EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

FACULTY OF
SCIENCE
Computer Graphics

Machine Learning in Graphics and Vision

- Advanced Topics in Deep Learning -

SoSe 2018

Hendrik Lensch

Overview

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Training of DNNs
- Backpropagation
- Optimizer: AdaGrad, RMSProp, Adam

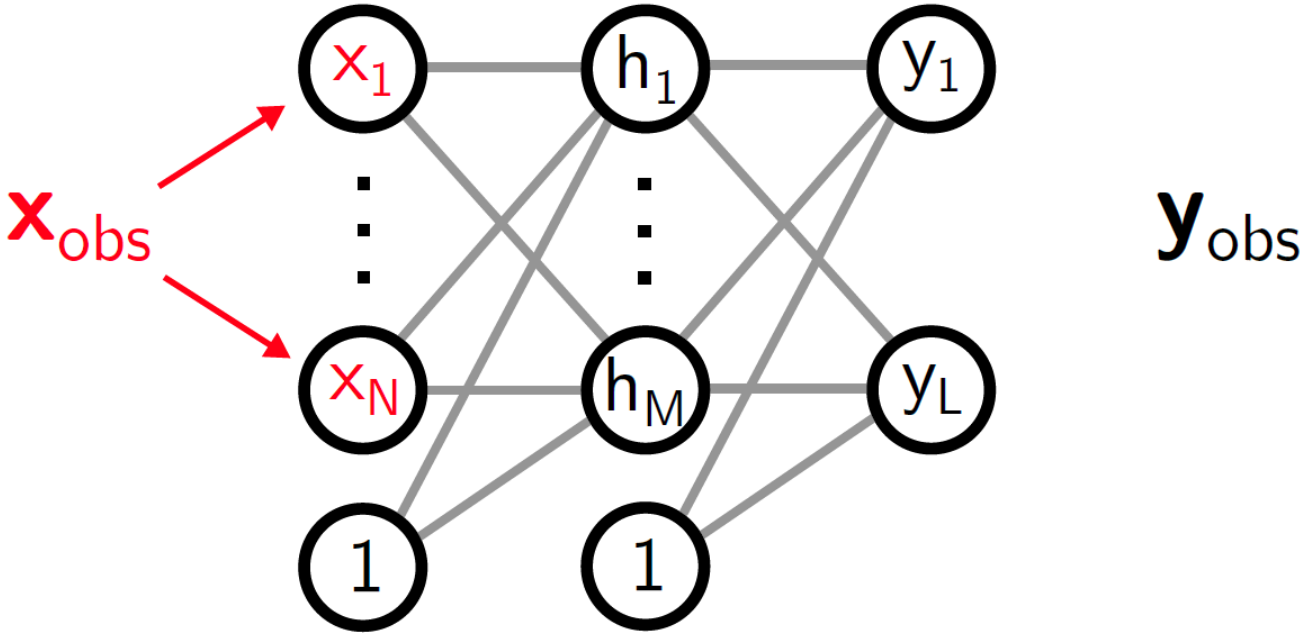
Processing Sequences
- RNN
- LSTM
- GRU
- Temporal Conv
- Video + DNNs
- 3D Convolution
- Graph Conv

Backpropagation

Algorithm for training a neural network using SGD:

1. Initialize $\mathbf{W}$, pick learning rate η and mini batch size M
2. Randomly pick M data points $\mathcal{X}_{\text{batch}} \subset \mathcal{X}$
3. For each data point $\mathbf{x}_m \in \mathcal{X}_{\text{batch}}$ do:
   - 3.1 Forward propagate $\mathbf{x}_m$ through network: $y_j = \sigma\left(\sum_i w_{ji} y_i\right)$ (here: $\{y_0\} = \mathbf{x}_m$)
   - 3.2 Evaluate errors δ_j at output nodes: $\delta_j = (y_j - t_j) y_j (1 - y_j)$
   - 3.3 Backpropagate errors δ_i through network: $\delta_i = y_i (1 - y_i) \sum_j \delta_j w_{ji}$
   - 3.4 Calculate derivatives: $\frac{\partial E_m}{\partial w_{ji}} = \delta_j y_i$
4. Update gradients: $\mathbf{W}^{t+1} = \mathbf{W}^t - \eta \sum_{m=1}^{M} \nabla_{\mathbf{W}} E_m |_{\mathbf{W}^t}$
5. If validation error decreases, go to step 2, otherwise stop

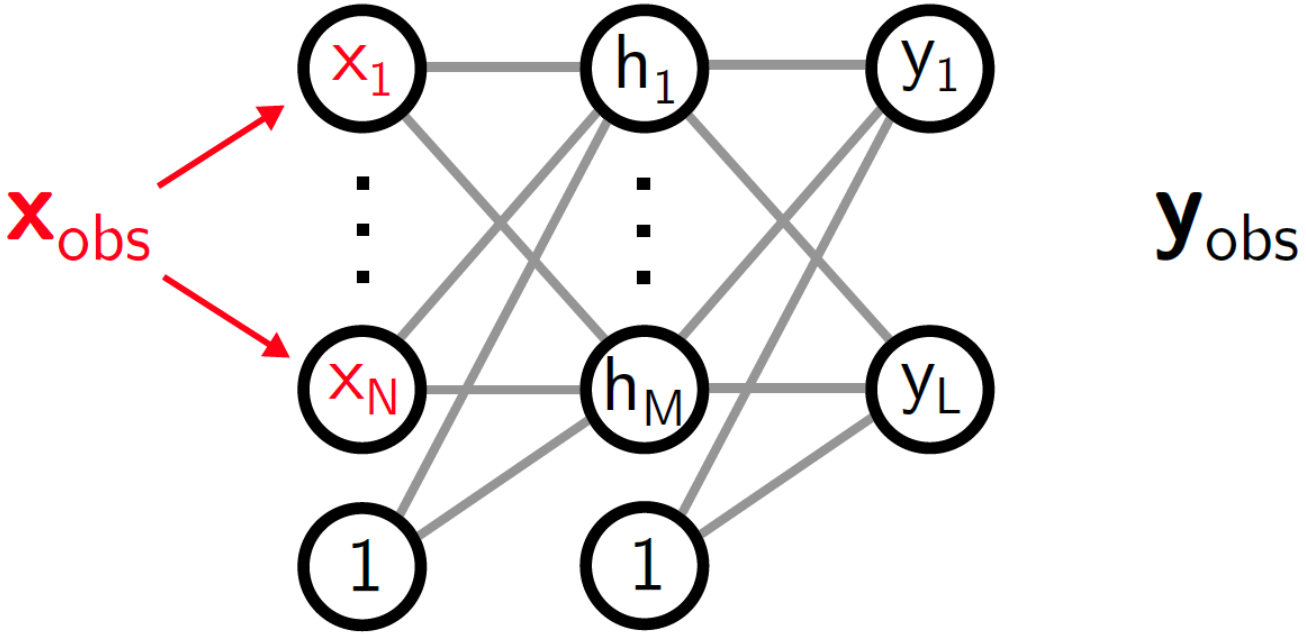
Forward Pass



[D. Rumelhart, G. Hinton and R. Williams: Learning representations by back-propagating errors. Nature, 1986]

- Forward pass: resembles processing in the brain
- Backward pass: relationship to neural learning less clear / less well understood

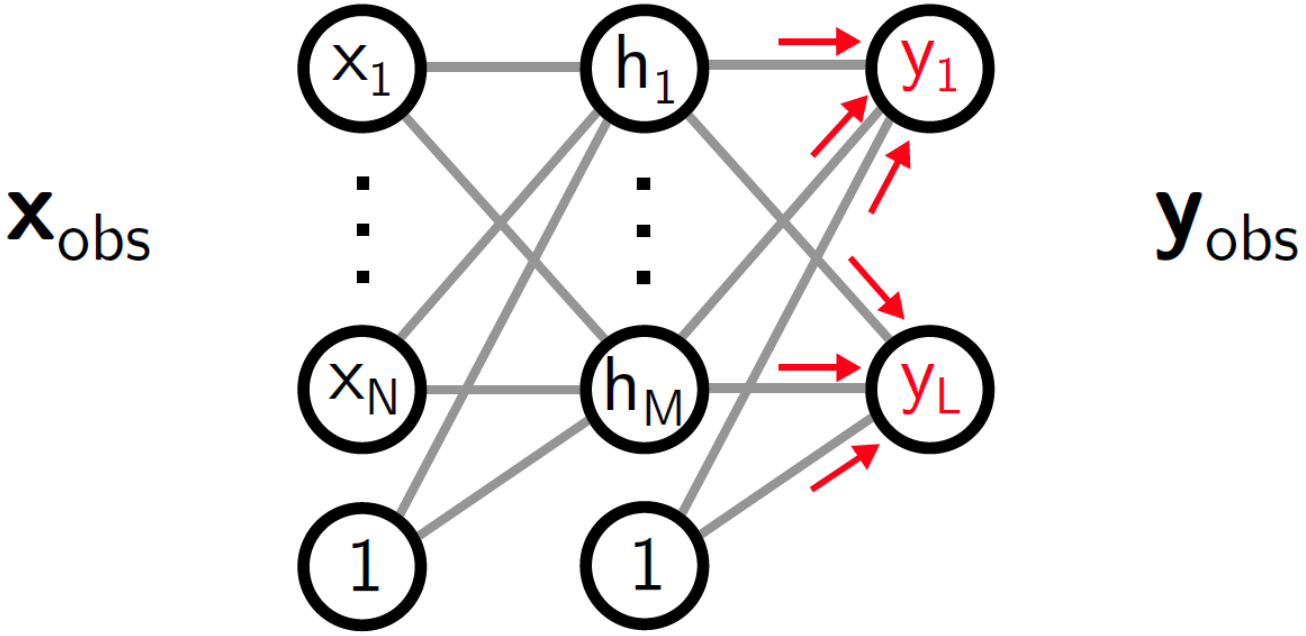
Forward Pass



[D. Rumelhart, G. Hinton and R. Williams: Learning representations by back-propagating errors. Nature, 1986]

- Forward pass: resembles processing in the brain
- Backward pass: relationship to neural learning less clear / less well understood

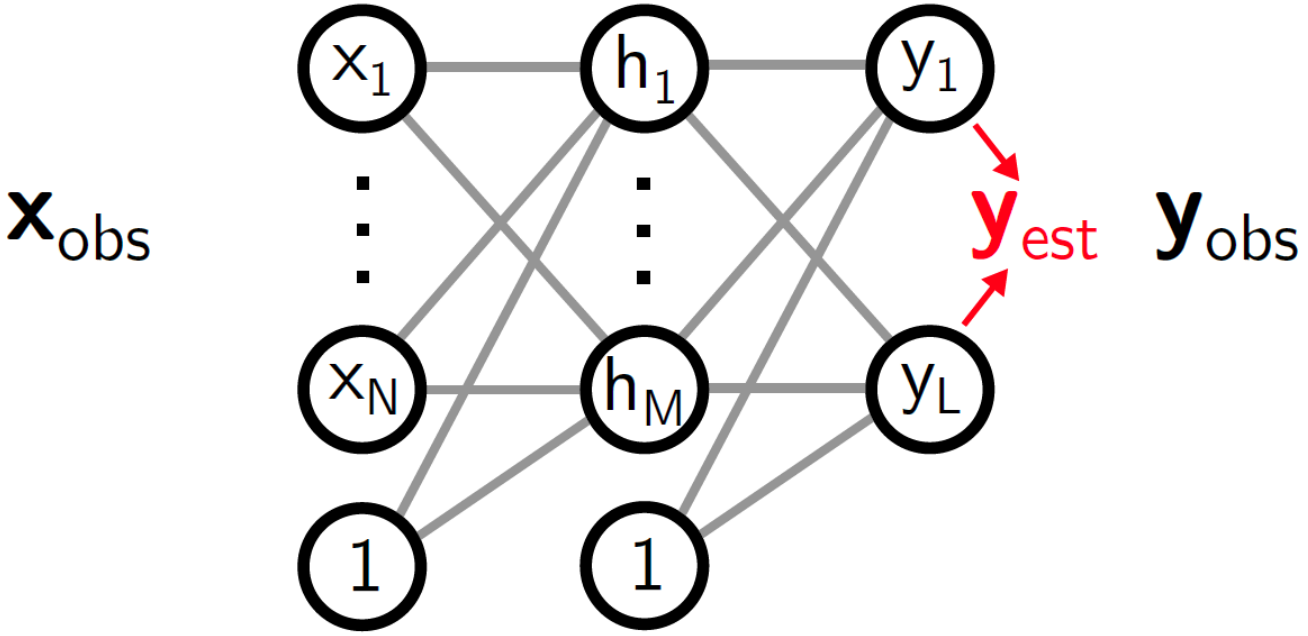
Forward Pass



[D. Rumelhart, G. Hinton and R. Williams: Learning representations by back-propagating errors. Nature, 1986]

- Forward pass: resembles processing in the brain
- Backward pass: relationship to neural learning less clear / less well understood

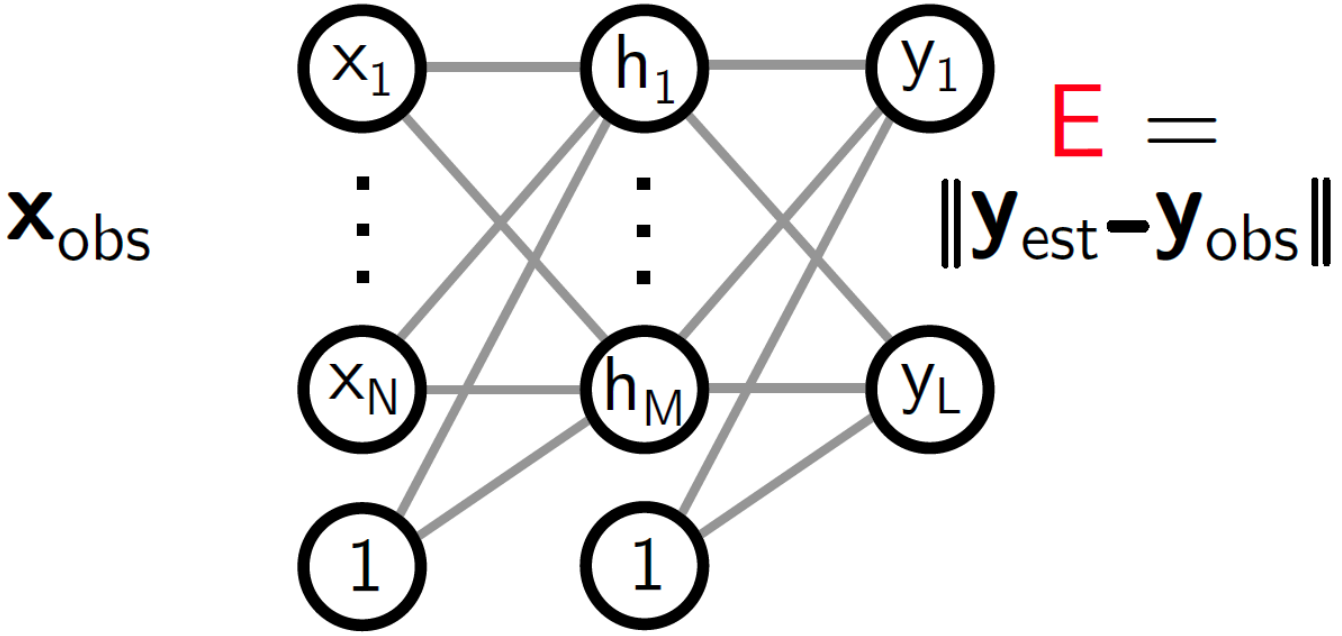
Forward Pass



[D. Rumelhart, G. Hinton and R. Williams: Learning representations by back-propagating errors. Nature, 1986]

- Forward pass: resembles processing in the brain
- Backward pass: relationship to neural learning less clear / less well understood

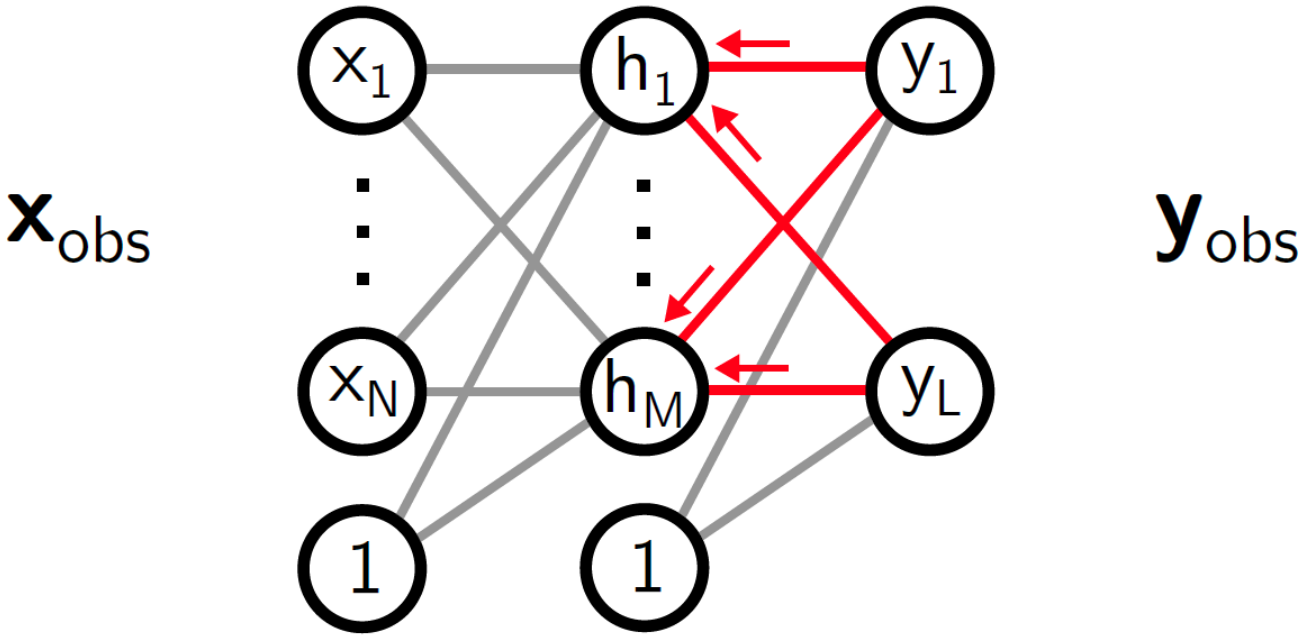
Error Calculation



[D. Rumelhart, G. Hinton and R. Williams: Learning representations by back-propagating errors. Nature, 1986]

- Forward pass: resembles processing in the brain
- Backward pass: relationship to neural learning less clear / less well understood

