Can Neural Networks Predict the Flow around Blunt Bodies?

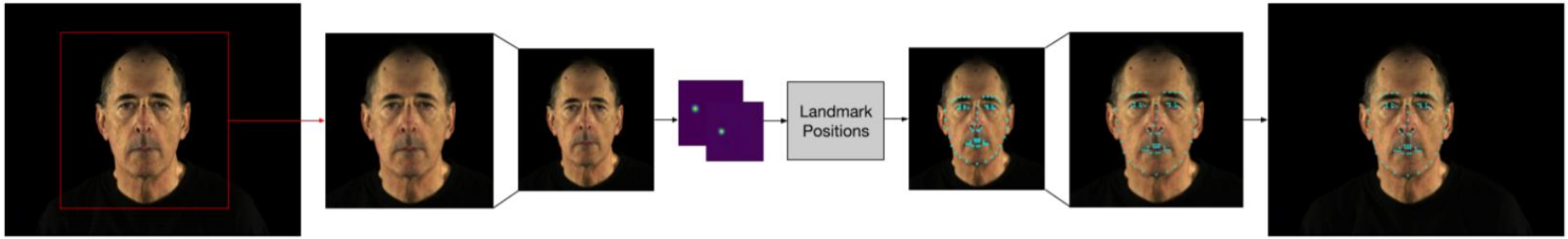
Ali Kashefi
Aashwin Mishra
Gianluca Iaccarino

Department of Mechanical Engineering & Center for Turbulence Research (CTR)

Motivation

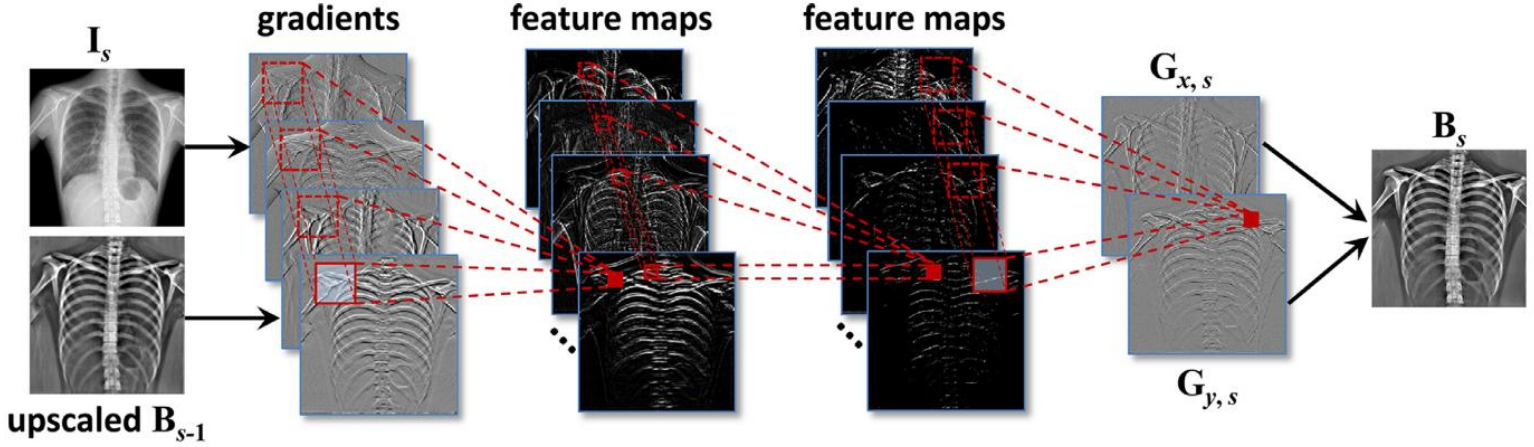
Computer Graphics + Deep Learning

(Bao et al. 2018, Stanford)



Medicine + Deep Learning

(Yang et al. 2016, Southern Medical University)

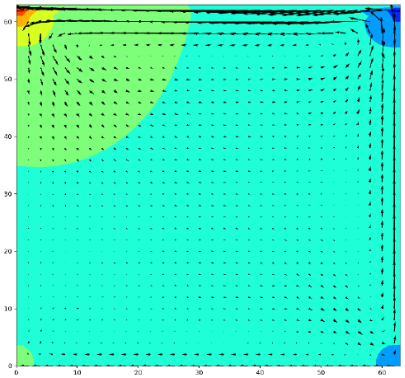


How about

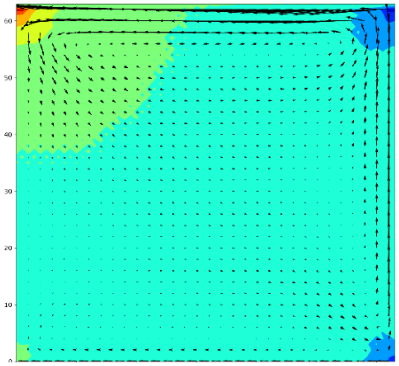
Computational Mechanics + Deep Learning?!

Review

Ground truth



Deep Learning Model

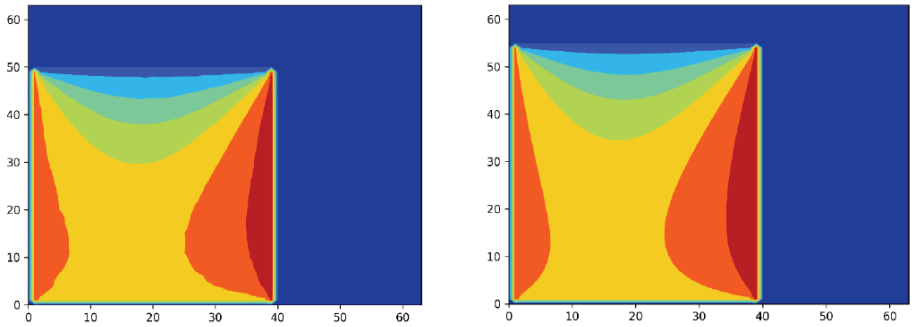


Cavity flow

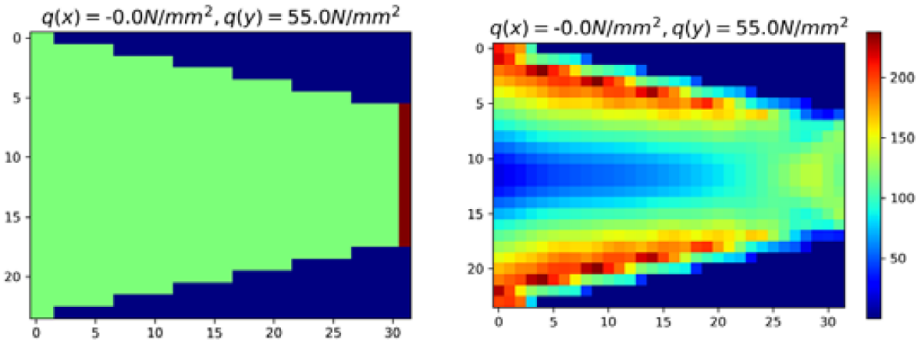
(Farimani et al. 2017, Stanford)

Deep Learning

Ground truth

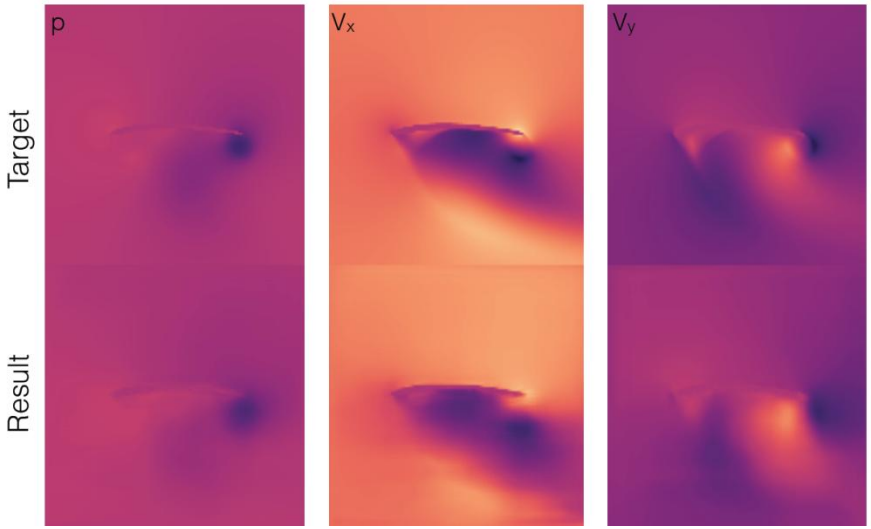


Heat conduction (Farimani et al. 2017)



