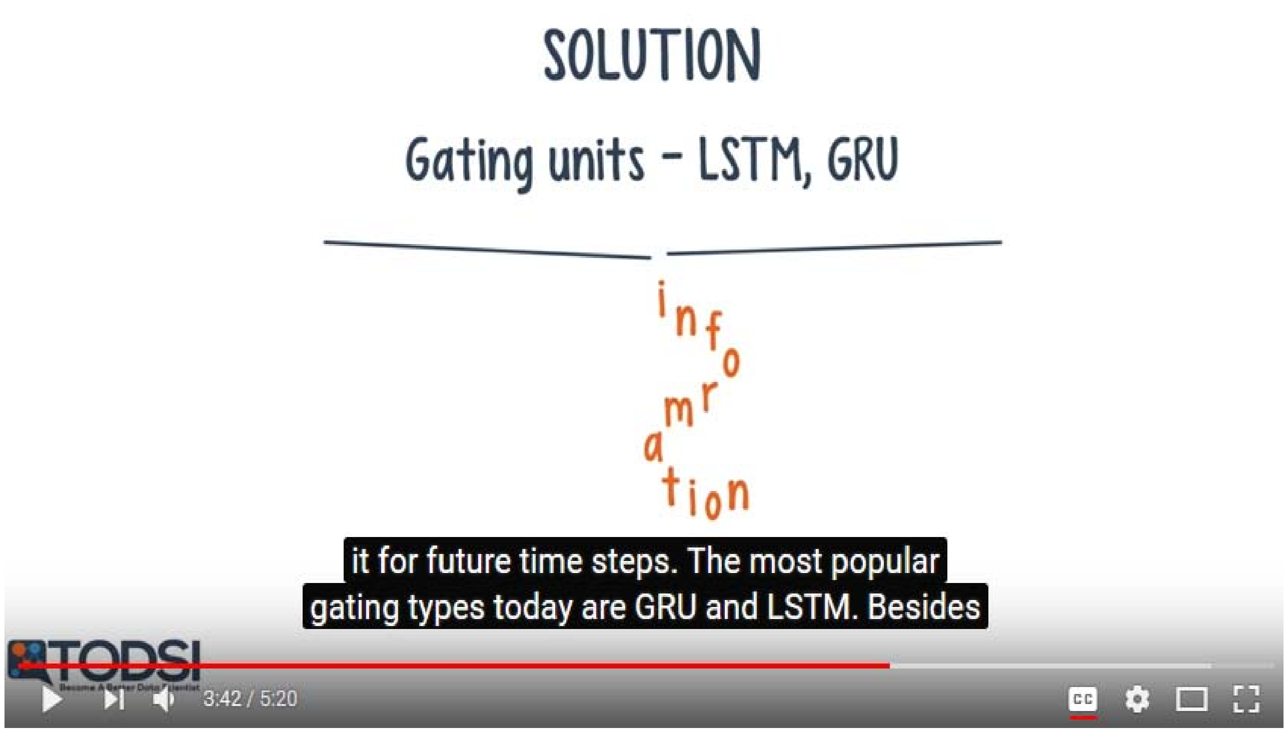




[12] Sequence Models

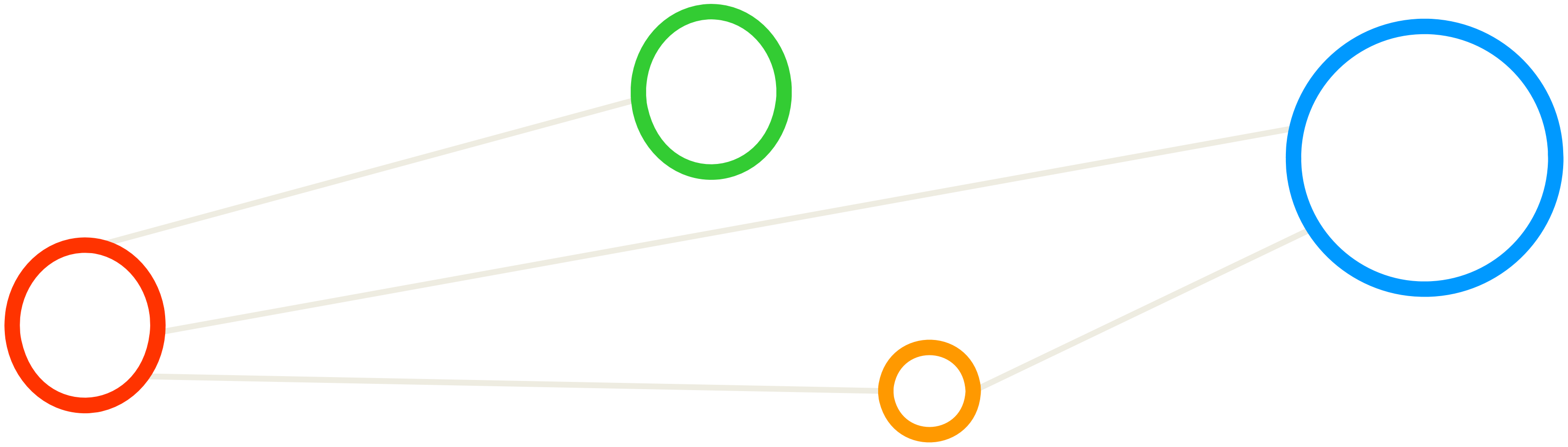
Exercises – Group Assignment – Check Status

