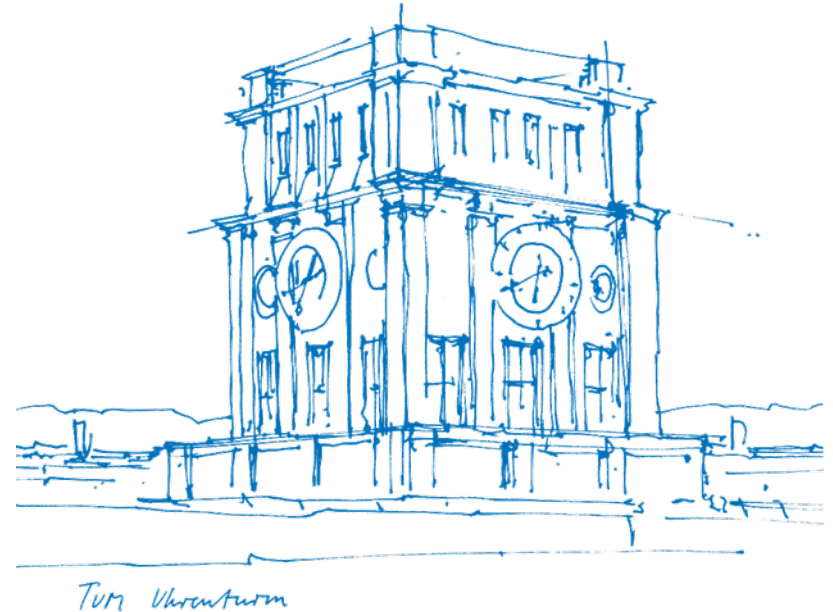
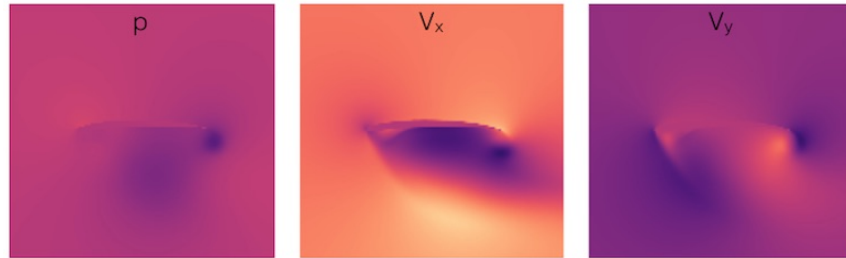


# Deep Learning Methods for Reynolds-Averaged Navier-Stokes Simulations of Airfoil Flows

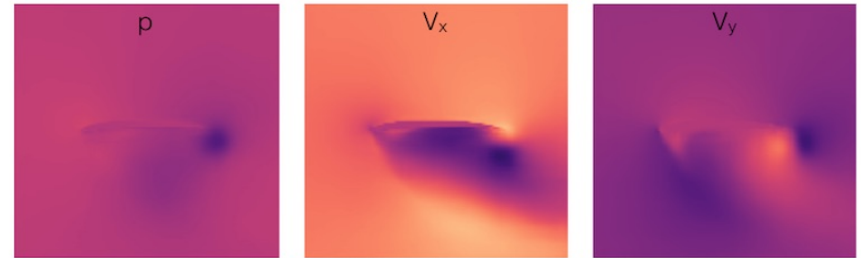
Julian Hohenadel  
Technical University of Munich  
Chair of Computer Graphics and Visualization  
Munich, 11. May 2020



# What this paper is about



Target / ground truth solution



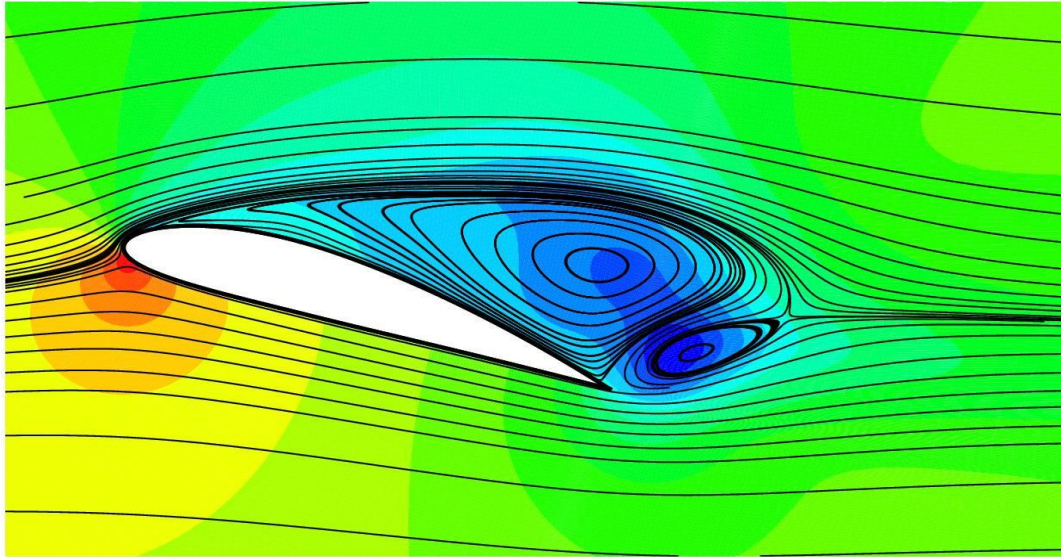
Solution inferred by trained CNN

Taken from <https://github.com/thunil/Deep-Flow-Prediction>

# What this paper is about

- Reynolds Averaged Navier-Stokes solutions
- Airfoil shapes
- Deep learning – CNN
- U-Net architecture derivative
- supervised training environment
- Inference of pressure and velocity fields
- Generalization
- Evaluation

# Background – RANS



Taken from <https://www.pinterest.ch/pin/615163630322034457/>

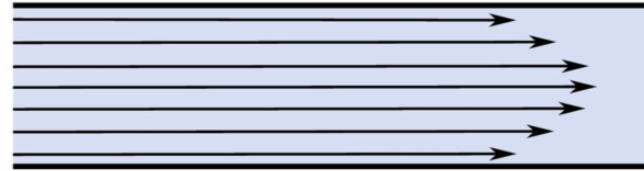
# Background – RANS

Nonlinear partial differential equation (PDE) system  
based on Navier-Stokes equations

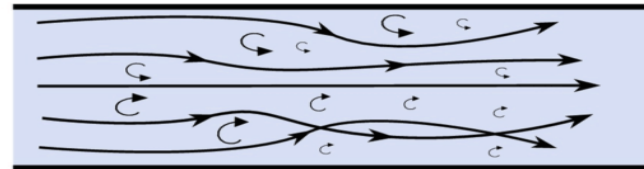
Used for the modeling of turbulent incompressible flows

Averages over time component

**laminar flow**



**turbulent flow**



Taken from <https://diffzi.com/laminar-flow-vs-turbulent-flow/>

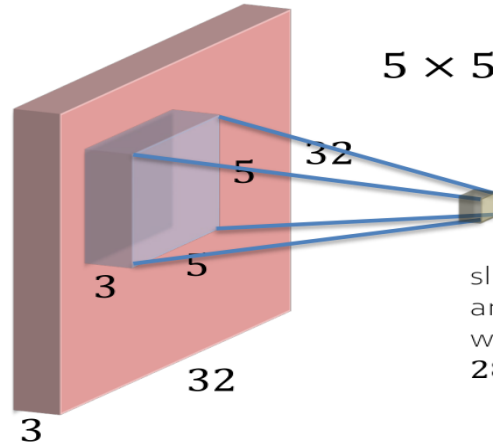
# Background – RANS

**Reynolds number  $Re$ :**

- dimensionless constant
- needed for calculation of turbulence models
- magnitude decides flow (laminar, turbulent)
- affects lift and drag coefficients

