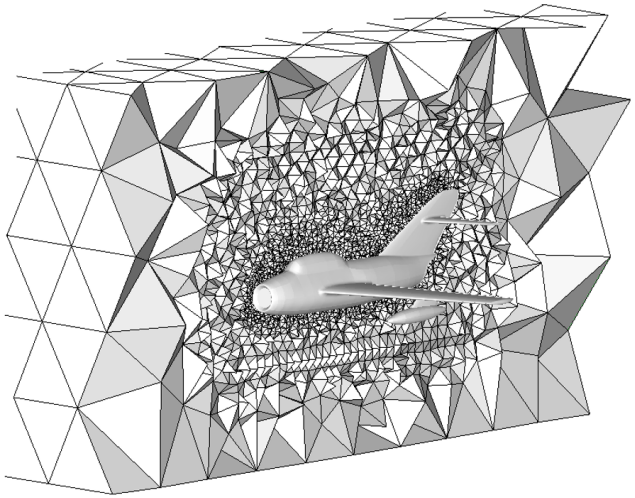
Deep Learning in CFD

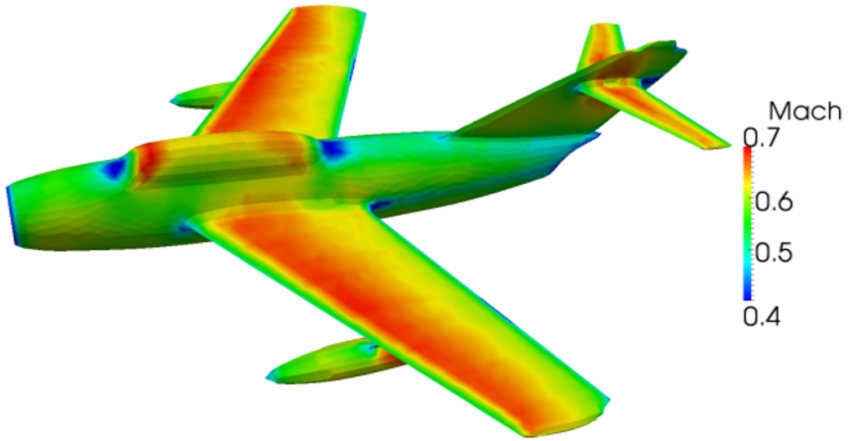
Václav Heidler, Ondřej Bublík, Aleš Pecka, Jan Vimmr

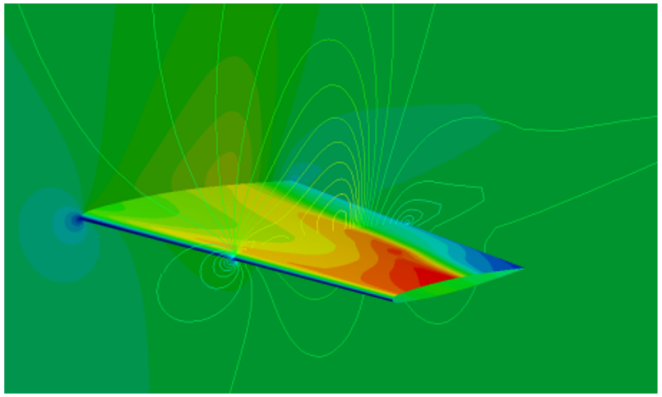
NTIS - New Technologies for the Information Society
Faculty of Applied Sciences, University of West Bohemia