[Video] RNN & LSTM



[5] Recurrent Neural Networks, YouTube

Summary



Exercises – Group Assignment – Check Status

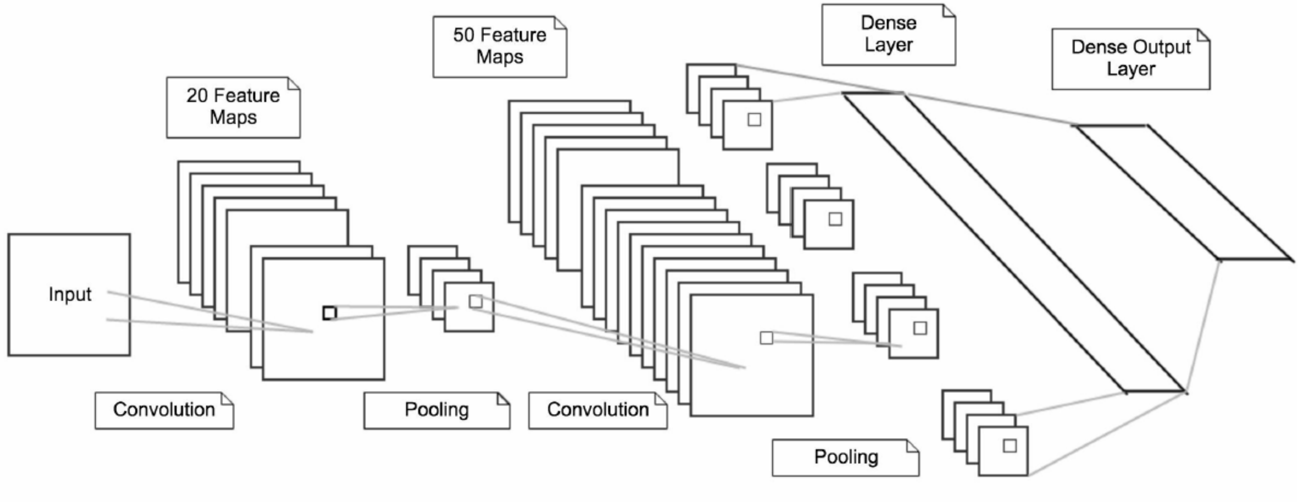
ANN – MNIST Dataset – Add Hidden Layers - Output

```
[vsc42544@gligar03 deeplearning]$ more KERAS_MNIST_ANN_HIDDEN.o1179466
60000 train samples
10000 test samples
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               100480
_________________________________________________________________
activation_1 (Activation)    (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               16512
_________________________________________________________________
activation_2 (Activation)    (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                1290
_________________________________________________________________
activation_3 (Activation)    (None, 10)                0
=================================================================
Total params: 118,282
Trainable params: 118,282
Non-trainable params: 0
_________________________________________________________________
Train on 48000 samples, validate on 12000 samples
Epoch 1/200

  128/48000 [..............................] - ETA: 4:29 - loss: 2.3122 - acc: 0.1094
 2176/48000 [>.............................] - ETA: 16s - loss: 2.2732 - acc: 0.1085
 4864/48000 [==>...........................] - ETA: 7s - loss: 2.2178 - acc: 0.1721
 7424/48000 [===>..........................] - ETA: 4s - loss: 2.1676 - acc: 0.2515
```

```
[vsc42544@gligar03 deeplearning]$ tail KERAS_MNIST_ANN_HIDDEN.o1179466

   32/10000 [..............................] - ETA: 0s
 2272/10000 [=====>........................] - ETA: 0s
 4544/10000 [============>.................] - ETA: 0s
 6784/10000 [===================>..........] - ETA: 0s
 9088/10000 [==========================>...] - ETA: 0s
10000/10000 [==============================] - 0s 22us/step
Test score: 0.0772481116249
Test accuracy: 0.9773
Working directory was /user/scratch/gent/vsc425/vsc42544/KERAS_MNIST_ANN_HIDDEN_1179466.master19.golett.gent.vsc
```

MNIST Dataset – CNN Model

```
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Activation, Flatten
from keras.utils import np_utils
from keras import backend as K
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.optimizers import SGD, RMSprop, Adam

# model
class CNN:
  @staticmethod
  def build(input_shape, classes):
    model = Sequential()
    model.add(Convolution2D(20, kernel_size=5, padding="same", input_shape=input_shape))
    model.add(Activation("relu"))
    model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
    model.add(Convolution2D(50, kernel_size=5, border_mode="same"))
    model.add(Activation("relu"))
    model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
    model.add(Flatten())
    model.add(Dense(500))
    model.add(Activation("relu"))
    model.add(Dense(classes))
    model.add(Activation("softmax"
    return model
```



[9] A. Gulli et al.

