ZESS Lectures:

Recent Advances in Machine Learning

Deep Learning Methods for Human Activity Recognition

Alexander Hölzemann and Kristof Van Laerhoven
Ubiquitous Computing Group, University of Siegen

Outlines

1. Application fields for time series and neural networks
2. Recap of neural networks
3. Discussing the paper „Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition“ of J. Ordoñoz and D. Roggen, University of Sussex – published 2016
4. Transfer Learning
5. Data Augmentation
6. Future Challanges
7. Project Proposal

Applications

Usages of neural networks and sensor based data leads to many fields of applications

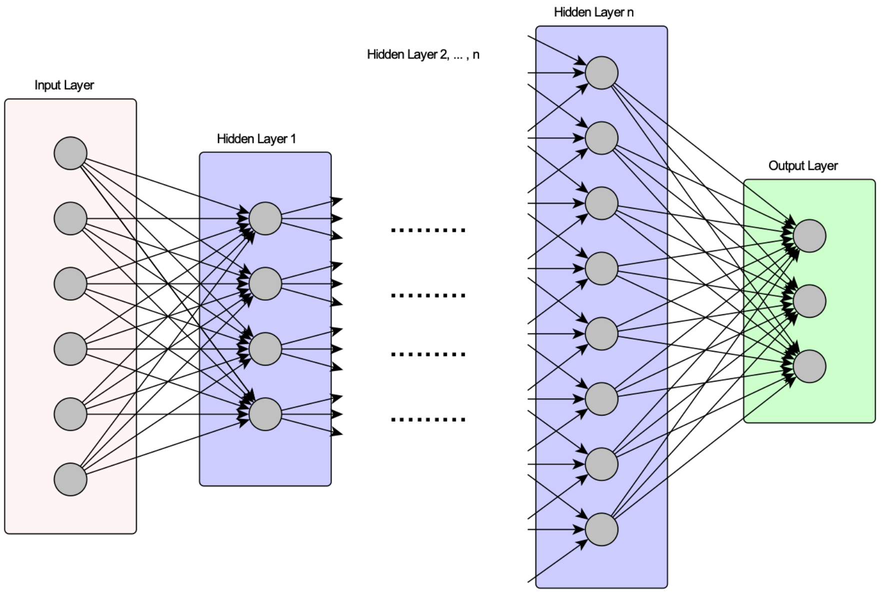
- Industrial
  - Supervision of machine status
  - Recommender systems for improving manufacturing processes
- Medical
  - Detecting the patients status during examinations
    - Sleep detection
  - Provides objective views on patients during studies
    - Smoke detection
- Consumer goods
  - Fitness Tracker
  - Condition monitoring of car drivers
  - ...

Architectures

- Deep Neural Network
  - Contains more Hidden Layers then a Artificial Neural Network
    → therefore it is more capable of learning from large data
- Convolutional Neural Network
  - Contains convolutional layers for subsampling the data
  - Traditionally comes from computer vision
  - Due to recent investigations finds it way into other ML related fields like Human Activity Recognition (HAR) or language processing
- Recurrent Neural Network
  - Are often combined with Long Short-Term Memory (LSTM) cells to model temporal correlations of data
  - LSTMs are serving as memory units
- Hybrid Models
  - CNN in combination with LSTMs [Ordóñez et al., 2016]

Layers – Input and Output Layer

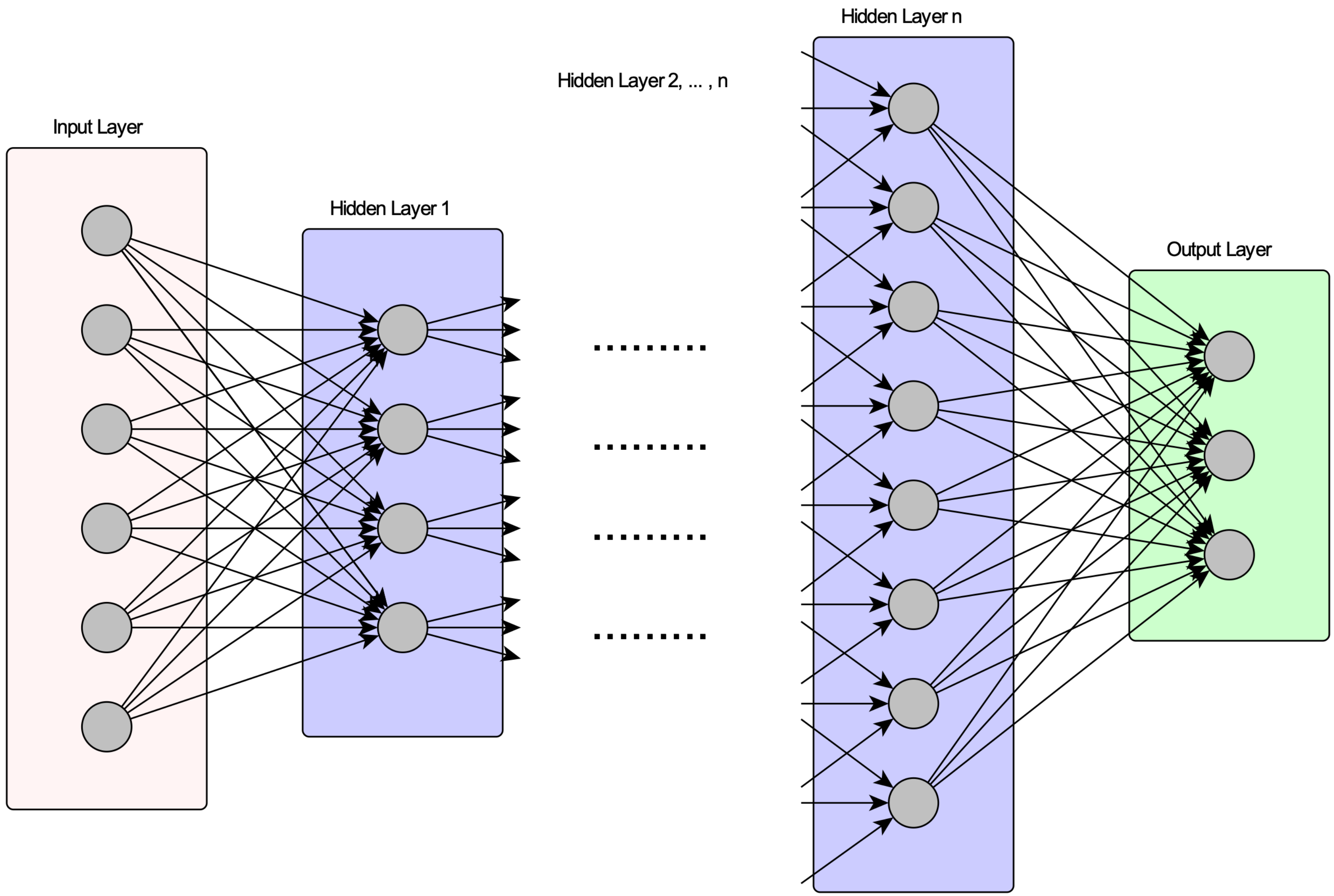
- Number of input neurons depends on the dataset
- Can be i.e. features of sample, channels of sensors or pixel of images
- Number of output neurons is described by the number of classes that should be recognized
- Layers in between vary depending on the implemented architecture



Layers - Hidden Layers

- Layers between the input and output layers are called hidden layers
- We distinguish different types of layers
  - Fully connected layer
  - Convolutional layer
  - Pooling layer
  - Recurrent layer
  - ....
- Are used as filters. I.e. in image processing for filtering edges or shapes

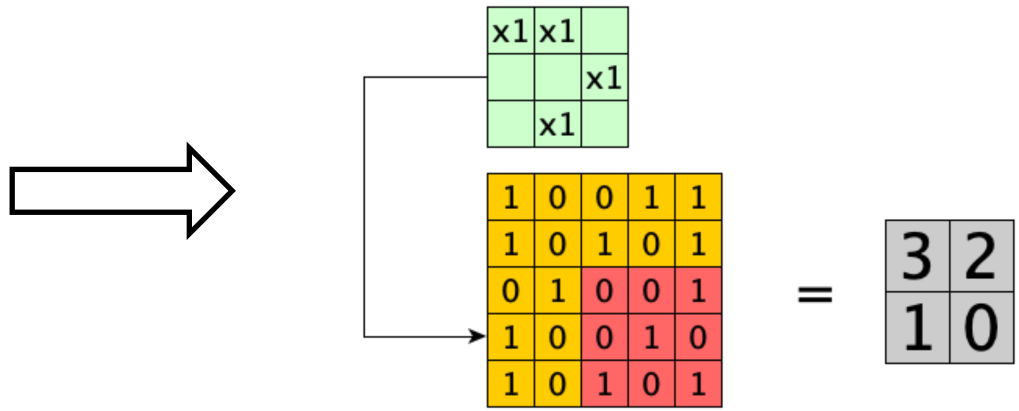
Fully Connected or Dense Layer



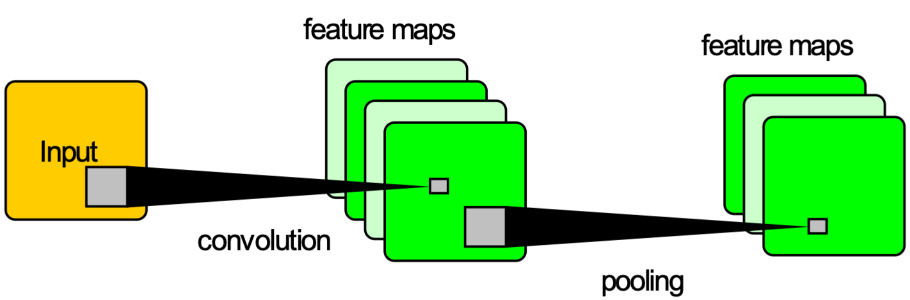
Convolution-Layer

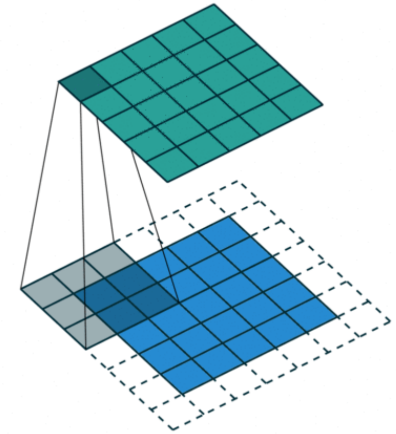
Convolution-Parameters: kernel= (3,3), padding=0, stride=2