Stress field for linear elasticity

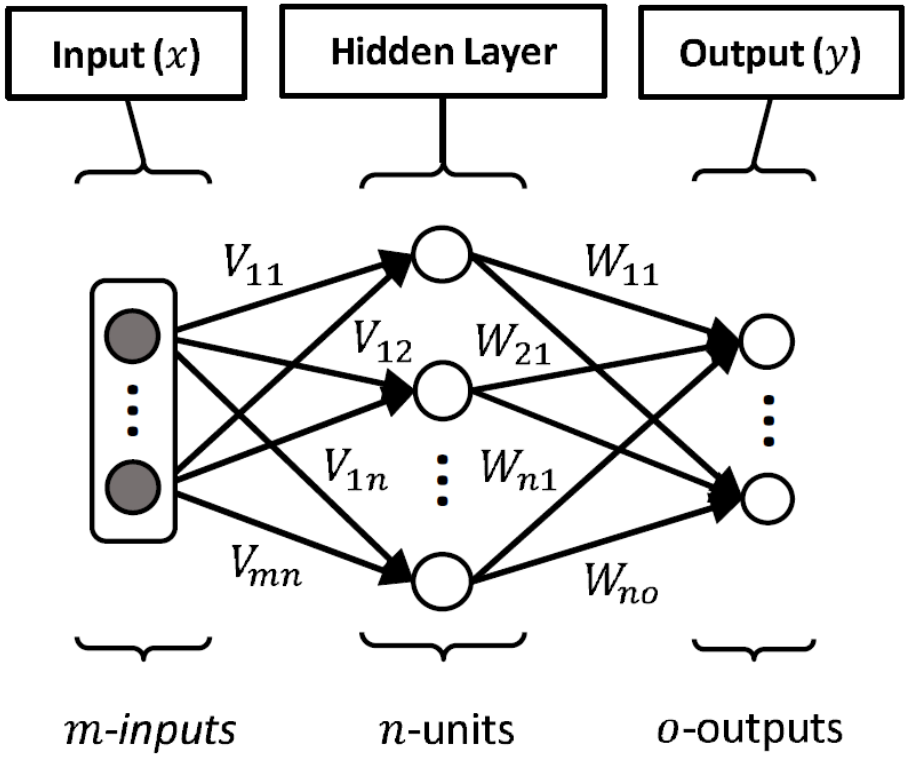
(Nie et al. 2018, Carnegie Mellon University)



Flow around airfoil

(Thuerey et al. 2018, University of Munich)

What is a neural network?



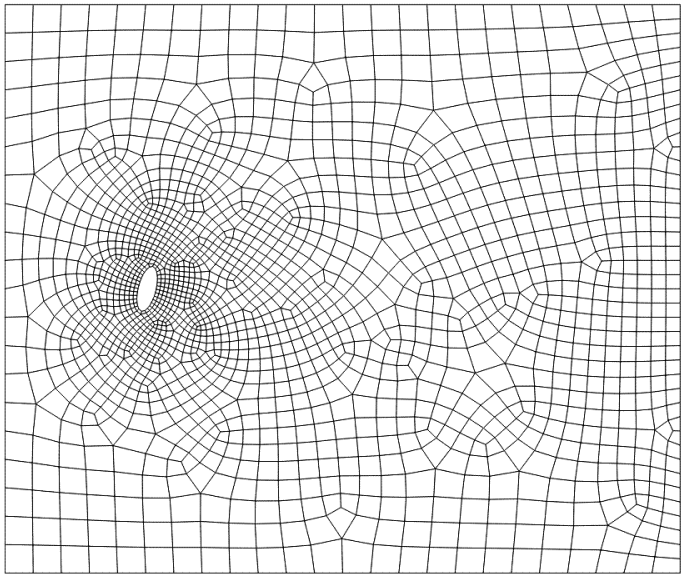
$$y_{\text{Prediction}} = f\left(x, V_{11}, \ldots, V_{mn}, W_{11}, \ldots, W_{no}\right)$$

$$\text{Cost Function} = \frac{1}{N_o}\sum\left(y_{\text{Truth}} - y_{\text{Prediction}}\right)^2$$

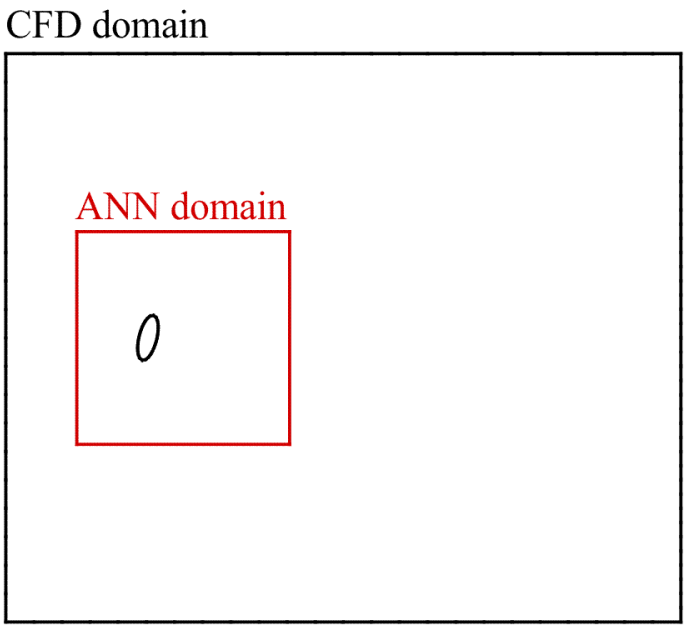
Problem formulation

- The prediction of laminar steady incompressible flows around bodies with **various shapes (17000):** **Circle, Hexagon, Rectangle, Ellipse, Triangle, Square, Pentagon,** and etc, with different sizes and orientations

(a)



(b)



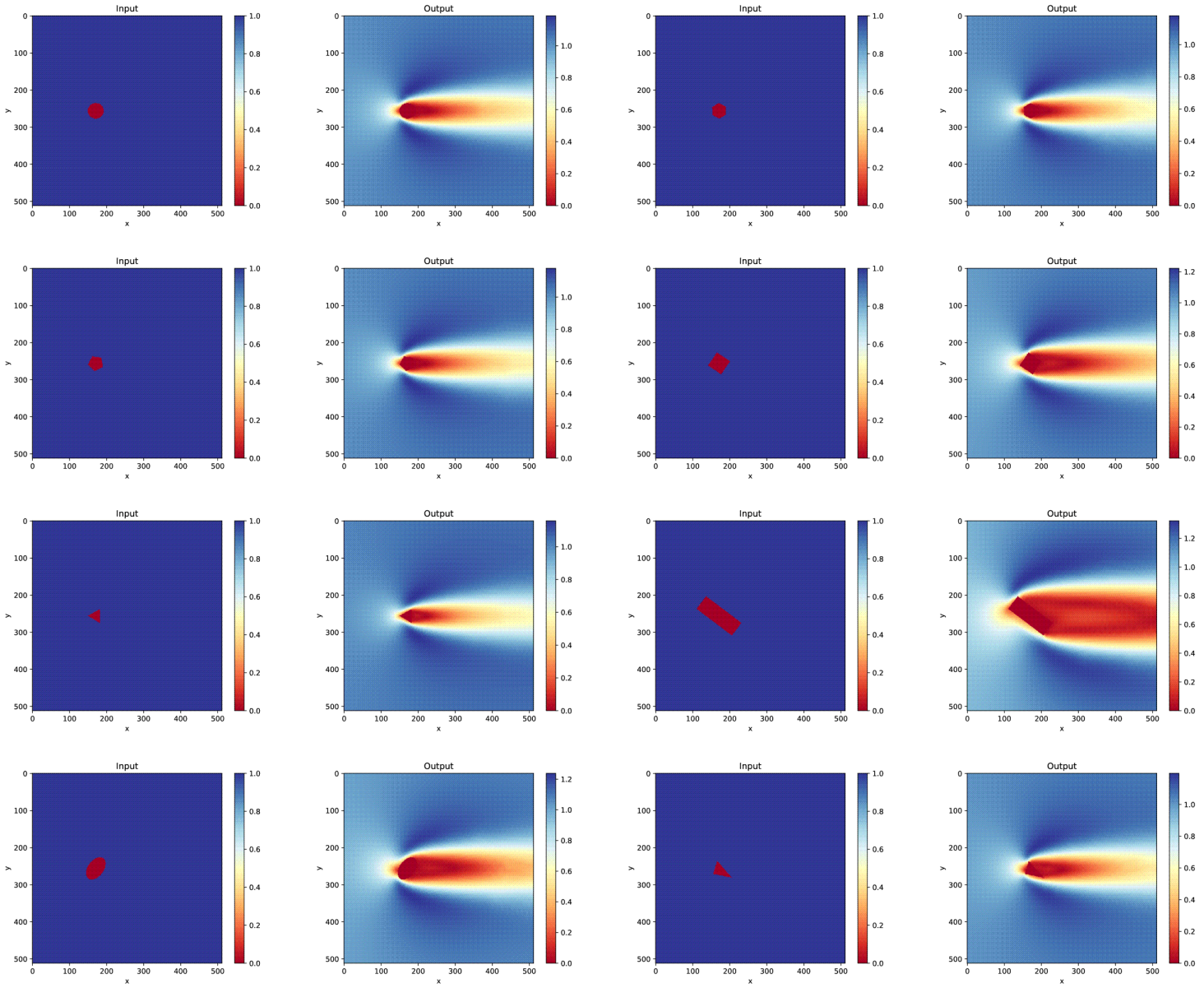
Data Generation

- Took **5 hours** to generate **17000 data** on the **certainty cluster (~120000 nodes)**