# Background – Convolutions

$32 \times 32 \times 3$  image

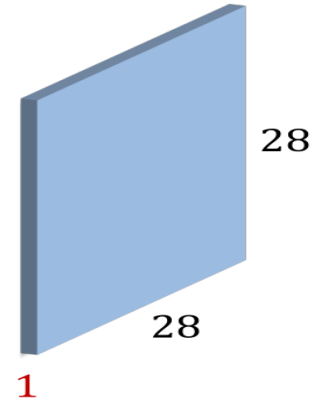


$5 \times 5 \times 3$  filter



slide over all spatial locations  $x_i$   
and compute all output  $z_i$   
w/o padding, there are  
 $28 \times 28$  locations

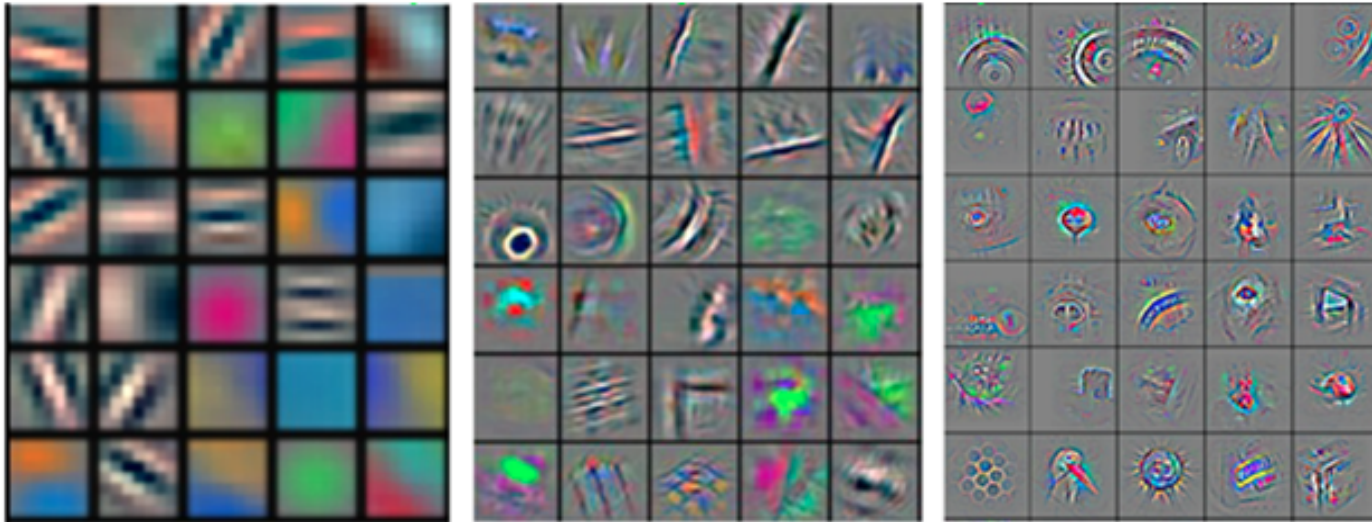
activation map  
(also feature map)



Taken from I2DL WS19/20 (TUM)

# Background – Convolutions

Low-Level Features, Mid-Level Features, High-Level Features: each filter captures different characteristics



Taken from <https://arxiv.org/pdf/1311.2901.pdf>



# Data Generation

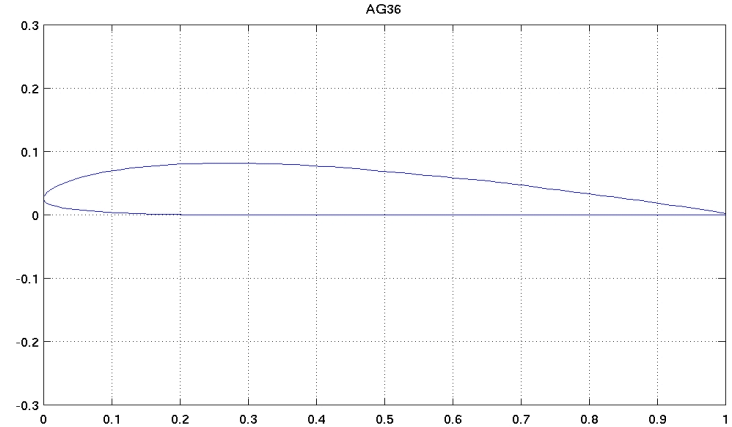
Airfoil shapes are provided by the UIUC database

Reynolds number:  $[0.5, 5] \cdot 10^6$  (highly turbulent)

Angle of attack:  $[-22.5, 22.5]$

Ground truth generated with OpenFOAM  
(pressure, x velocity, y velocity)

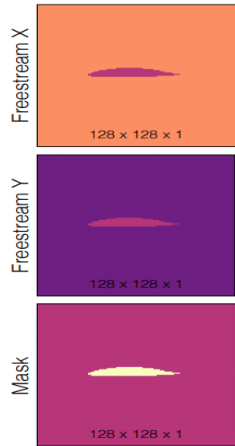
Training data resolution:  $3 \times 128 \times 128$   
(Inference region  $128 \times 128 < \text{full simulation domain}$ )



Taken from <https://m-selig.ae.illinois.edu/ads/afplots/ag35.gif>

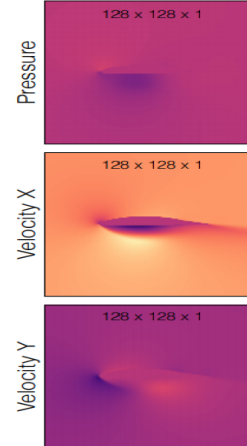
# Pre-processing – Data

## Input channels



Reynolds number encoded as differently scaled freestream velocity vectors wrt. their magnitude

## Target channels



Data from the RANS solution

# Pre-processing – Normalization

Motivation: Flatten space of solutions, accelerate learning by simplifying the learning task for the NN

Normalization of target channels by division with freestream magnitude (vector norm, default: L2):

This makes pressure and velocity dimensionless

$$\tilde{v}_o = \frac{v_o}{\|v_i\|}, \quad \tilde{p}_o = \frac{p_o}{\|v_i\|^2} - \text{important to remove quadratic scaling of pressure}$$

For a better understanding:

Pressure:  $[p]_{SI} = 1 \text{ Pa} = 1 \frac{\text{kg}}{\text{m} \cdot \text{s}^2}$

Density:  $[\rho]_{SI} = 1 \frac{\text{kg}}{\text{m}^3}$  – constant in incompressible flow

Velocity:  $[v]_{SI} = \frac{\text{m}}{\text{s}}$

# Pre-processing – Offset removal & value clamping

Motivation: eliminate ill-posed learning goal & improve numerical precision

Spatially move pressure distribution into the origin – RANS typically only needs  $\nabla_p$  for computation