1	0	0	1	1
1	0	1	0	1
0	1	0	0	1
1	0	0	1	0
1	0	1	0	1

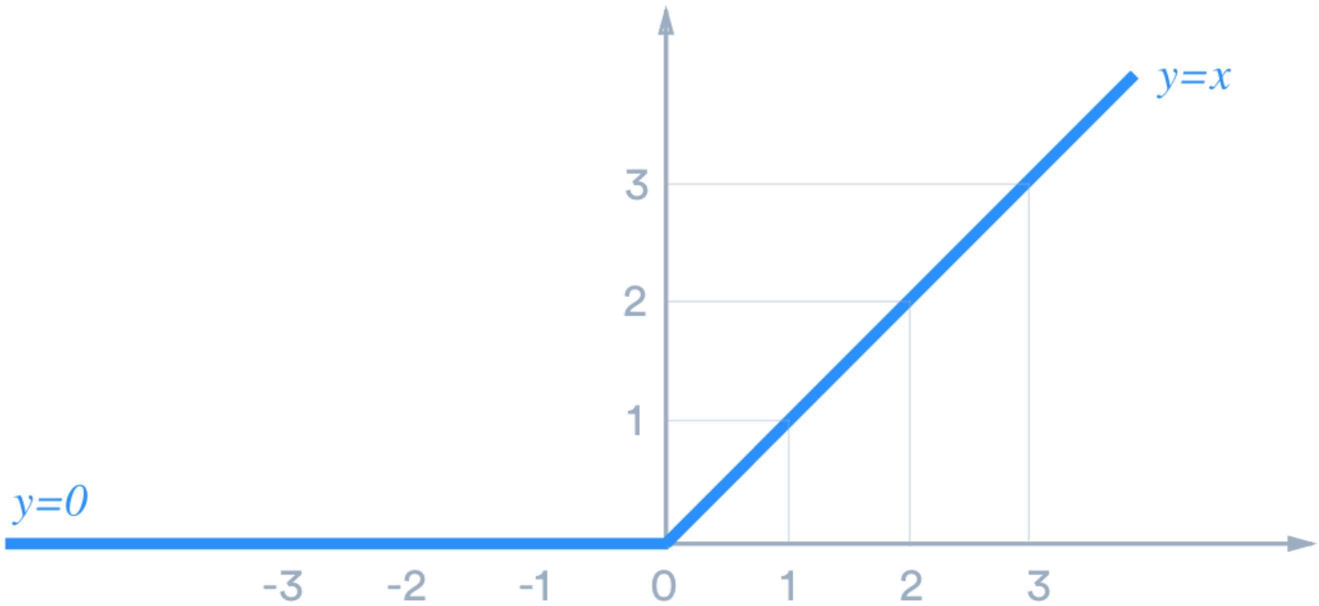


Convolution-Parameters:
kernel=(3,3), padding=1, stride=1



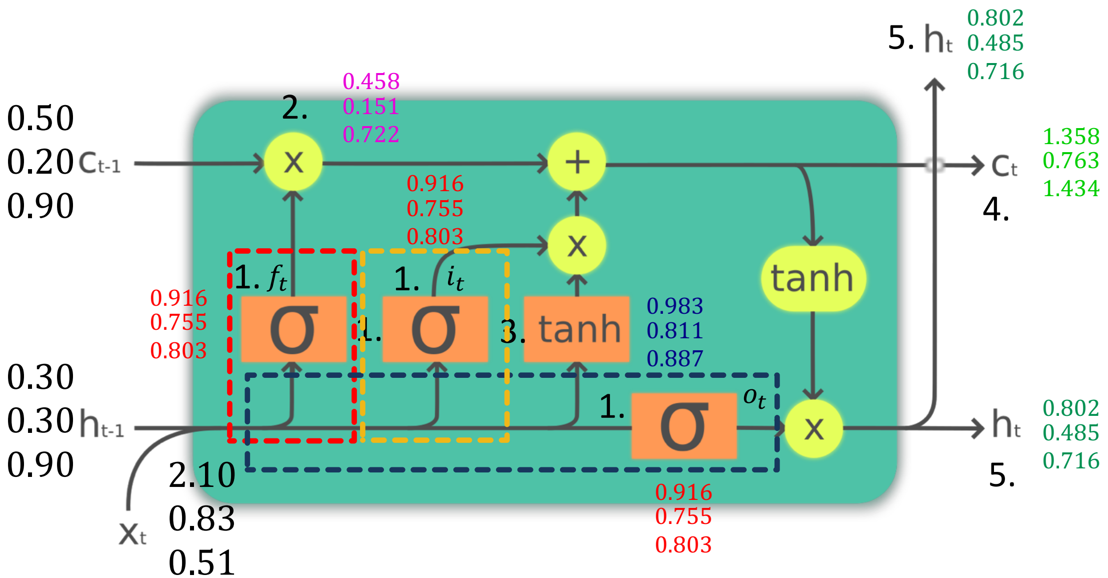


Rectified Linear Unit Function (ReLU)



- Activation-Function that got popular in recent scientific publications
- Definition: $f(y)\ \max(0, y)$
  - If y <0 ==> 0, else y
- Sigmoid function projects values between 0.0 and 1.0, ReLU between 0 and ∞.

LSTM Cells

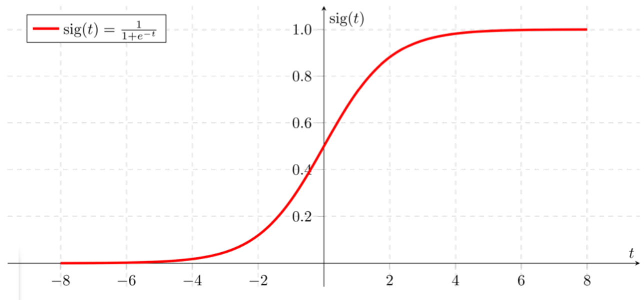


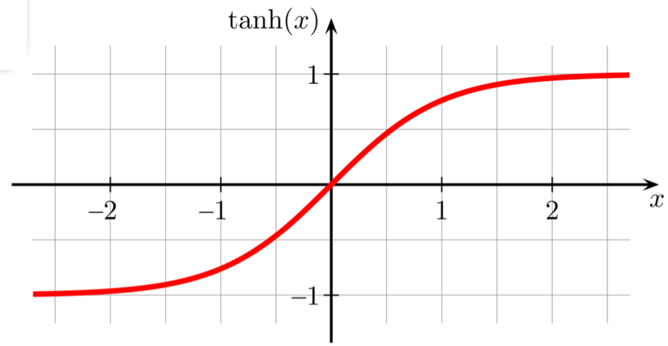
Usage of calculations for next cell or layer is controlled by **gates**

$f_t = forget\ gate$ | is memory set to 0?

$i_t = input\ gate$ | is cell updated? (which values)

$o_t = output\ gate$ | is current info visible?





$c_{t-1} = Input\ cell\ state\ vector$

$h_{t-1} = Input\ hidden\ state\ vector$

$c_t = output\ cell\ state\ vector$

$h_t = output\ hidden\ state\ vector$

$x_t = Input\ vector$

$\sigma = Sigmoid\ function$

$tanh = \tanh\ function$

$$1.)\ \begin{pmatrix}0.30 & & 2.10\\ 0.30 & + & 0.83\\ 0.90 & & 0.51\end{pmatrix} * \sigma = \begin{matrix}2.40\\1.13\\1.41\end{matrix} * \sigma = \begin{matrix}0.916\\0.755\\0.803\end{matrix}$$

$$2.)\ \begin{matrix}0.50\\0.20\\0.90\end{matrix} * \begin{matrix}0.916\\0.755\\0.803\end{matrix} = \begin{matrix}0.458\\0.151\\0.722\end{matrix}$$

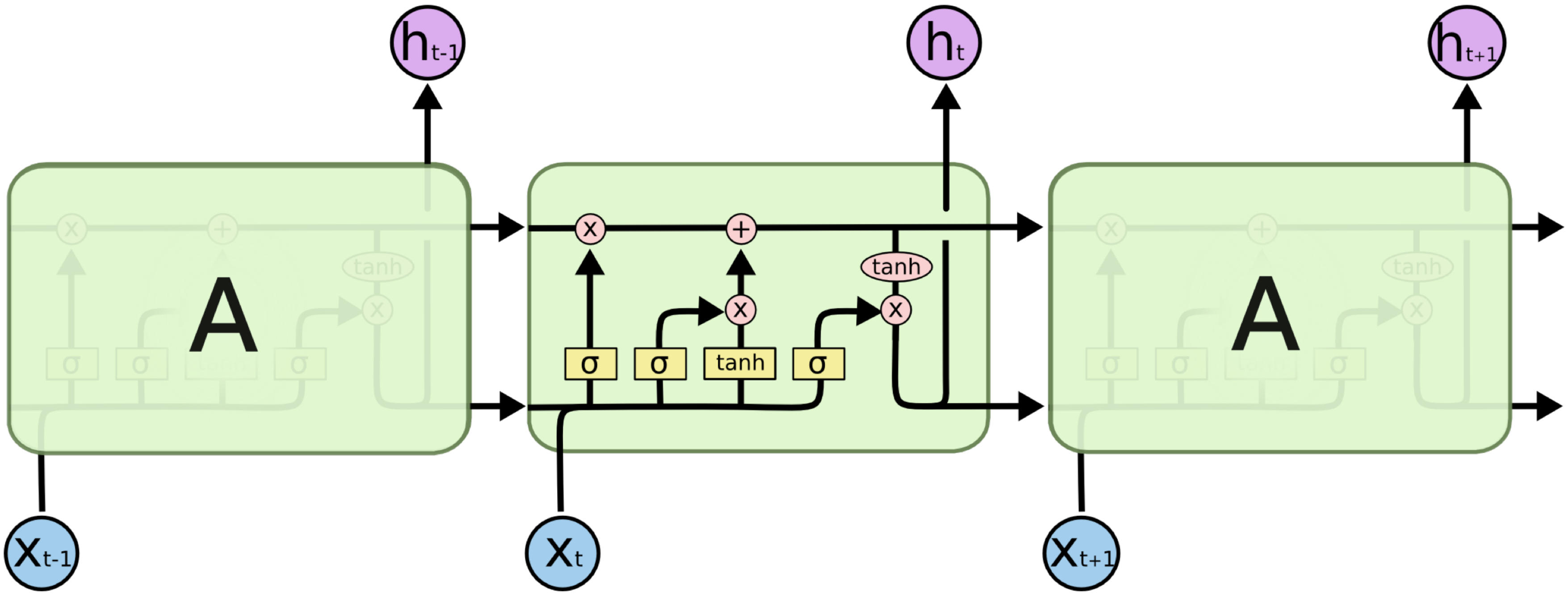
$$3.)\ \tanh\begin{pmatrix}0.30 & & 2.10\\ 0.30 & + & 0.83\\ 0.90 & & 0.51\end{pmatrix} = \begin{matrix}0.983\\0.811\\0.887\end{matrix}$$

$$4.)\ \begin{matrix}0.983\\0.811\\0.887\end{matrix} * \begin{matrix}0.916\\0.755\\0.803\end{matrix} + \begin{matrix}0.458\\0.151\\0.722\end{matrix} = \begin{matrix}1.358\\0.763\\1.434\end{matrix} = c_t$$

$$5.)\ \tanh\begin{pmatrix}1.358\\0.763\\1.434\end{pmatrix} * \begin{matrix}0.916\\0.755\\0.803\end{matrix} = \begin{matrix}0.802\\0.485\\0.716\end{matrix} = h_t$$