- Took **3 weeks** to write a sequence of **C++/batch files** to make the procedure automatic

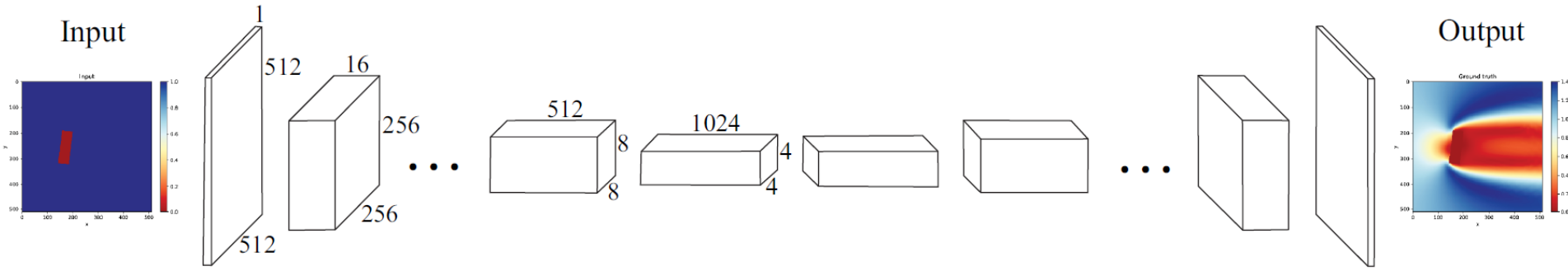
Data Generation

- Examples of input/output of the network



Network design

CNN DCNN

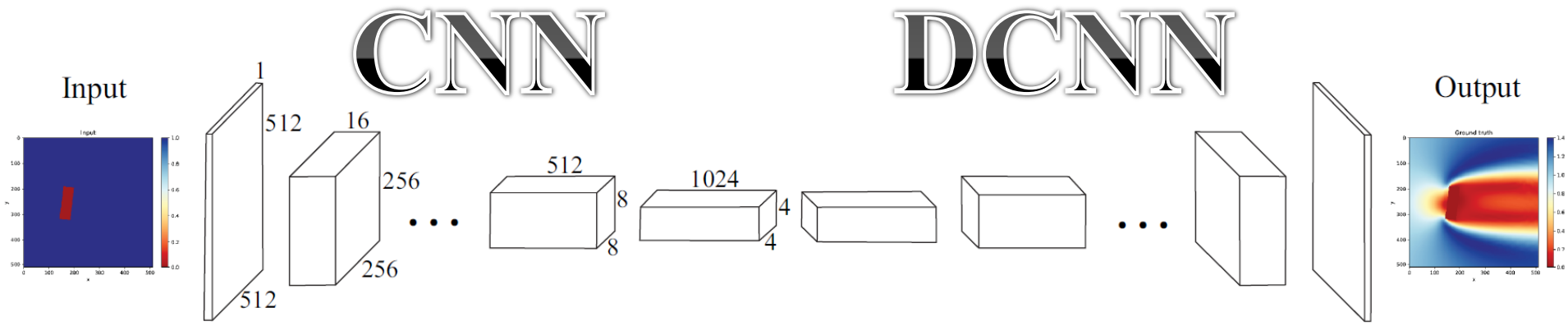


python™

Keras

TensorFlow

Network design



$$y_{\text{Prediction}} = f\left(x, V_{11}, \ldots, V_{mn}, W_{11}, \ldots, W_{no}\right)$$

- **39,158,080** hyper-parameters
- **No fully-connected layer**
- **No max-pooling** and **no up-sampling**
- **Leaky ReLU activation** in the **CNN** layers
- **ReLU activation** in the **DCNN** layers

Network design

Layer#1 = Input

Layer#2 $C(f=16, k=4, s=2)$, Leaky ReLU

Layer#3 $C(f=32, k=4, s=2)$, Leaky ReLU, Batch Normalization

Layer#4 $C(f=64, k=4, s=2)$, Leaky ReLU, Batch Normalization

Layer#5 $C(f=128, k=4, s=2)$, Leaky ReLU, Batch Normalization

Layer#6 $C(f=256, k=4, s=2)$, Leaky ReLU, Batch Normalization

Layer#7 $C(f=512, k=4, s=2)$, Leaky ReLU, Batch Normalization

Layer#8 $C(f=1024, k=4, s=2)$, Leaky ReLU, Batch Normalization

Layer#9 $CT(f=1024, k=4, s=1)$, ReLU, Batch Normalization

Layer#10 $CT(f=512, k=4, s=2)$, ReLU, Batch Normalization

Layer#11 $CT(f=256, k=4, s=2)$, ReLU, Batch Normalization

Layer#12 $CT(f=128, k=4, s=2)$, ReLU, Batch Normalization

Layer#13 $CT(f=64, k=4, s=2)$, ReLU, Batch Normalization

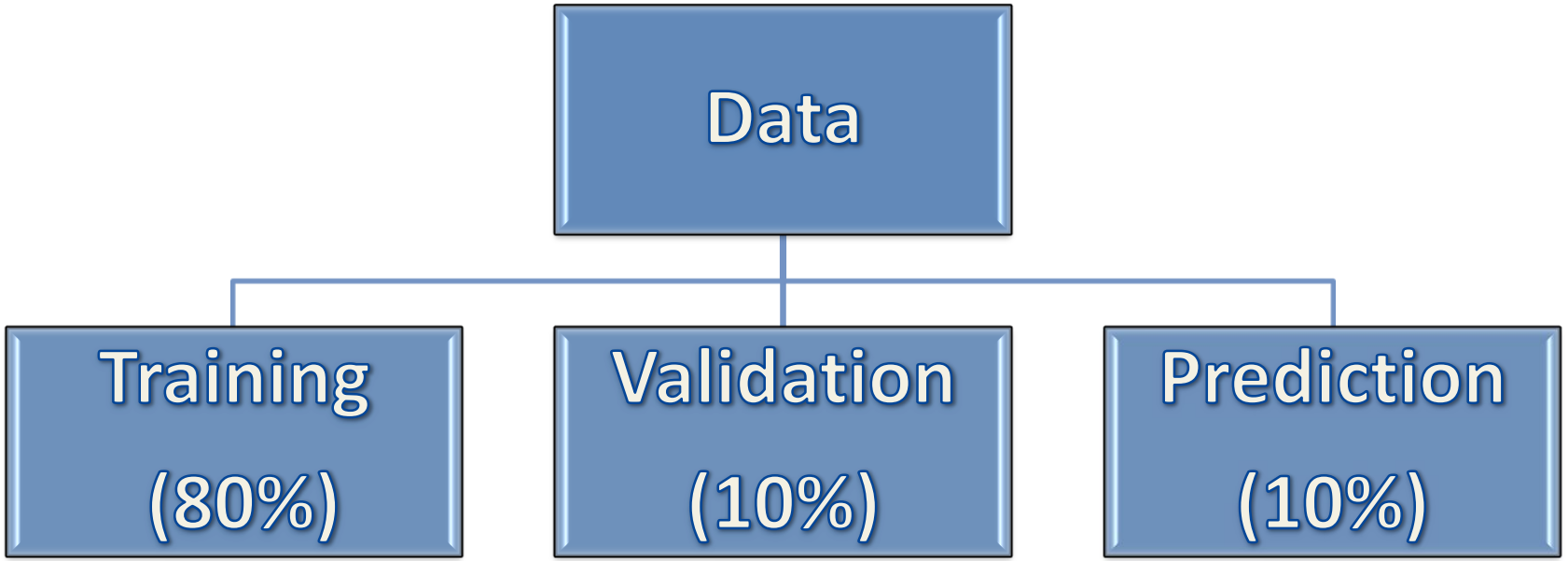
Layer#14 $CT(f=32, k=4, s=2)$, ReLU, Batch Normalization

Layer#15 $CT(f=16, k=4, s=2)$, ReLU, Batch Normalization

Layer#16 Output = $CT(f=1, k=4, s=2)$, ReLU

Training

- Data distribution (randomly chosen)