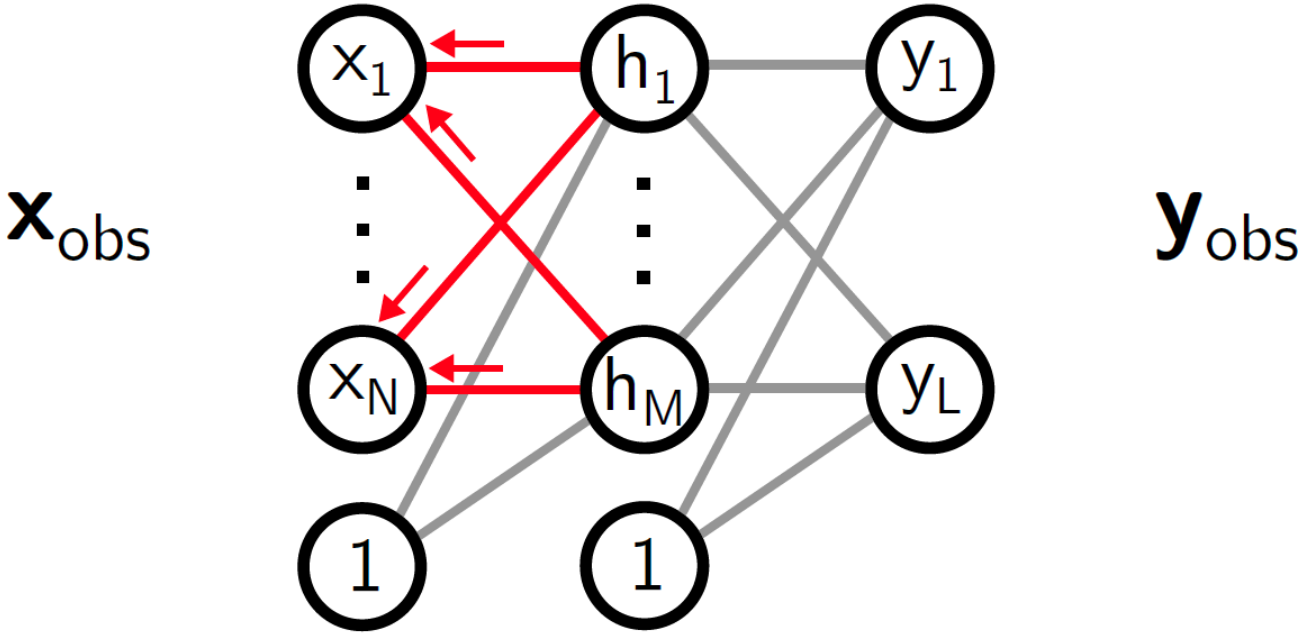
Error Backpropagation



[D. Rumelhart, G. Hinton and R. Williams: Learning representations by back-propagating errors. Nature, 1986]

- Forward pass: resembles processing in the brain
- Backward pass: relationship to neural learning less clear / less well understood

Error Backpropagation



[D. Rumelhart, G. Hinton and R. Williams: Learning representations by back-propagating errors. Nature, 1986]

- Forward pass: resembles processing in the brain
- Backward pass: relationship to neural learning less clear / less well understood

Backpropagation

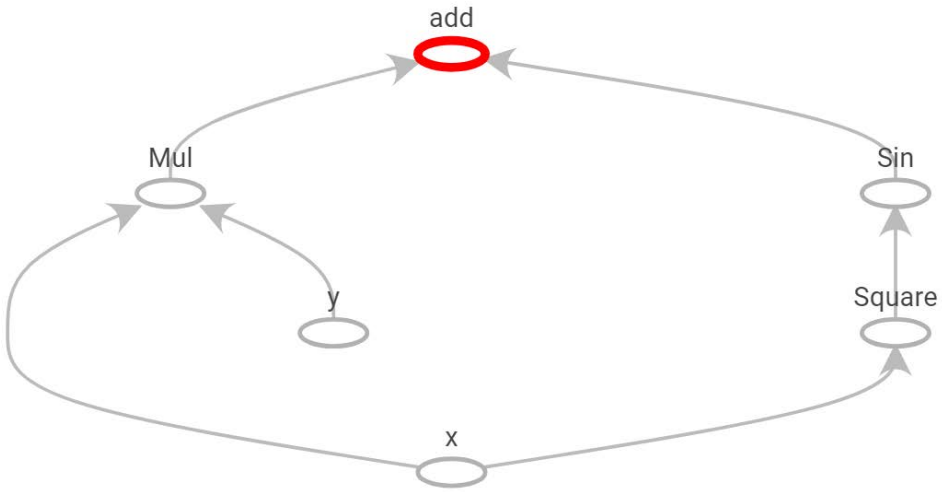
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Visualization
  - What is backpropagation really doing? | Chapter 3, deep learning video
  - Backpropagation calculus | Appendix to deep learning chapter 3 video

Auto-Diff/Backpropagation (TensorFlow)

Find derivative of $f(x, y) = x \times y + \sin(x^2)$ wrt. to x, y

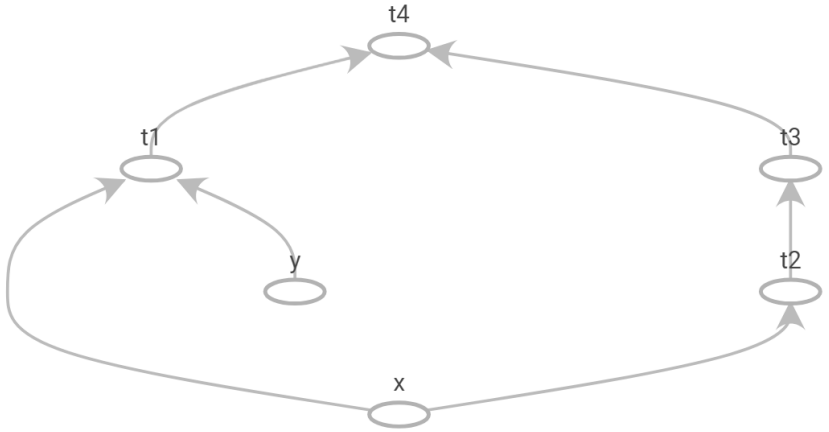
```
x = tf.placeholder(dtype=tf.float32, name="x")
y = tf.placeholder(dtype=tf.float32, name="y")

f = tf.multiply(x, y) + tf.sin(tf.square(x))

with tf.Session() as sess:
    f = sess.run(f, {x: 4., y:3.})
    print('f(%f, %f) = %f' % (4., 3., f))
```



Auto-Diff/Backpropagation (TensorFlow)

Let's rewrite

$$
\begin{aligned}
f(x,y) &= xy + \sin(x^2) \\
&= t_1 + \sin(x^2) && \text{using} \quad t_1 = xy \\
&= t_1 + \sin(t_2) && \text{using} \quad t_2 = x^2 \\
&= t_1 + t_3 && \text{using} \quad t_3 = \sin(t_2) \\
&= t_4 && \text{using} \quad t_4 = t_1 + t_3
\end{aligned}
$$




```
x = tf.placeholder(dtype=tf.float32, name='x')
y = tf.placeholder(dtype=tf.float32, name='y')

t1 = tf.multiply(x, y, name='t1')
t2 = tf.square(x, name='t2')
t3 = tf.sin(t2, name='t3')
t4 = tf.add(t1, t3, name='t4')

f = t4
```

Auto-Diff/Backpropagation (TensorFlow)

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

What is $\frac{\partial f}{\partial t_4}$?

Simply

$$\frac{\partial f}{\partial t_4} = \frac{\partial t_4}{\partial t_4} = 1$$

```
with tf.Session() as sess:
    print(sess.run(tf.gradients(f, t4), feed_dict)[0])
```

```
1.0
```

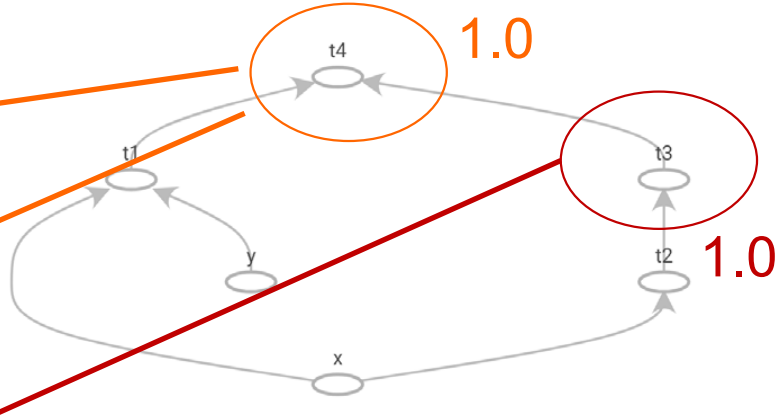
What is $\frac{\partial f}{\partial t_3}$?

Simply

$$\frac{\partial f}{\partial t_3} = \frac{\partial f}{\partial t_4} * \frac{\partial t_4}{\partial t_3} = 1 * \frac{\partial t_4}{\partial t_3} = 1 * 1$$

```
with tf.Session() as sess:
    print(sess.run(tf.gradients(f, t3), feed_dict)[0])
```

```
1.0
```



$$\begin{aligned} f(x,y) &= xy + \sin(x^2) \\ &= t_1 + \sin(x^2) && \text{using} \quad t_1 = xy \\ &= t_1 + \sin(t_2) && \text{using} \quad t_2 = x^2 \\ &= t_1 + t_3 && \text{using} \quad t_3 = \sin(t_2) \\ &= t_4 && \text{using} \quad t_4 = t_1 + t_3 \end{aligned}$$

Auto-Diff/Backpropagation (TensorFlow)

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

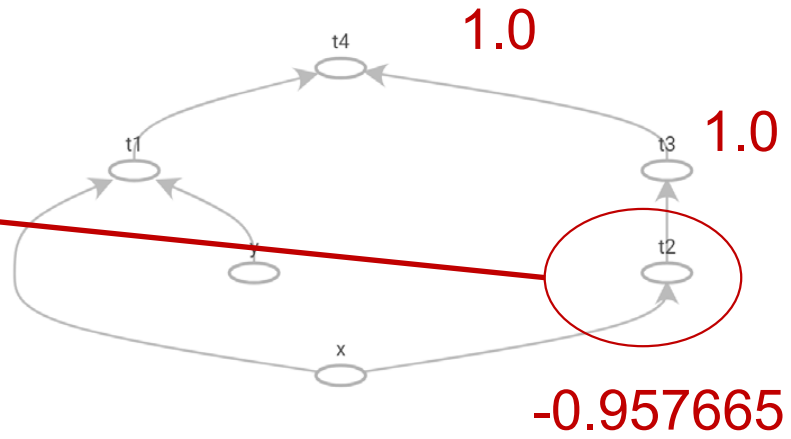
What is $\frac{\partial f}{\partial t_2}$?

Simply

$$\frac{\partial f}{\partial t_2} = \frac{\partial f}{\partial t_4} * \frac{\partial t_4}{\partial t_3} * \frac{\partial t_3}{\partial t_2} = 1 * \frac{\partial t_4}{\partial t_3} * \frac{\partial t_3}{\partial t_2} = 1 * 1 * \frac{\partial t_3}{\partial t_2} = 1 * 1 * \cos(x^2) \approx -0.95765948032$$

```
with tf.Session() as sess:
    print(sess.run(tf.gradients(f, t2), feed_dict)[0])
```

```
-0.9576595
```



$$\begin{aligned} f(x,y) &= xy + \sin(x^2) \\ &= t_1 + \sin(x^2) && \text{using} \quad t_1 = xy \\ &= t_1 + \sin(t_2) && \text{using} \quad t_2 = x^2 \\ &= t_1 + t_3 && \text{using} \quad t_3 = \sin(t_2) \\ &= t_4 && \text{using} \quad t_4 = t_1 + t_3 \end{aligned}$$

Auto-Diff/Backpropagation (TensorFlow)

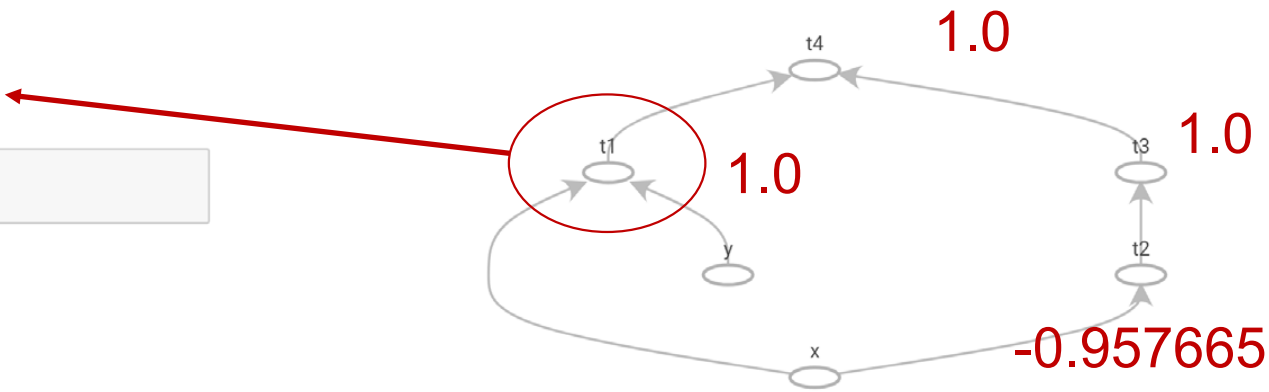
What is $\frac{\partial f}{\partial t_1}$?

Simply

$$\frac{\partial f}{\partial t_1} = \frac{\partial f}{\partial t_4} * \frac{\partial t_4}{\partial t_1} = 1 * \frac{\partial t_4}{\partial t_1} = 1 * 1 = 1$$

```
with tf.Session() as sess:
    print(sess.run(tf.gradients(f, t1), feed_dict)[0])
```

```
1.0
```



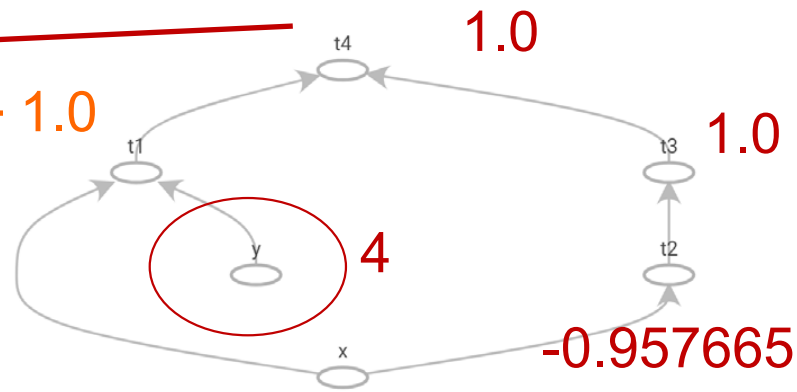
$$\begin{aligned} f(x,y) &= xy + \sin(x^2) \\ &= t_1 + \sin(x^2) && \text{using} \quad t_1 = xy \\ &= t_1 + \sin(t_2) && \text{using} \quad t_2 = x^2 \\ &= t_1 + t_3 && \text{using} \quad t_3 = \sin(t_2) \\ &= t_4 && \text{using} \quad t_4 = t_1 + t_3 \end{aligned}$$

Auto-Diff/Backpropagation (TensorFlow)

What is $\frac{\partial f}{\partial y}$?

Simply

$$\begin{aligned}\frac{\partial f}{\partial y} &= \frac{\partial f}{\partial t_4}\frac{\partial t_4}{\partial t_1}\frac{\partial t_1}{\partial y} \\ &= 1 * 1 * \frac{\partial t_1}{\partial x} \\ &= 1 * 1 * x \\ &= 1 * 1 * 4 \\ &= 4\end{aligned}$$




```
with tf.Session() as sess:
    print(sess.run(tf.gradients(f, y), feed_dict)[0])
```

```
4.0
```

$$\begin{aligned}f(x,y) &= xy + \sin(x^2) \\ &= t_1 + \sin(x^2) && \text{using} \quad t_1 = xy \\ &= t_1 + \sin(t_2) && \text{using} \quad t_2 = x^2 \\ &= t_1 + t_3 && \text{using} \quad t_3 = \sin(t_2) \\ &= t_4 && \text{using} \quad t_4 = t_1 + t_3\end{aligned}$$

Auto-Diff/Backpropagation (TensorFlow)

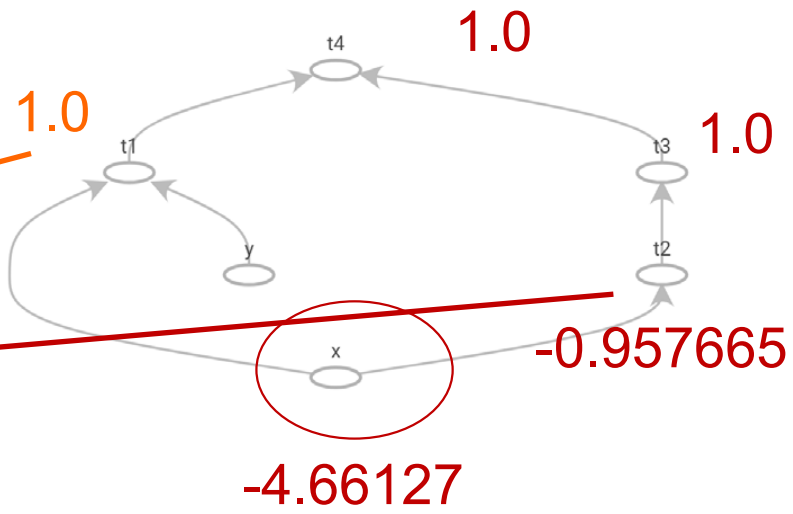
```
with tf.Session() as sess:
    print(sess.run(tf.gradients(f, t1), feed_dict)[0])
```

```
1.0
```

What is $\frac{\partial f}{\partial x}$?