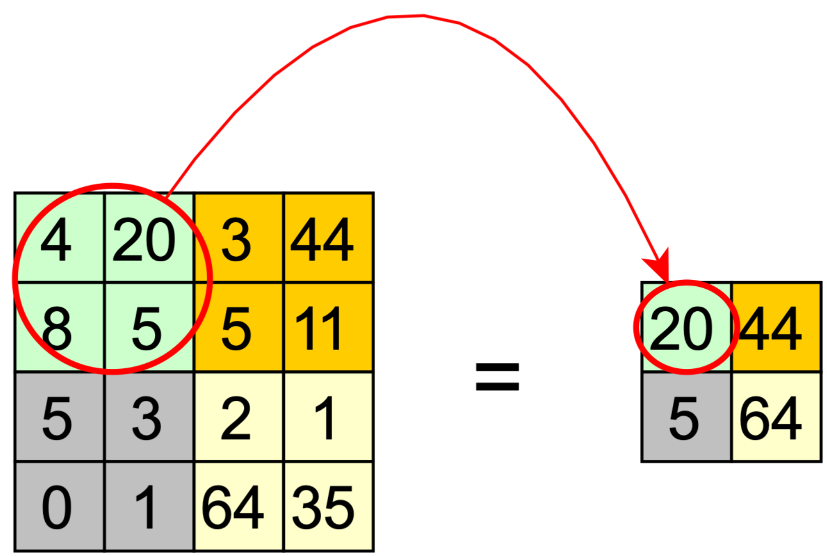
Source: https://commons.wikimedia.org/

LSTM Cells in a Recurrent Layer

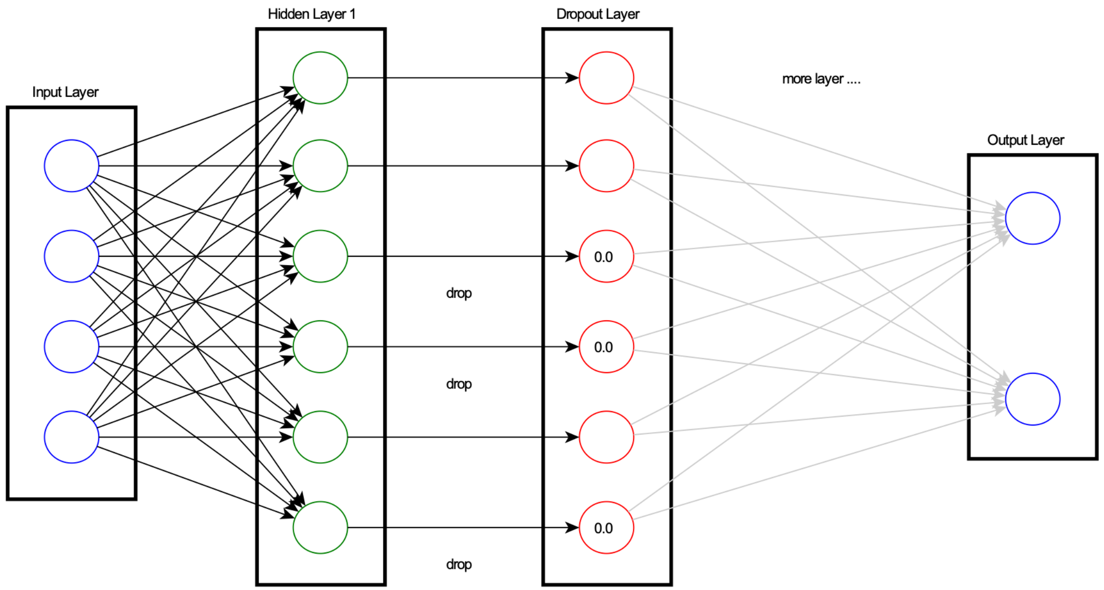


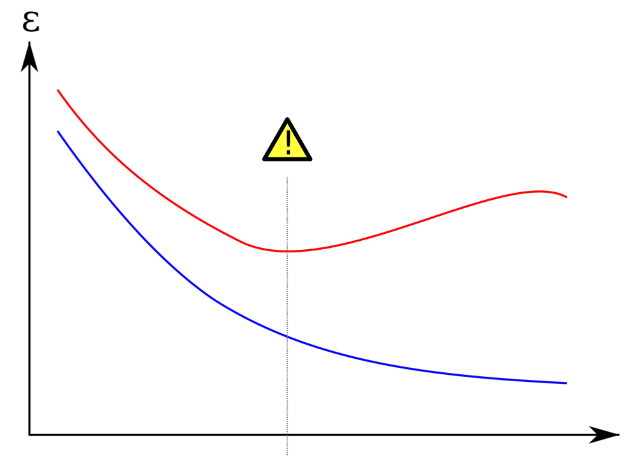
Pooling-Layer

- (Max) – Pooling
- Used to reduce data
- Example with: kernel = 2 and stride = 2
  - Used for subsampling the data, without loosing too much Important information
- More pooling algorithms
  - Min pooling
  - Mean pooling
  - ...



Dropout-Layer and Overfitting

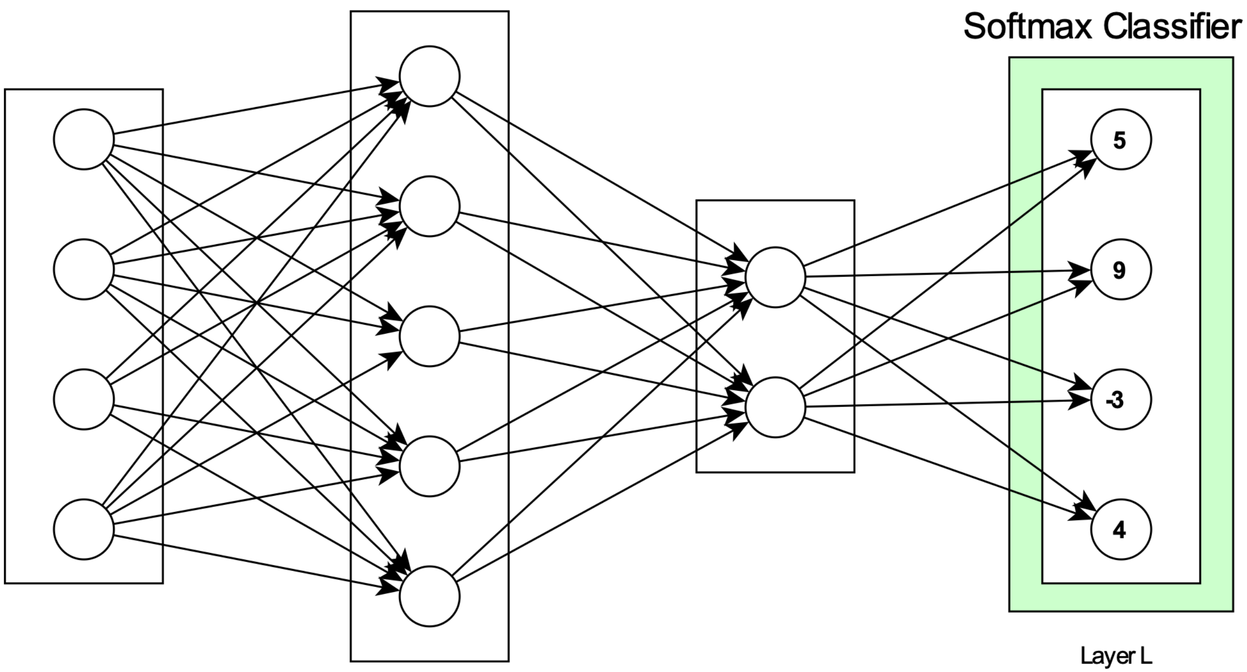




- Regulation-Method for preventing overfitting of a cnn
- Drops out a percentage of neurons per layer, i.e. 50%
- These neurons gets choosen randomly and will not be regarded for the next calculation step by setting them to 0.0

- Overfitting means an overspecification of a certain model
- the model is not able to generalize
- Blue line represents the error rate for training data, red the error for test data.
- Red lines raises and blue line falls

→ Overfitting !!



Softmax-Activation-Function is defined as: $\boldsymbol{t_i = e^{(x^L)}}$ $\boldsymbol{and\ a^L = \frac{t_i}{\sum_{i=1}^{||L||} t_i}}$

Example:

||L|| = (4,1), $t = \ (\, e^5 \; e^9 \; e^{-3} \; e^4 \,) =$ (148.1 8103.1 0.049 55.6)

Now we need to normalize the results in values between 0 and 1.