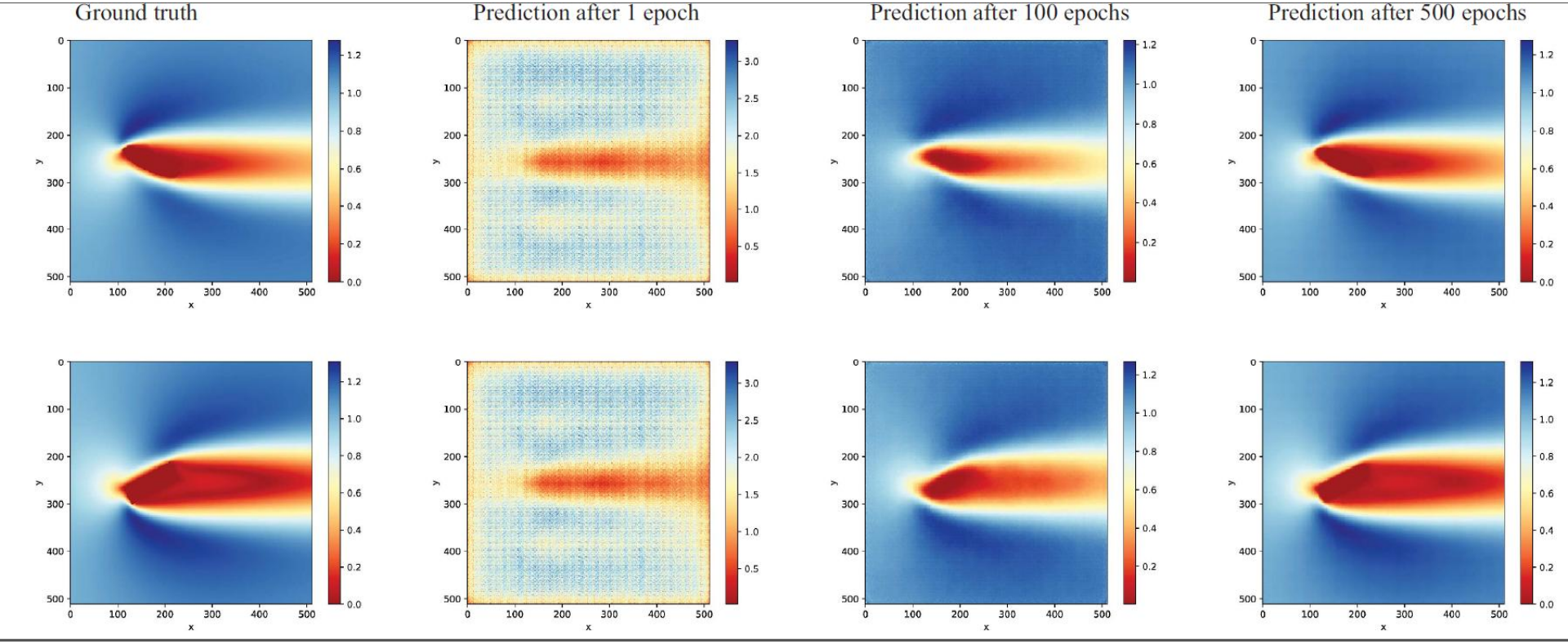


$$\mathrm{MSE}=\frac{1}{N_x N_y}\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\left(\frac{u_{i,j}^{CFD}-u_{i,j}^{ANN}}{U_\infty}\right)^2$$

Training took ~2h on GPU or (equivalently 9 days on CPU)

Training

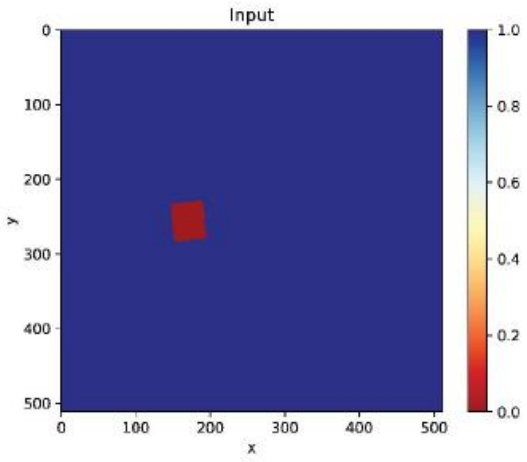
- Prediction after 1, 100, and 500 epochs, and a comparison with the ground truth



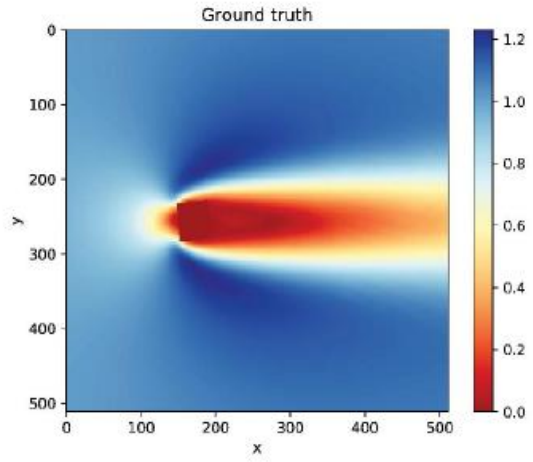
Results

Prediction over **200 unseen data** in **1 minute (on CPU)** with the average **mean square error of 0.1 %**