Simply

$$
\begin{aligned}
\frac{\partial f}{\partial x} &= \frac{\partial f}{\partial t_4}\frac{\partial t_4}{\partial t_1}\frac{\partial t_1}{\partial x} + \frac{\partial f}{\partial t_4}\frac{\partial t_4}{\partial t_3}\frac{\partial t_3}{\partial t_2}\frac{\partial t_2}{\partial x} \\
&= 1 * \frac{\partial t_1}{\partial x} - 0.95765948032\frac{\partial t_2}{\partial x} \\
&= 1 * y - 0.95765948032\frac{\partial t_2}{\partial x} \\
&= 1 * y - 0.95765948032 * 2 * x \\
&= 1 * 3 - 0.95765948032 * 2 * 4 \\
&= -4.66127584256
\end{aligned}
$$



Backpropagation is AutoDiff

$$\frac{\partial f}{\partial t_i} = \sum_{p \in \text{parents}(t_i)} \frac{\partial f}{\partial p}\frac{\partial p}{\partial t_i}$$

$$
\begin{aligned}
f(x,y) &= xy + \sin(x^2) & & \\
&= t_1 + \sin(x^2) & \text{using} \quad & t_1 = xy \\
&= t_1 + \sin(t_2) & \text{using} \quad & t_2 = x^2 \\
&= t_1 + t_3 & \text{using} \quad & t_3 = \sin(t_2) \\
&= t_4 & \text{using} \quad & t_4 = t_1 + t_3
\end{aligned}
$$

Backpropagation

Algorithm for training a neural network using SGD:

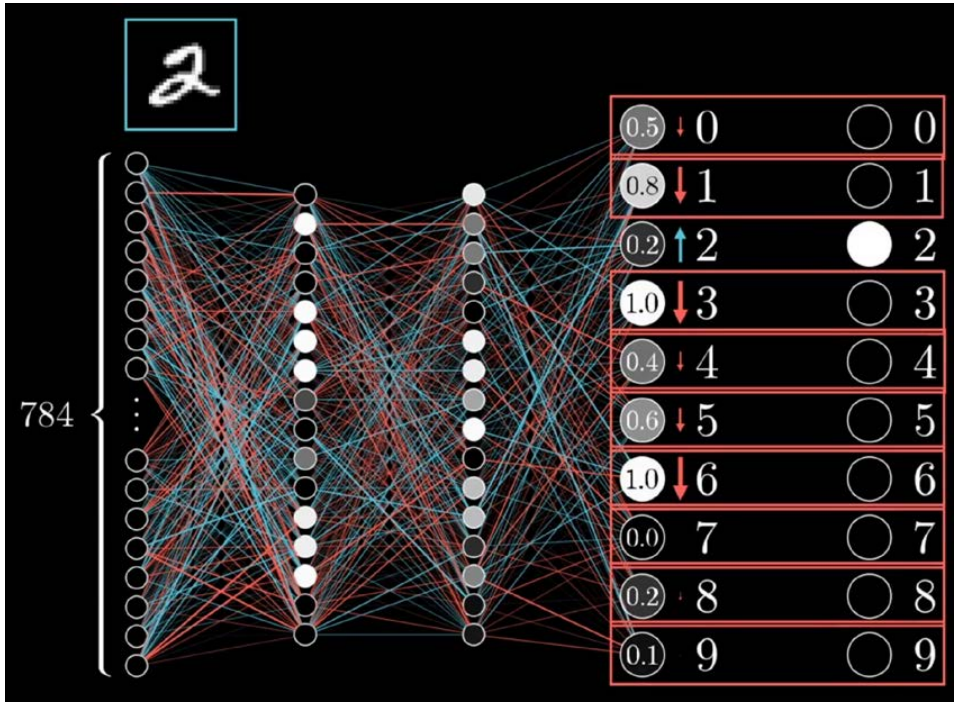
1. Initialize $\mathbf{W}$, pick learning rate η and mini batch size M
2. Randomly pick M data points $\mathcal{X}_{\text{batch}} \subset \mathcal{X}$
3. For each data point $\mathbf{x}_m \in \mathcal{X}_{\text{batch}}$ do:
   - 3.1 Forward propagate $\mathbf{x}_m$ through network: $y_j = \sigma\left(\sum_i w_{ji} y_i\right)$ (here: $\{y_0\} = \mathbf{x}_m$)
   - 3.2 Evaluate errors δ_j at output nodes: $\delta_j = (y_j - t_j) y_j (1 - y_j)$
   - 3.3 Backpropagate errors δ_i through network: $\delta_i = y_i (1 - y_i) \sum_j \delta_j w_{ji}$
   - 3.4 Calculate derivatives: $\frac{\partial E_m}{\partial w_{ji}} = \delta_j y_i$
4. Update gradients: $\mathbf{W}^{t+1} = \mathbf{W}^t - \eta \sum_{m=1}^{M} \nabla_{\mathbf{W}} E_m |_{\mathbf{W}^t}$
5. If validation error decreases, go to step 2, otherwise stop

[Andreas Geiger]

Mini Batch

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Size will determine training behavior
- Size will be depending on available GPU memory
  - Backprop requires storing of **activation and derivatives (for accumulation) for each neuron and each sample** in the batch
  - **Limited by memory**
- Each mini batch should reflect the distribution of the entire data set
- Small subsets: bad approximation of the distribution → bad / noisy approximation of the gradient
- Training still possible with batch size 1



Stochastic Gradient Descent (SGD)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Error function over all data points $\{(x_n, t_n)\}_{n=1}^{N}$ and all elements j in last layer, e.g.

$$E = \frac{1}{2}\sum_n \sum_j \left(y_{jn} - t_{jn}\right)^2$$

- Form a random mini-batch M (M << N) of all training examples N

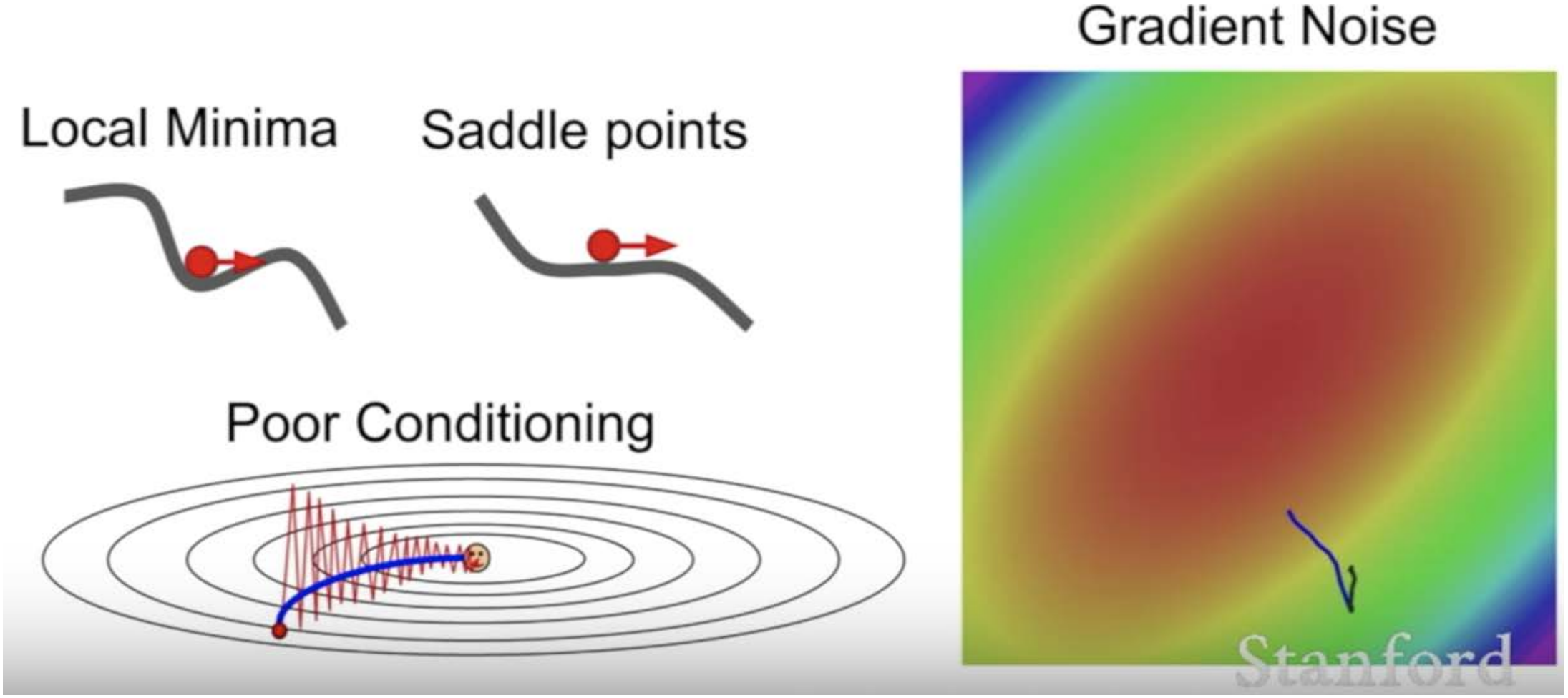
- Compute a (noisy) gradient estimate from the mini batch

$$\nabla E \approx \sum_m \nabla E_m$$

- Update weights according to learning rate α

$$w_{i+1} = w_i - \alpha \nabla E_m(w_i)$$

Problems with vanilla SGD



SGD + Momentum

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Gradient descent

$$w_{i+1} = w_i - \alpha \nabla E(w_i)$$

- Adding momentum
  - Build up „velocity“ with friction ρ

$$v_{i+1} = \rho v_i + \nabla E(w_i)$$

$$w_{i+1} = w_i - \alpha v_{i+1}$$

  - Smoothes the gradient
  - Can bridge plataeus

- **Problem**
  - Overshooting: Not homing in on the minimum because of the large accumulated momentum

AdaGrad - Adaptive Subgradient Methods

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Adapt the learning rate by the accumulated squared gradient
- Slow down close to the minimum

$$g_{i+1}^2 = g_i^2 + \nabla E(w_i) \cdot \nabla E(w_i)$$

$$w_{i+1} = w_i - \frac{\alpha}{\sqrt{g_{i+1}^2 + \epsilon}} \nabla E(w_i)$$

- Adapts to the local geometry of the error function

RMSProp

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Proposed by Geoff Hinton
- Use exponentially decaying accumulation (e.g. $\beta = 0{,}9$)

$$g_{i+1}^2 = \beta g_i^2 + (1-\beta)\nabla E(w_i) \cdot \nabla E(w_i)$$

$$w_{i+1} = w_i - \frac{\alpha}{\sqrt{g_{i+1}^2 + \epsilon}} \nabla E(w_i)$$

- Limits the influence of old gradients
- (almost the same as AdaDelta)

ADAM

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- … combine first (AdaGrad) and second order (RMSProp) momentum

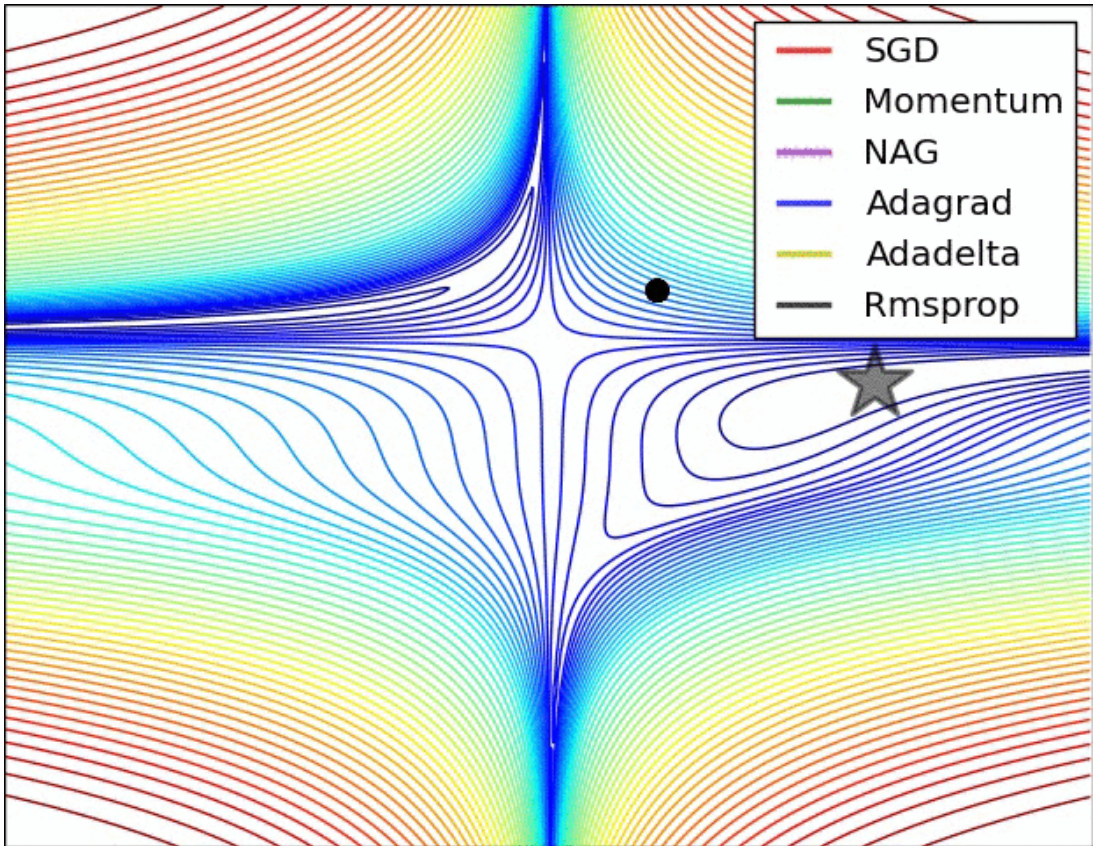
$$v_{i+1} = \rho v_i + (1-\rho)\nabla E(w_i)$$

$$g_{i+1}^2 = \beta g_i^2 + (1-\beta)\nabla E(w_i) \cdot \nabla E(w_i)$$

$$w_{i+1} = w_i - \frac{\alpha}{\sqrt{g_{i+1}^2 + \epsilon}} v_{i+1}$$

- Additional correction to fight bias: use $\hat{v} = \frac{v}{1-\rho}$ and $\hat{g}^2 = \frac{g^2}{1-\beta}$

Optimization Performance on Error Landscape

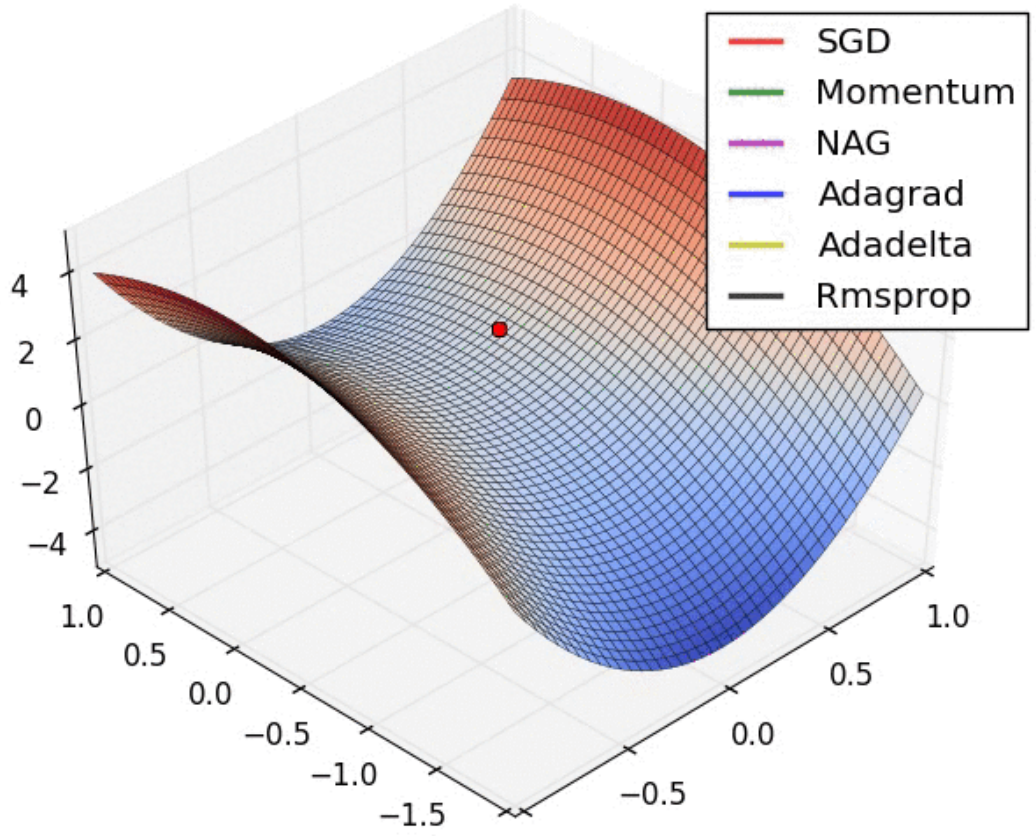


[Sebastian Ruder]

Optimization Performance on Error Landscape

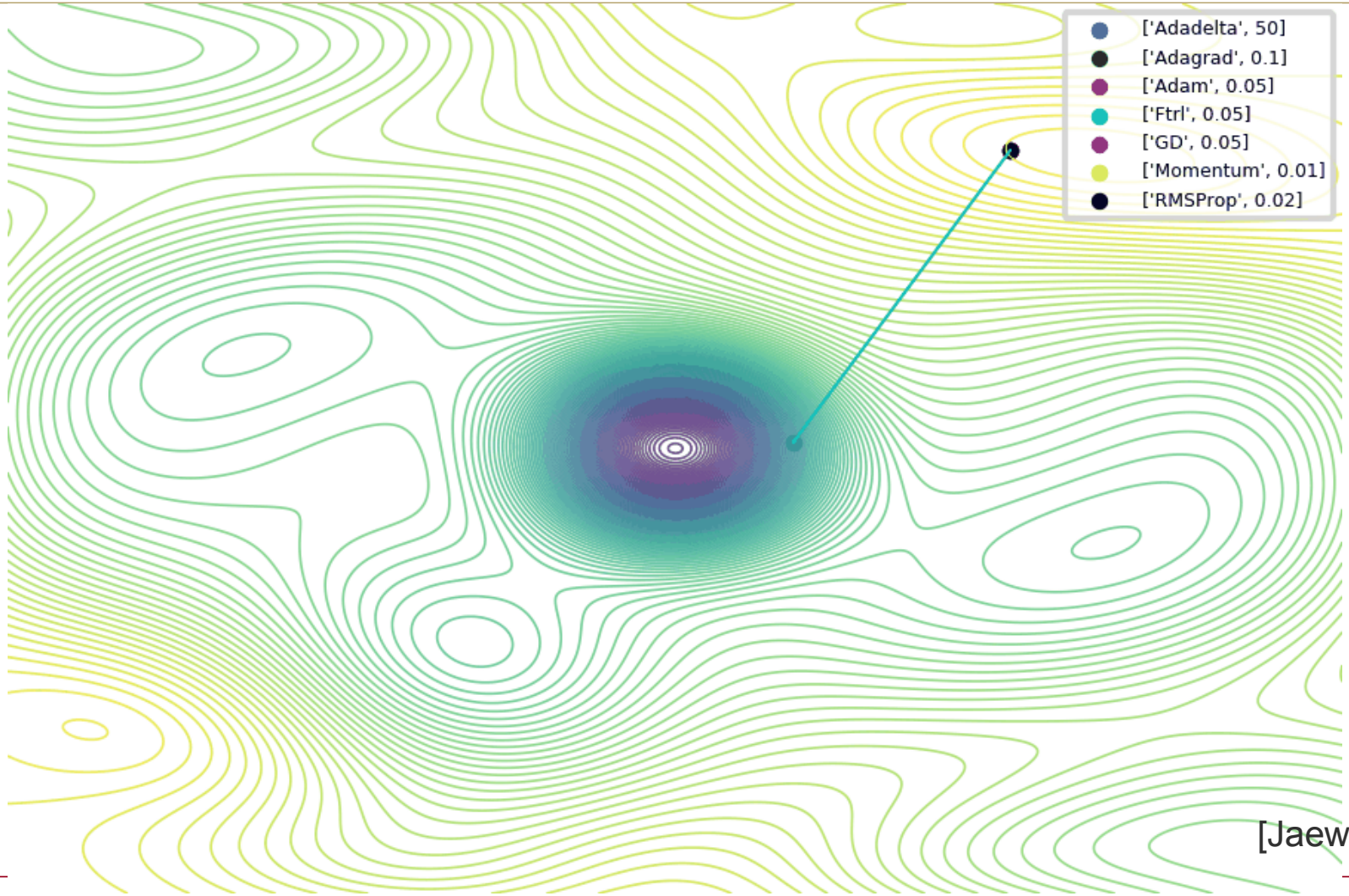
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Saddle point