MNIST Dataset – CNN Model – Output

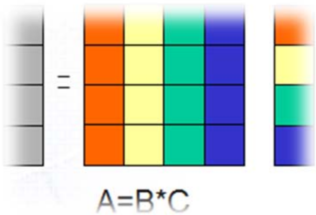
```
[vsc42544@gligar01 deeplearning]$ head KERAS_MNIST_CNN.o1179880
60000 train samples
10000 test samples
Train on 48000 samples, validate on 12000 samples
Epoch 1/20

  128/48000 [..............................] - ETA: 10:06 - loss: 2.2997 - acc: 0.1250
  256/48000 [..............................] - ETA: 7:46 - loss: 2.2578 - acc: 0.1992
  384/48000 [..............................] - ETA: 6:58 - loss: 2.2127 - acc: 0.2083
  512/48000 [..............................] - ETA: 6:35 - loss: 2.1632 - acc: 0.2598
  640/48000 [..............................] - ETA: 6:20 - loss: 2.0934 - acc: 0.3234
```

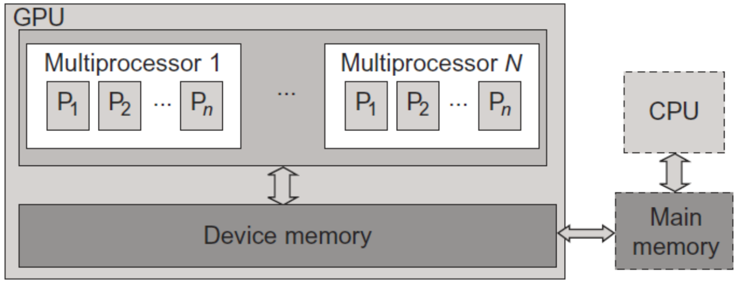
```
[vsc42544@gligar01 deeplearning]$ tail KERAS_MNIST_CNN.o1179880
 9824/10000 [============================>.] - ETA: 0s
 9856/10000 [============================>.] - ETA: 0s
 9888/10000 [============================>.] - ETA: 0s
 9920/10000 [============================>.] - ETA: 0s
 9952/10000 [============================>.] - ETA: 0s
 9984/10000 [============================>.] - ETA: 0s
10000/10000 [==============================] - 41s 4ms/step
Test score: 0.0483192791523
Test accuracy: 0.99
Working directory was /user/scratch/gent/vsc425/vsc42544/KERAS_MNIST_CNN_1179880.master19.golett.gent.vsc
```

GPU Acceleration

- **CPU acceleration means that GPUs accelerate computing due to a massive parallelism with thousands of threads compared to only a few threads used by conventional CPUs**
- **GPUs are designed to compute large numbers of floating point operations in parallel**

- GPU accelerator architecture example (e.g. NVIDIA card)
  - GPUs can have 128 cores on one single GPU chip
  - Each core can work with eight threads of instructions
  - GPU is able to concurrently execute 128 * 8 = 1024 threads
  - Interaction and thus major (bandwidth) bottleneck between CPU and GPU is via memory interactions
  - E.g. applications that use matrix – vector multiplication



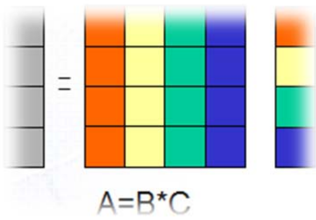


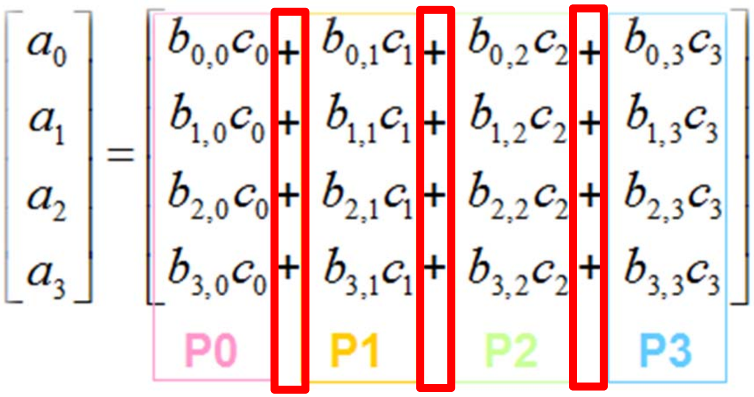
[7] Distributed & Cloud Computing Book

(other well known accelerators & many-core processors are e.g. Intel Xeon Phi → run 'CPU' applications easier)