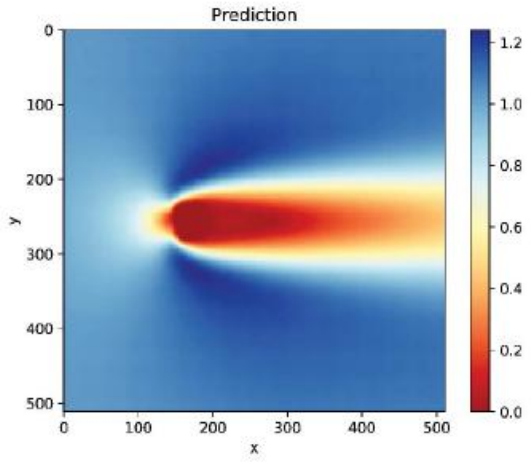
Input

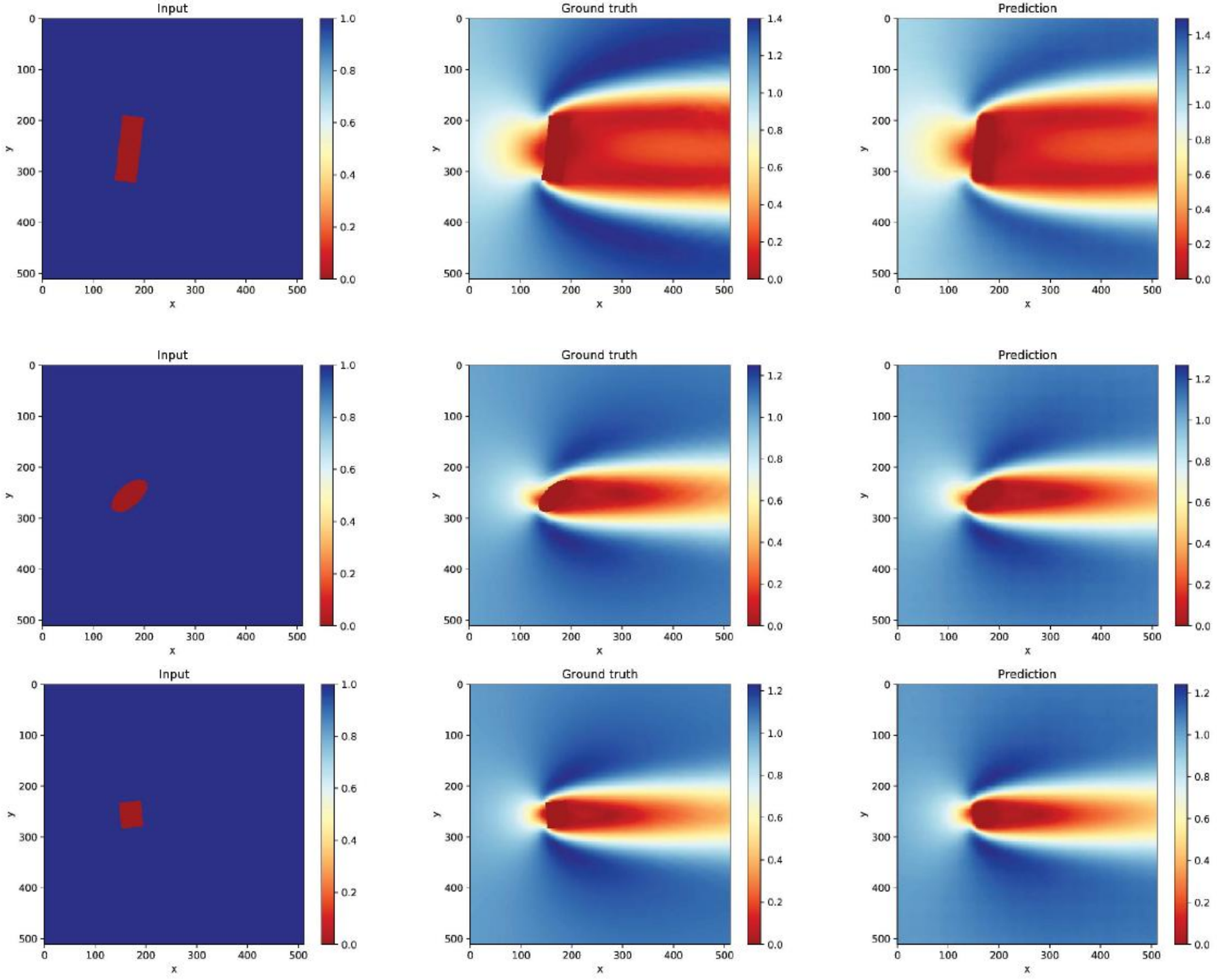


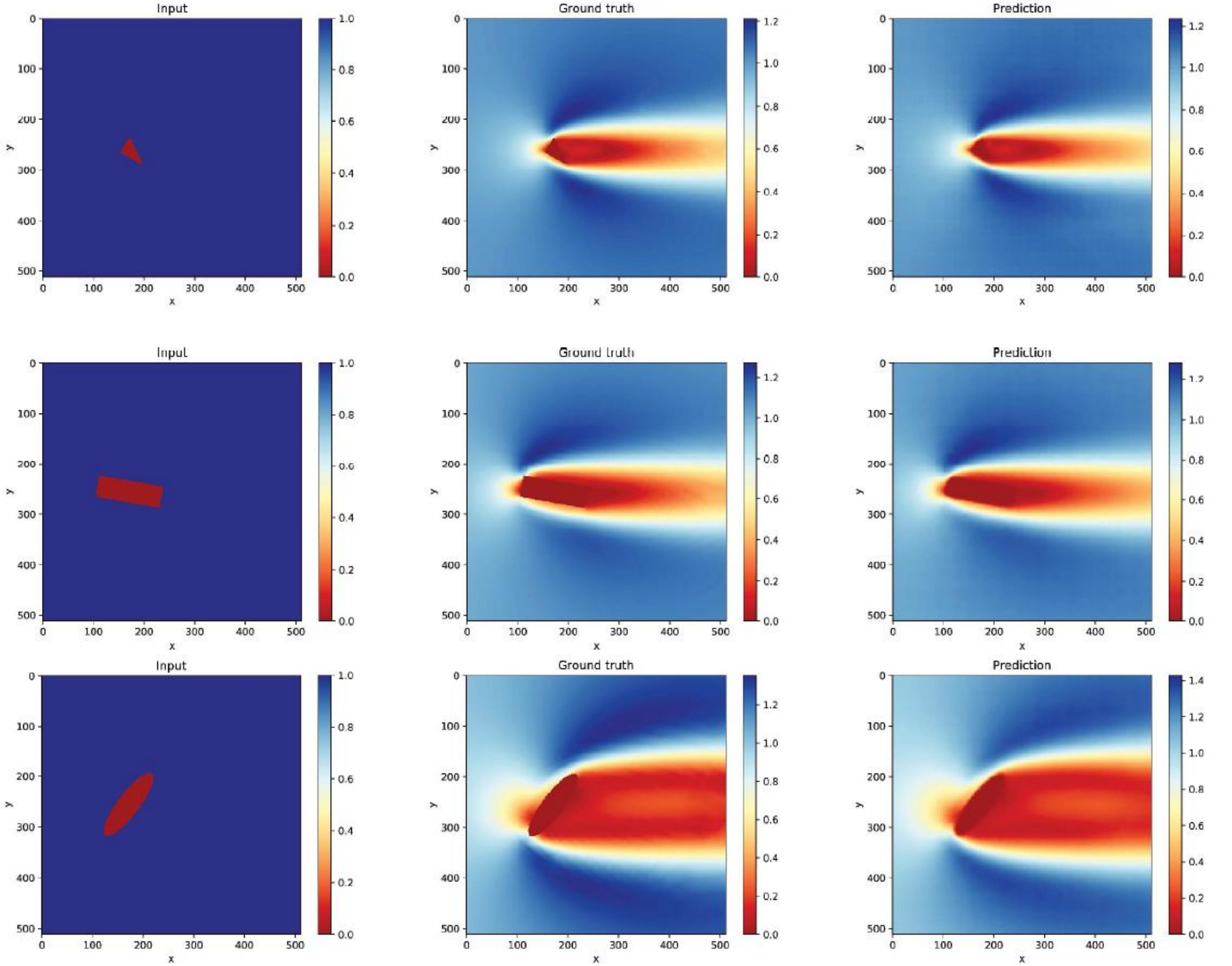
Truth

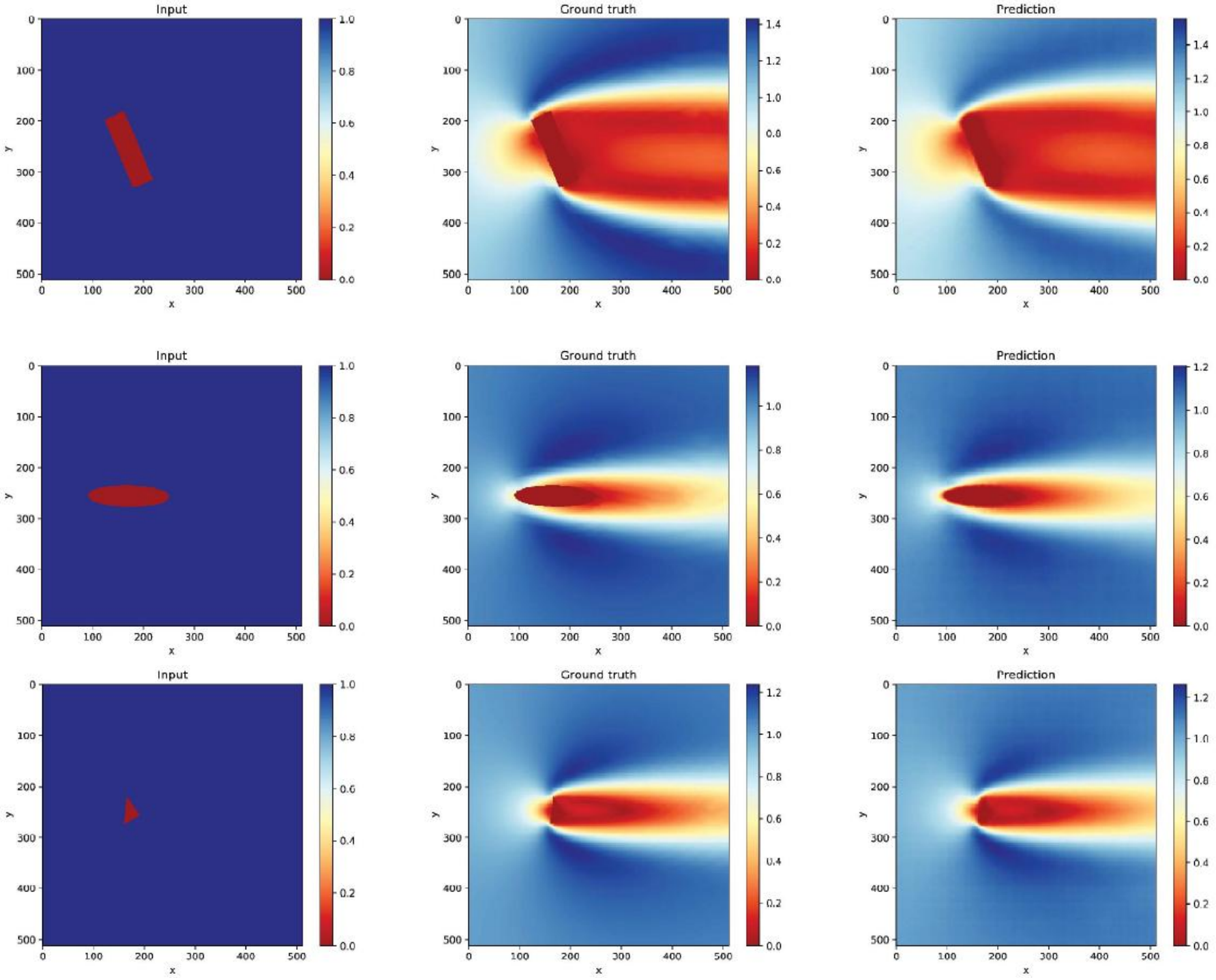


Prediction




Input
Ground truth
Prediction
x
y

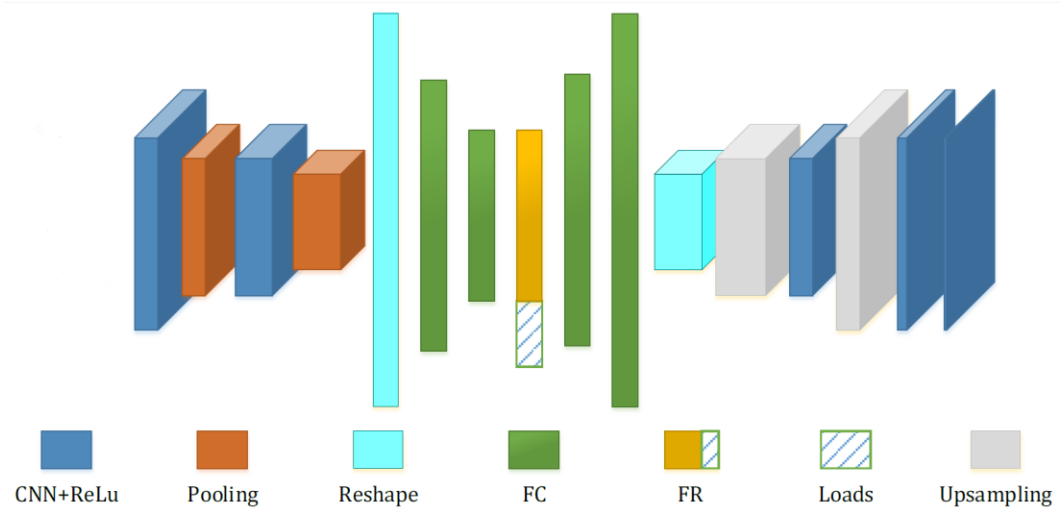

Input
Ground truth
Prediction
Input
Ground truth
Prediction
Input
Ground truth
Prediction
x
y

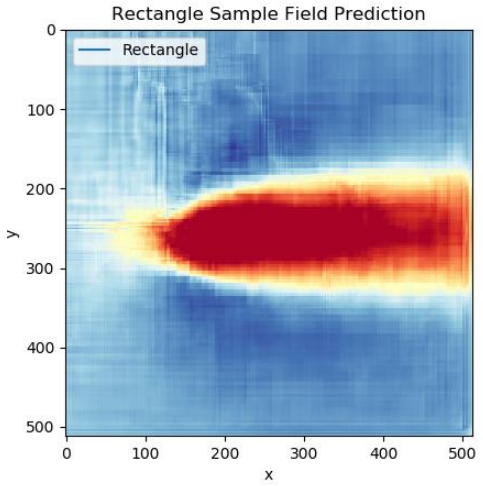