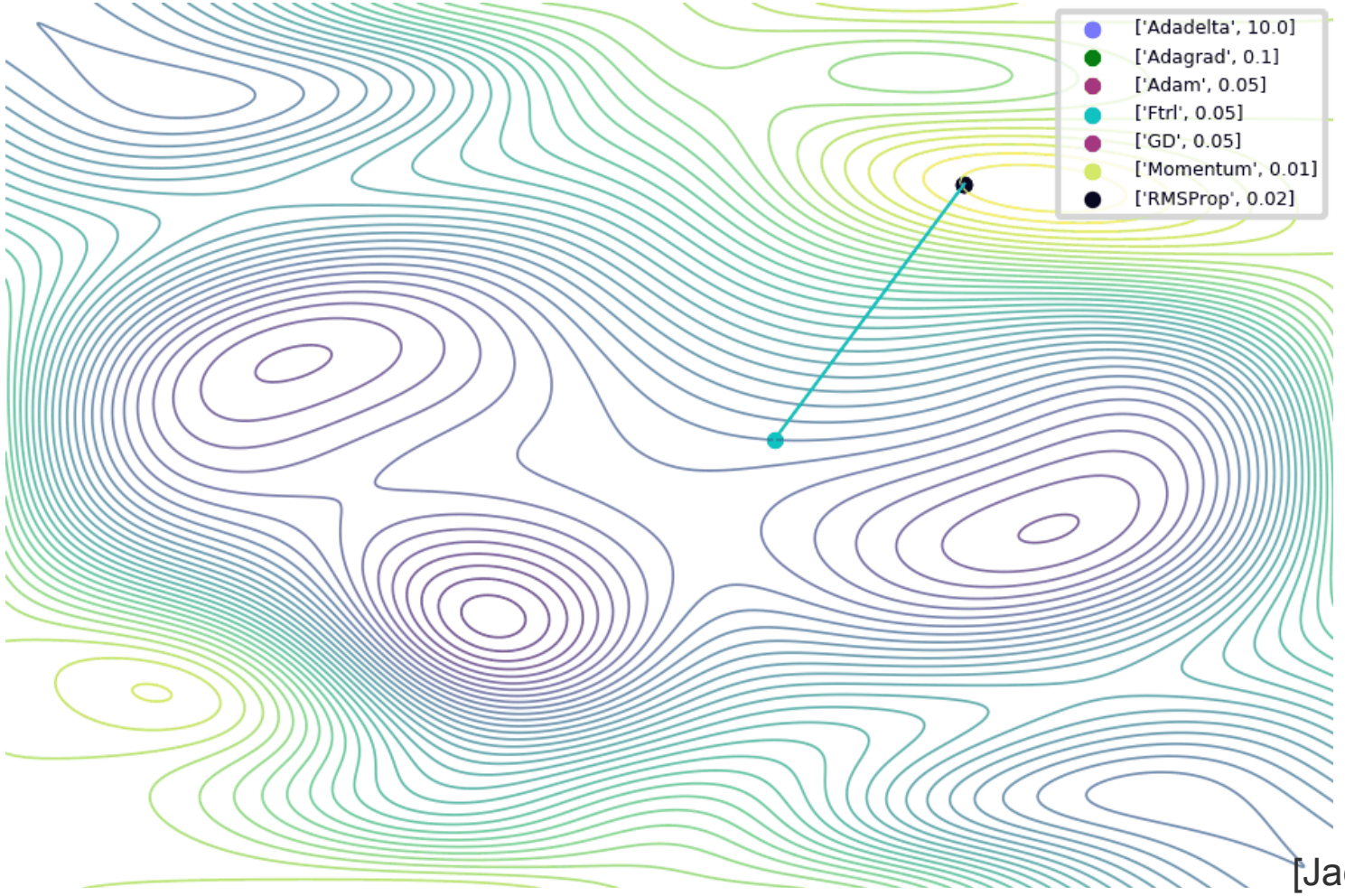
[Sebastian Ruder]

Optimization Performance on Error Landscape



[Jaewan Yun]

Optimization Performance on Error Landscape



[Jaewan Yun]

Learning on Sequences

Recurrent Neural Networks (RNN)

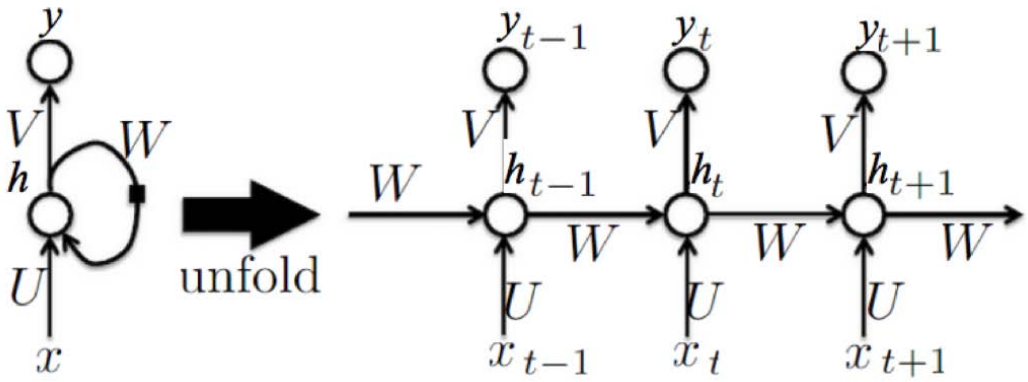
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Sequences:
  - Text
  - Speech
  - Time series
  - Sound
  - Omics-data
  - (Video)

- Exploit correlation / relationship between different time steps

- Three widely used recurrent units
  - Traditional tanh unit
  - Long short-term memory (LSTM) unit
  - Gated recurrent unit (GRU)

tanh-based RNN



- x_t is the input at time step t.
- h_t is the hidden state at time step t.
- h_t is calculated based on the previous hidden state and the input at the current step:

$$h_t = \sigma(Ux_t + Wh_{t-1})$$

- y_t is the output at step t.
- E.g., if we wanted to predict the next word in a sentence it would be a vector of probabilities across our vocabulary
- Often: $\sigma = tanh$

- **Problem of vanishing gradients /exploding gradients:** $f(x)^k$

Long Short-Term Memory (LSTM)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- [Hochreiter and Schmidhuber 1997]

- Address the vanishing gradient problem/exploding gradient problem.

Closer to how humans process information: Introduce **gates**

- Control how much of the previous hidden state to forget
- Control how much of new input to take

Long Short-Term Memory (LSTM)

- Forget Gate (gate 0, forget past)

$$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1})$$

- Input Gate (current activation matters

$$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1})$$

- New memory cell

$$\tilde{c}_t = tanh(W^{(c)}x_t + U^{(c)}h_{t-1})$$

- Final memory cell

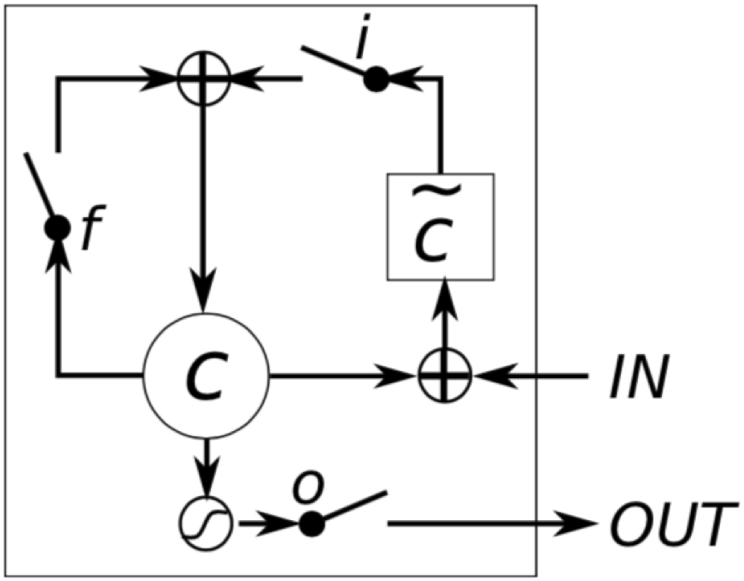
$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t$$

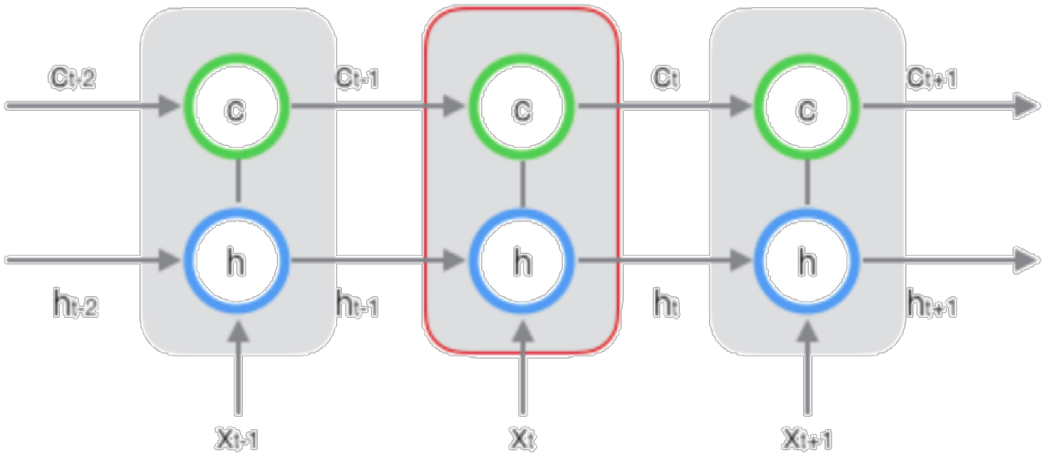
- Output Gate (how much the cell is exposed)

$$o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1})$$

- Final hidden state

$$h_t = o_t \cdot tanh(c_t)$$





Gated Recurrent Unit (GRU)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- LSTMs work well but unnecessarily complicated
- GRU is a variant of LSTM

- Approach:
  - Combine the forgetting gate and input gate in LSTM into a single "Update Gate".
  - Combine the Cell State and Hidden State.

- Computationally less expensive
  - less parameters, less complex structure

- Performance is at least as good as LSTM

Gated Recurrent Unit (GRU)

- Reset gate: determines how to combine the new input with the previous memory

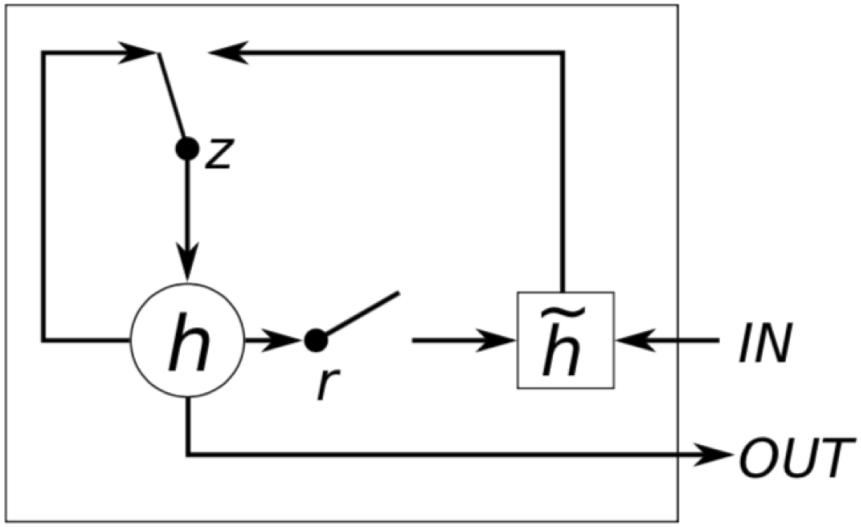
$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$

- Update gate: decides how much of the previous memory to keep around

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$$

- Candidate hidden layer

$$\tilde{h}_t = tanh(W^{(h)}x_t + r_t \cdot U^{(h)}h_{t-1})$$



- Final memory at time step combines current and previous time steps:

$$h_t = z_t \cdot h_{t-1} + (1 - z_t) \cdot \tilde{h}_t$$

Advantage of LSTM/GRU

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- It is easy for each unit to remember the existence of a specific feature in the input stream for a long series of steps.

- The shortcut paths allow the error to be back-propagated easily without too quickly vanishing
  - Error pass through multiple bounded nonlinearities, which reduces the likelihood of the vanishing gradient.

LSTM vs. GRU

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

LSTM	GRU
Three gates	Two gates
Control the exposure of memory content (cell state)	Expose the entire cell state to other units in the network
Has separate input and forget gates	Performs both of these operations together via update gate
More parameters	Fewer parameters

RNN – Comparison Evaluation

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

The authors built models for each of the three test units (LSTM, GRU, tanh) along the following criteria:

- **Similar numbers of parameters** in each network, for fair comparison
- RMSProp optimization
- Learning rate chosen to maximize the validation
- The models are tested across four music datasets and two speech datasets.

Unit	# of Units	# of Parameters
Polyphonic music modeling		
LSTM	36	$\approx 19.8 \times 10^3$
GRU	46	$\approx 20.2 \times 10^3$
tanh	100	$\approx 20.1 \times 10^3$
Speech signal modeling		
LSTM	195	$\approx 169.1 \times 10^3$
GRU	227	$\approx 168.9 \times 10^3$
tanh	400	$\approx 168.4 \times 10^3$

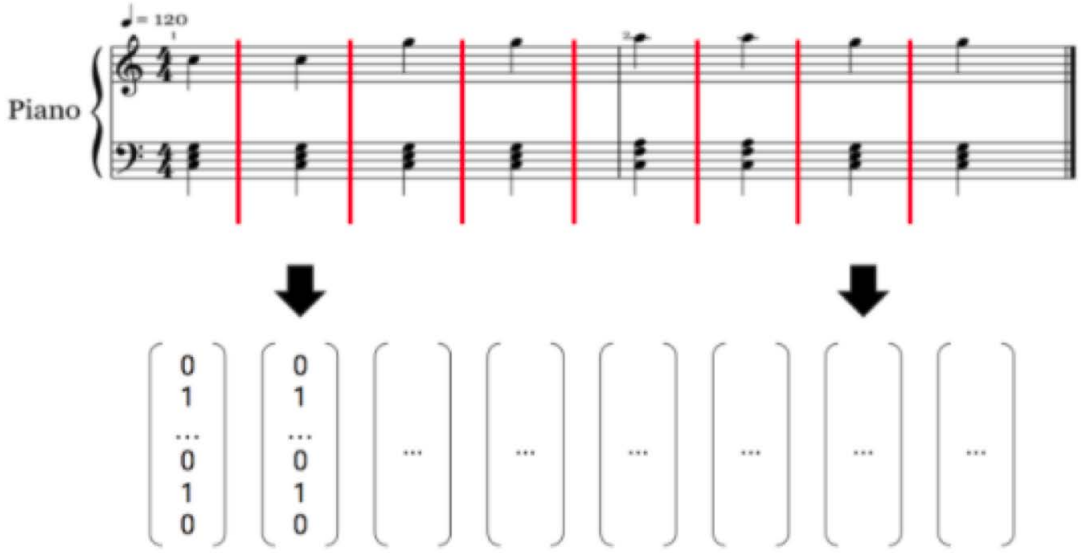
Table 1: The sizes of the models tested in the experiments.

[Cho et al. 2014]

Tasks

Music dataset
- Input: the sequence of vectors
- Output: predict the next time step of the sequence

Speech signal dataset:
- Look at 20 consecutive samples to predict the following 10 consecutive samples
- Input: one-dimensional raw audio signal at each time step
- Output: the next time 10 consecutive step of the sequence



Result - average negative log-likelihood

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Music datasets

- The GRU-RNN outperformed all the others (LSTM-RNN and tanh-RNN)
- All the three models performed closely to each other

Ubisoft datasets

- the RNNs with the gating units clearly outperformed the more traditional tanh-RNN

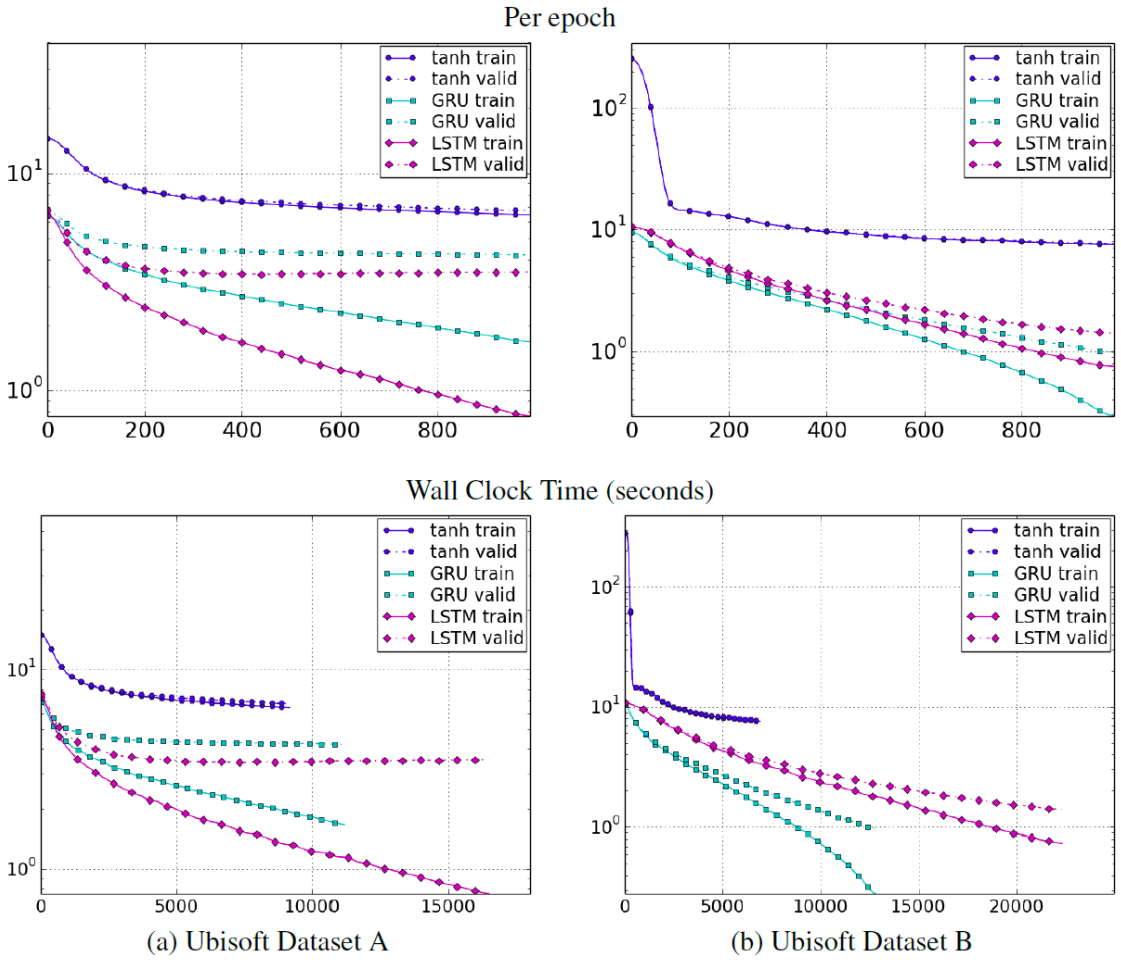
			tanh	GRU	LSTM
Music Datasets	Nottingham	train	3.22	2.79	3.08
		test	**3.13**	3.23	3.20
	JSB Chorales	train	8.82	6.94	8.15
		test	9.10	**8.54**	8.67
	MuseData	train	5.64	5.06	5.18
		test	6.23	**5.99**	6.23
	Piano-midi	train	5.64	4.93	6.49
		test	9.03	**8.82**	9.03
Ubisoft Datasets	Ubisoft dataset A	train	6.29	2.31	1.44
		test	6.44	3.59	**2.70**
	Ubisoft dataset B	train	7.61	0.38	0.80
		test	7.62	**0.88**	1.26

Table 2: The average negative log-probabilities of the training and test sets.

[Cho et al. 2014]

Result – Learning curves

- The gated units (LSTM and GRU) well outperformed the tanh unit
- The GRU-RNN once again producing the best results



[Cho et al. 2014]

Take home points

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Music datasets
- The GRU-RNN reached the inching better performance.
- All of the models performed relatively closely

Speech datasets
- The gated units well outperformed the tanh unit
- The GRU-RNN produce the best results both in terms of accuracy and training time.

- Gated units are superior to recurrent neural networks (RNNs)
- The performance of the two gated units (LSTM and GRU) cannot be clearly distinguished.