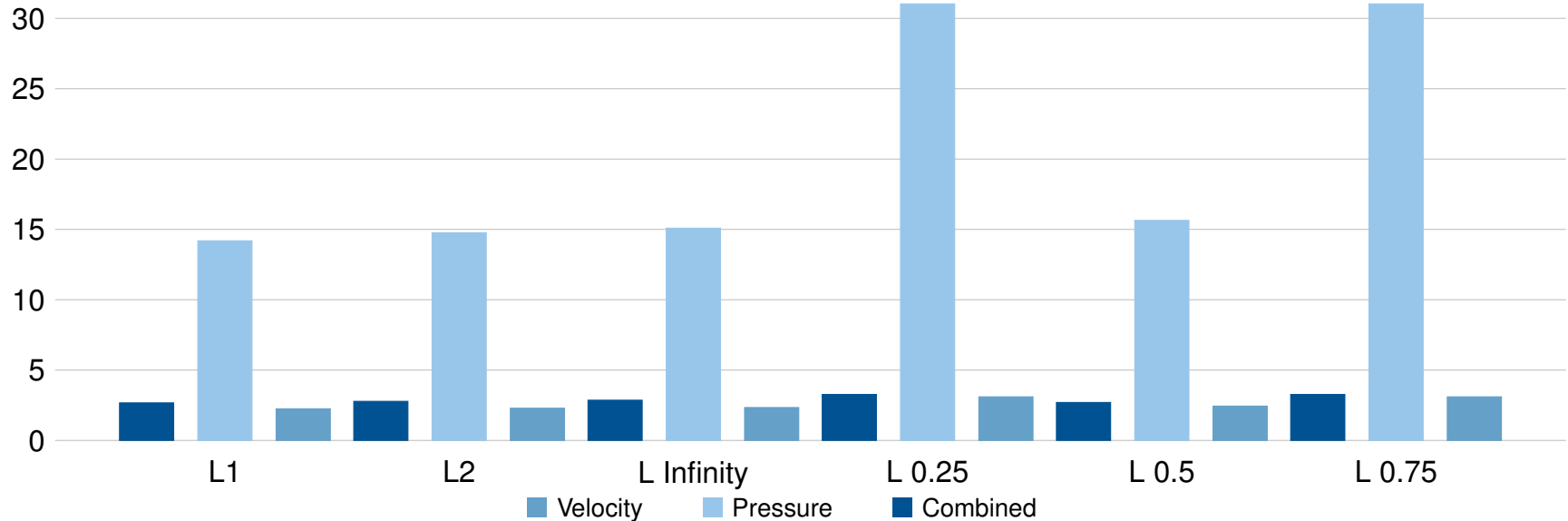
$$\hat{p}_o = \tilde{p}_o - p_{mean}$$

Clamp both input and target channels into  $[-1, 1]$  range by diving by the maximum absolute value

# Pre-processing – Evaluation

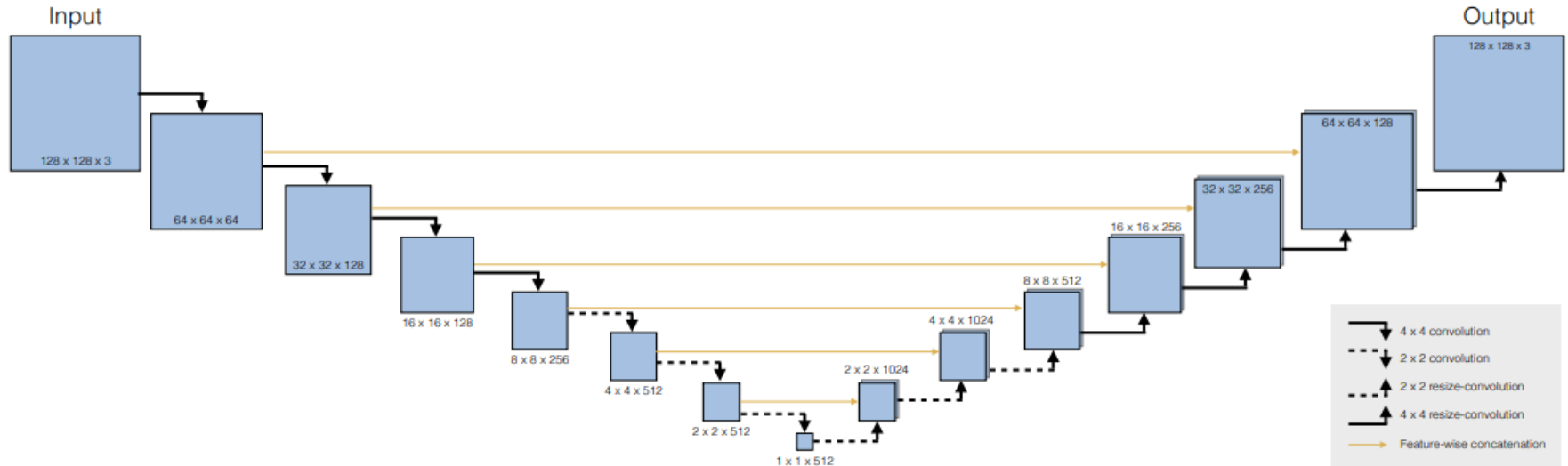
Vector norms used in pre-processing comparison wrt. error, default: L2 (in %)

L1 normalization achieves the best error rates (p, vel, combined: **14.19%**, **2.251%**, **2.646%** – L2: 14.76%, 2.291%, 2.780%)



# Architecture

U-Net derivative proposed in the paper:



Taken from <https://arxiv.org/pdf/1810.08217.pdf>

# Architecture – Convolutional blocks

## Encoder

1. Activation – Leaky ReLu (0.2)
2. Convolution – Width down, Depth up
3. Batch normalization
4. Dropout (1%)

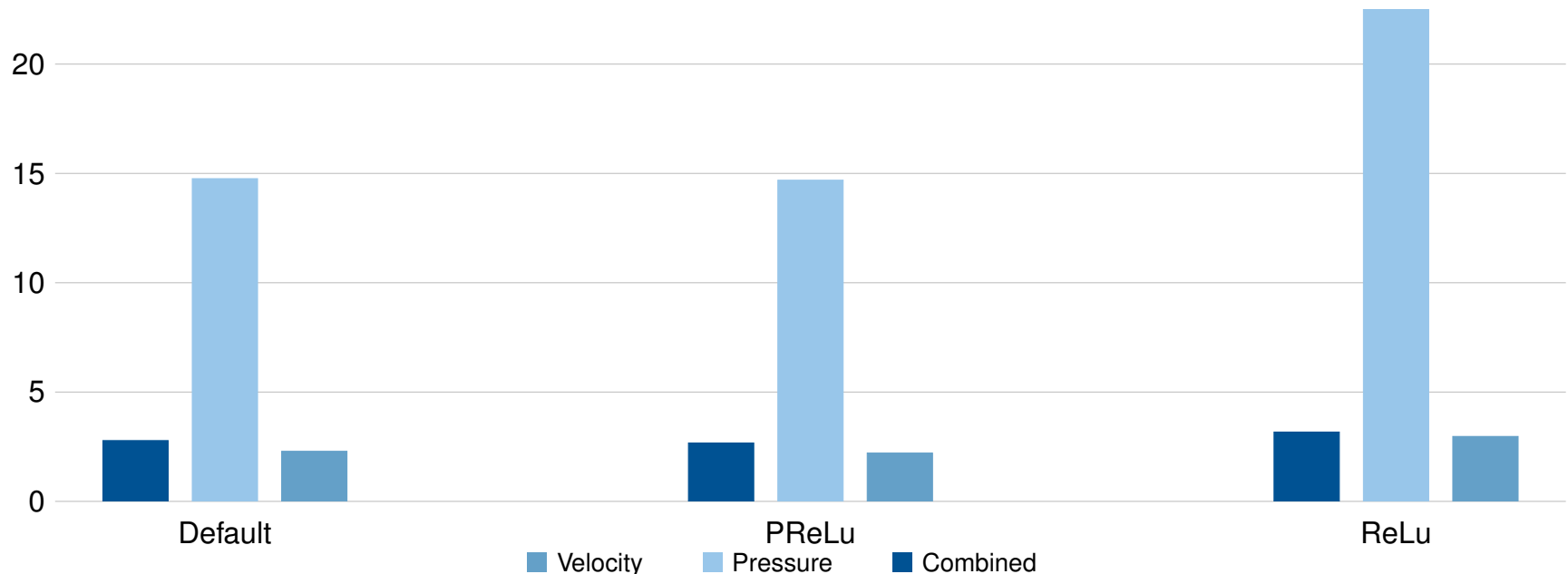
## Decoder

1. Activation – ReLu
2. Upsampling – linear (2.0)
3. Convolution – Width up, Depth down
4. Batch normalization
5. Dropout (1%)

# Architecture – Evaluation

Error percentage of different activation functions after 160k iterations (266 epochs).