GPU Application Example – Matrix-Vector Multiplication

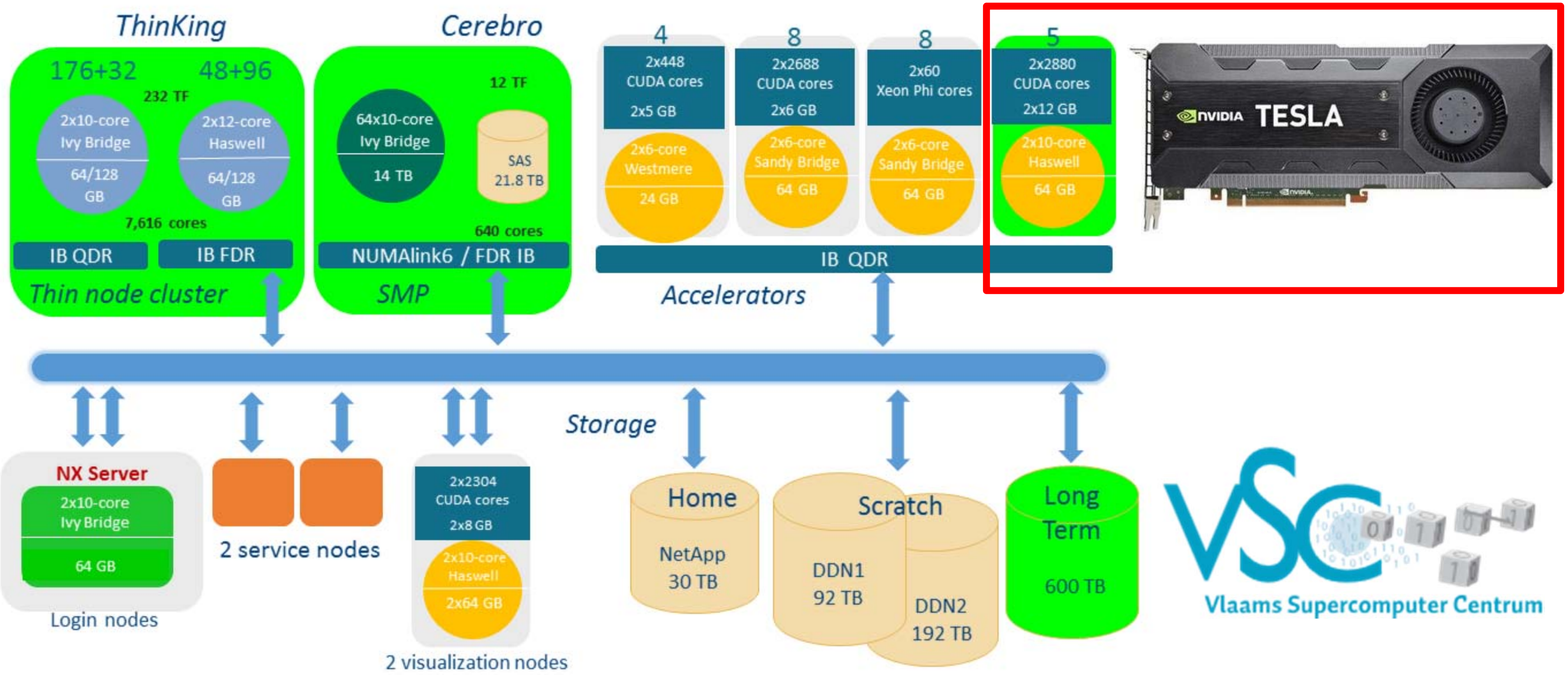
- Many machine learning problems include matrix multiplications