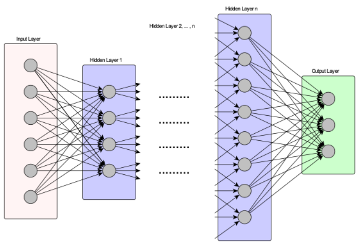
$\sum_{i=1}^{4} = 148.1 + 8103.1 + 0.049 + 55.6 = 8306.849$

→ ((148.1 / 8306,849) (8103.1 / 8306,849) (0.049 / 8306,849) (55.6 / 8306,849))

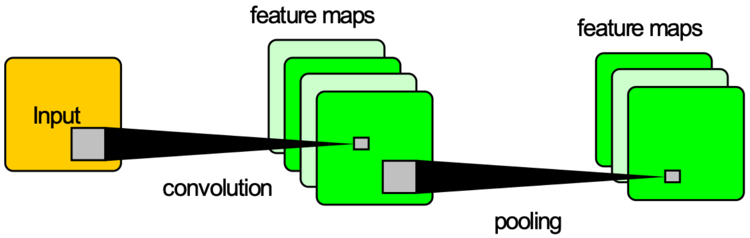
= (0.0178 0.975 0.000005 0.0066) $\Longrightarrow$ with a probability of 97.5 % it‘s class 2

Summary

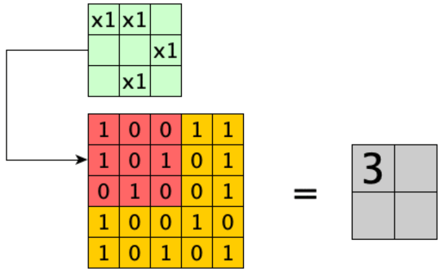
- **Layer:** One calculation step of a neural network. Different types of layers are usable.



- **Feature maps:** output of a subsampling layer (e.g. convolution, pooling, ...)



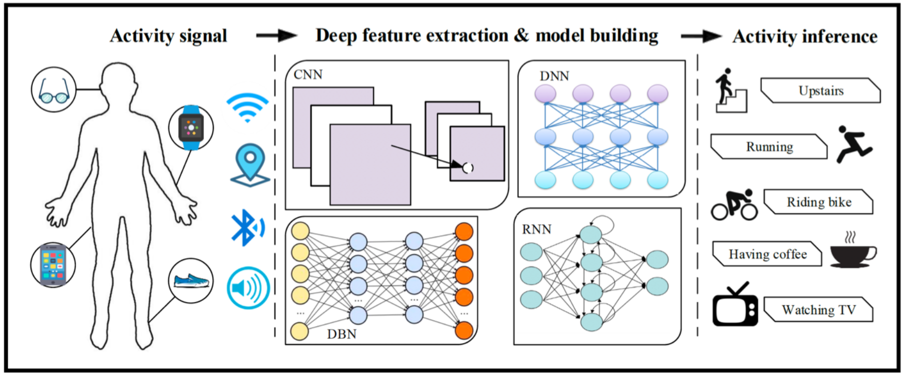
- **Hyperparameters:**
  - **Kernel:** defines the filter-size, (e.g 3x3, 5x5x3)
  - **stride**: defines how many fields get shifted
  - **padding:** defines whether a frame of zeros should be added.



0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	10	1	2	5	3	0	0
0	0	1	3	33	12	7	0	0
0	0	42	1	35	76	1	0	0
0	0	43	65	4	87	0	0	0
0	0	1	22	1	99	1	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Why Deep Learning for HAR?

- Deep Learning works fine for computer vision or text analysis
- Tremendous progress for conventional PR and ML approaches within the last years, but as always leaves room for improvements.
- in PR and classical ML features are hand-crafted.
  - Features depending on human experience
  - Only shallow features can be learned (e.g. mean, variance, ...).
  - Shallow features can only be used to recognize low level activities like running or walking. Drinking coffee for example is a more complex activity
- ML needs labeled data for initial training → Existing Deep Learning Networks are able to explore unlabeled data and use them for learning.
- ML data is mostly static → but most real life applications are coming with streamed data → works better with DL



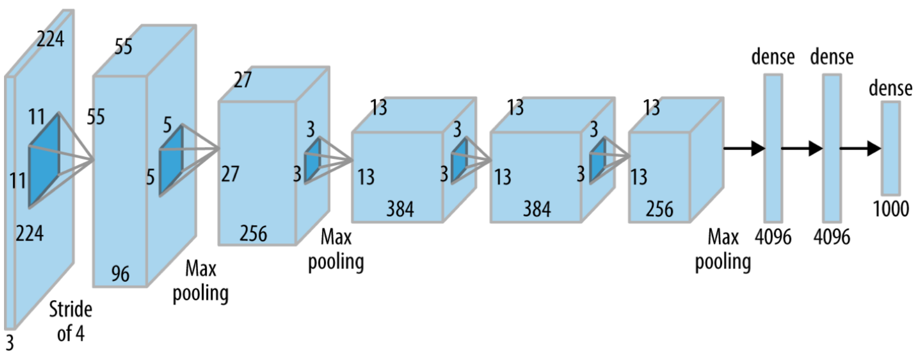
Sensor Modality

- Different types of Sensors are usable for (H)AR.
- We need to distinguish between:
  - Attached Sensors
    - Accelerometer, gyroscope, magnetometer
    - Can also be attached to objects
      - Accelerometer used as a vibration sensor attached to a industrial machine, RFID tags in smart home enviroments
  - Ambient Sensor
    - Light, temperature, humidity, pressure sensor, radar, ...
    - Often used for daily activity recognition in a smart home enviroment.
  - Hybrid Sensor
    - Acceleracion sensor combined with acoustic informations can improve HAR [Hayashi et al., 2015]
    - Combination of ambient and object sensors or body-worn Sensors is also widely used

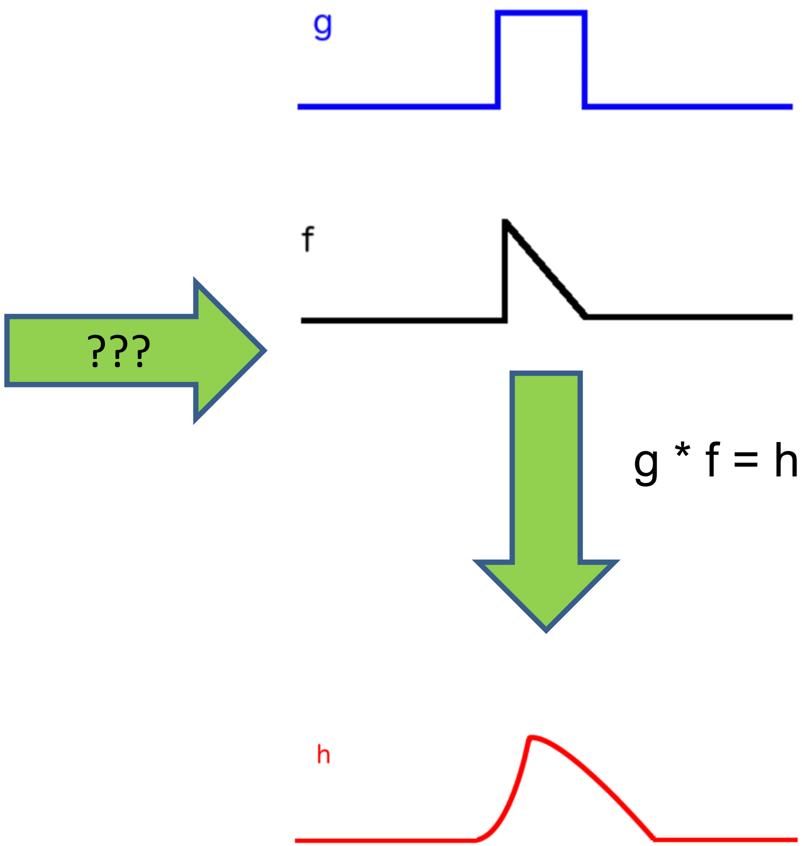
Recently published paper

- Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition
  - Francisco Javier Ordoñez* and Daniel Roggen
  - Published in Sensors 16(1), Special Issue "Advances on Data Transmission and Analysis for Wearable Sensors Systems“, 2016
  - doi: 10.3390/s16010115
  - Proposes a very efficient CNN Architecture for Activity Recognition with IMU-data.

- Question: **Is there an equivalent for a 2D convolution in 1D for time series data?**