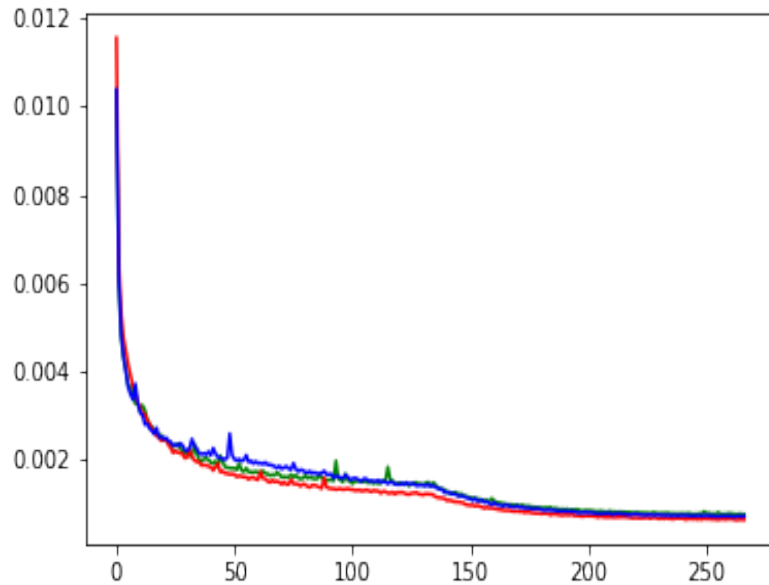
PReLU achieves the best error rates (p, vel, combined: **14.69%**, **2.216%**, **2.676%** – Default: 14.76%, 2.296%, 2.787%)



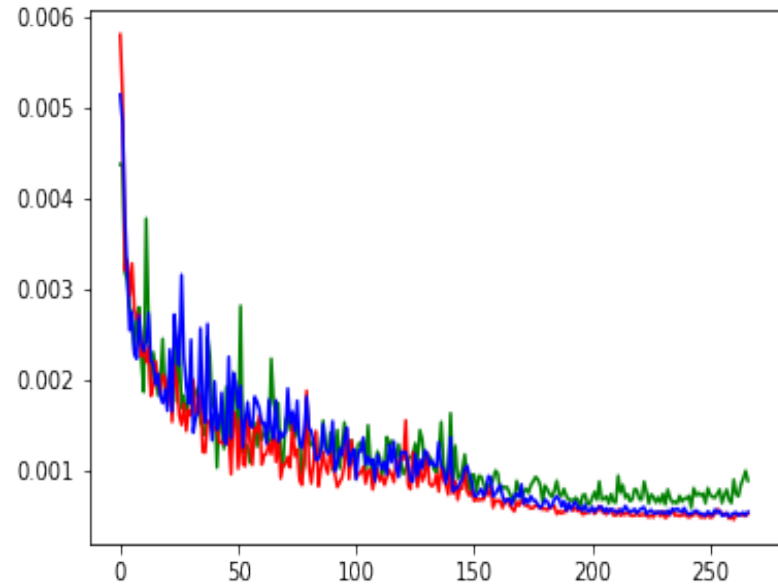


# Architecture – Evaluation

Training loss



Validation loss



# Transfer

Motivation: Can the network architecture adapt to other PDE systems and how well will it perform?

Another use case for PDE systems like RANS is predicting wave propagation on shallow water

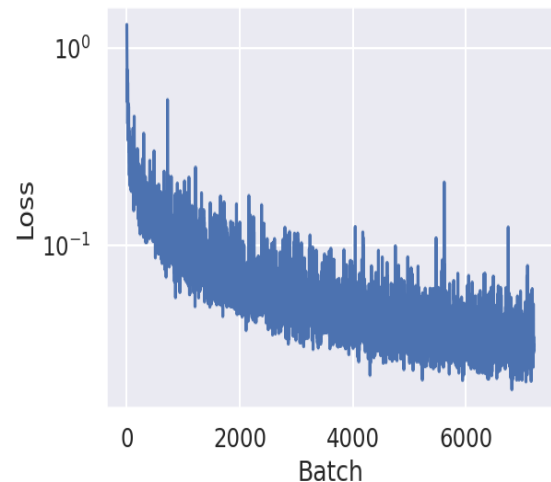
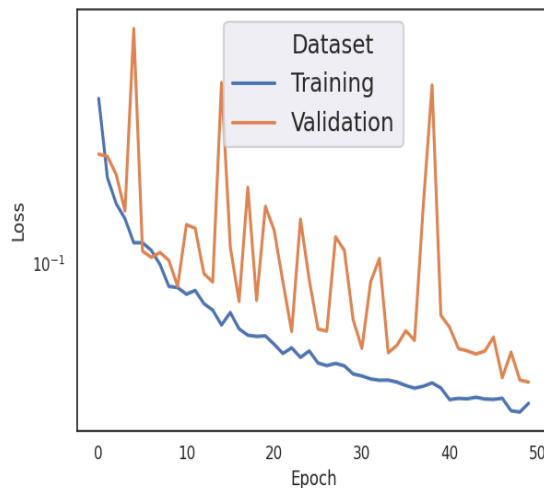
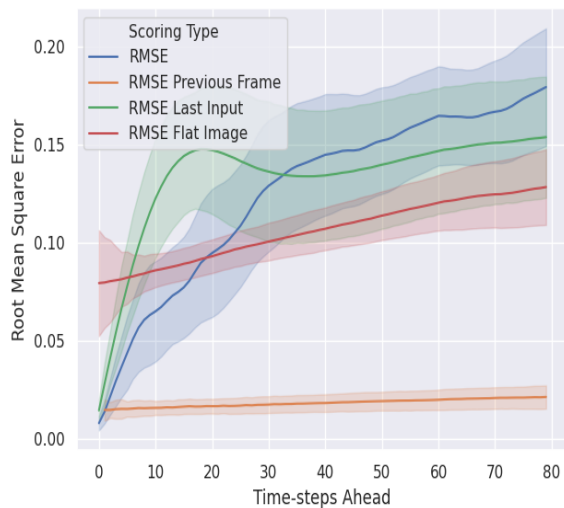
Wave propagation, in this case, is governed by the Saint-Venant equations (related with Navier-Stokes equations)

U-Net architecture changes:

- Input channels – contain the last  $n$  time steps
- Output channels – predict the next  $m$  time steps
- Output is refeeded as input to predict time series

# Transfer – Evaluation

RMSE with variance, validation loss and batch loss on Bigger Tub environment:

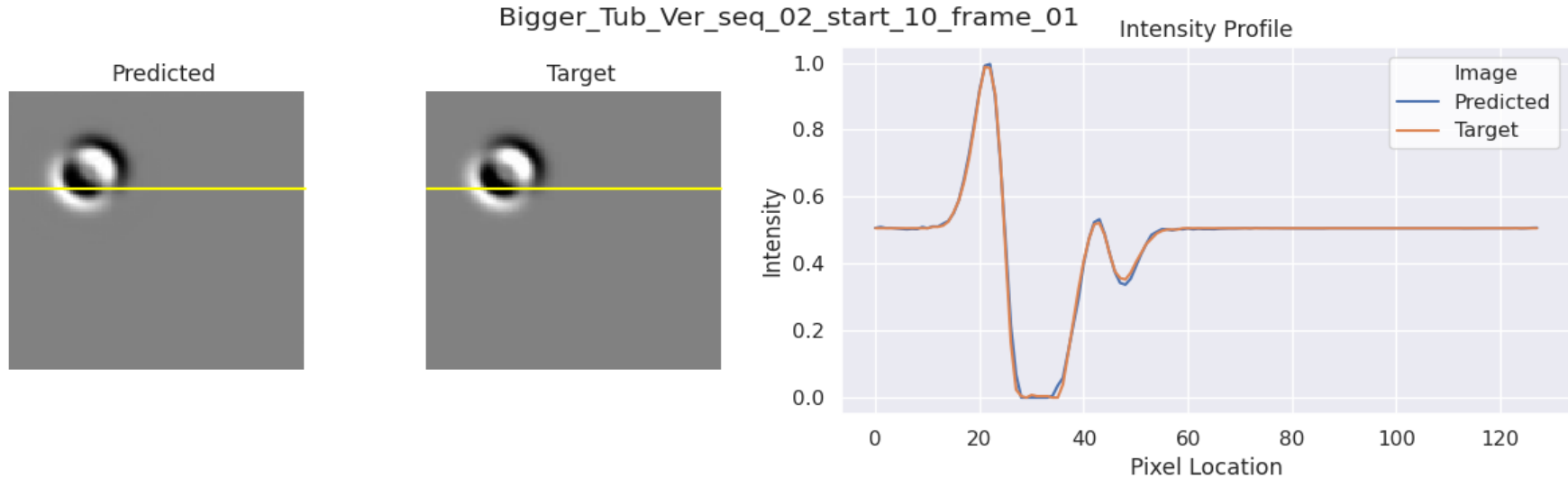


All plots and training in Transfer were made with [https://github.com/stathius/wave\\_propagation](https://github.com/stathius/wave_propagation)

# Transfer – Evaluation

Wave propagation prediction

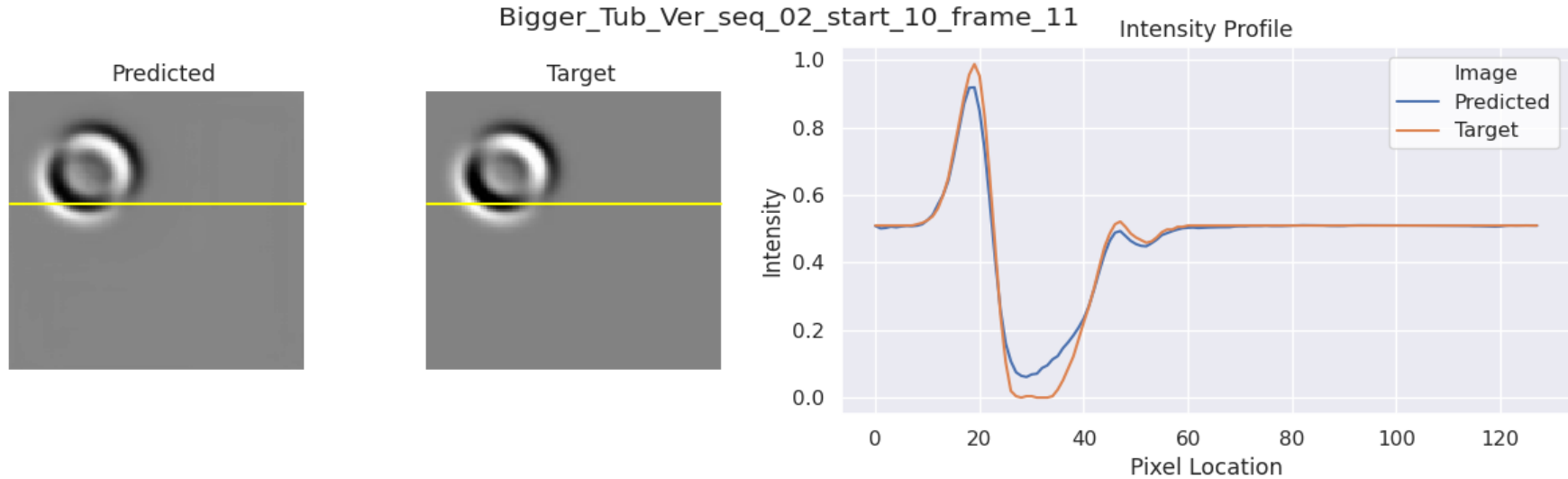
Intensity profile on scanline – Frame 1



# Transfer – Evaluation

Wave propagation prediction

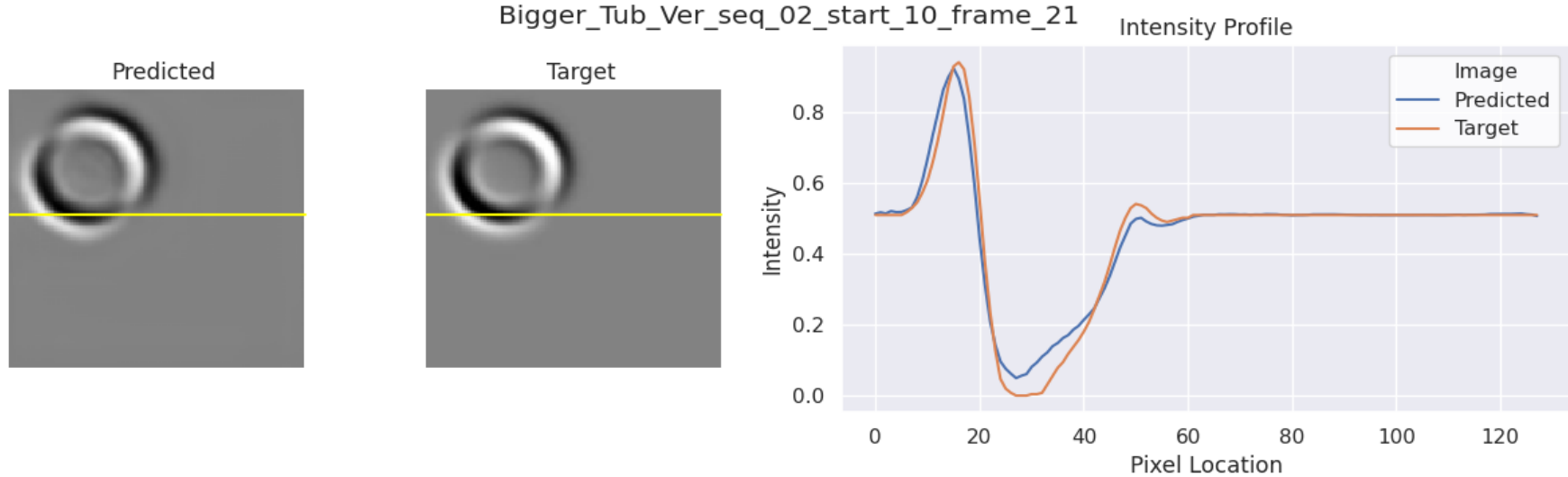
Intensity profile on scanline – Frame 11