HPC System KU Leuven – GPUs

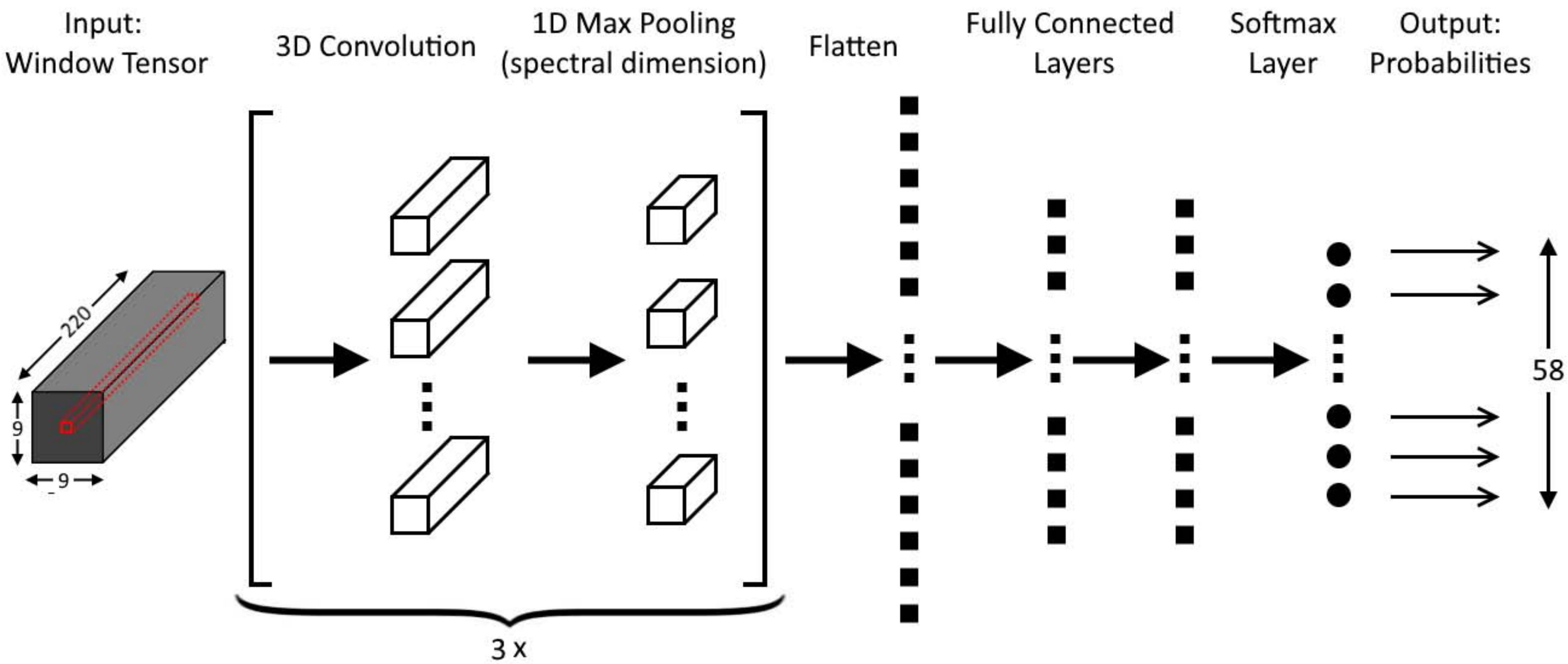
- Accelerators
  - Nodes with two 10-core "Haswell" Xeon E5-2650v3 2.3GHz CPUs, 64 GB of RAM and 2 GPUs Tesla K40



modified from [8] HPC System KU Leuven

CNN Architecture for Remote Sensing Application

- Classify pixels in a hyperspectral remote sensing image having groundtruth/labels available
- Created CNN architecture for a specific hyperspectral land cover type classification problem
- Used dataset of Indian Pines (compared to other approaches) using all labelled pixels/classes
- Performed no manual feature engineering to obtain good results (aka accuracy)



Small Data – Outputs

```
- - - - - - - Begin - Information - - - - - - -

--> Data:
> Number of classes: 16, HS-channels: 220, Window-size: 9
> Mean: 2524.4013671875 , Standard deviation: 1603.017822265625
> Excluded labels: []
> Number of training samples: 1036
> Number of test samples: 9330
--> Learning:
> Epochs: 1000, Batch size: 50
> LR: 1, Momentum: 0, LR-decay: 5e-06
> Activation Functions: relu, Loss Function: mean_squared_error
> Optimizer: SGD
--> Regularization:
> Dropout: 0.0
> L2 regularization with factor: 0.0

- - - - - - - End - Information - - - - - - -
```

Layer (type)	Output Shape	Param #
conv3d_1 (Conv3D)	(None, 7, 7, 216, 48)	2208
max_pooling3d_1 (MaxPooling3	(None, 7, 7, 72, 48)	0
zero_padding3d_1 (ZeroPaddin	(None, 7, 7, 76, 48)	0
conv3d_2 (Conv3D)	(None, 5, 5, 72, 32)	69152
max_pooling3d_2 (MaxPooling3	(None, 5, 5, 24, 32)	0
zero_padding3d_2 (ZeroPaddin	(None, 5, 5, 28, 32)	0
conv3d_3 (Conv3D)	(None, 3, 3, 24, 32)	46112
max_pooling3d_3 (MaxPooling3	(None, 3, 3, 12, 32)	0
flatten_1 (Flatten)	(None, 3456)	0
dense_1 (Dense)	(None, 128)	442496
dense_2 (Dense)	(None, 128)	16512
dense_3 (Dense)	(None, 16)	2064