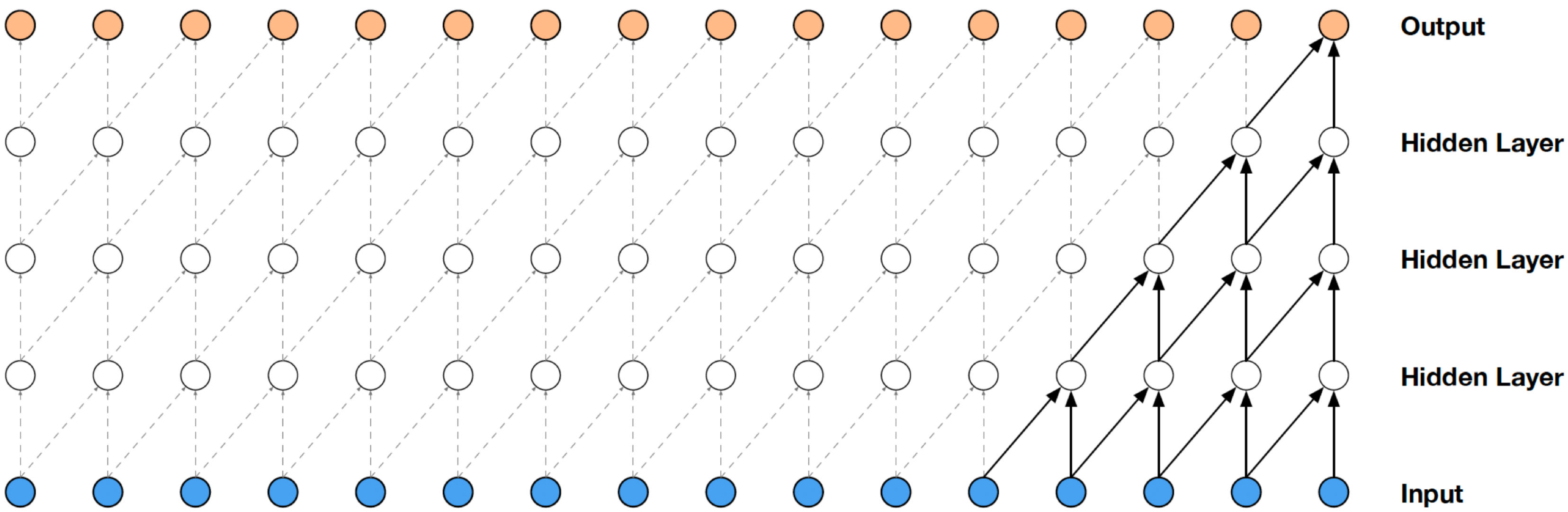


AlexNet – Architecture



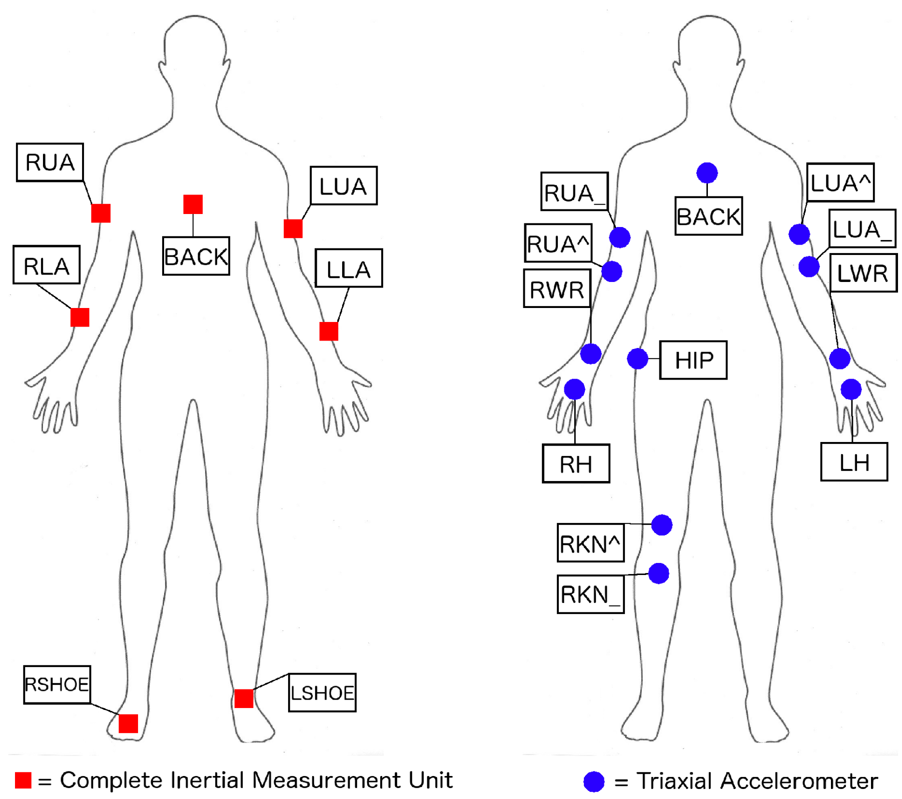
Example: Convolute over windows

- Architecture of WaveNet, a neural network for generating raw audio waveforms [Oord_2016]



- Only conv. Layers are used
- 3 convolution layers
- 5 windows at input convolute to 1 output neuron
- Input size is the same as output size
- Definition: $p(x) = \prod_{t=1}^{T} p\left(x_t \mid x_1, \dots, x_{t-1}\right)$

OPPORTUNITY Dataset

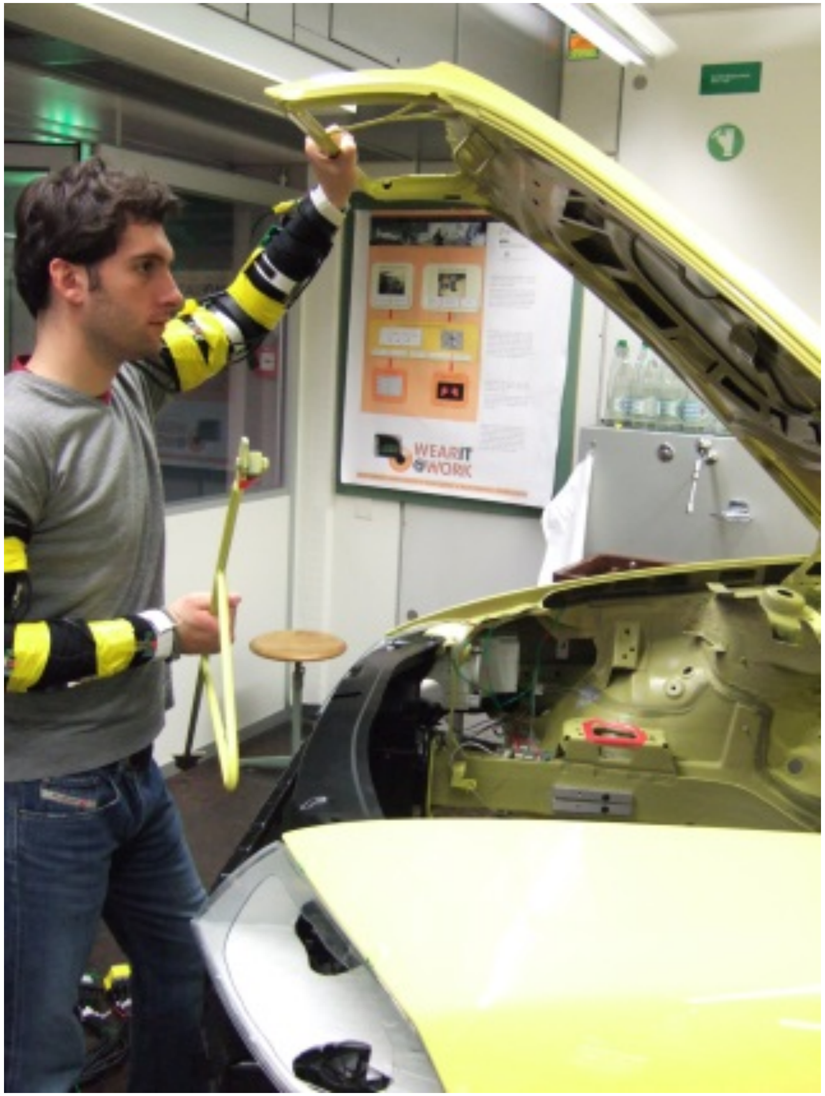


- 4 subjects performing morning activities of daily living
- created in a laboratory

- 16 activities with 20 repititions (gestures and modes of locomotion)
- Includes null class (void, trash)

- IMUs + triaxial accelerometer
- 5 commercial IMUs located in a custom-made jacket (red dots)
- 2 IMU, one placed on each foot (red dots)
- 12 acc-sensors placed on the limbs (blue dots)
- 113 Channels of input space
- 32 Hz sampling rate

Skoda Checkpoint Dataset



- Activities of assembly-line workers in a car production environment
- 1 subject
- 20 3D-accelerometers on both arms
- 10 gestures,
- 3 hours of recording of 70 repetitions per gestures
- No null class

- Data used for evaluation is restricted to 10 sensors placed on the right arm.

Overview Datasets

Dataset	#Subjects	Type	S.Rate	#Activities	#Samples	Sensors	Reference
OPPORTUNITY	4	ADL	32 Hz	16	701.366	A,G,M,O,AM	Ordóñez et al., 2016
Skoda Checkpoint	1	Factory	96 Hz	10	22.000	A	Plötz et al., 2011

(A=accelerometer, G=gyroscope, M=magnetometer, O=object sensor, AM=ambient sensor)

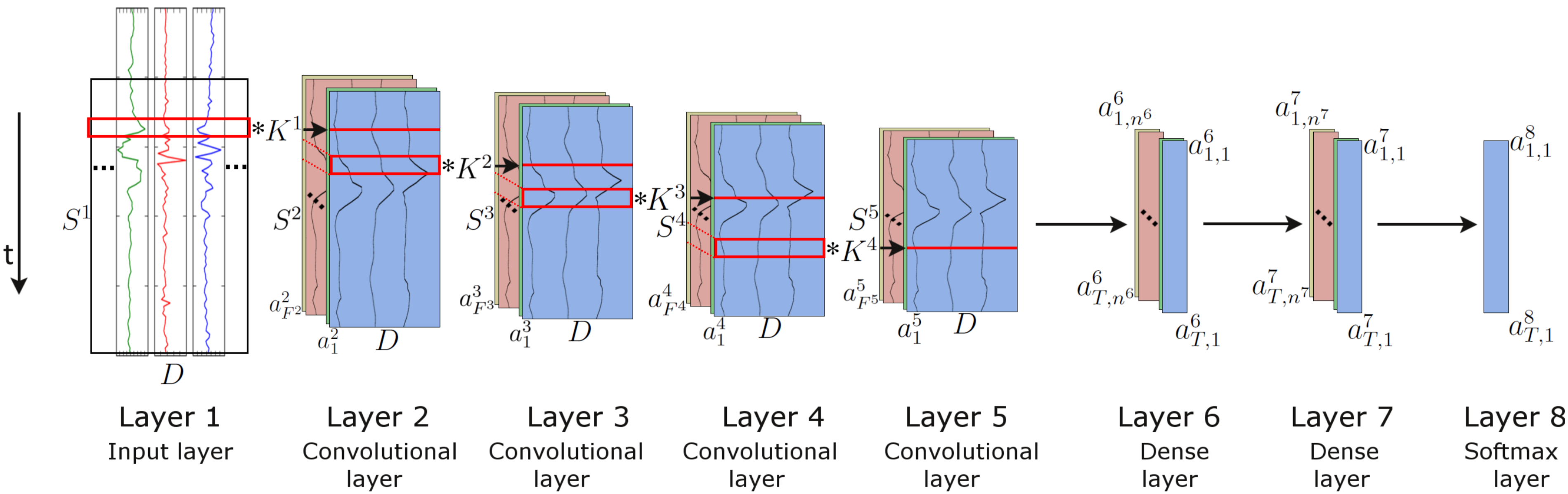
OPPORTUNITY						Skoda		
Gestures			Modes of Locomotion					
Name	# of Repetitions	# of Instances	Name	# of Repetitions	# of Instances	Name	# of Repetitions	# of Instances
Open Door 1	94	1583	Stand	1267	38,429	Write on Notepad	58	20,874
Open Door 2	92	1685	Walk	1291	22,522	Open Hood	68	24,444
Close Door 1	89	1497	Sit	124	16,162	Close Hood	66	23,530
Close Door 2	90	1588	Lie	30	2866	Check Gaps Door	67	16,961
Open Fridge	157	196	*Null*	283	16,688	Open Door	69	10,410
Close Fridge	159	1728				Check Steering Wheel	69	12,994
Open Dishwasher	102	1314				Open and Close Trunk	63	23,061
Close Dishwasher	99	1214				Close both Doors	69	18,039
Open Drawer 1	96	897				Close Door	70	9783
Close Drawer 1	95	781				Check Trunk	64	19,757
Open Drawer 2	91	861						
Close Drawer 2	90	754						
Open Drawer 3	102	1082						
Close Drawer 3	103	1070						
Clean Table	79	1717						
Drink from Cup	213	6115						
Toggle Switch	156	1257						
Null	1605	69,558						

- Many more datasets are available, most of them from activities of daily life (ADL)
- [Wang_2017] gives a good overview about the available datasets