# Transfer – Evaluation

Wave propagation prediction

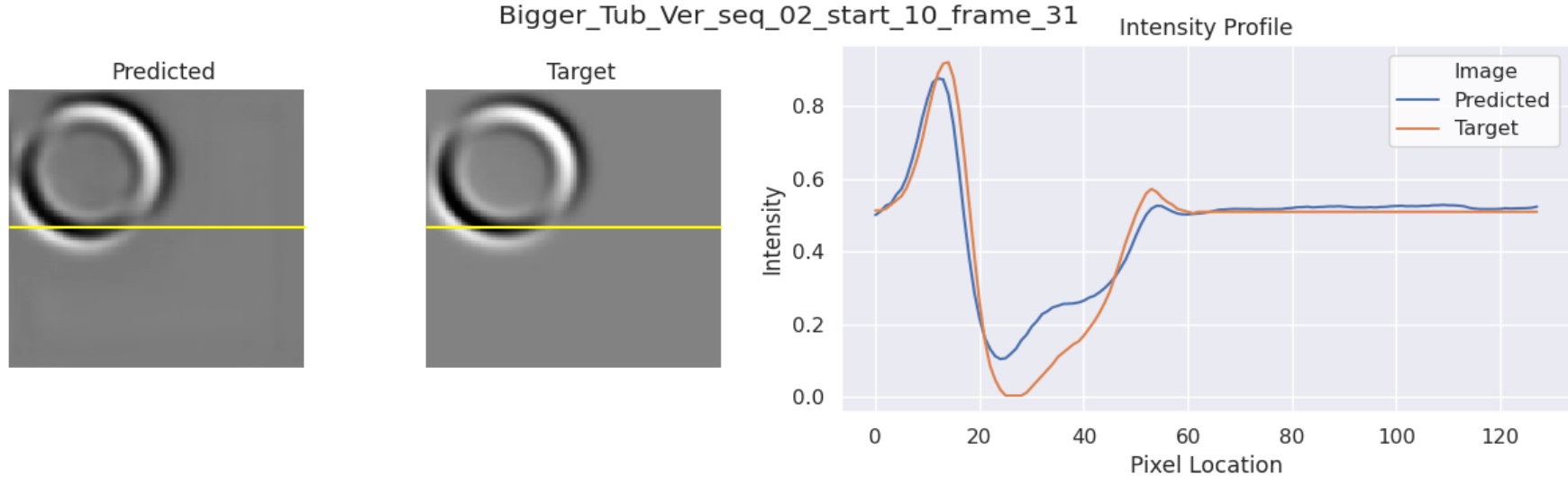
Intensity profile on scanline – Frame 21



# Transfer – Evaluation

Wave propagation prediction

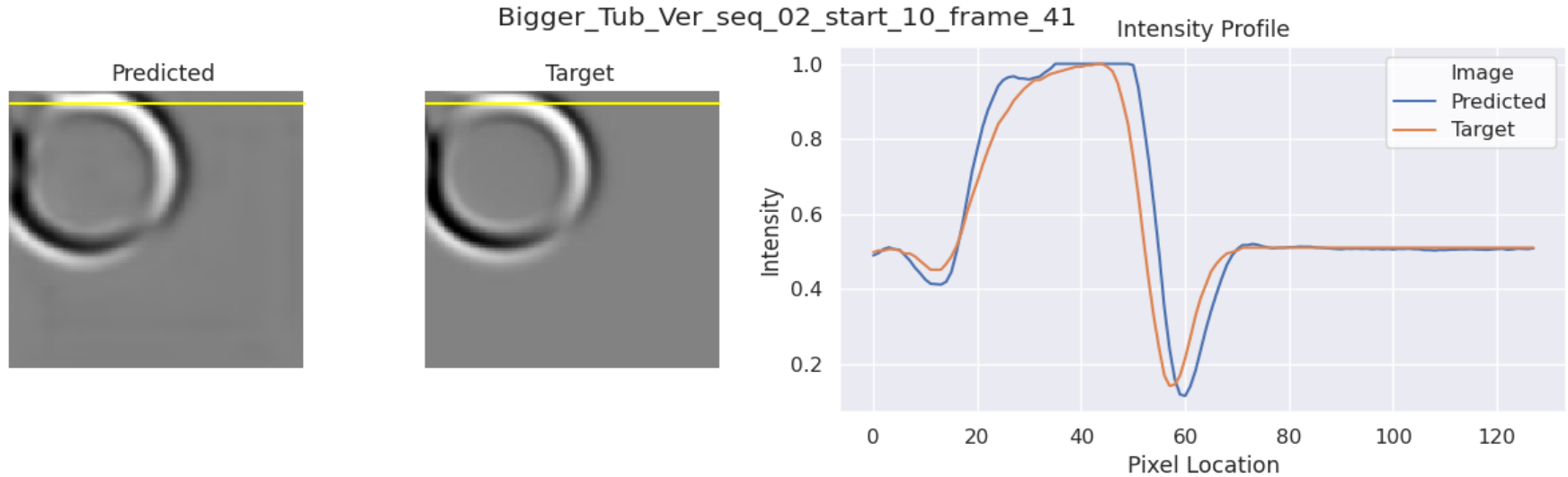
Intensity profile on scanline – Frame 31