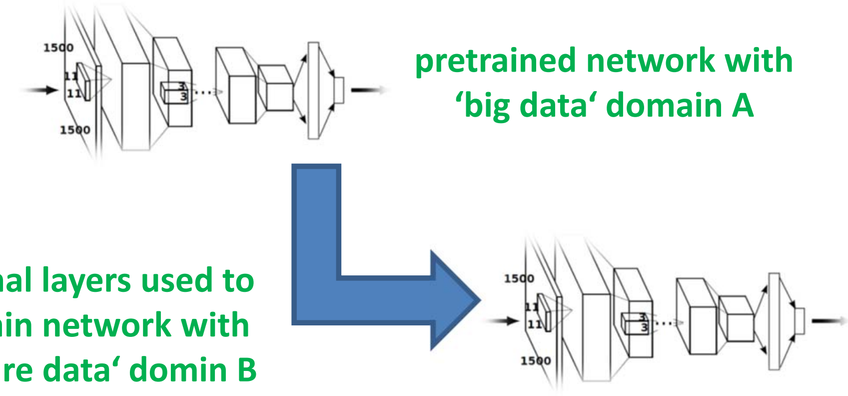
Total params: 578,544

```
loss: 0.027155295770896957 - acc: 0.7379421221609412
New loss is bigger --> dont save model

1036/1036 [==============================] - 6s 5ms/step - loss: 8.0197e-04 - acc: 0.9894
 > Time needed for learning and testing: 0.49032413981337514 hours
```

Full Data – Output (2)

```
[vsc42544@gligar02 .deep_learning_private]$ sed -n '906765,906824p' IndianPines_full_2GPU.o20657803
300821/300821 [==============================] - 137s 457us/step
loss: 0.004567355140805838 - acc: 0.8353838329111958
New loss is bigger --> dont save model

33424/33424 [==============================] - 182s 5ms/step - loss: 4.8412e-04 - acc: 0.9762
 > Time needed for learning and testing: 16.111324077533144 hours

 -   -   -   -   -   -   -   Begin - Information -   -   -   -   -   -   -

 --> Data:
 > Number of classes: 58, HS-channels: 220, Window-size: 9
 > Mean: 2424.492431640625 , Standard deviation: 1431.9654541015625
 > Excluded labels: []
 > Number of training samples: 33424
 > Number of test samples: 300821
 --> Learning:
 > Epochs: 1000, Batch size: 50
 > LR: 1, Momentum: 0, LR-decay: 5e-06
 > Activation Functions: relu, Loss Function: mean_squared_error
 > Optimizer: SGD
 --> Regularization:
 > Dropout: 0.0
 > L2 regularization with factor: 0.0

 -   -   -   -   -   -   -   End - Information   -   -   -   -   -   -   -
```

Transfer Learning Results – Transferability



- **Data randomly taken from various city images and used with the trained CNN using pre-trained ImageNet**
- **Even on unseen data from complete different datasets transfer learning is working well**
- **Shown for scene-wide classification, not much for pixel-wise classification**

[10] D. Marmanis et al., 'Deep Learning Earth Obervation Classification Using ImageNet Pretrained Networks', 2016

Problem of Overfitting – Impacts on Learning

- **The higher the degree of the polynomial (cf. model complexity), the more degrees of freedom are existing and thus the more capacity exists to overfit the training data**

- Understanding deterministic noise & target complexity
  - Increasing target complexity increases deterministic noise (at some level)
  - Increasing the number of data N decreases the deterministic noise
- Finite N case: $\mathcal{H}$ tries to fit the noise
  - Fitting the noise straightforward (e.g. Perceptron Learning Algorithm)
  - Stochastic (in data) and deterministic (simple model) noise will be part of it
- Two 'solution methods' for avoiding overfitting
  - Regularization: 'Putting the brakes in learning', e.g. early stopping (more theoretical, hence 'theory of regularization')
  - Validation: 'Checking the bottom line', e.g. other hints for out-of-sample (more practical, methods on data that provides 'hints')

High-level Tools – Keras – Regularization Techniques

- Keras is a high-level deep learning library implemented in Python that works on top of existing other rather low-level deep learning frameworks like Tensorflow, CNTK, or Theano
- The key idea behind the Keras tool is to enable faster experimentation with deep networks
- Created deep learning models run seamlessly on CPU and GPU via low-level frameworks

```
keras.layers.Dropout(rate,
                     noise_shape=None,
                     seed=None)
```

- Dropout is randomly setting a fraction of input units to 0 at each update during training time, which helps prevent overfitting (using parameter rate)

```
from keras import regularizers
model.add(Dense(64, input_dim=64,
kernel_regularizer=regularizers.l2(0.01),
activity_regularizer=regularizers.l1(0.01)))
```

- L2 regularizers allow to apply penalties on layer parameter or layer activity during optimization itself – therefore the penalties are incorporated in the lost function that the network optimizes

Keras

[11] Keras Python Deep Learning Library

Remote Sensing - Experimental Setup @ JSC – Revisited

- CNN Setup
  - Table overview
- HPC Machines used
  - Systems JURECA and JURON
- GPUs
  - NVIDIA Tesla K80 (JURECA)
  - NVIDIA Tesla P100 (JURON)
  - While Using MathWorks' Matlab for the data
- Frameworks
  - Keras library (2.0.6) was used
  - Tensorflow (0.12.1 on Jureca, 1.3.0rc2 on Juron) as back-end
  - Automated usage of the GPU's of these machines via Tensorflow