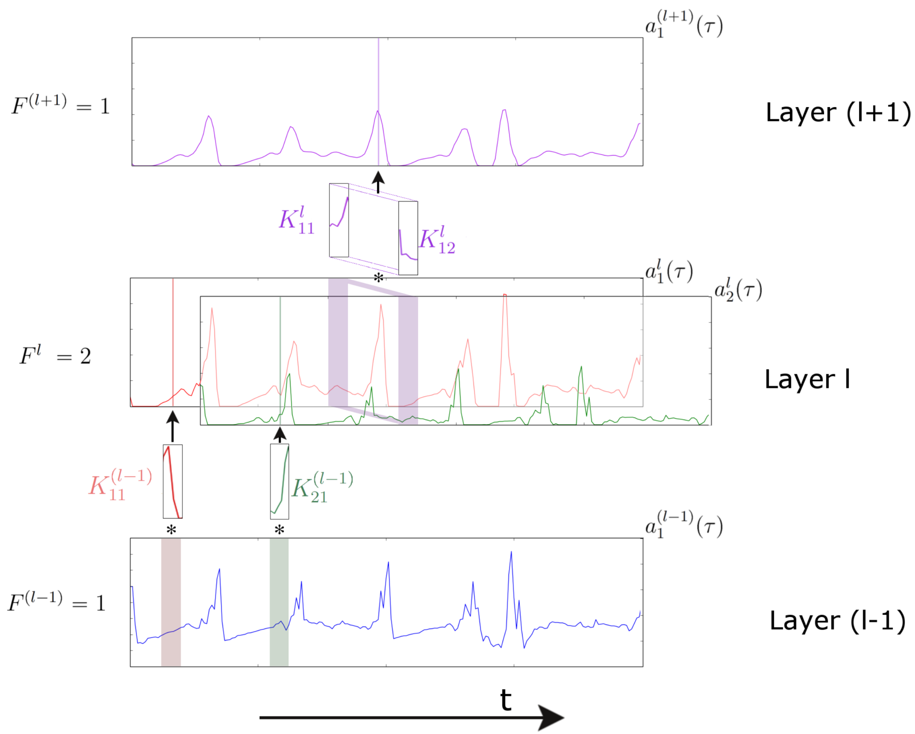
Comparison of used Datasets

- Opportunity Dataset:
  - i.e. 79 examples of cleaning table, but 213 examples of drinking from cup.
    - Dataset seems to be unbalanced
      - Gets countered by weighting classes according to their sample proportions
  - 1605 examples of null class
- Skoda Dataset:
  - Balanced amount of repititions per activity but a null class is missing
  - Question: Is a null class important?

Ordóñez et. al 2016: Network Architecture



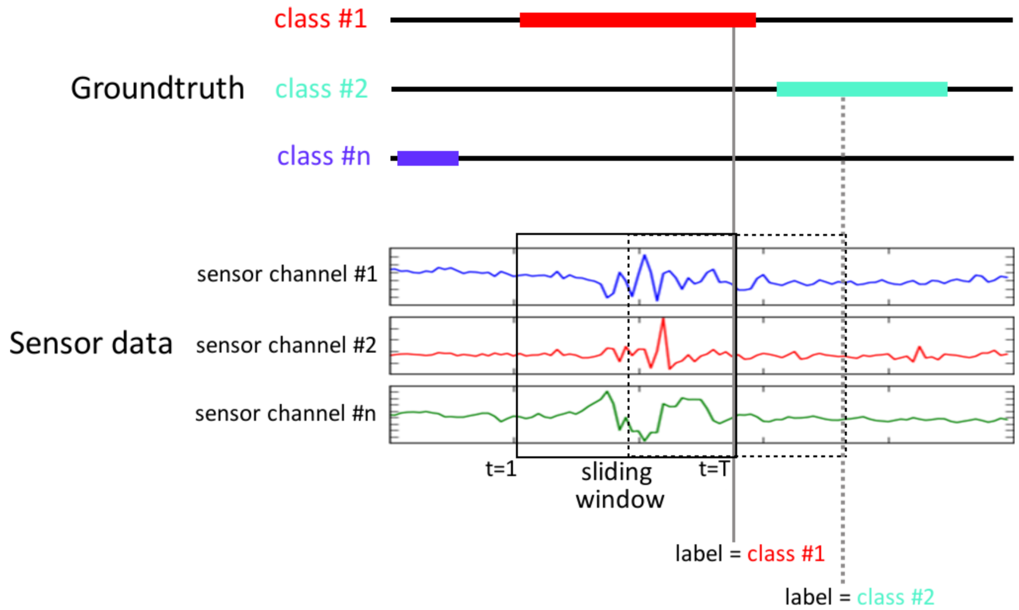
- Evaluated a Baseline CNN vs DeepConvLSTM net
  - Both share the same architecture (for comparison)
  - In the recurrent network the dense layers have the LSTM cells
- Input layer, 4 convolutional layers (64 convolutions), 2 (recurrent) dense layers with 128 LSTM cells, softmax classifier.
- Vertical time axis t
- Model is trained supervised
- Same number of sensor channels D for every feature map F^l .
- S^l length of feature map in layer *l*
- K^l kernel of layer *l.*
- a_i^l denotes the activation for feature map i in layer l
- $a_{T,n}^l$ denotes to the activation of the recurrent unit n at time T



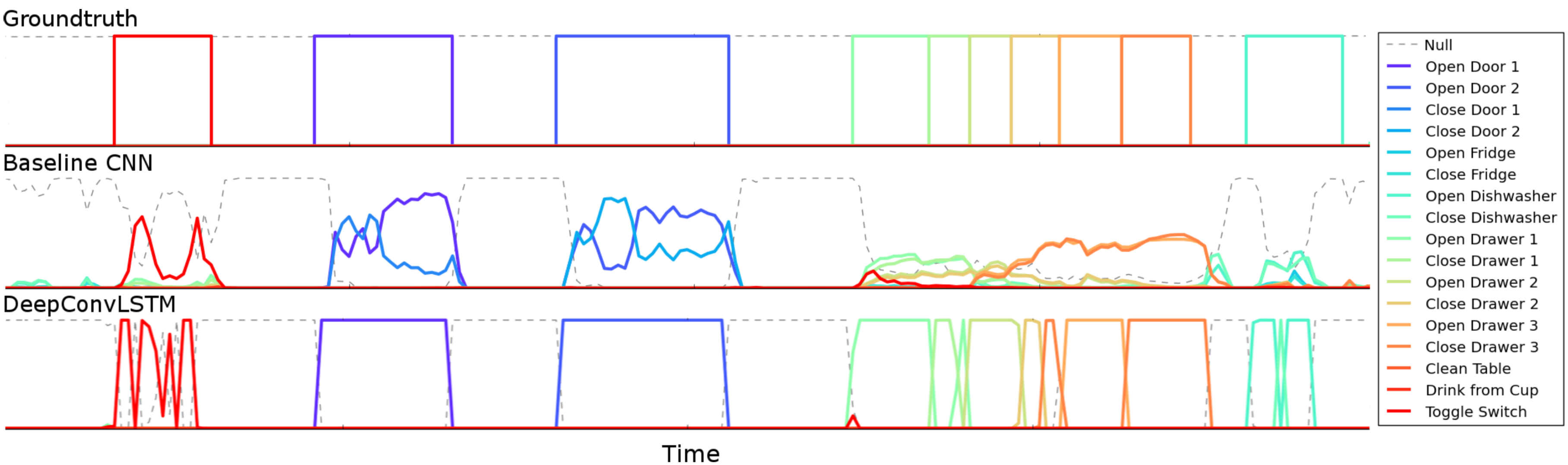
- Temporal convolution over n batches.
- Sliding window with 500ms size and 250ms step size → 50% overlap
- F defines the feature map (2D time series)
- l defines level of layer
- l-1 is the data at the input layer
- l+2 is the data at next or output layer
- $K_{11}^{(l-1)}$ and $K_{21}^{(l-1)}$ are two different kernels that are detecting 2 different features.
  - $K_{11}^{(l-1)}$ detects minima
  - $K_{21}^{(l-1)}$ detects maxima
- Deeper Layers create more feature maps through convolutional filter
- Last convolution merges all feature maps into a dense layer

Classification

- Label of the last sample in the window is label of the class
- How much sense does this make?
- DeepConvLSTM should predict class probality after observing whole window, therefore several approaches exist:
  - Window gets classified by last sample of 500ms sequence → every 250ms a label
  - Max-pooling the prediction of the sequence
  - Summing all of the sequence prediction over time and returning the most frequent



Basic CNN vs. DeepConvLSTM



DeepConvLSTM provides a better identifying of start and beginning of gesture

- DeepConvLSTM outperforms common used CNN-Architectures.
  - $F_1 - Score$: **Baseline CNN** 0.884 (Skoda Dataset), 0.883 (OPPORTUNITY)
  - $F_1 - Score$: **DeepConvLSTM** 0.958 (Skoda Dataset), 0.915 (OPPORTUNITY)
  - *Training and Testing Time (GPU) (OPPERTUNITY):*
    - **BaselineCNN** 282.2 min, 5.43 sec and **DeepConvLSTM** 340.3 min, 6.68 sec
- Even longer gestures has been successfully been classified.
- How is this possible with a window length of 500ms?
  - They speculate that this is due to the fact that longer gestures consists of several shorter characteristic patterns.
  - It seems that DeepConvLSTMs are able to classify the gesture even without a complete view of it.