Input
Ground truth
Prediction
Input
Ground truth
Prediction
Input
Ground truth
Prediction
x
y

Challenges

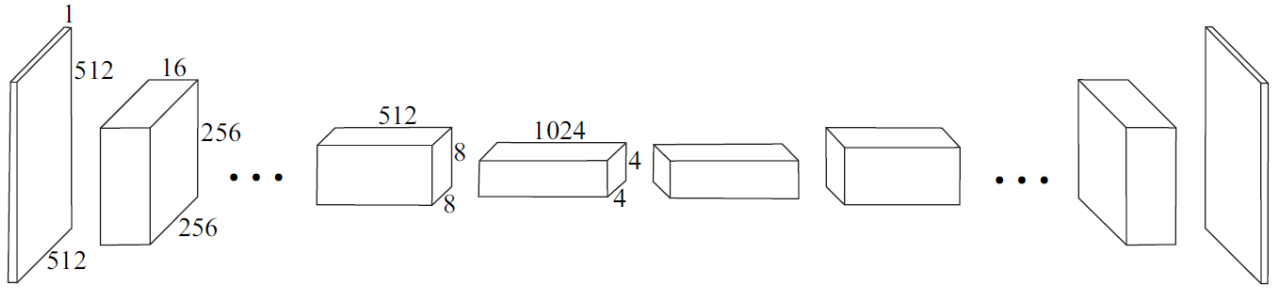
- The effect of **Choosing/Designing Neural Network** on the accuracy of the model

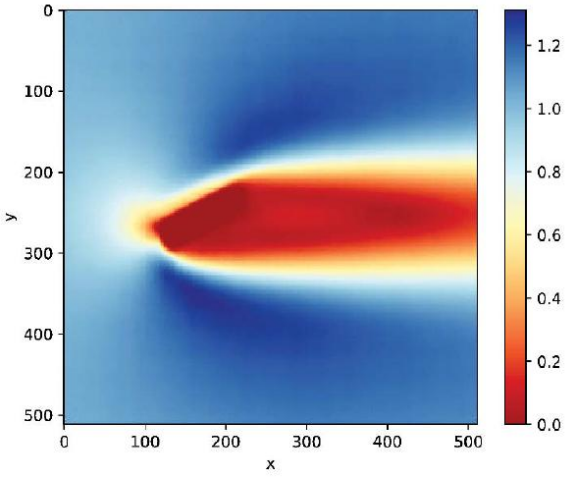
➢ The network proposed by scientists at CMU:





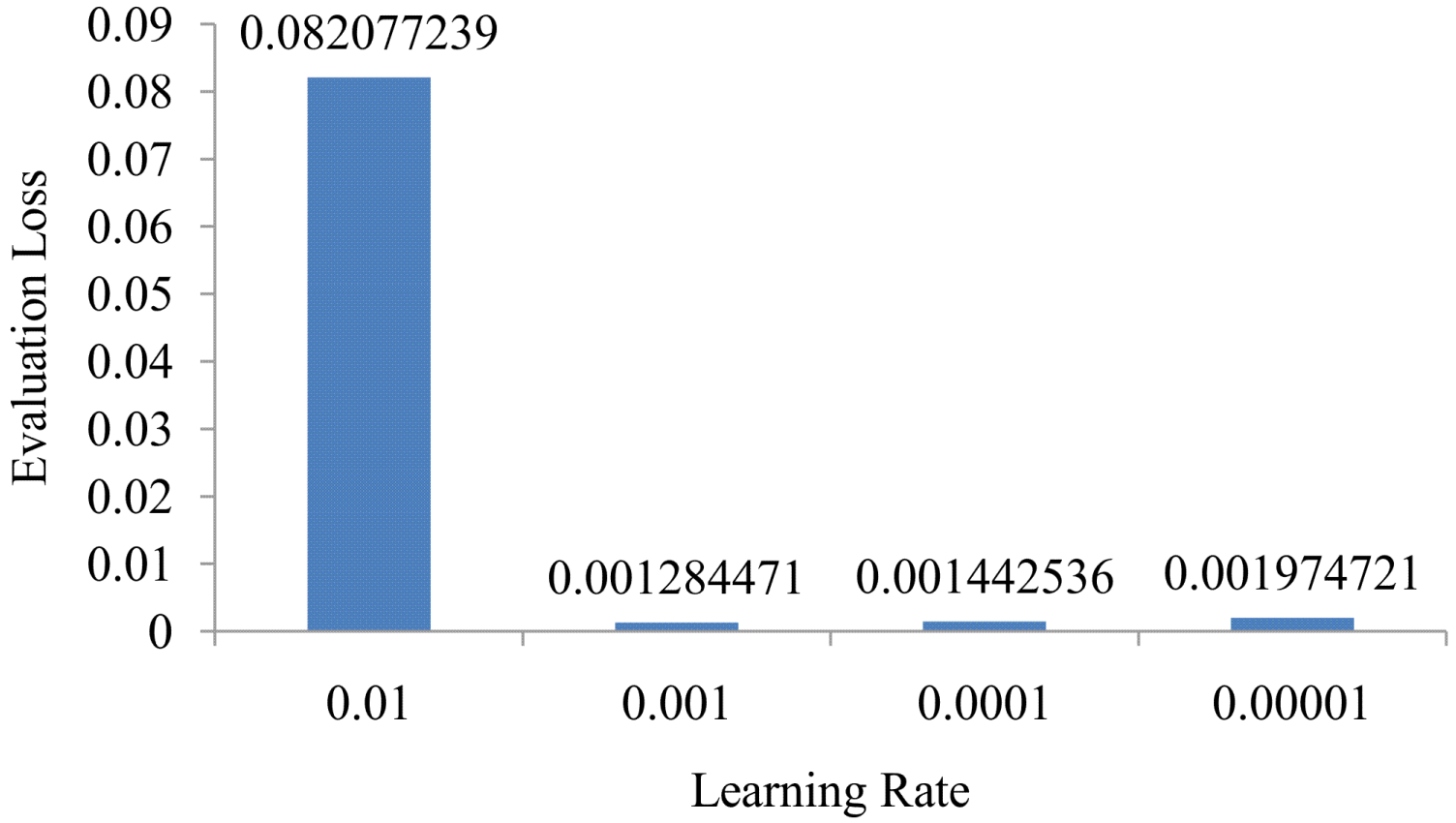
➢ Our network:





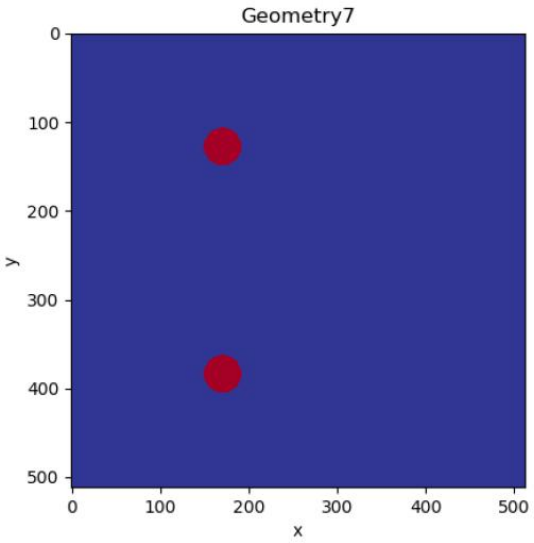
Challenges

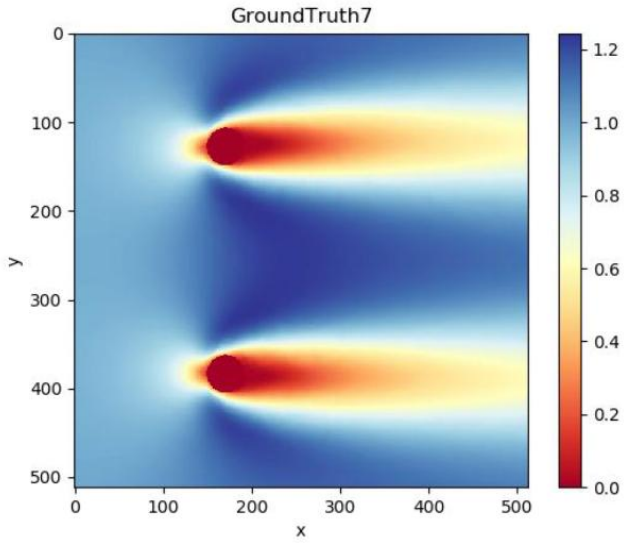
- The effect of the **Learning Rate** on the accuracy of the model
- **Adam** optimizer parameters: {Learning rate, $\beta_1, \beta_2, \varepsilon$}



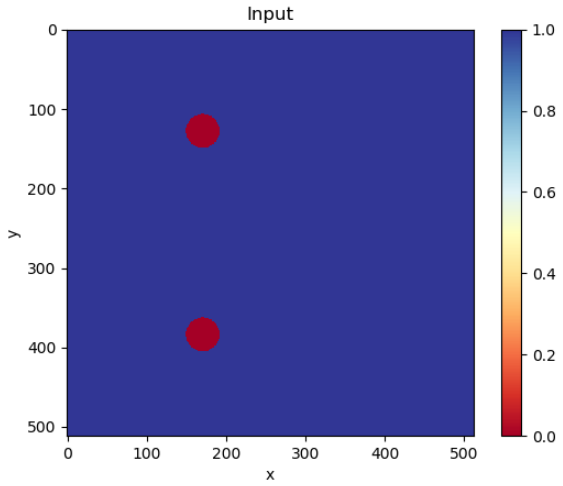
Challenges

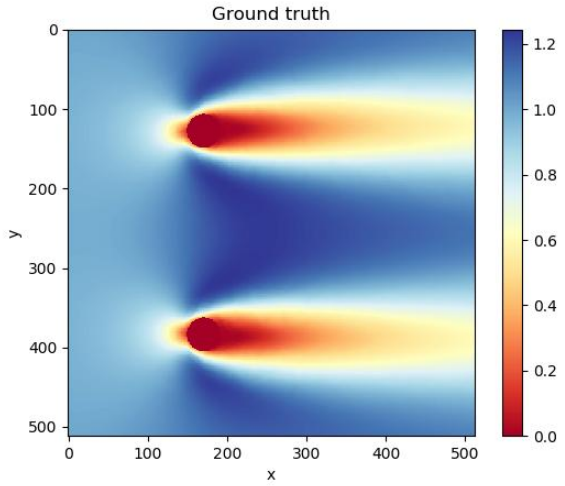
- Examine the capability of our network to predict the flow around combination of two shapes, for instance “two separated cylinders”

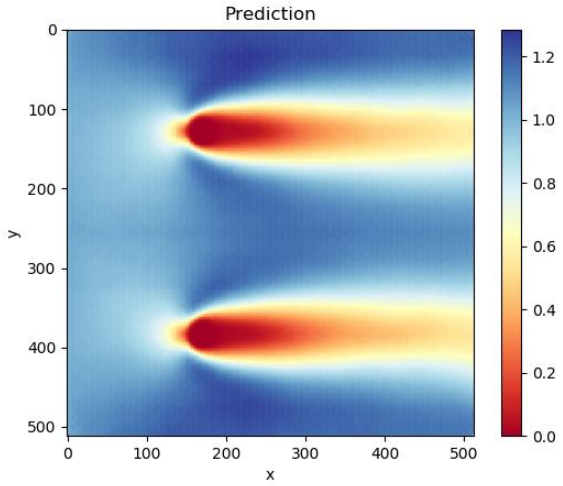




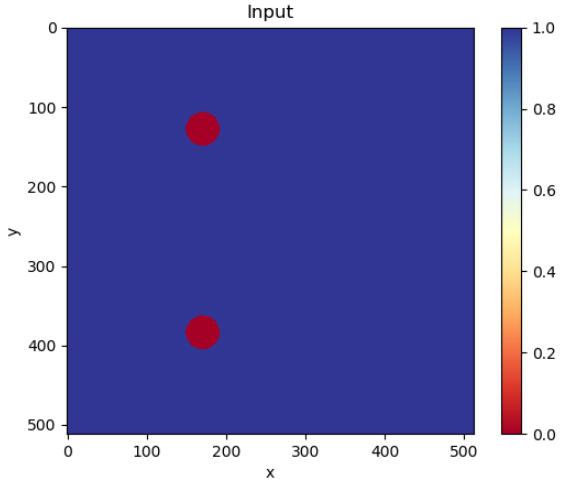
Prediction?