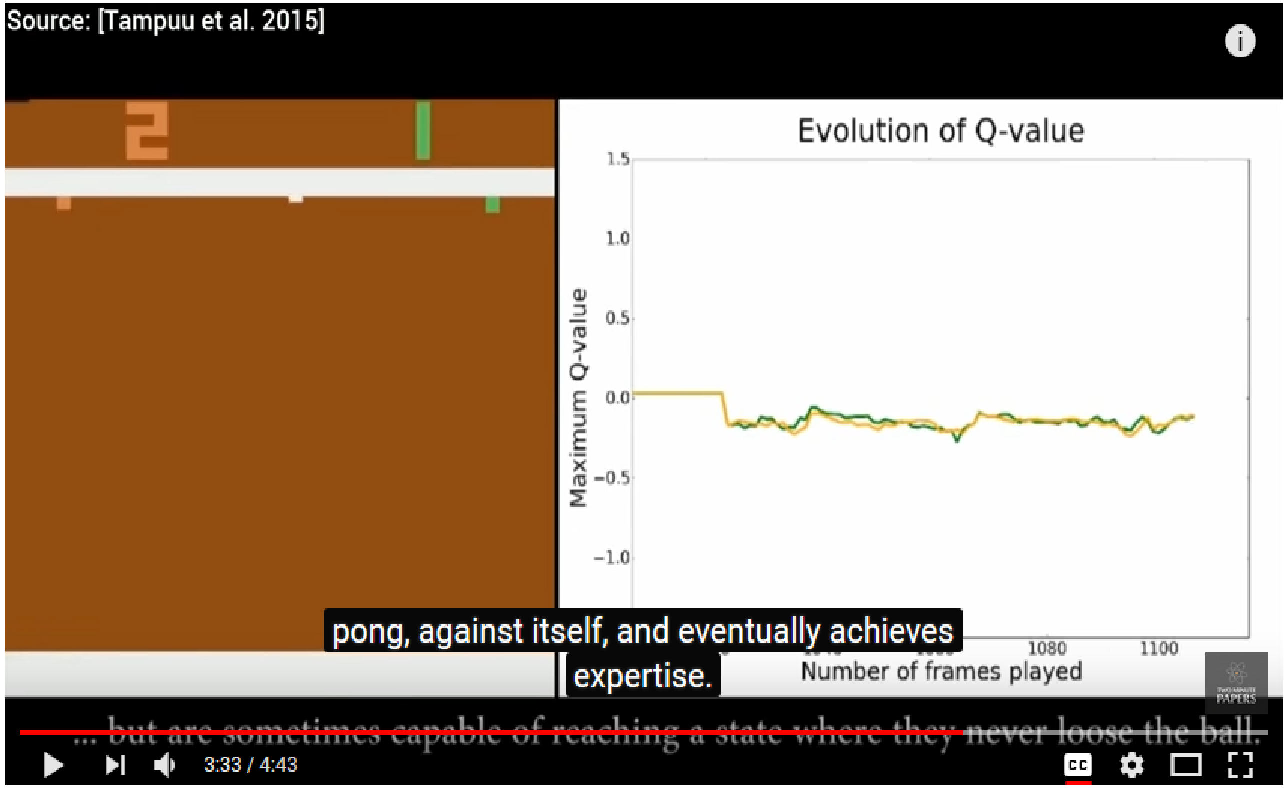
Feature	Representation / Value
Conv. Layer Filters	48, 32, 32
Conv. Layer Filter size	(3,3,5), (3,3,5), (3,3,5)
Dense Layer Neurons	128, 128
Optimizer	SGD
Loss Function	mean squared error
Activation Functions	ReLU
Training Epochs	600
Batch Size	50
Learning Rate	1
Learning Rate Decay	5×10^{-6}

(adding regularization values adds even more complexity in finding the right parameters)

(having the validation with the full grid search of all parameters and all combinations is quite compute – intensive → ~infeasable)