Ordóñez et al 2016: Benefits of the paper

- First paper that successfully introduced a mix of Convolution and LSTM-Layer for HAR.
- Demonstrated that LSTM-Cells improve the recognition of HAR-tasks. Due to their memory they are able to learn temporal dynamics
- This is fundamental to distinguish gestures of similar kind

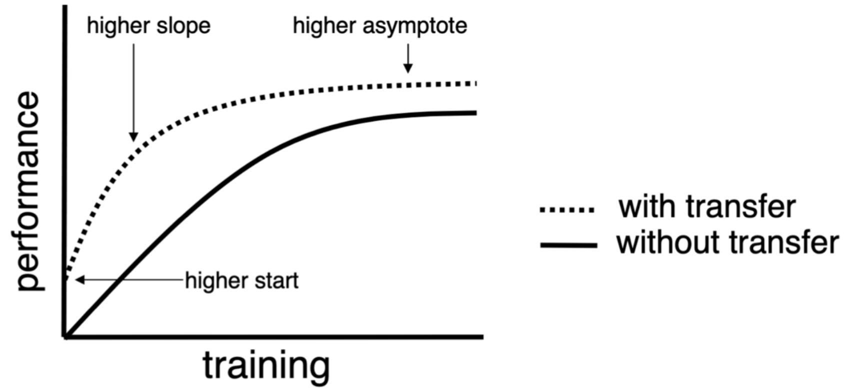
Code is available at https://github.com/sussexwearlab/DeepConvLSTM

Transfer Learning

Transfer Learning

- ML trains every model isolated and depending on his task
- Transfer Learning means to reuse a model for other related tasks

- TL brings several improvements:
  - Improved baseline performance
  - Decreased model-development-time
  - Improved final performance



Possible benefit of transfer learning (Source: Torrey_2009)

Transfer Learning

From Handball

To

Basketball

Fundamentals

Before we can start with transfering models we need to answer the following questions:

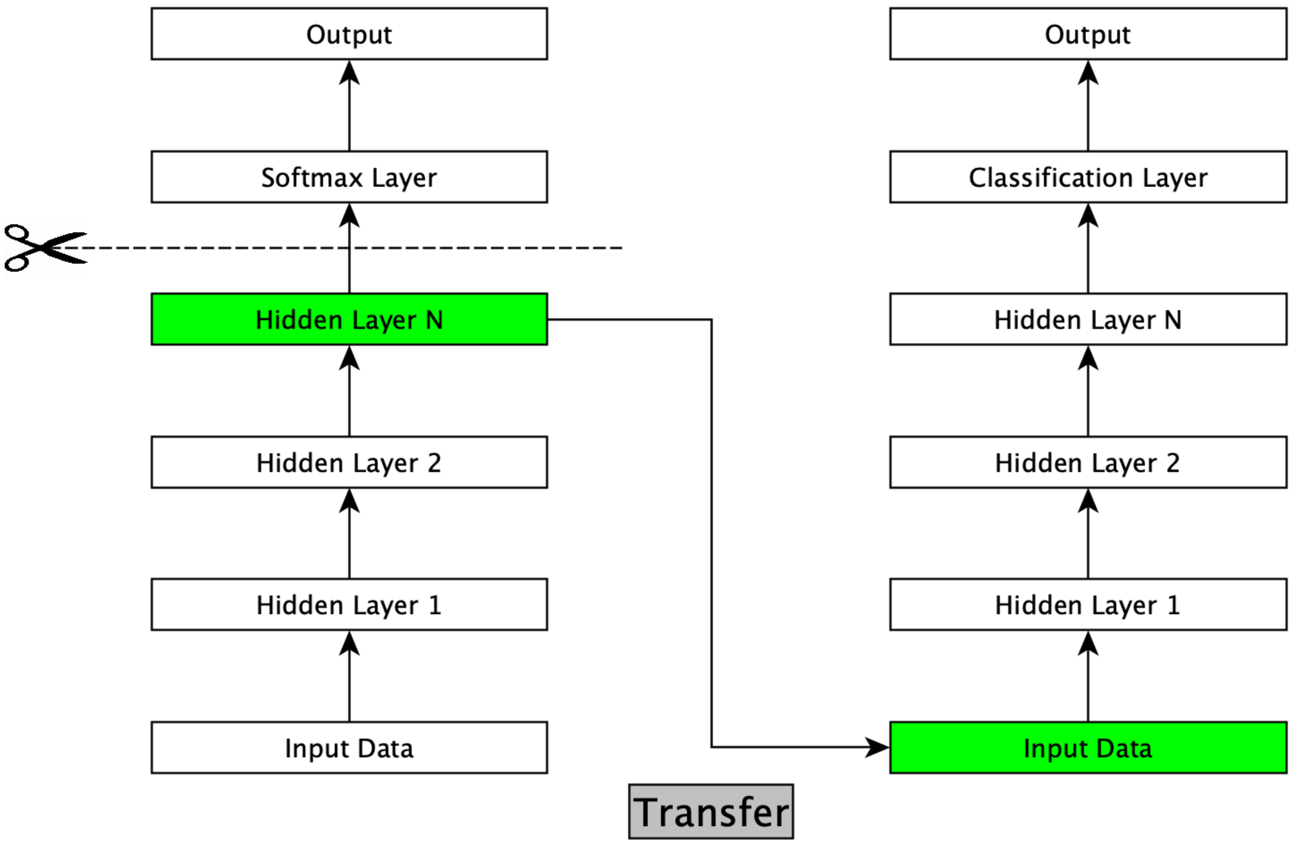
- What to transfer?
  - Instance transfer
  - Feature-representation transfer
  - Parameter transfer
  - Relational-knowledge transfer
- When to transfer?
  - As the last step before the output layer?
  - Between the hidden layers?
- How to transfer?
  - Inductive transfer
  - Unsupervised transfer
  - Transductive transfer

What can be transfered?

- Instance transfer:
  - source domain data cannot be reused directly.
  - but several instances of trained models are reused to improve the classification results.
  - According to [Wang_2018] the similarity between source and target domain needs to be measured.
- Feature-representation transfer:
  - Approach identifies usefull feature representations from the source domain. These features can be reused in the target domain.
- Parameter transfer:
  - Source and target domain share some parameters, that need to be identified and transfered.
- Relational-knowledge transfer:
  - Knowledge from data that is not independent and identically distributed can be transfered. Social Networks are working massively. with this approach

When do we transfer?

For example:



How do we transfer?

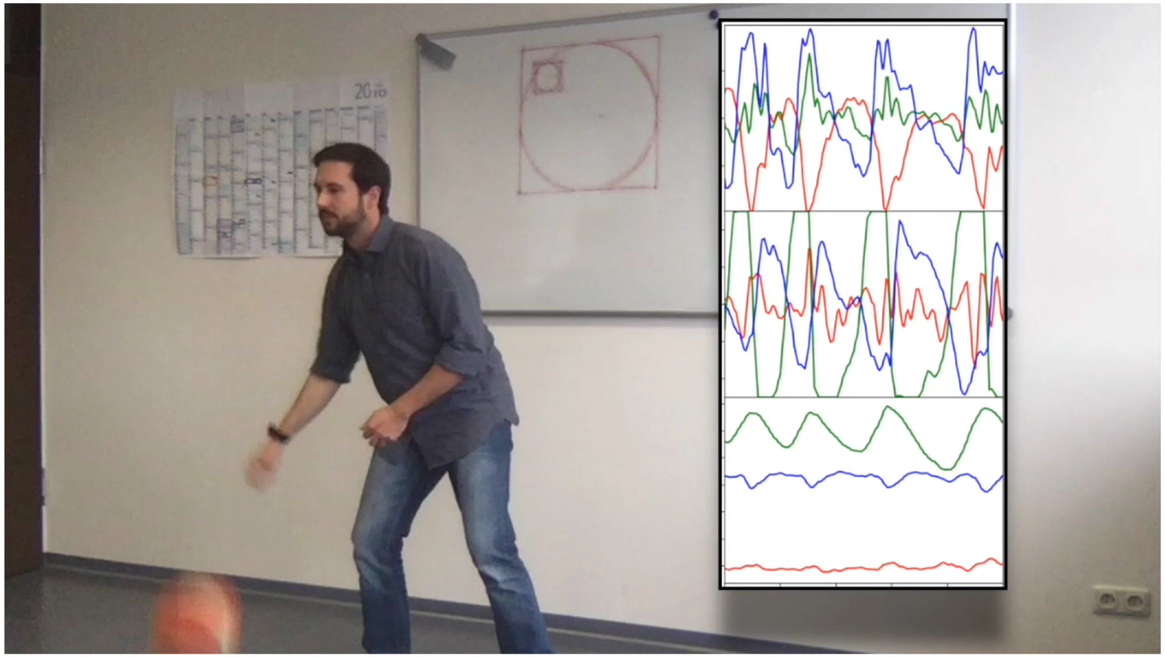
- Inductive Learning:
  - Source and Target domain are the same
  - Algorithm utilize the biases of the source domain to help improve the target task
  - I.e. multitask learning, self-taught learning
- Unsupervised Learning
  - Similar to Inductive Learning, but with focus on unsupervised learning techniques
- Transductive Learning
  - Similarities between tasks, but from different domains
  - Source domain has labeled data, but target domain has none.

Challenges of transfer learning

- Negative transfer:
  - Refers to a scenario where transfer learning can lead to a drop of performance.
  - A reason can be that the relationship between source and target domain is not close enough.
- Transfer bounds
  - To gauge the amount of transfer it is important to know what exactly and how much knowledge should be transfered.
    - For example [Mahmud_2007] tries to find out these bounderies based on the Kolmogorov complexity

Basketball Activity Recognition

- 5 classes:
  - low dribbling
  - high dribbling
  - crossover
  - (jump shot)
  - void-class



Current Research:

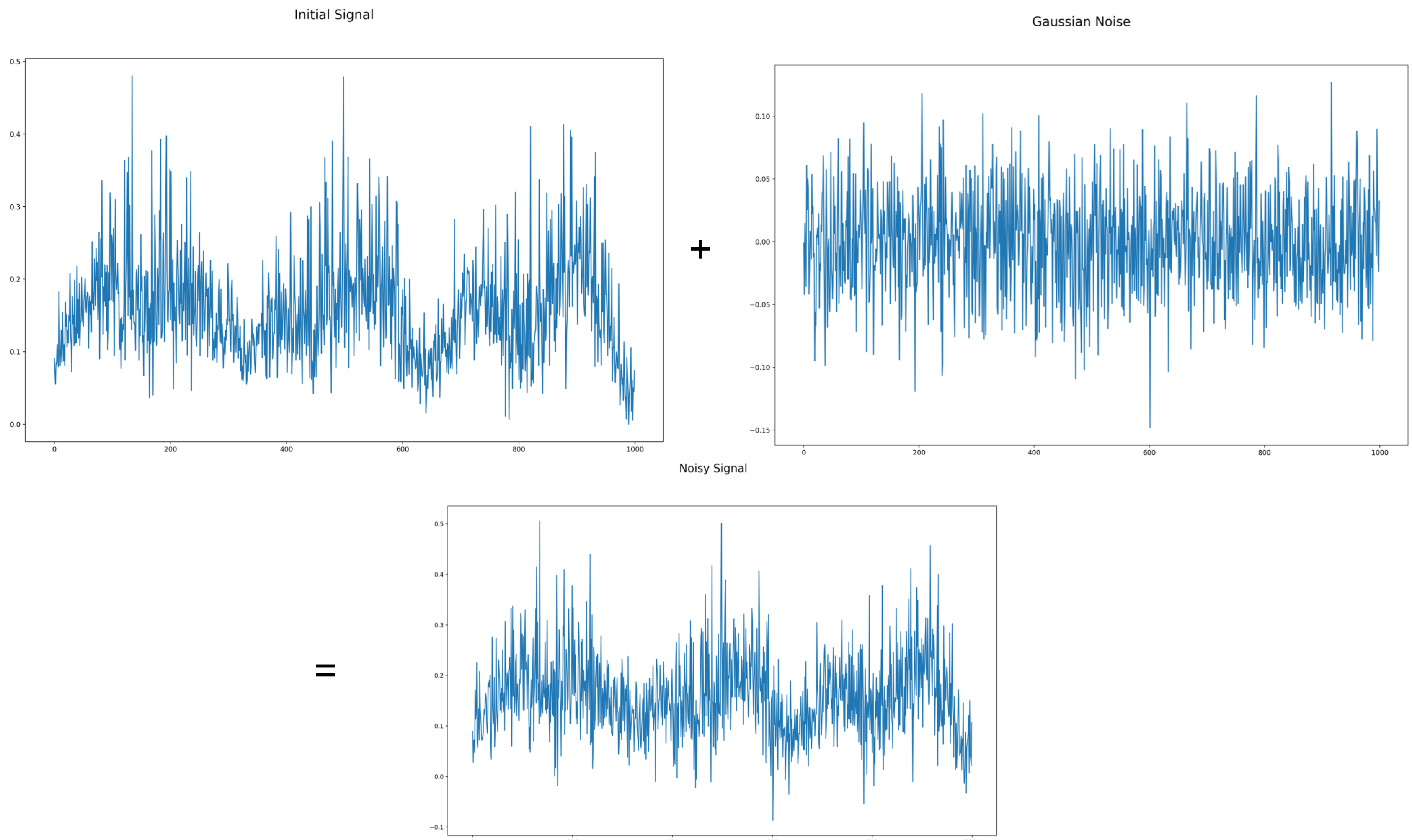
Transferring a model that was trained with lab data, so that it recognizes activities recorded from players in a real environment.

Data Augmentation

Data Augmentation

- Common problem for Neural Networks is, that you need huge datasets
- Techniques for increasing data are:
  1. Oversampling
     - Copying the data
  2. for images you can flip, rotate, crop, translate, change colors, scale the data and so on...
     - According to [Um_2017] also partially applicable on sensor based data
  3. Add **gaussian noise** on the data. That means a randomly assigned noise with mean 0
  4. Interpolating or extrapolating the data

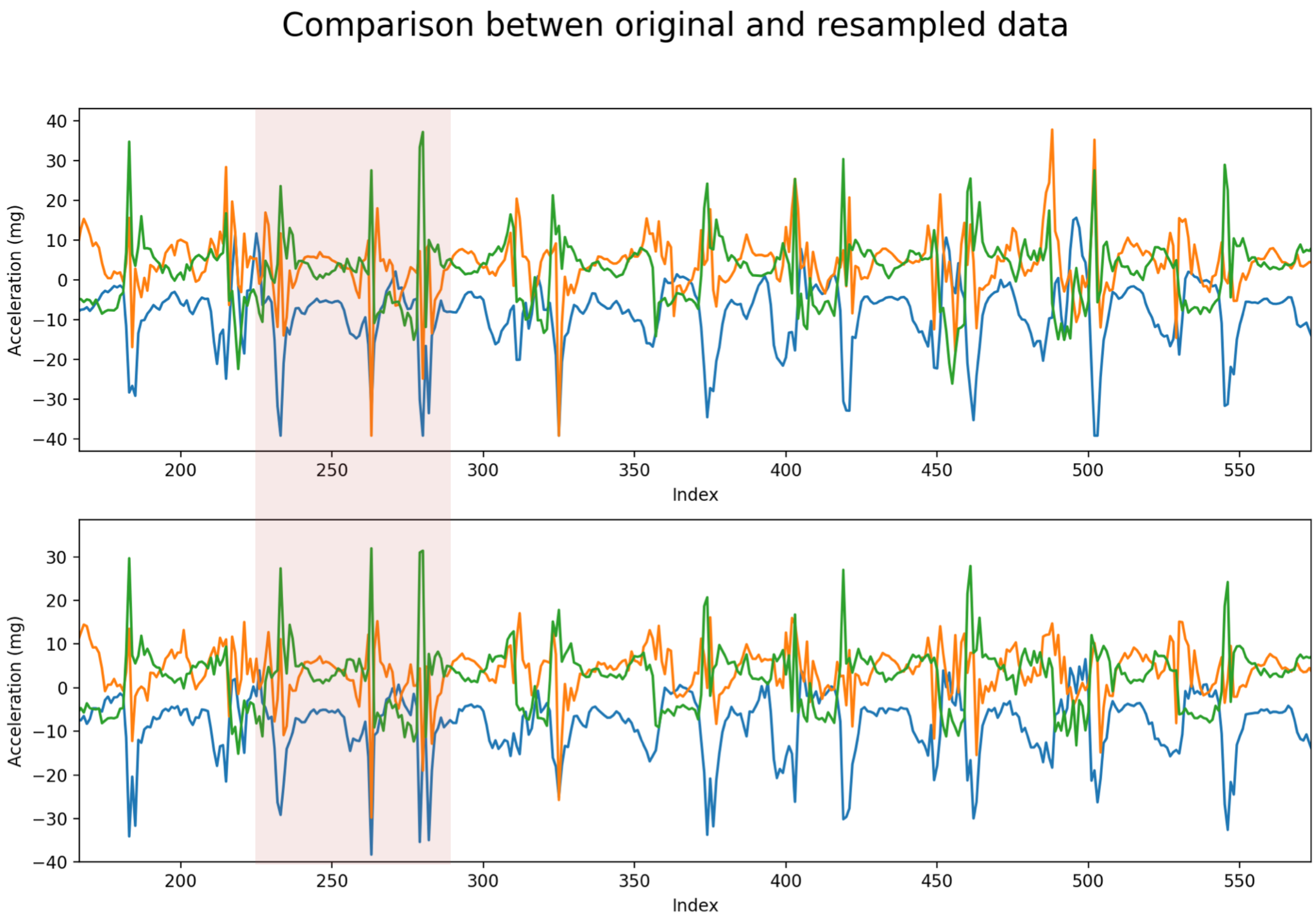
Data Augmentation for HAR: Gaussian Noise



- Introduced by Chawla et. Al, 2002 - SMOTE: Synthetic Minority Oversampling Technique
  - $c_i = (c_k - c_j)\,\lambda + c_j$
  - $c_e = (c_j - c_k)\,\lambda + c_j$
    - $0.0 < \lambda < 1.0$
- The result will be a new signal very similar to the original signal.
  - High sampling rate results in a noisier signal
  - Downsampling before resampling results in a smooth signal

Data Augmentation

Resampled with n = 6 and lambda = 0.5



Data Augmentation

Resampled with n = 6 and lambda = 0.5

Future Challenges

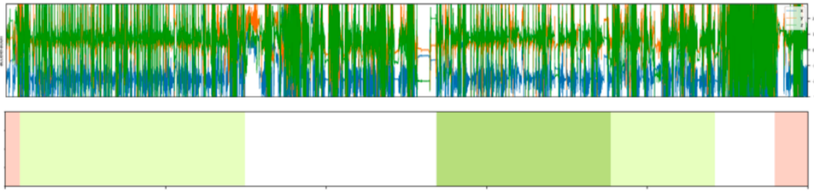
- Using neural networks on real sensor based data in real life situations and working enviroments
- Implementing power saving ml-algorithms for mobile devices or embedded hardware
- Combining hand-crafted and deep learning features
- Using context based information for improving classification methods
- Transfer learning on sensor based data

Project Proposal

Implement a CNN with our basketball activity dataset

Optional: Implement transfer learning.

- For example transferring a trained model of known activities to recognize a yet unknown activity.
  - Can we use the knowledge of how fast dribbling looks like during charge to recognize a slower dribbling frequency?



References

- Ordonez_2016: Ordóñez FJ, Roggen D. **Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition**. Sensors [Internet]. 2016;16(1). Available from: http://www.mdpi.com/1424-8220/16/1/115
- Torrey_2009: L. Torrey & J. Shavlik (2009). Transfer Learning. In E. Soria, J. Martin, R. Magdalena, M. Martinez & A. Serrano, editor, Handbook of Research on Machine Learning Applications. IGI Global. Hayashi_2015:
- Mahmud_2007: Mahmud MMH, Ray SR. **Transfer Learning Using Kolmogorov Complexity**: **Basic Theory and Empirical Evaluations**. In: Proceedings of the 20th International Conference on Neural Information Processing Systems [Internet]. USA: Curran Associates Inc.; 2007. p. 985–992. (NIPS'07). Available from: http://dl.acm.org/citation.cfm?id=2981562.2981686
- Oord_2016: 1. Oord A van den, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, et al. **WaveNet: A Generative Model for Raw Audio**. arXiv:160903499 [cs] [Internet]. 2016 Sep 12 [cited 2019 Apr 13]; Available from: http://arxiv.org/abs/1609.03499
- Wang_2017: Wang J, Chen Y, Hao S, Peng X, Hu L. **Deep learning for sensor-based activity recognition: A survey**. Pattern Recognition Letters. 2019 Mar;119:3–11.
- Um_2017: Um TT, Pfister FMJ, Pichler D, Endo S, Lang M, Hirche S, et al. **Data Augmentation of Wearable Sensor Data for Parkinson's Disease Monitoring using Convolutional Neural Networks**. Proceedings of the 19th ACM International Conference on Multimodal Interaction - ICMI 2017. 2017;216–20.
- Hölzemann_2018: Hölzemann A, Van Laerhoven K. **Using Wrist-Worn Activity Recognition for Basketball Game Analysis**. In: Proceedings of the 5th international Workshop on Sensor-based Activity Recognition and Interaction - iWOAR '18 [Internet]. Berlin, Germany: ACM Press; 2018 [cited 2018 Oct 10]. p. 1–6. Available from: http://dl.acm.org/citation.cfm?doid=3266157.3266217
- Wang_2018: Wang T, Huan J, Zhu M. **Instance-based Deep Transfer Learning**. arXiv:180902776 [cs] [Internet]. 2018 Sep 8 [cited 2019 May 15]; Available from: http://arxiv.org/abs/1809.02776