Temporal Convolution Network

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Bai et al. [2018] show that convolutional approaches can outperform recurrent settings on many sequence problems:

- simple and clearer than canonical recurrent networks
- combines modern convolutional architectures
- retains longer memory and longer history

Sequence modeling:

$$\hat{y}_0, \ldots, \hat{y}_T = f(x_0, \ldots, x_T)$$

Temporal Convolution Network

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

The distinguishing characteristics of TCNs are

1. The convolutions in the architecture are causal, each temporal convolution filter only convolves over samples from the past.
2. The architecture can take a sequence of any length and map it to an output sequence of the same length
3. Very large temporal footprint by combining deep architectures and dilated convolution

Temporal Convolution Network

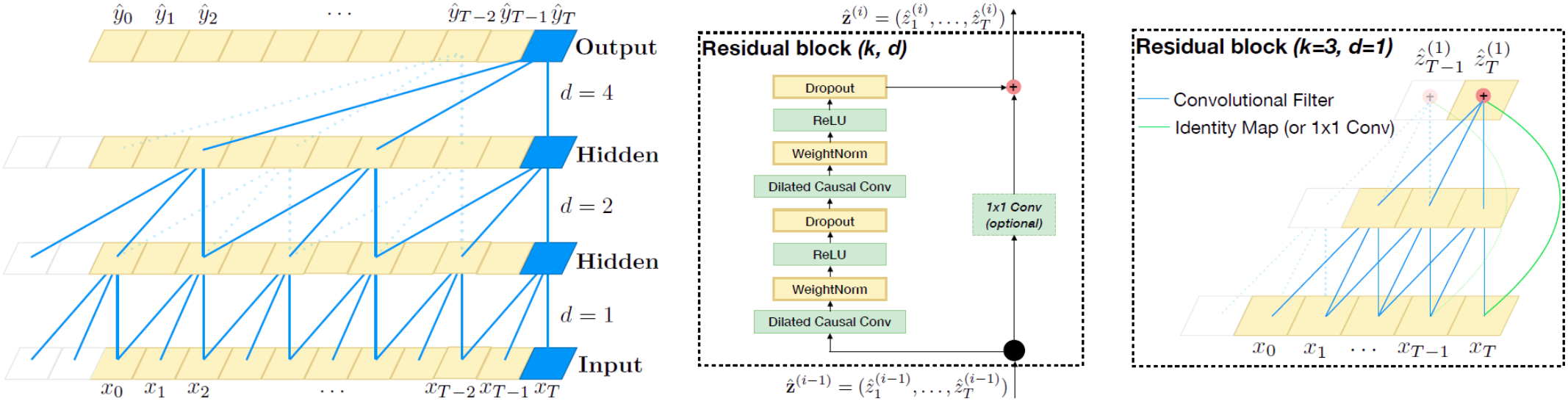


Figure 1. Architectural elements in a TCN. (a) A dilated causal convolution with dilation factors $d = 1, 2, 4$ and filter size $k = 3$. The receptive field is able to cover all values from the input sequence. (b) TCN residual block. An 1x1 convolution is added when residual input and output have different dimensions. (c) An example of residual connection in a TCN. The blue lines are filters in the residual function, and the green lines are identity mappings.

- dilated convolutions (van den Oord et al., 2016) to enable an exponentially large receptive field (Yu & Koltun, 2106)

$$F(s) = (\mathbf{x} *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot \mathbf{x}_{s-d \cdot i}$$

Discussion - Advantages

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Parallelism
- Flexible receptive field size
- Stable Gradients comparing to RNN
  - RNN and gated RNNs are complicated to train
- TCN avoids exploding/vanishing gradients because TCN has a backpropagation path different from the temporal direction of the sequence.
- Low memory requirement for training
- Variable length inputs

Discussion - Disadvantages

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Data storage during evaluation
  - RNN can use less memory on evaluation compare to training process
- Potential parameter change for a transfer of domain
  - **TCN** may not perform well on transfer of domain
    - when little memory need -> large memory need
    - for not having a sufficiently large receptive field

Results

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Sequence Modeling Task	Model Size ($\approx$)	Models			
		LSTM	GRU	RNN	**TCN**
Seq. MNIST (accuracy^h)	70K	87.2	96.2	21.5	**99.0**
Permuted MNIST (accuracy)	70K	85.7	87.3	25.3	**97.2**
Adding problem T=600 (loss$^\ell$)	70K	0.164	**5.3e-5**	0.177	**5.8e-5**
Copy memory T=1000 (loss)	16K	0.0204	0.0197	0.0202	**3.5e-5**
Music JSB Chorales (loss)	300K	8.45	8.43	8.91	**8.10**
Music Nottingham (loss)	1M	3.29	3.46	4.05	**3.07**
Word-level PTB (perplexity$^\ell$)	13M	**78.93**	92.48	114.50	88.68
Word-level Wiki-103 (perplexity)	-	48.4	-	-	**45.19**
Word-level LAMBADA (perplexity)	-	4186	-	14725	**1279**
Char-level PTB (bpc$^\ell$)	3M	1.36	1.37	1.48	**1.31**
Char-level text8 (bpc)	5M	1.50	1.53	1.69	**1.45**

Results – arbitrarily long „memory“

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Copy sequence task

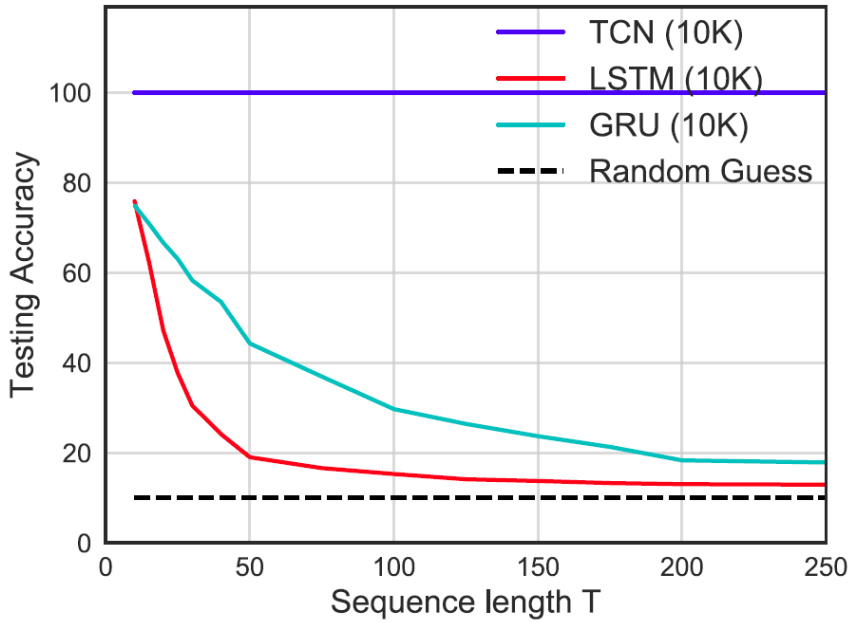


Figure 5. Accuracy on the copy memory task for sequences of different lengths T. While TCN exhibits 100% accuracy for all sequence lengths, the LSTM and GRU degenerate to random guessing as T grows.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

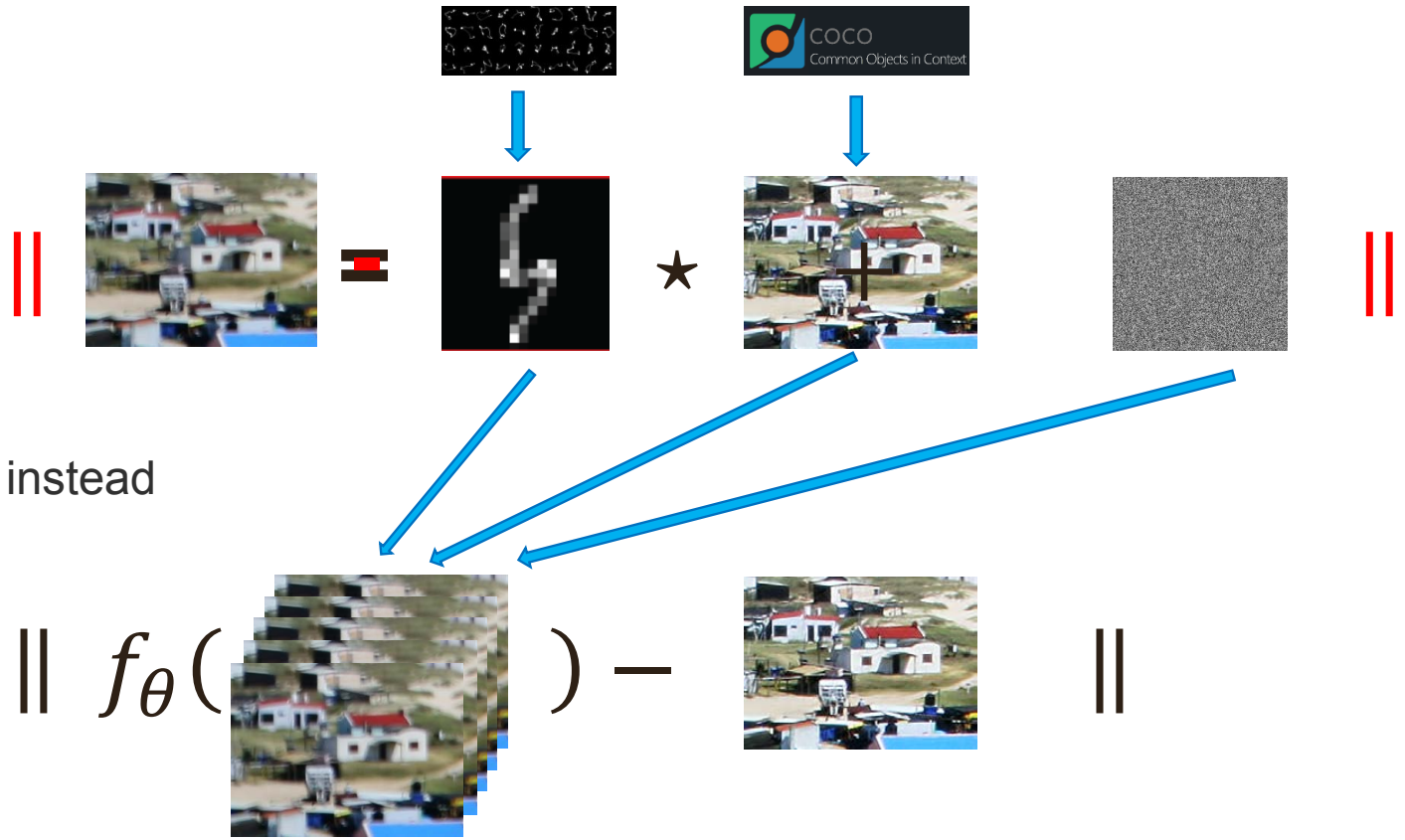
FACULTY OF
SCIENCE
Computer Graphics

Learning Blind Motion Deblurring

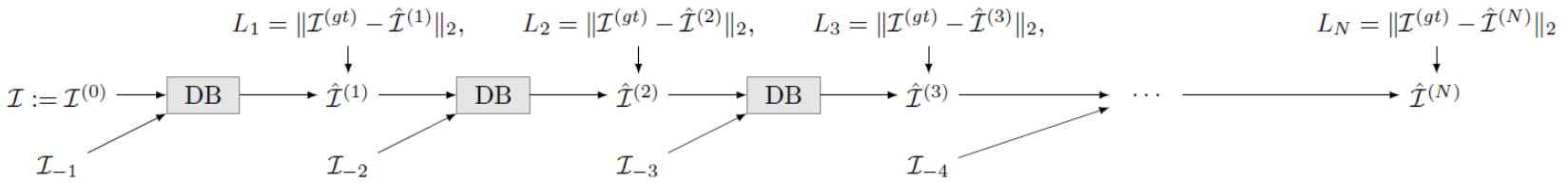
Patrick Wieschollek, Michael Hirsch, Bernhard Schölkopf, Hendrik Lensch

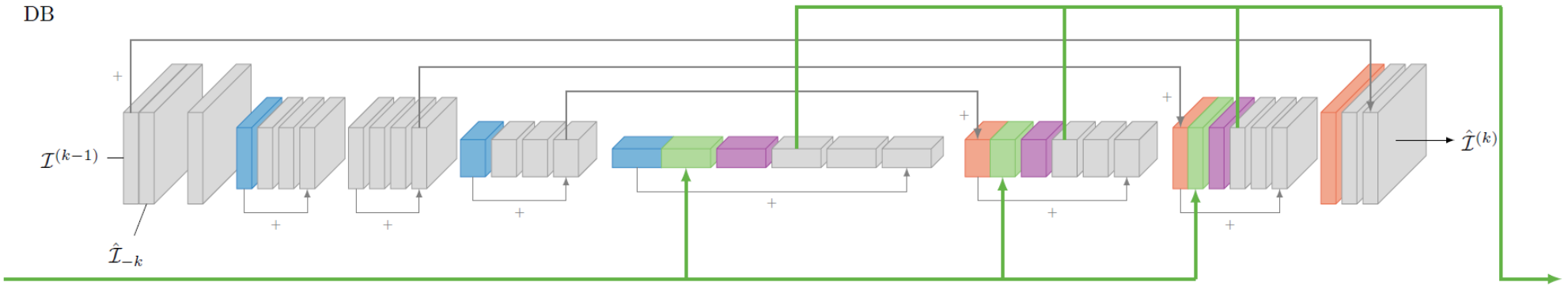
ACCV 2016, ICCV 2017

Forward Model / Optimization



Deblurring on Arbitrary Sequences





Training

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Triplets of input, learned reconstruction, ground truth

Training Set – Data Augmentation

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Training need tens of thousands of examples

- How to generate training data?
  - Start with a good data set
  - Perform data augmentation to enrich the data set by synthetically varied examples

- Data Augmentation
  - Noise
  - Blur
  - Scale
  - Translation
  - Reflection
  - Intensity
  - Shear
  - …

- Here: Input 4k videos, optical flow to enhence blur, per-frame camera shake simulation

Results

Image and Video Deblurring

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- input

Image and Video Deblurring

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- [Delbracio et al. 2015]

Image and Video Deblurring

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- [Wieschollek et al. 2016 - ours]

Image and Video Deblurring

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- [Wieschollek et al. 2017 - ours]

Image and Video Deblurring

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Ground truth

Previous Work

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

	Non-linear Playback	Appearance Changes	Temporal Alignment	Handling Detours
a Evangelidis et al. (2011) 	✓	✗	✓	✗
b Torii et al. (2015) 	✗	✓	✗	✗
c Douze et al. (2016) 	✗	✓	✓	✗
d 	✓	✓	✓	✓

Efficiency: Sparse Feature-Extraction (260h Data)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

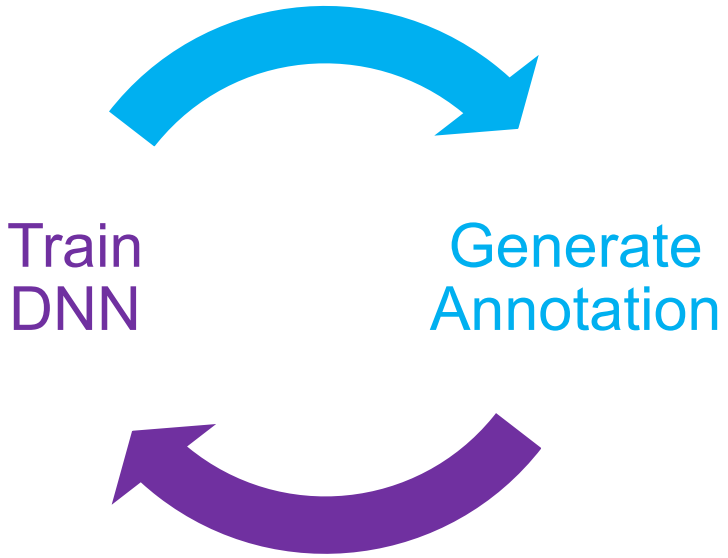
Approach	FPS	Total Time
Evangelidis et al. (2011)	0.11	0.81 years (~42 weeks)
Evangelidis et al. (2013)	0.11	0.81 years (~42 weeks)
Wang et al. (2014)	3.77	8.6 days
Ours	140	5.57 hours
Ours (12 GPUS)	12 * 140	28 minutes

x1272 speed up

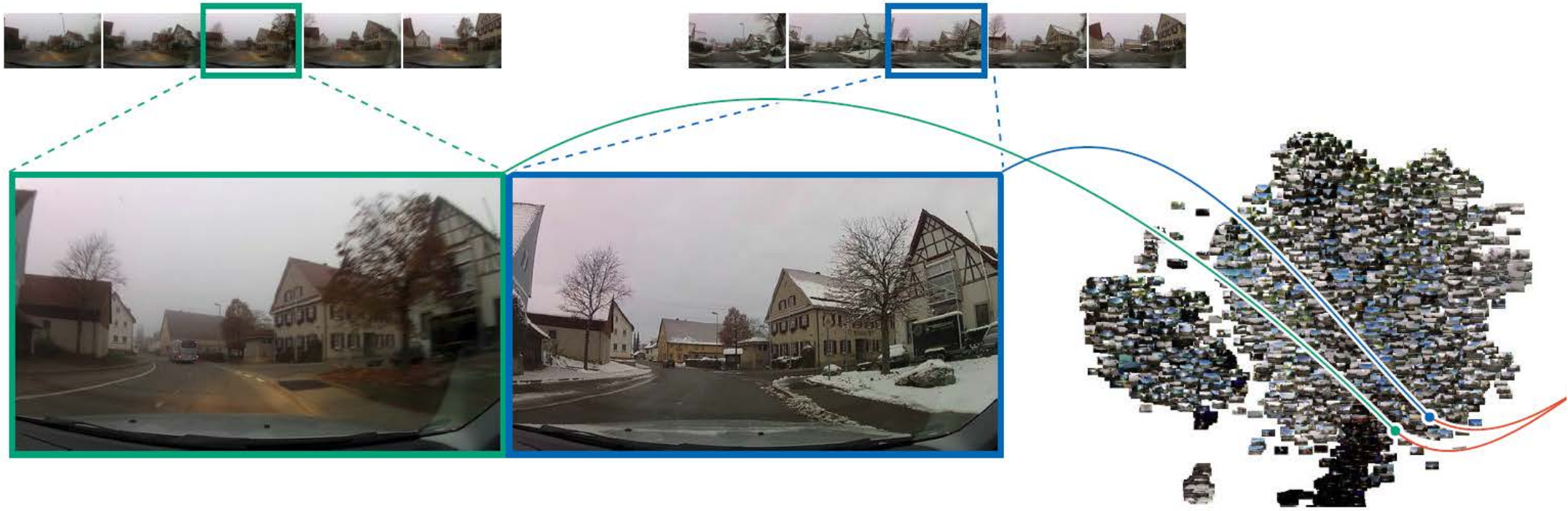
Data-Driven Approaches without Annotations

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Chicken-Egg-Problem



Encoding each Frame

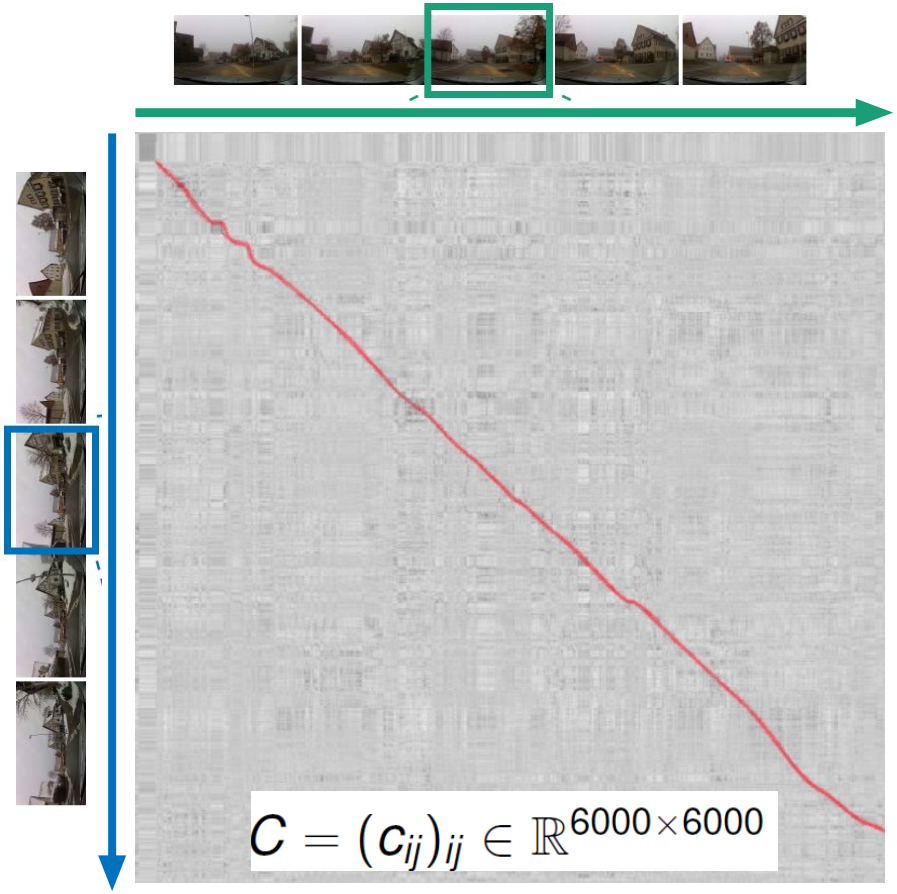


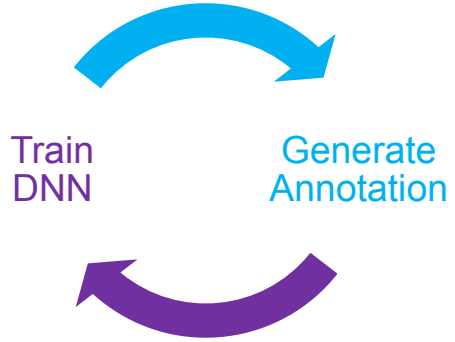
- Visualization (t-SNE): mapping 1024 dimensions to 2 dimensions
- Euclidean distance between two frames in high-dimensional encoding-space represents their similarity.