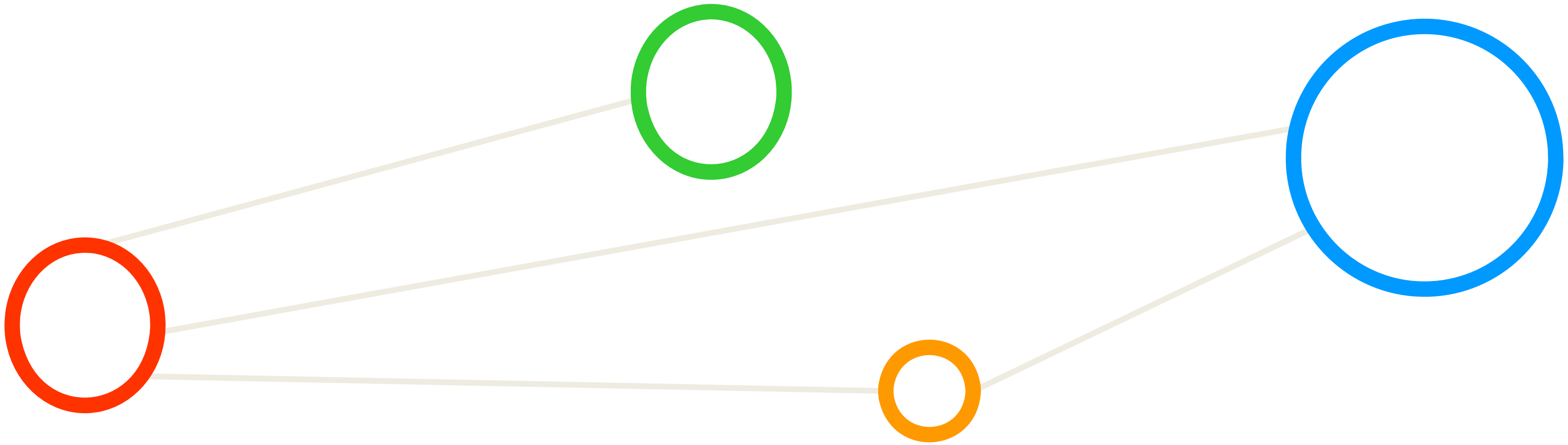
Exercises – Group Assignment – Check Status

[Video] Deep Learning Applications



[6] Deep Learning Applications, YouTube

Lecture Bibliography



Lecture Bibliography (1)

- [1] Keras Python Deep Learning Library,
  Online: https://keras.io/
- [2] Theano Deep Learning Framework,
  Online: https://github.com/Theano/Theano
- [3] LSTM Networks for Sentiment Analysis,
  Online: http://deeplearning.net/tutorial/lstm.html
- [4] Tensorflow Deep Learning Framework,
  Online: https://www.tensorflow.org/
- [5] YouTube Video, 'Recurrent Neural Networks - Ep. 9 (Deep Learning SIMPLIFIED)',
  Online: https://www.youtube.com/watch?v=_aCuOwF1ZjU&t=7s
- [6] YouTube Video, '9 Cool Deep Learning Applications | Two Minute Papers #35',
  Online: https://www.youtube.com/watch?v=Bui3DWs02h4
- [7] K. Hwang, G. C. Fox, J. J. Dongarra, 'Distributed and Cloud Computing', Book,
  Online: http://store.elsevier.com/product.jsp?locale=en_EU&isbn=9780128002049
- [8] HPC System KU Leuven,
  Online: https://www.vscentrum.be/infrastructure/hardware/hardware-kul
- [9] A. Gulli and S. Pal, 'Deep Learning with Keras' Book, ISBN-13 9781787128422, 318 pages,
  Online: https://www.packtpub.com/big-data-and-business-intelligence/deep-learning-keras

Lecture Bibliography (2)

- [10] Dimitrios Marmanis et al., 'Deep Learning Earth Obervation Classification Using ImageNet Pretrained Networks', IEEE Geoscience and Remote Sensing Letters, Volume 13 (1), 2016, Online: http://ieeexplore.ieee.org/document/7342907/
- [11] Keras Python Deep Learning Library, Online: https://keras.io/
- [12] YouTube Video,'Sequence Models and the RNN API (TensorFlow Dev Summit 2017)', Online: https://www.youtube.com/watch?v=RIR_-Xlbp7s


services
measurement
within
fusion
new
funding
Policy-based
concepts
device
Analysis
forms
cross-disciplinary
resources
Centre
climate
Computer
expertise
Approaches
construction
Fusion
year
like
Computational
Test
E.g
Computing
PB
community
disciplines
results
Enable
different
Cross-Disciplinary
EU
methods
High
big
nodes
Environment
Modelling
Science
centers
Key
scientific
important
files
etc
DKRZ
Use
storage
basis
Infrastructure
SMAQ
international
model
engineering
References
formats
simulations
ENES
technologies
brain
increasing
via
EUDAT
physical
IPCC
Big
Project
Data
processing
Common
TB
MapReduce
create
Methods
computing
Using
European
databases
GB
Supercomputing
Work
images
codes
analysis
DLCL
stored
Resources
analyze
Large
Summary
data-intensive
Service
Scientific
levels
often
use
Level
Research
Matters
Concrete
external
performance
JSC
computational
datasets
research
e.g
science
HPC
Juelich
system
Landscape
acquisition
large-scale
compute
management
System
Climate
modelling
AR5
Hardware
distributed
cores
access
hundreds
Nutshell
Services
network
current
DLCLs
Understanding
structures
Technologies
Earth
Structure
Germany
Simulation
directory
project
General
Provide
NASA
Energy
systems
day
Processing
Health