# Transfer – Evaluation

Wave propagation prediction

Intensity profile on scanline – Frame 41

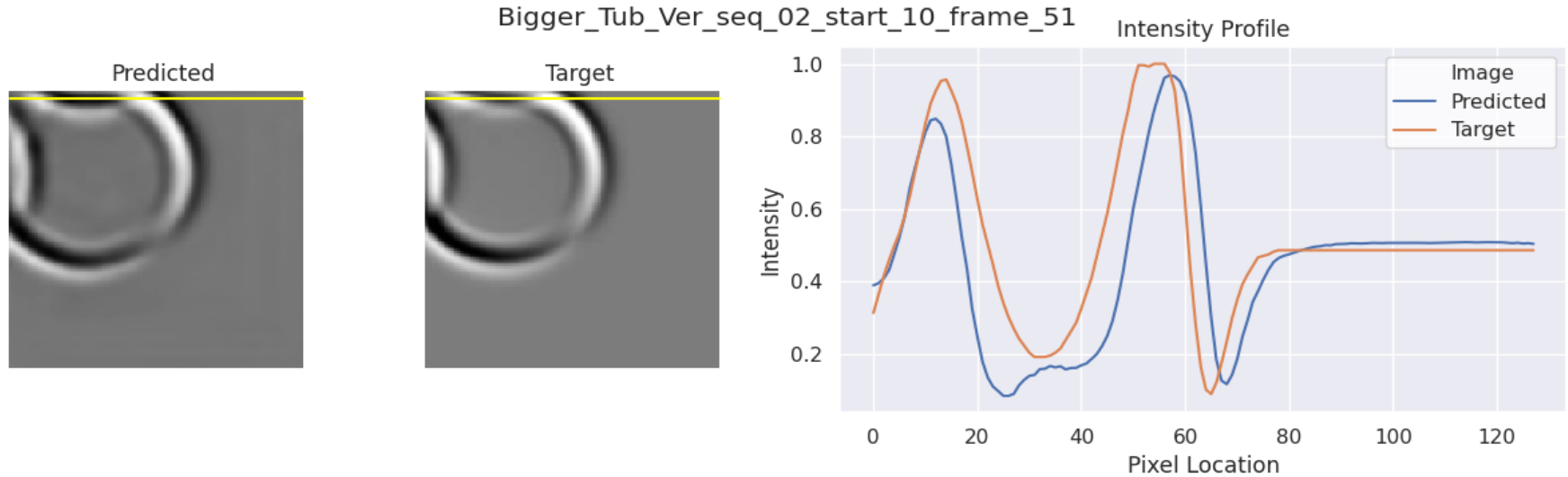




# Transfer – Evaluation

Wave propagation prediction

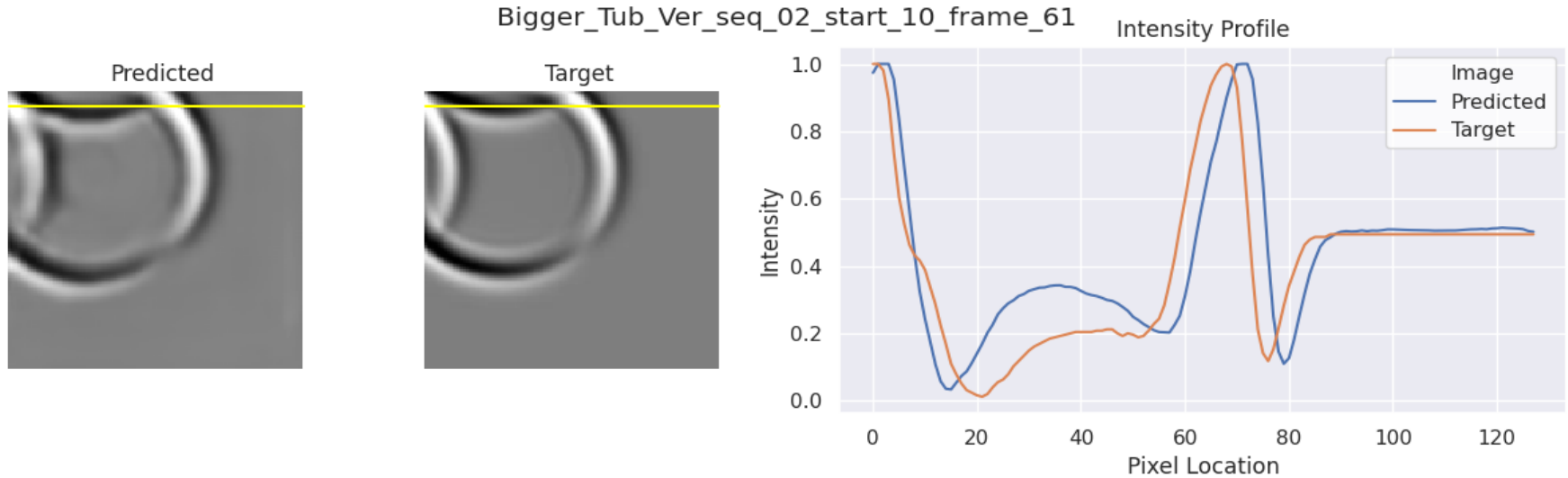
Intensity profile on scanline – Frame 51



# Transfer – Evaluation

Wave propagation prediction

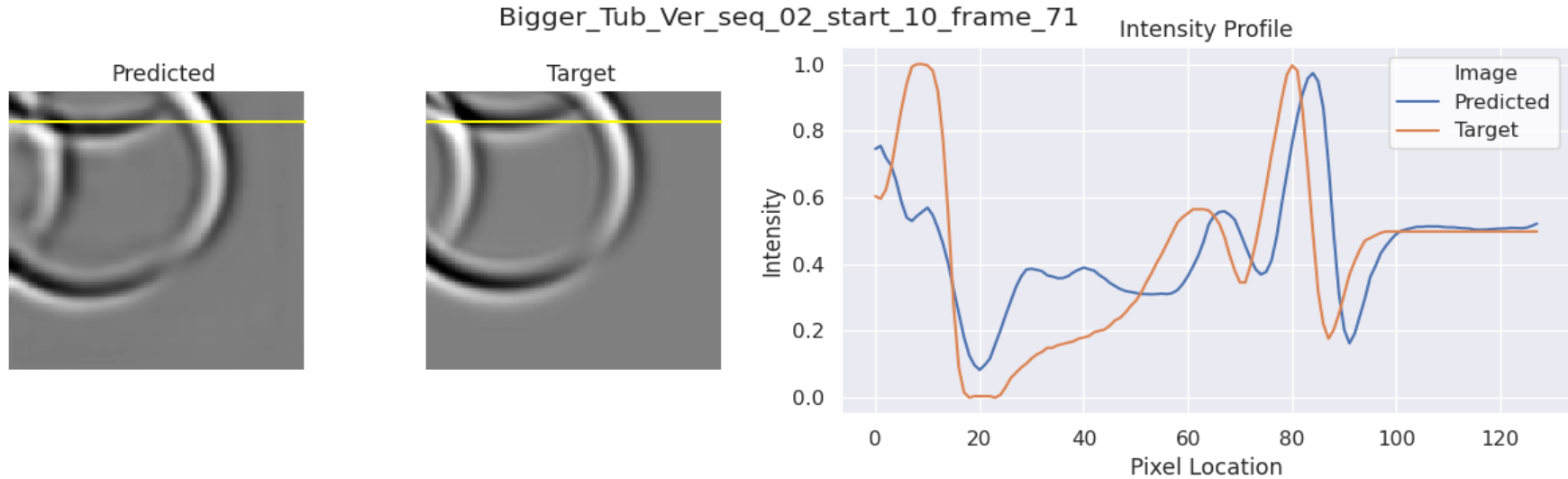
Intensity profile on scanline – Frame 61



# Transfer – Evaluation

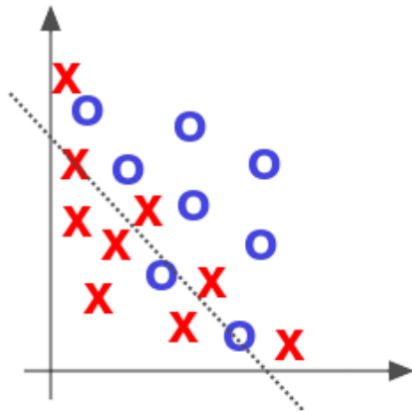
Wave propagation prediction

Intensity profile on scanline – Frame 71

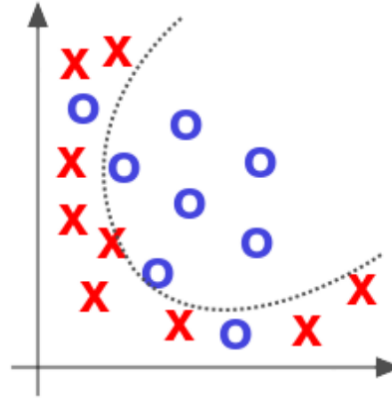


# Generalization

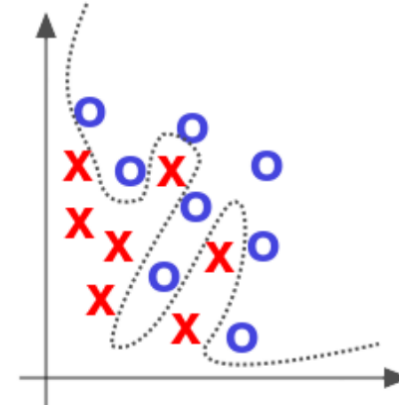
Motivation: Key question in deep learning: How well does my NN perform on unseen data?