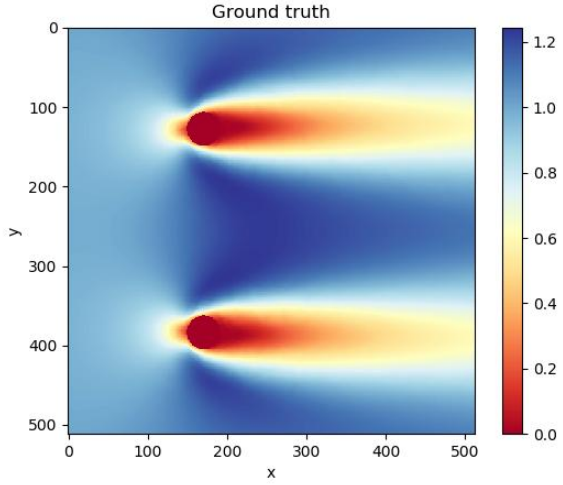

Input
y
x
0
100
200
300
400
500
1.0
0.8
0.6
0.4
0.2
0.0

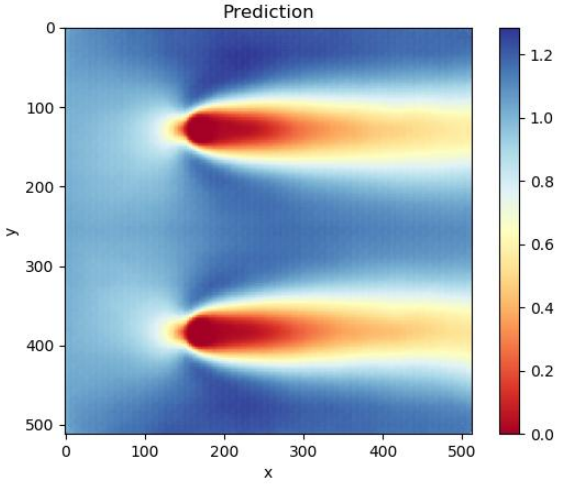

Ground truth
y
x
0
100
200
300
400
500
1.2
1.0
0.8
0.6
0.4
0.2
0.0


Prediction
y
x
0
100
200
300
400
500
1.2
1.0
0.8
0.6
0.4
0.2
0.0

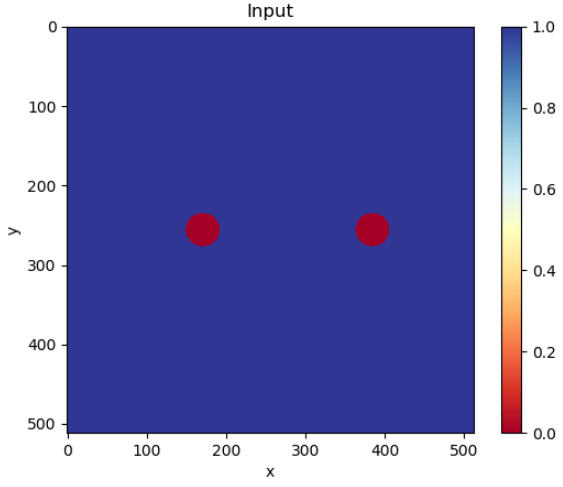
MSE = 0.5%

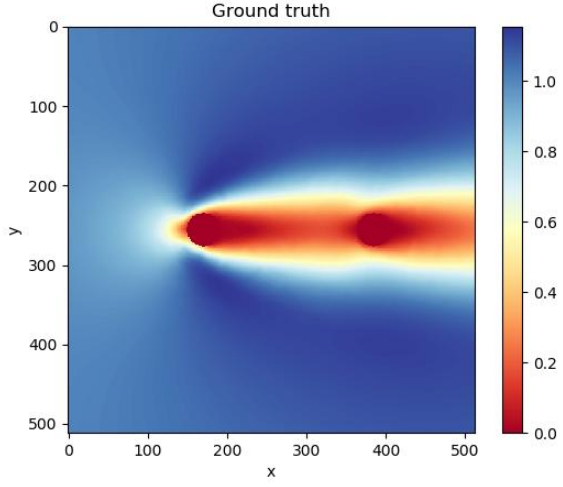

Input
y
x

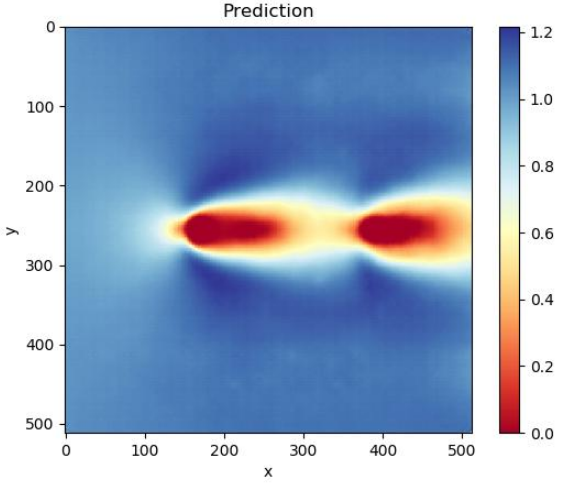

Ground truth
y
x


Prediction
y
x

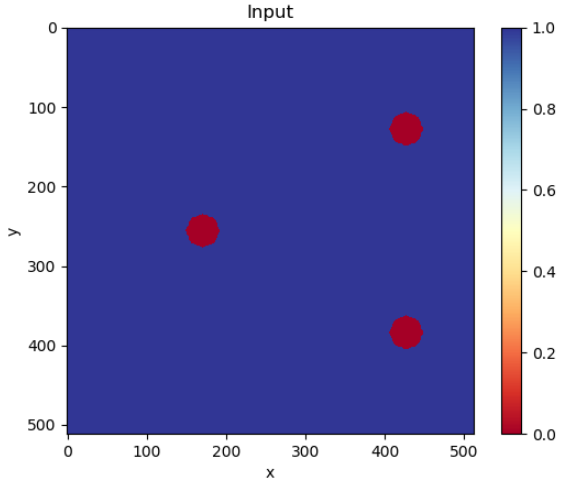
MSE = 0.5%

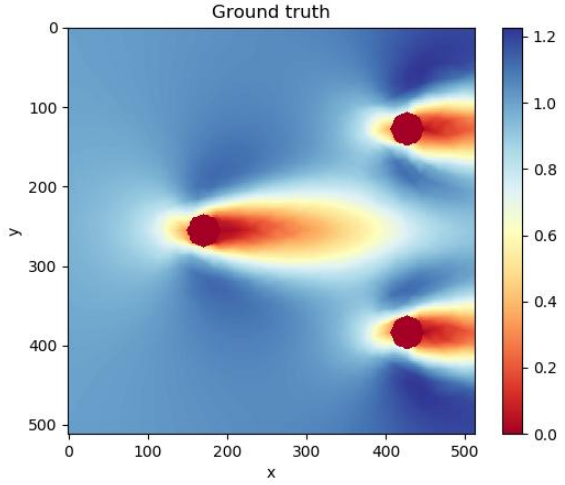

Input
y
x

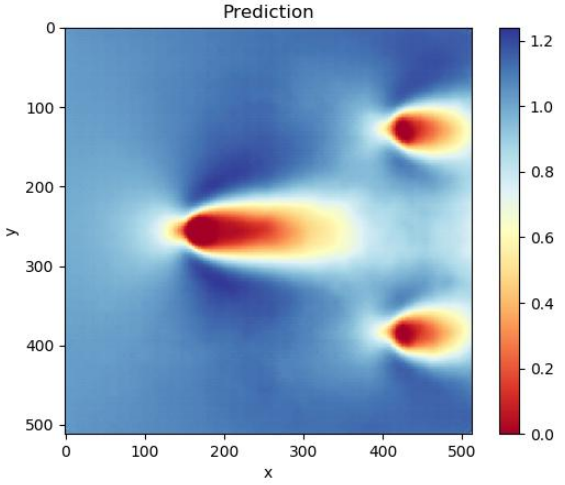

Ground truth
y
x


Prediction
y
x

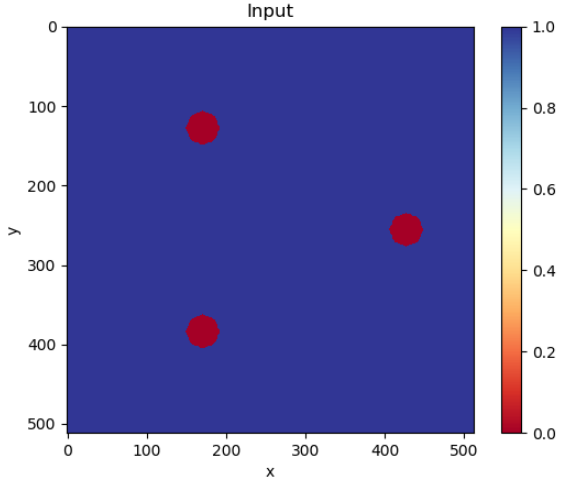
MSE = 0.8%

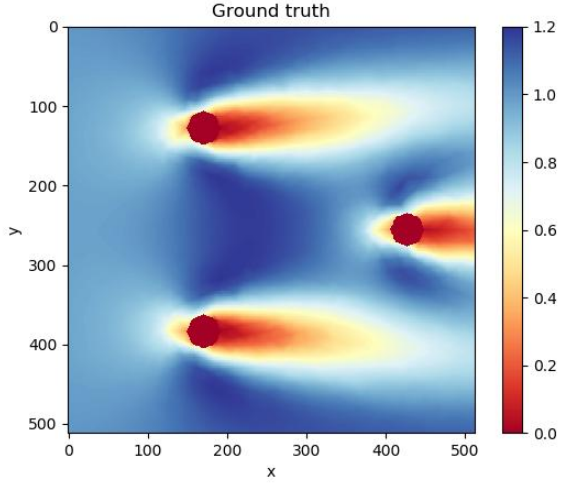

Input
0
100
200
300
400
500
y
x
1.0
0.8
0.6
0.4
0.2
0.0

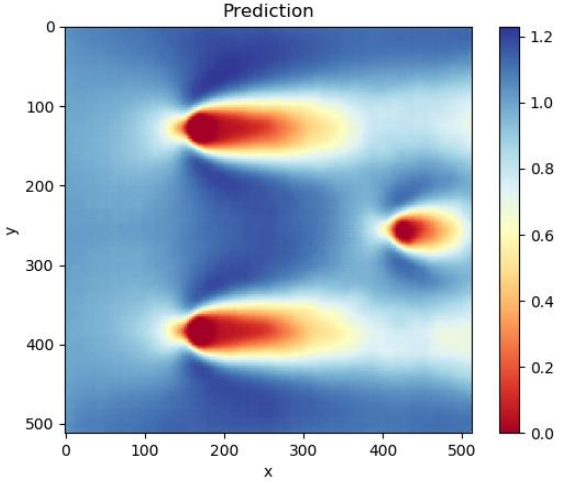

Ground truth
0
100
200
300
400
500
y
x
1.2
1.0
0.8
0.6
0.4
0.2
0.0


Prediction
0
100
200
300
400
500
y
x
1.2
1.0
0.8
0.6
0.4
0.2
0.0

MSE = 0.7%


Input
0
100
200
300
400
500
y
x
1.0
0.8
0.6
0.4
0.2
0.0


Ground truth
0
100
200
300
400
500
y
x
1.2
1.0
0.8
0.6
0.4
0.2
0.0


Prediction
0
100
200
300
400
500
y
x
1.2
1.0
0.8
0.6
0.4
0.2
0.0

MSE = 0.9%

Thank you!