Phase: Generate Annotation

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Assume trained Deep Neural Network is given



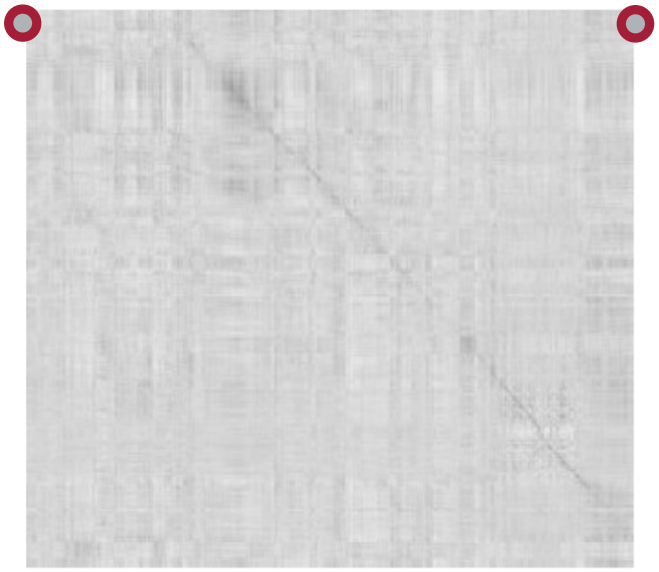


1. pair-wise distances C‘
2. low-rank approximation
3. de-correlated distances C
4. find syncronization path

Finding a Synchronization Path

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

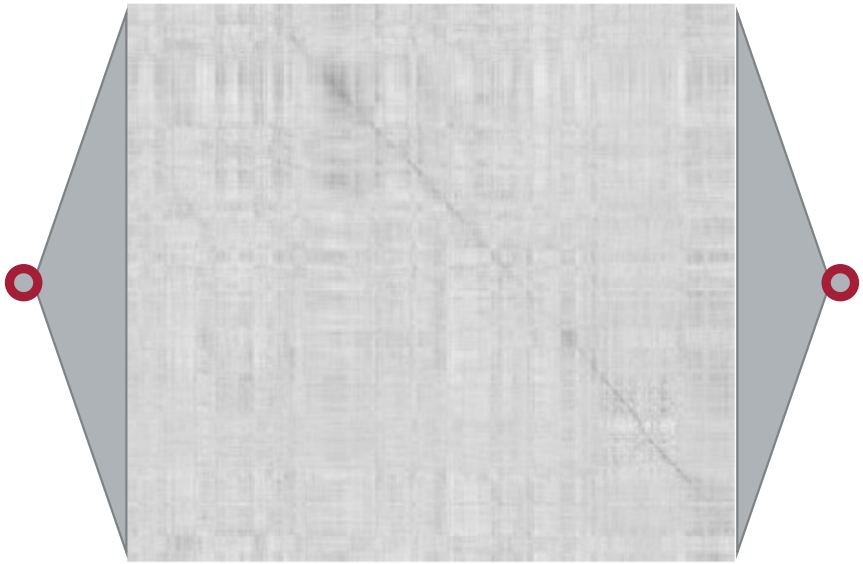
- Dijkstra's algorithm finds shortest path between two given nodes.
- We do not know the start or end node!



Finding a Synchronization Path

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

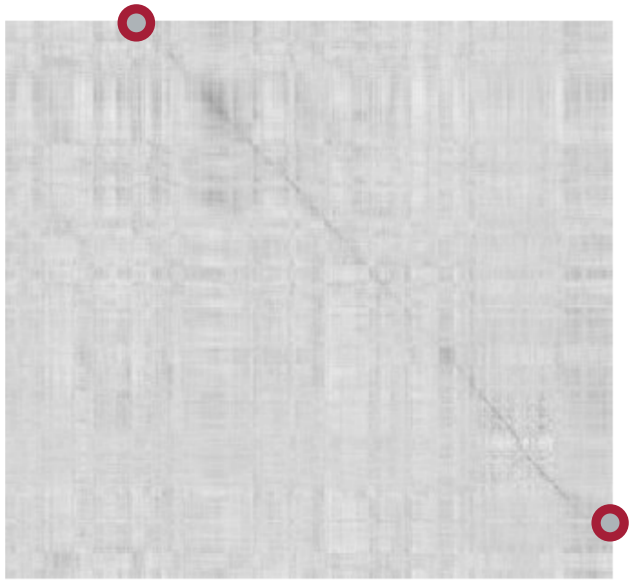
- Dijkstra's algorithm finds shortest path between two given nodes.
- We do not know the start or end node!



Finding a Synchronization Path

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

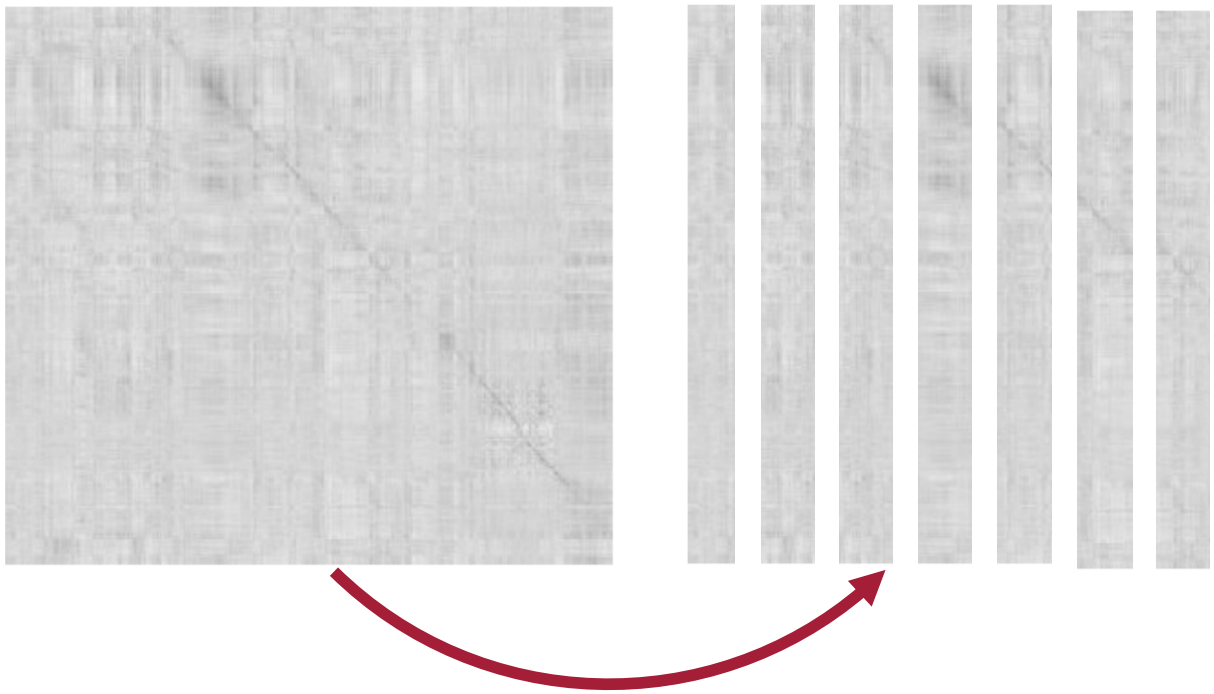
- Dijkstra's algorithm finds shortest path between two given nodes.
- We do not know the start or end node!
- We even do not know if there is any path!



Finding a Synchronization Path

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

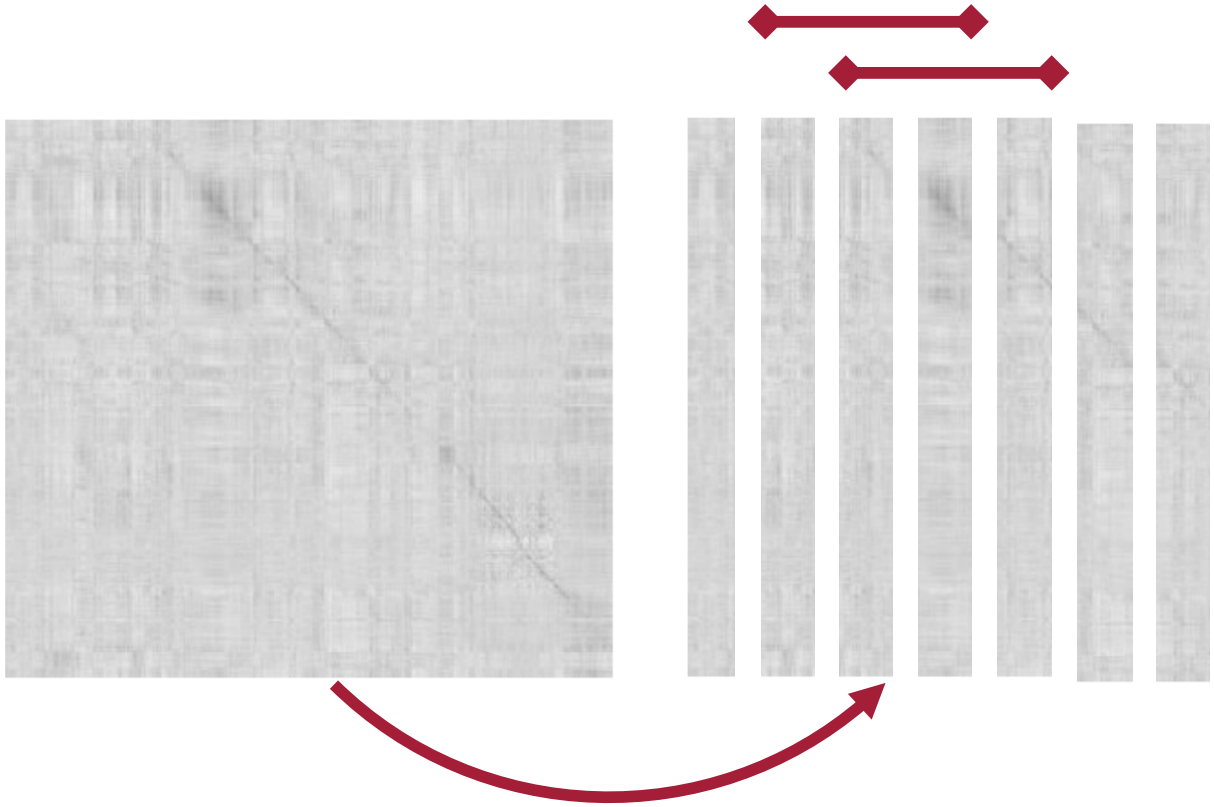
- Dijkstra's algorithm finds shortest path
- Partition into strips



Finding a Synchronization Path

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

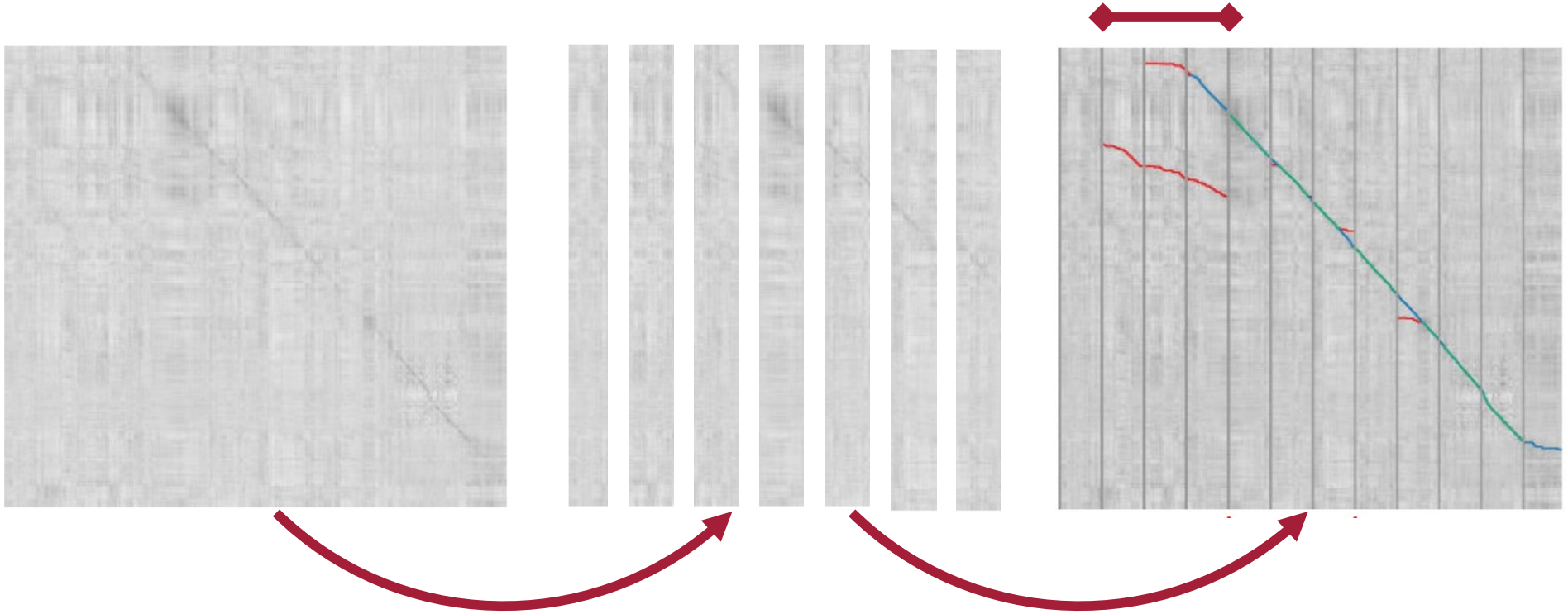
- Dijkstra's algorithm finds shortest path
- Partition into strips
- Find shortest path for overlapping strips



Finding a Synchronization Path

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

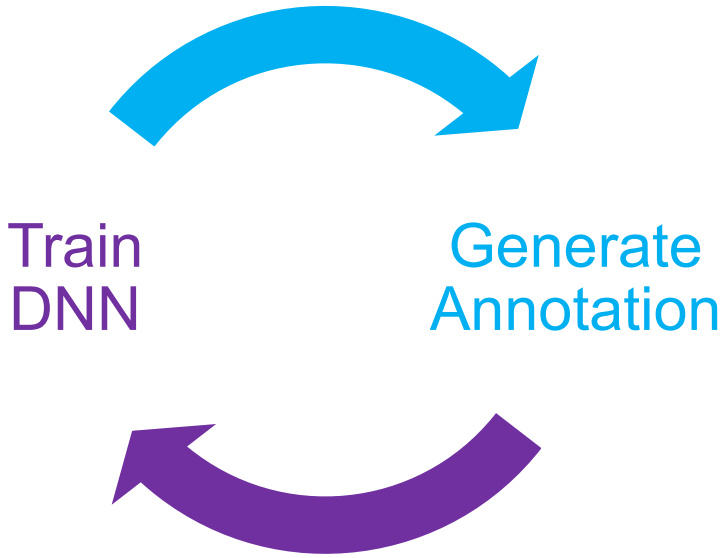
- Dijkstra's algorithm finds shortest path
- Partition into strips
- Find shortest path for overlapping strips
- Judge consistency in overlap regions



Data-Driven Approaches without Annotations

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

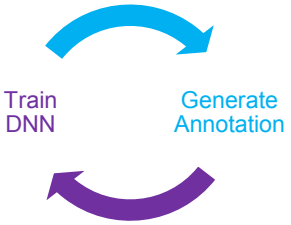
- Chicken-Egg-Problem



Phase: Train DNN

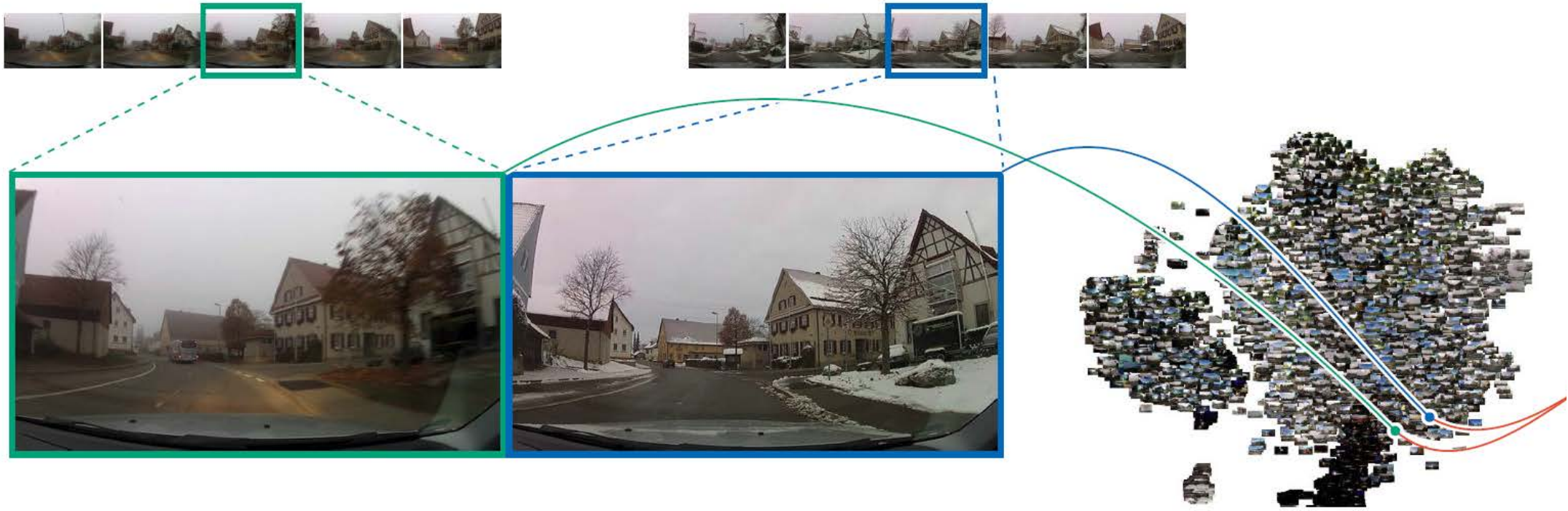
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Assume ground-truth annotations are given
- Optimize triplet loss



$$\max\{0, m + \|\Phi_a - \Phi_p\|_2^2 - \|\Phi_a - \Phi_n\|_2^2\}$$

Encoding each Frame

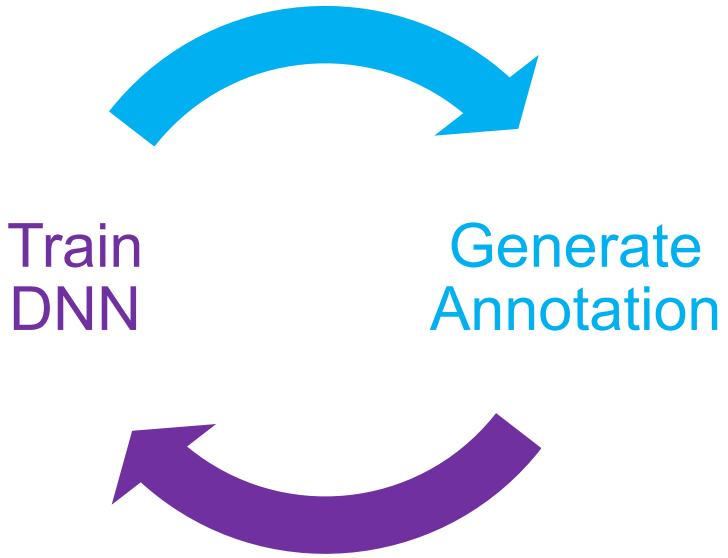


- Encoding clearly is not perfect
- Interplay of CNN encoding and shortest path on cleaned matrix render it a stable process