Underfitted



Appropriate



Overfitted

Taken from Deep Learning by Adam Gibson, Josh Patterson, O'Reilly Media Inc., 2017

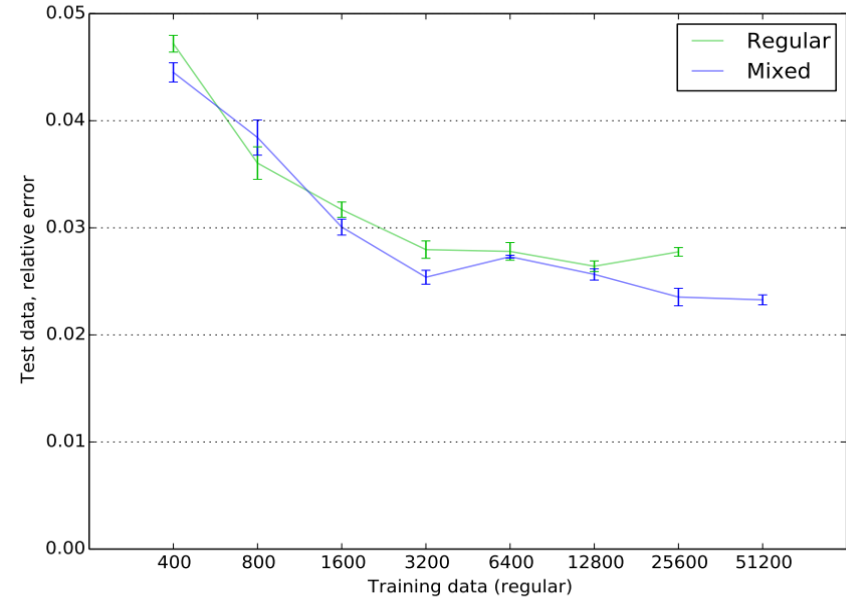
# Generalization

Split up training data:

- Regular
- Mixed (50% regular, 50% sheared ( $\pm 15$  degrees))

The plot shows training with a  $30.9 \cdot 10^6$  parameter model

The high capacity supports training with the mixed dataset, achieving a even lower error



Taken from <https://arxiv.org/pdf/1810.08217.pdf>

# Generalization – Evaluation

To test generalization new datasets need to be created

Generation of 1 training sample:  $\approx 70$  seconds (using Google Colab)

Generation of 12.8k training samples:  $> 10$  days – not feasible

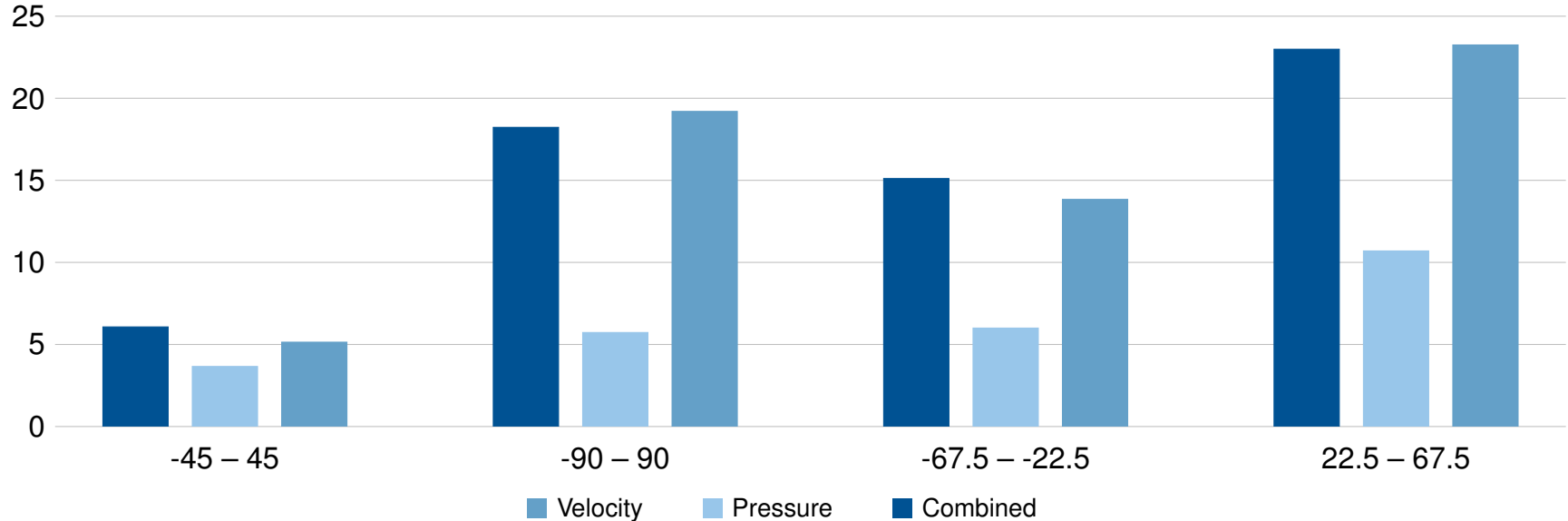
$\implies$  Generate only new test sets containing 90 samples:  $< 2$  hours – feasible

Extrapolation with different angle of attack intervals, default:  $[-22.5, 22.5]$ :

- $[-45, 45]$
- $[-90, 90]$
- $[-67.5, -22.5]$
- $[22.5, 67.5]$

# Generalization – Evaluation

Error increase of different angle of attack intervals wrt. ground truth  $[-22.5, 22.5]$



# Discussion

## Positiv

- Relative error  $< 3\%$
- Convolutions paired with an encoder decoder structure seem to catch regions of interest fast and reliable
- U-Nets can outperform LSTM's in accuracy as well as in speed with a fraction of capacity (in time-series problems)
- accuracy does not suffer too much from models with a lot less capacity mostly affects sharpness of solutions
- Inference speed is  $1000\times$  when compared with OpenFOAM solver
- Accuracy improvements still possible (bigger models, more training data)



# Discussion

## Negativ

- Prior knowledge needed for proper pre-processing
- Solvers needed for dataset generation
- Extrapolation yields mediocre results
- Fresh Training needed for other shapes (e.g. cars in wind tunnel)
  - transfer learning unlikely
- Trade off: training speed – grid resolution
- Possible data loss from transformation:
  - adaptive grid (solver)  $\implies$  cartesian grid (NN)
- no guarantee for correctness
- Accuracy improvements computationally expensive likely requires
  - tailored architectures and loss functions

# Summary

Investigate the accuracy of U-Net models for the inference of Reynolds-Averaged Navier-Stokes solutions

## Data Generation $6 \times 128 \times 128$

- Input (encodes Reynolds number): Bit Mask, x & y velocity
- Target (RANS solution): Pressure, x & y velocity

## Pre-Processing

- Make data dimensionless, flatten space of solutions
- Pressure offset removal, numerical precision

## Architecture

- U-Net – Encoder - Bottleneck - Decoder structure
- Activations highly depend on current task

## Transfer

- U-Net as time-series prediction NN for wave propagation
- Input: last  $n$  frames, Output: next  $m$  frames, refeed

## Generalization

- NN performance on unseen data: Different angels of attack
- mediocre performance on wider intervals (velocity)

## Discussion

- low error (improvable), speed up, even with low capacity
- prior knowledge and solvers needed, poor extrapolation

# Backup slides

# Backup slides – Training Setup

Adam optimizer ( $\beta_1 = 0.5, \beta_2 = 0.999$ )

Learning rate: 0.0004

Learning rate decay: On

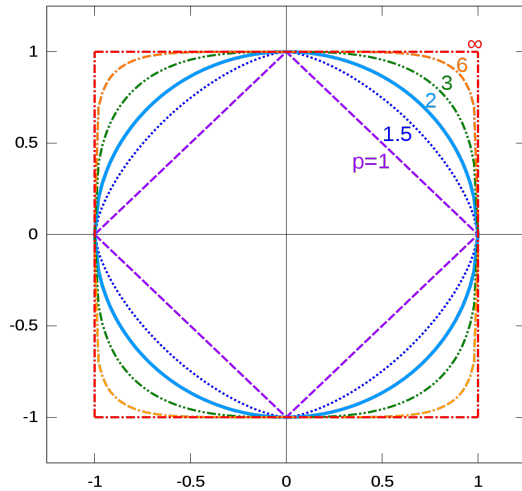
Batch size: 10

Iterations: 80000

Model parameters:

122.979, 487.107, 1.938.819, 7.736.067, 30.905.859

# Backup slides – Norms on unit circle



Taken from: [https://de.wikipedia.org/wiki/Norm\\_\(Mathematik\)#/media/Datei:Vector-p-Norms\\_qtl1.svg](https://de.wikipedia.org/wiki/Norm_(Mathematik)#/media/Datei:Vector-p-Norms_qtl1.svg)