Data-Driven Approaches without Annotations

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

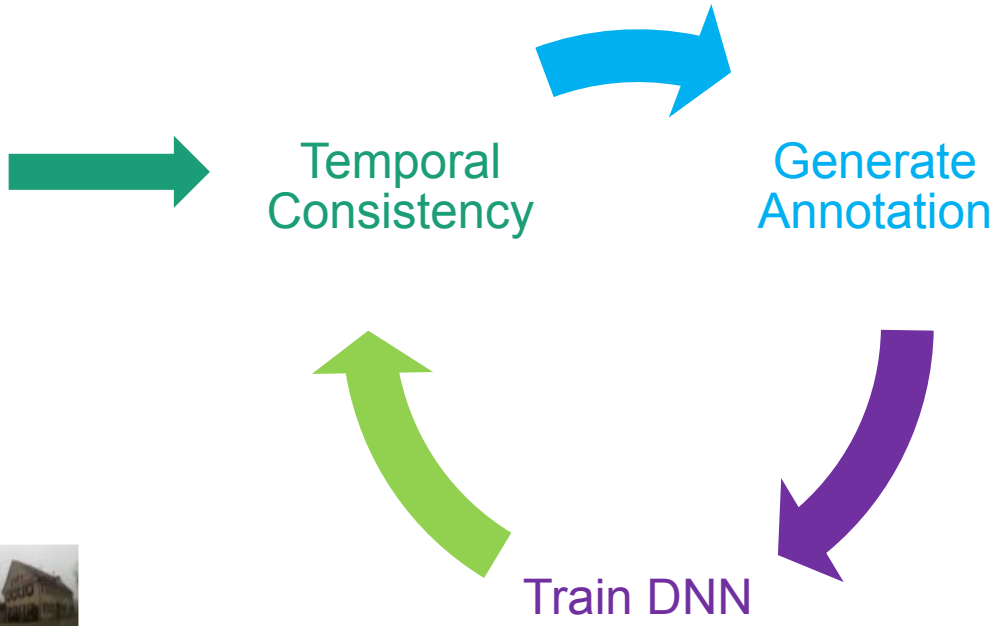
- Chicken-Egg-Problem



Data-Driven Approaches without Annotations

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- Solve the Chicken-Egg-Problem

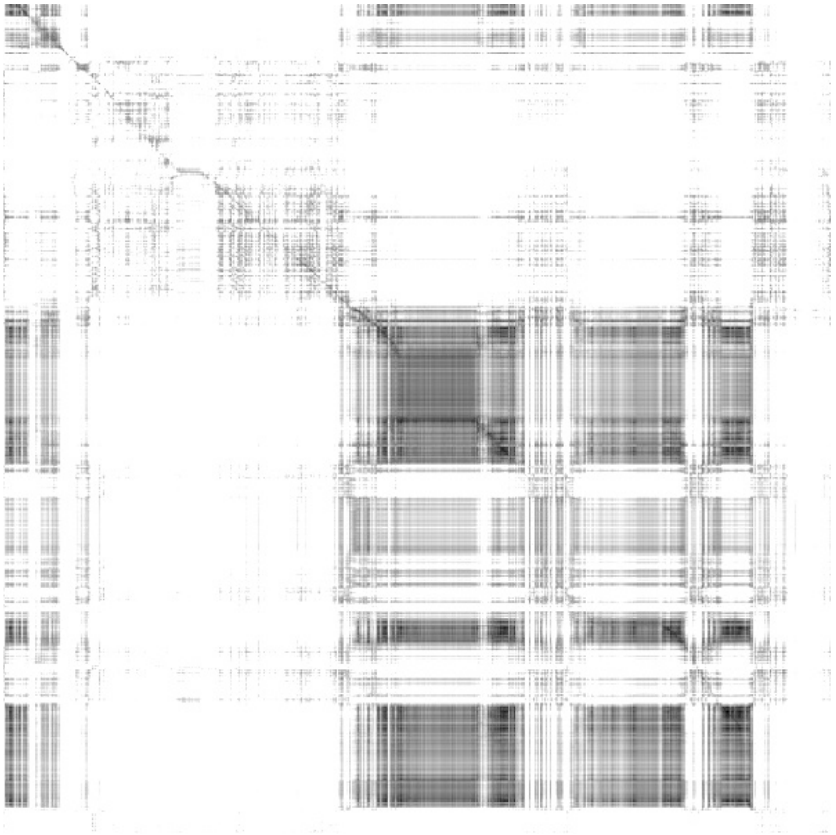


- Causality of Time

Curriculum Learning

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Start with examples which are not too difficult and not too obvious:



1. Intra-Video Sampling
   consecutive frames are similar
2. Inter-Video Sampling
   ensure temporal consistency to reject false positives
3. Transitive Inter-Video Sampling
   $v/_{\sim}$ is equivalence relation $a \sim b \sim c$

Important Features

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Why are local features not useful?
- What is the network looking for?

HARRIS

used in Evangelidis et al. 2011, 2013

SIFT

used in Wang et al. 2014

Learned Features (Saliency Maps)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Input | ResNet (ImageNet) | Ours

Saliency Maps

Robustness

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

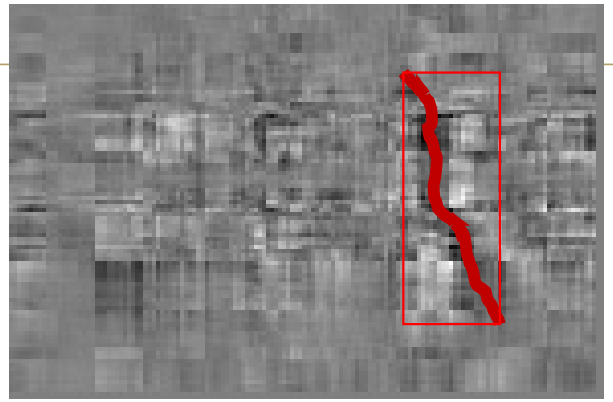
after tree-felling

before tree-felling

Application to other Events

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Golf swing – same neural network
- Event boundaries visible

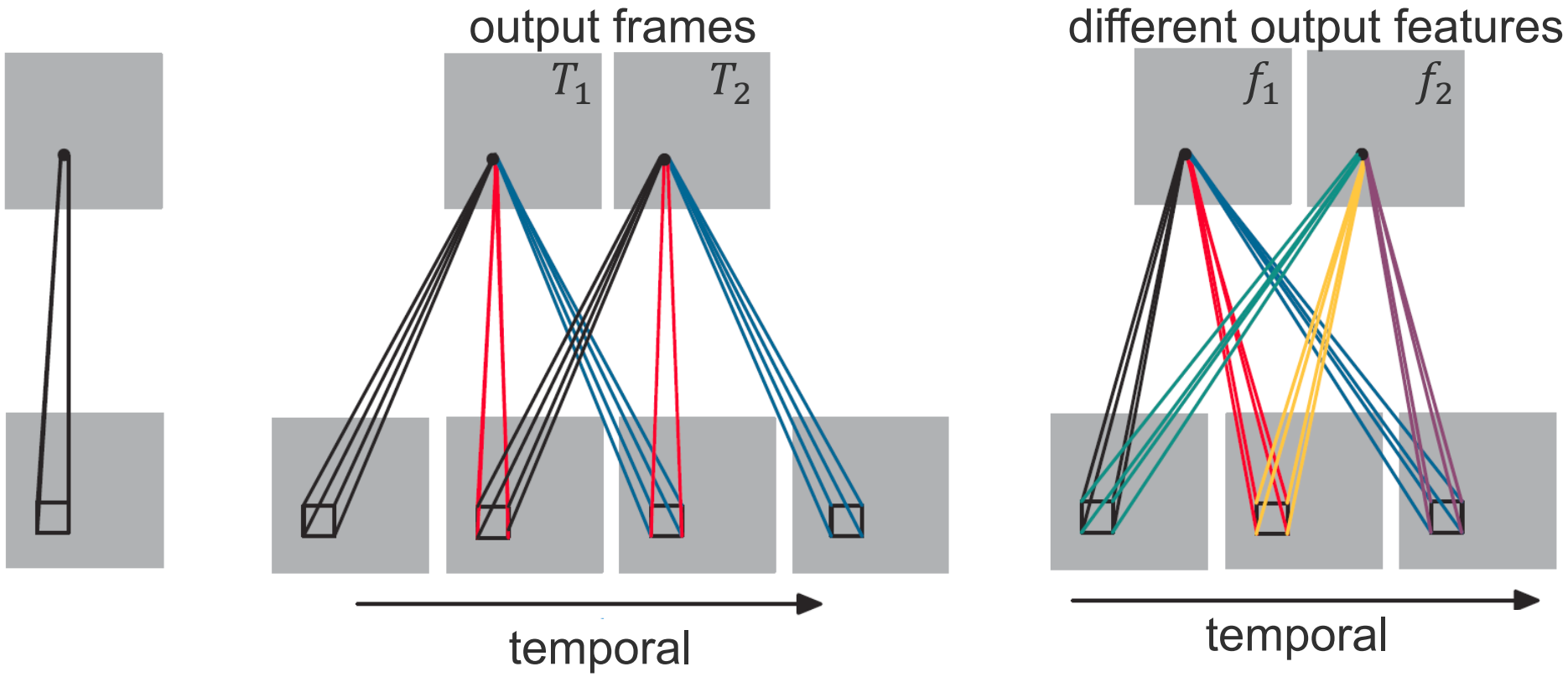


3D Convolution

3D Convolution for Temporal Data

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- [Ji et al. PAMI 2013]
- Extend 2D convolution to include the temporal domain -> 3D convolution



Human Action Recognition

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- 3D convolutional neural networks for human action recognition

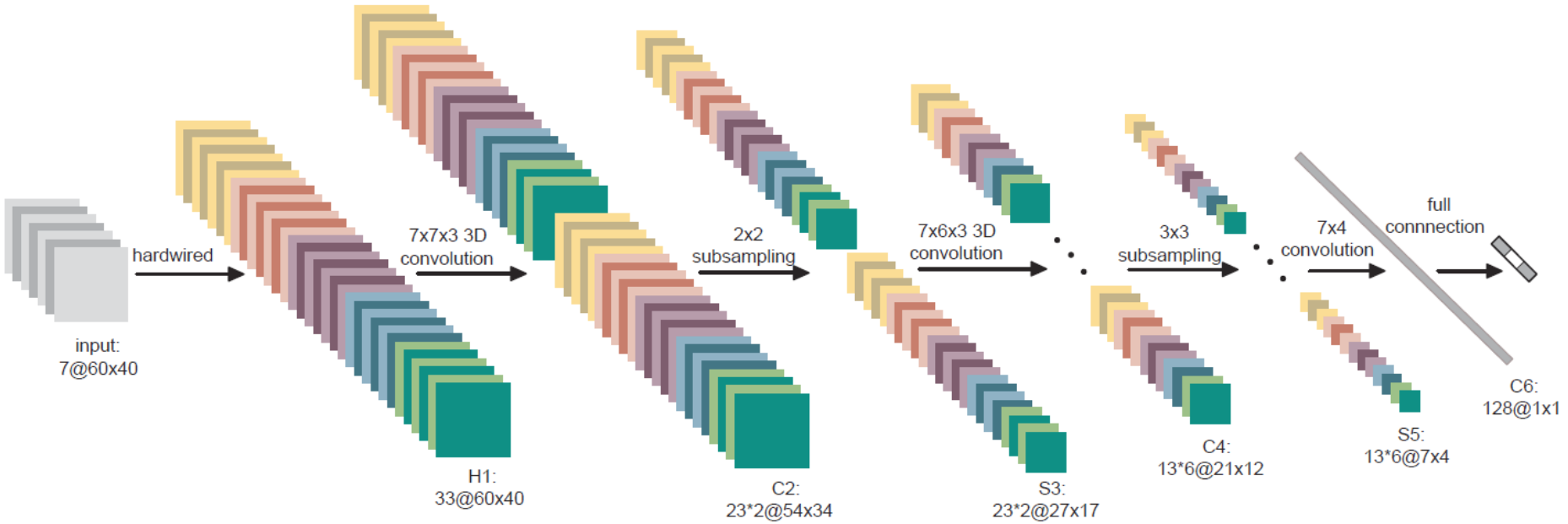


Figure 3. A 3D CNN architecture for human action recognition. This architecture consists of 1 hardwired layer, 3 convolution layers, 2 subsampling layers, and 1 full connection layer. Detailed descriptions are given in the text.

- Inherent memory limitation: cubic volume max. 256x256x256, often smaller

Results

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Tracking

- Recognition accuracy on KTH data set

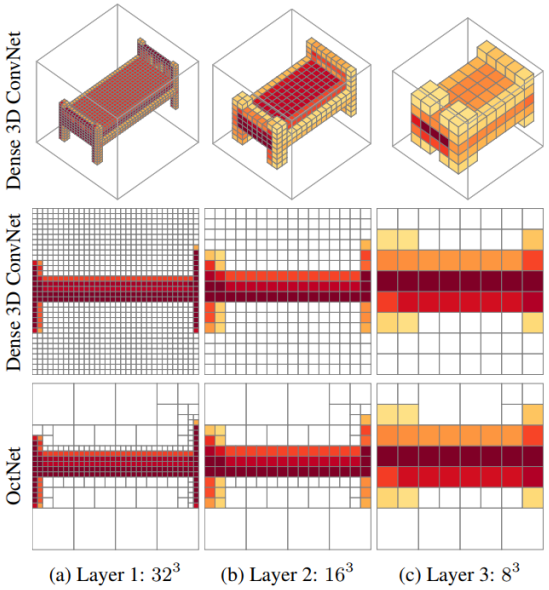
Method	Boxing	Handclapping	Handwaving	Jogging	Running	Walking	Average
3D CNN	90	94	97	84	79	97	90.2
Schüldt	97.9	59.7	73.6	60.4	54.9	83.8	71.7
Dollár	93	77	85	57	85	90	81.2
Niebles	98	86	93	53	88	82	83.3
Jhuang	92	98	92	85	87	96	91.7
Schindler	–	–	–	–	–	–	92.7

3D Deep Learning

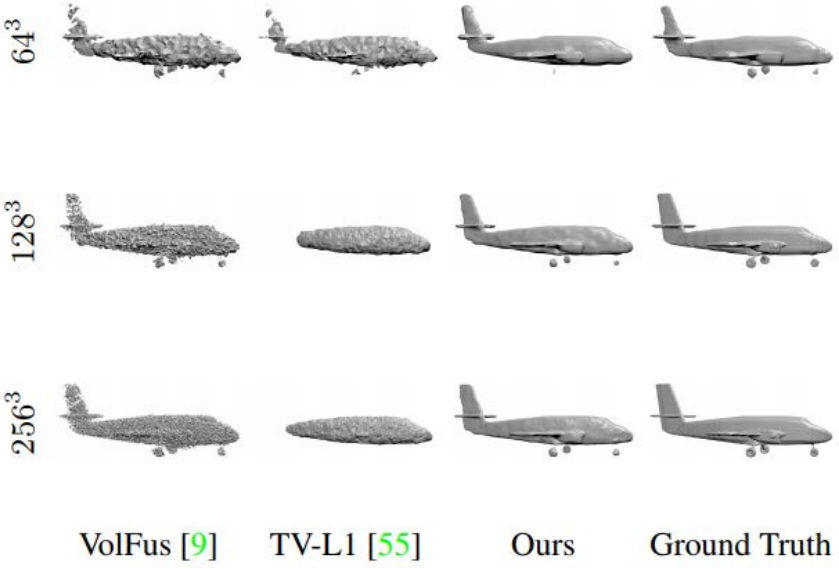
[Andreas Geiger et al.]

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- **OctNet**: 3D ConvNet exploiting sparsity of input (mesh, point cloud) to reach high resolutions (512^3 voxels) without hitting memory limits by using octrees
- **OctNetFusion**: Architecture for learning depth map fusion at high resolution building on the data structures of OctNet



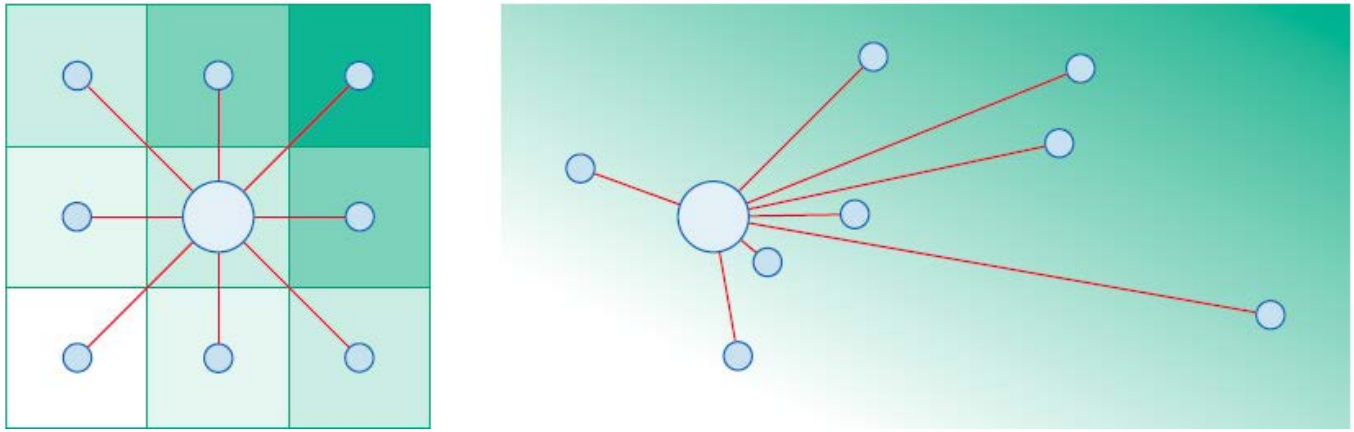
OctNet



OctNetFusion

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

FACULTY OF
SCIENCE
Computer Graphics



Flex-Convolution
Deep Learning Beyond Grid-Worlds

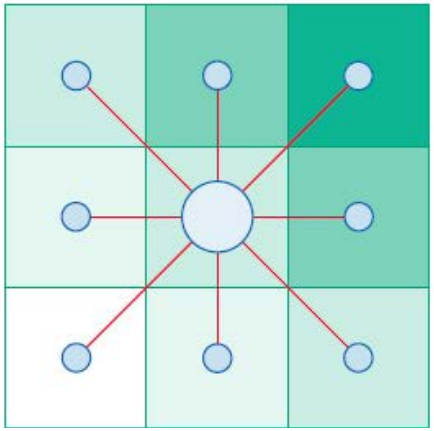
Fabian Groh, Patrick Wieschollek, Hendrik P. A. Lensch

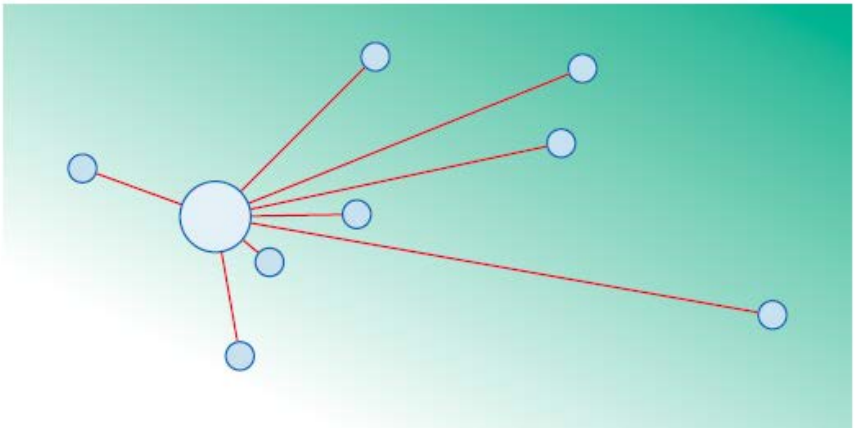
submitted

Flex-Convolution

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Convolution on k-nearest neighbors
- Should be similarly powerful as 2D convolution but for arbitrary dimensions
- Requires kernel weight for continous locations / distances



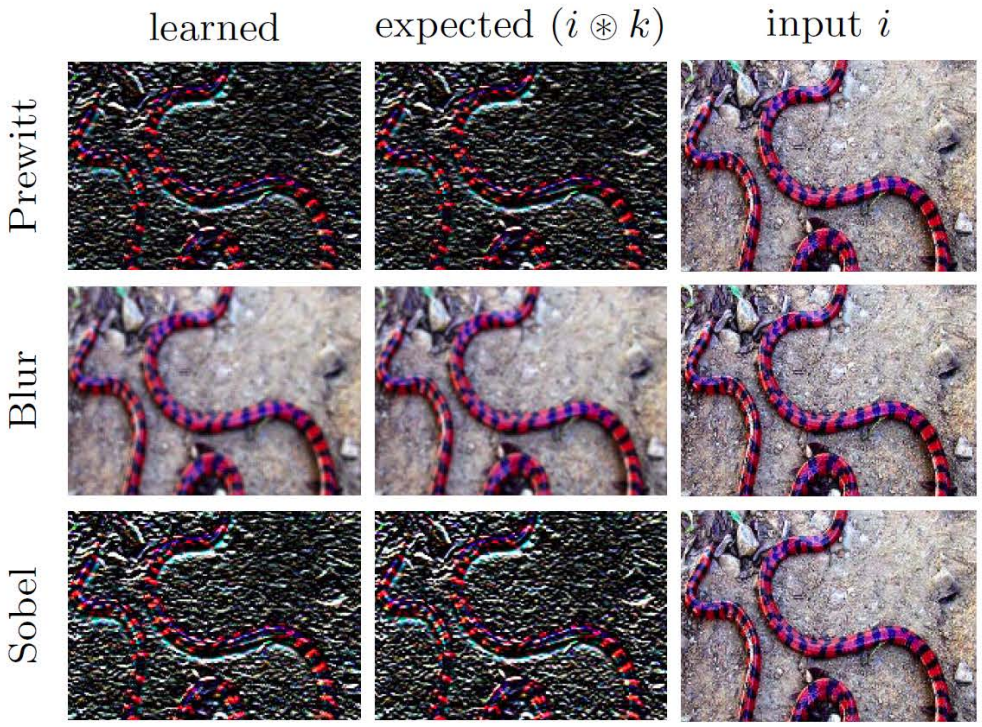


$$f'(c', \ell^{(i)}) = \sum_{c \in C} \sum_{\mathcal{N}_K(\ell^{(i)})} \tilde{w}(c, \ell^{(i)}, \ell_k^{(i)}) \cdot f(c, \ell_k^{(i)})$$

$$\tilde{w}(c, \ell^{(i)}, \ell_k^{(i)} \,|\, \theta_{\ell_c}, \theta_{b_c}) = \left\langle \theta_{\ell_c}, \ell^{(i)} - \ell_k^{(i)} \right\rangle + \theta_{b_c}$$

Flexible Filter Ability

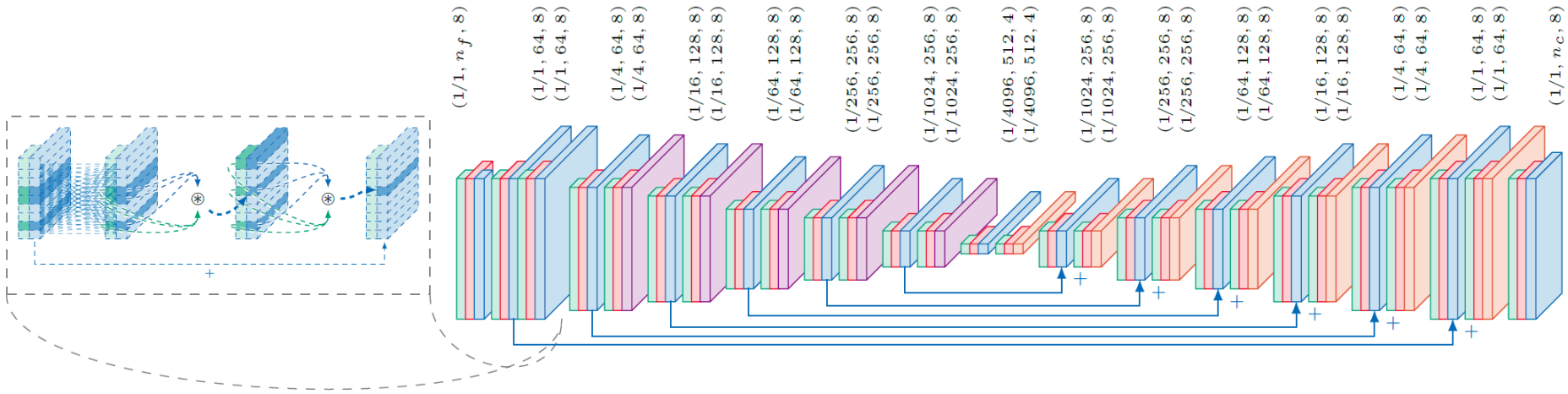
- Flex-Conv can mimic typical image filters



	ground-truth filter k	learned parameters
Prewitt	$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$	$\theta_x = 0.0000$, $\theta_y = 1.0058$, $\theta_b = -0.0004$
Blur	$\frac{1}{N}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	$\theta_x = 0.0002$, $\theta_y = -0.0022$, $\theta_b = 0.1108$
Sobel	$\frac{1}{N}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$	$\theta_x = 0.0018$, $\theta_y = 1.3523$, $\theta_b = -0.0002$

Network Structure

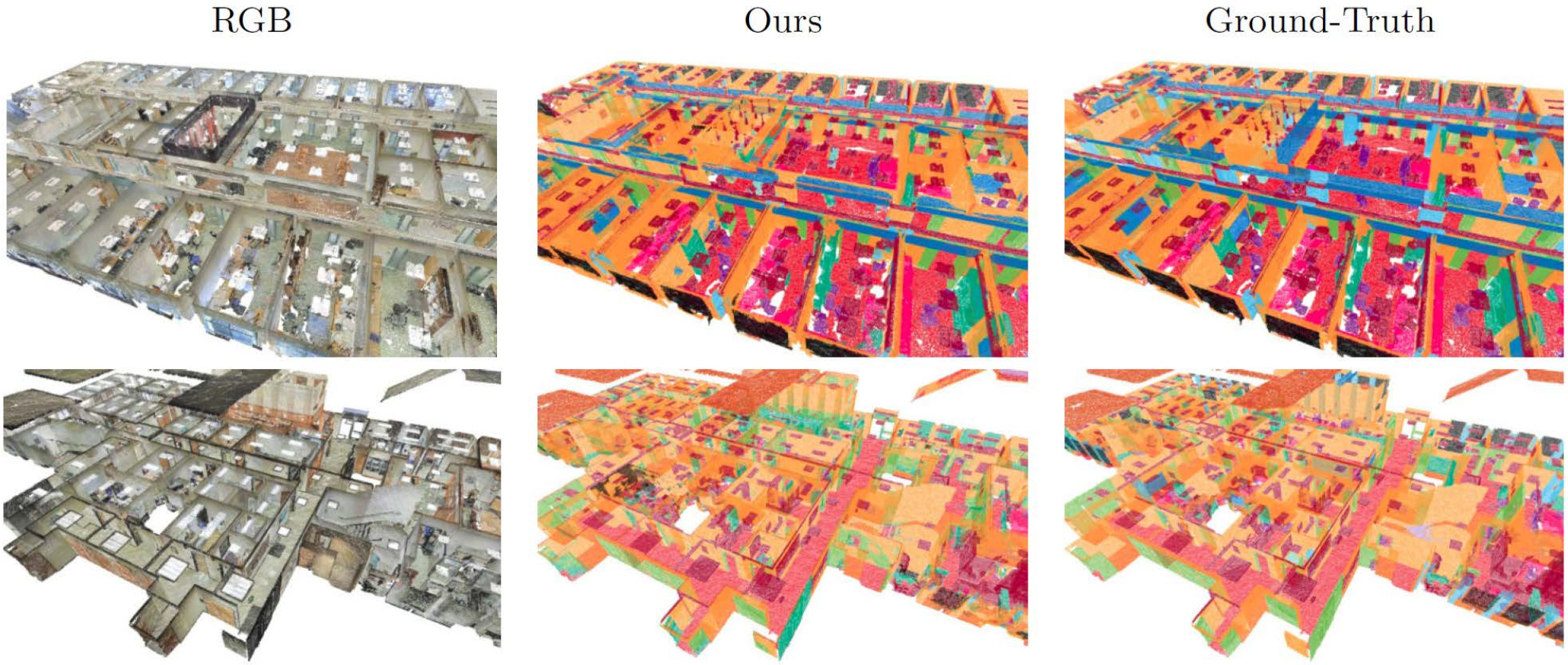
- Main goal of the flex-conv approach is to allow for exactly the same structures as with standard image convolution CNNs
- Here: SegNet with U-Net-Style and Skip Connections
- Specific operators for flexible down and upsampling



Results

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

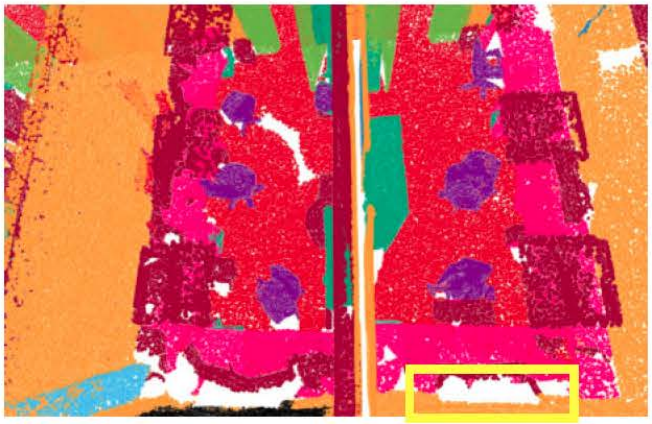
- First CNN-based approach to work on point clouds of million points



Classification Results

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

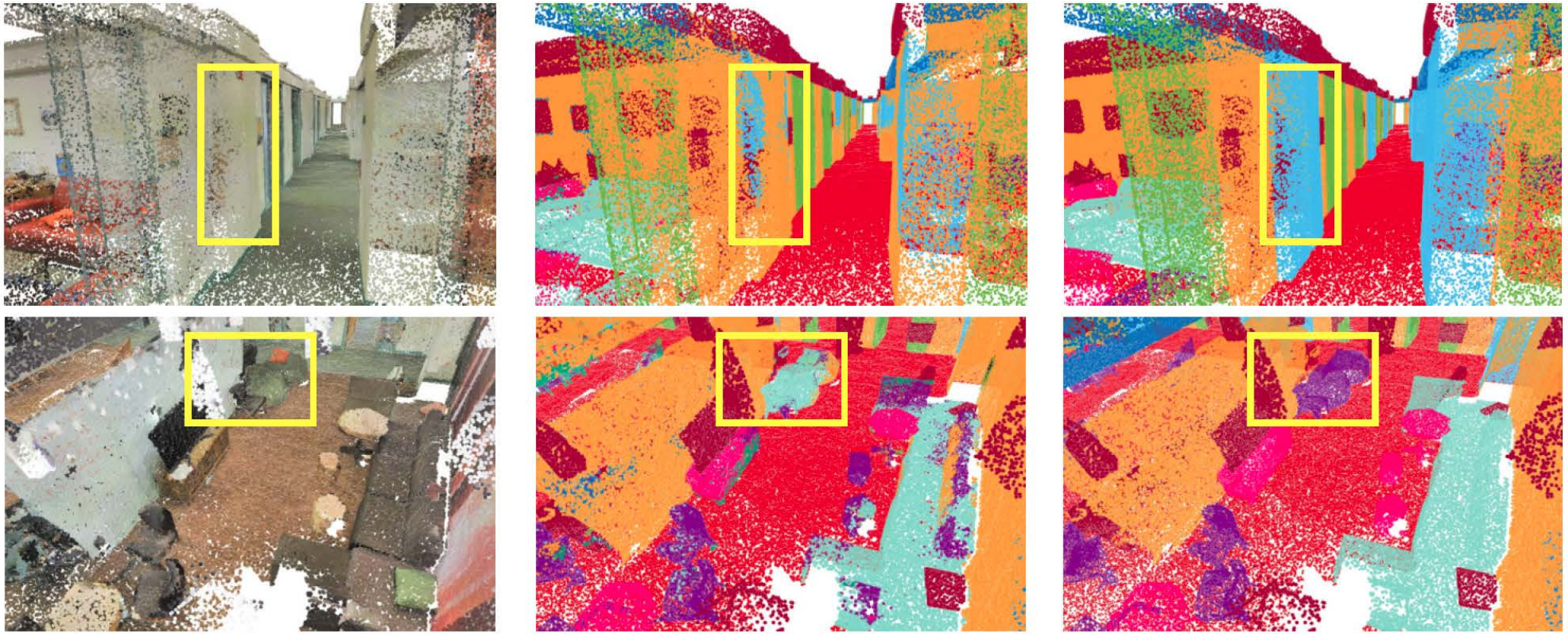
- „wrong“ classification because auf closed blinds



Classification Results

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

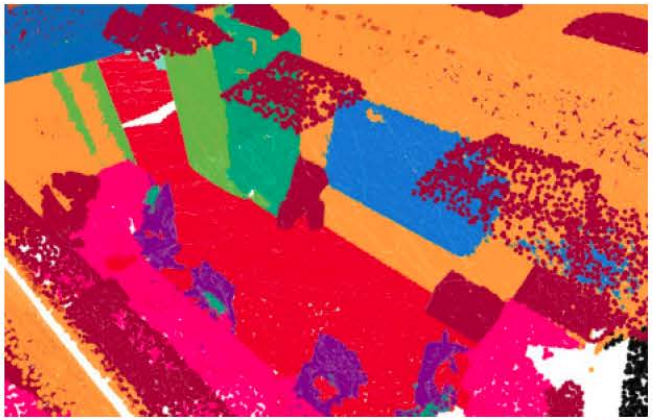
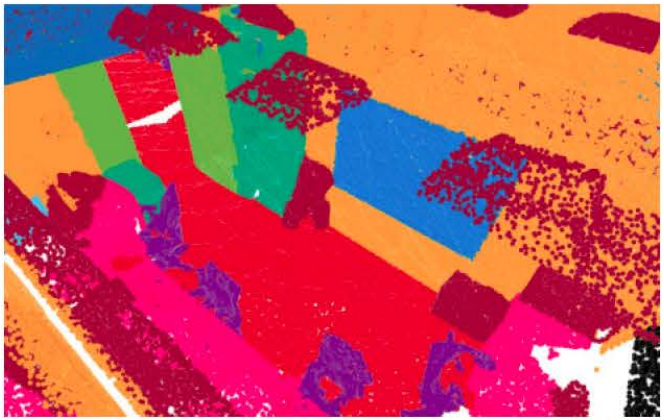
- Columns hard to distinguish from wall - would need propagation of semi-global context
- Seating bag classified as sofa instead of chair

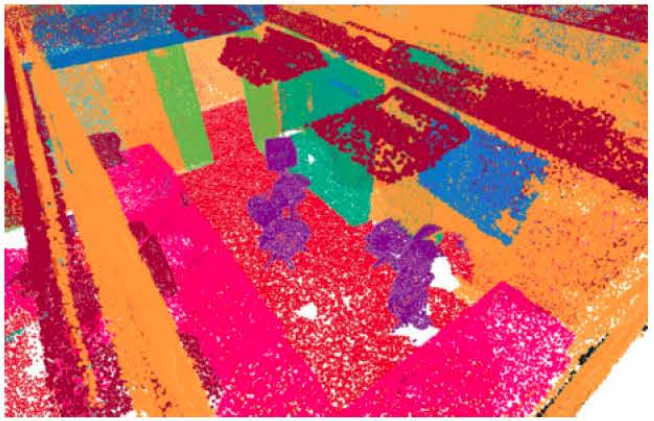
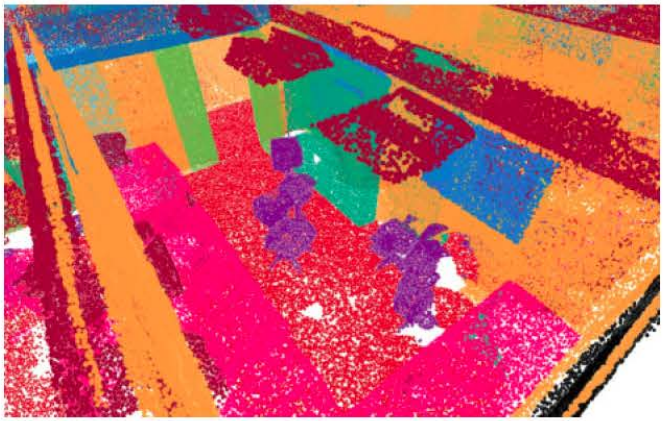


Results

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Almost perfect results in office spaces

References

- Sebastian Ruder. An overview of gradient descent optimization algorithms,
- Jaewan Yun, https://github.com/Jaewan-Yun/optimizer-visualization
- Junyoung Chung, Caglar Gulcehre, KyungHyun Cho and Yoshua Bengio, Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.
- S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Computation, 9(8):1735–1780, 1997
- K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259, 2014. (GRU)
- Shaojie Bai, J. Zico Kolter, Vladlen Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling
- S Ji, W Xu, M Yang, K Yu, 3D convolutional neural networks for human action recognition IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 35, Issue: 1, Jan. 2013)
- Wieschollek P., Hirsch M., Schölkopf B., Lensch H. P.A.(2017). Learning Blind Motion Deblurring. International Conference on Computer Vision (ICCV), pp. 231–240.
- Wieschollek P., Schölkopf B., Lensch H.P.A., Hirsch M. (2016). End-to-End Learning for Image Burst Deblurring. Asian Conference on Computer Vision (ACCV), pp. 35-51.
- Fabian Groh, Patrick Wieschollek, Hendrik P. A. Lensch, Flex-Convolution Deep Learning Beyond Grid-Worlds, arXiv 2018