February 2024

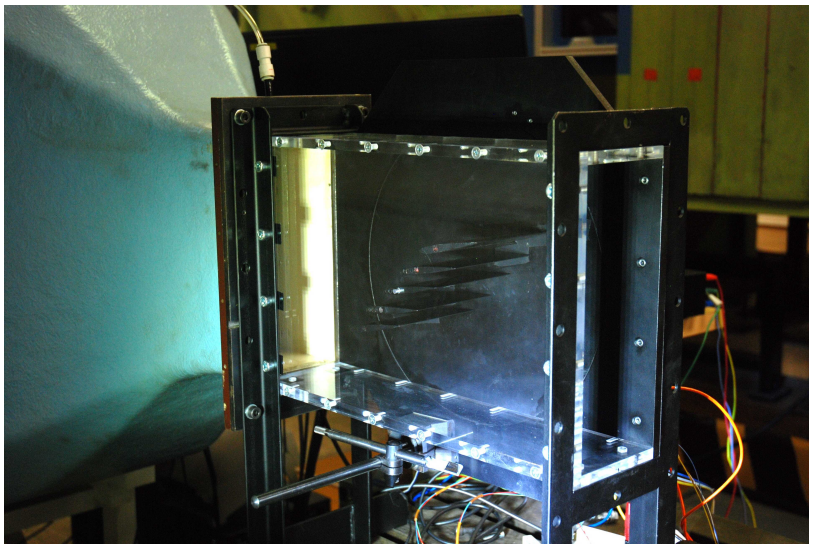
CFD Introduction

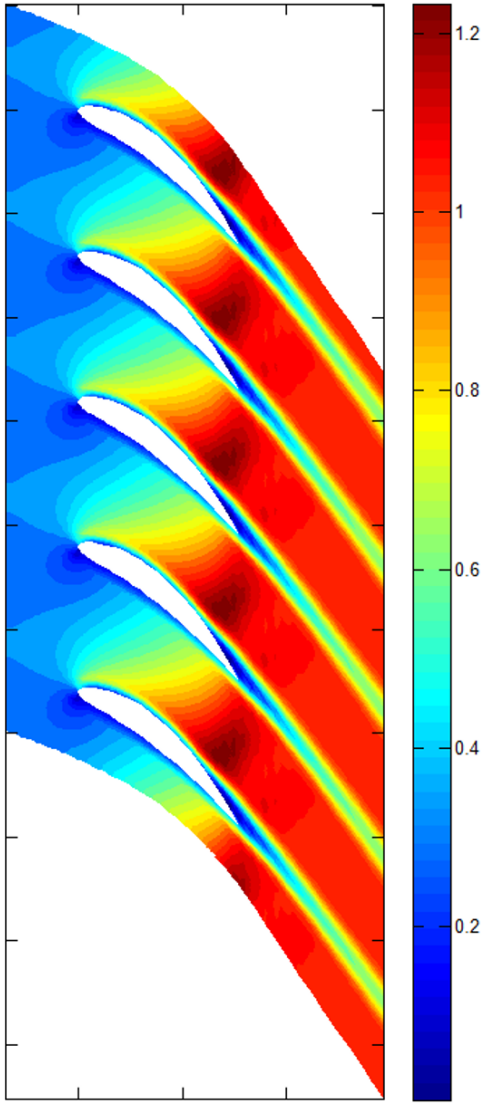






CFD Introduction





Navier-Stokes Equations - incompressible flow

$$\frac{\partial u_i}{\partial x_i} = 0$$

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} + \frac{\partial p}{\partial x_i} = \frac{1}{\mathrm{Re}} \frac{\partial^2 u_i}{\partial x_j \partial x_j}, \qquad i = 1, \ldots, D$$

- p is pressure
- u_i is the velocity vector
- Re is the Reynolds number

Navier-Stokes Equations - Compressible flow

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i) = 0$$

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_i u_j + p\delta_{ij})}{\partial x_j} = \frac{\partial \tau_{ij}}{\partial x_j}, \qquad i = 1, \ldots, D$$

$$\frac{\partial(\rho e)}{\partial t} + \frac{\partial(\rho u_i e + p u_i)}{\partial x_i} = \frac{\partial}{\partial x_i}\left(\tau_{ij} u_i - q_i\right)$$

- τ_{ij} is the viscous stress tensor

$$\tau_{ij} = \frac{1}{\mathrm{Re}}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right)$$

- q_i is the heat flux vector

$$q_i = -\frac{\gamma}{\gamma - 1}\frac{1}{\mathrm{Pr}\,\mathrm{Re}}\frac{\partial}{\partial x_i}\left(\frac{p}{\rho}\right)$$

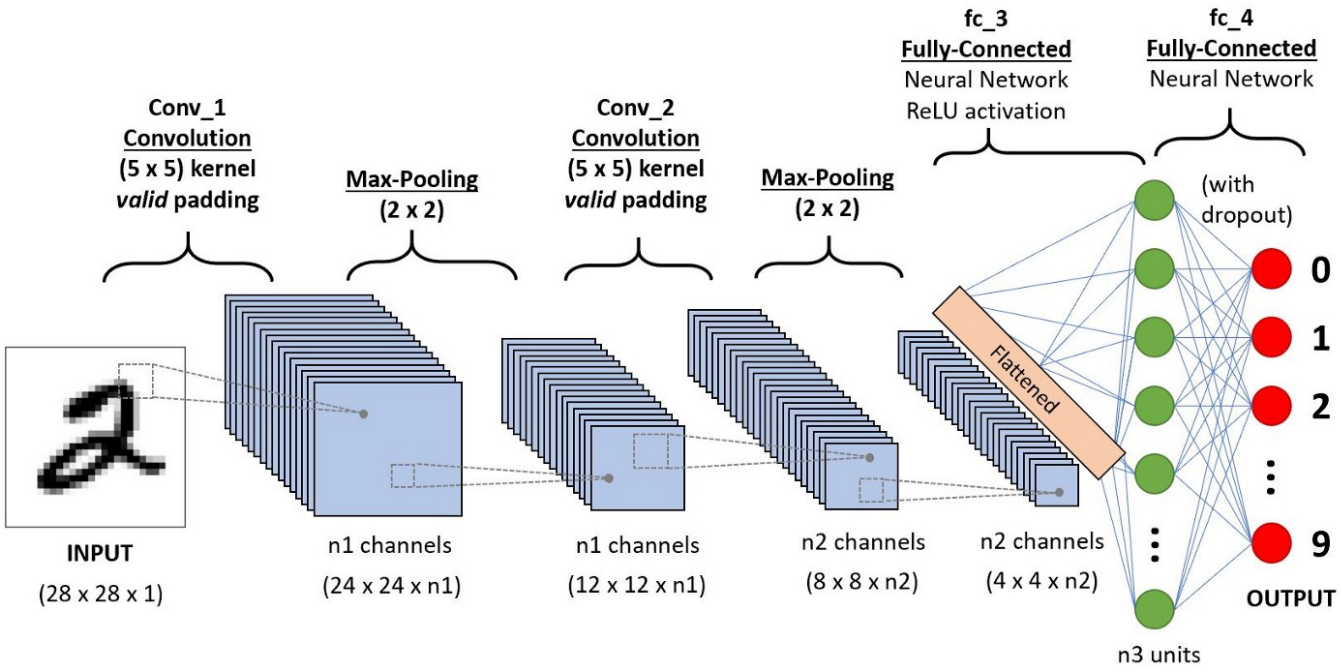
- ρ is density
- u_i is the velocity vector
- e is the total energy per unit mass
- p is pressure
- γ is the heat capacity ratio
- Re is the Reynolds number
- Pr is the Prandtl number

The system is completed by the relation for the total

Convolution Neural Networks

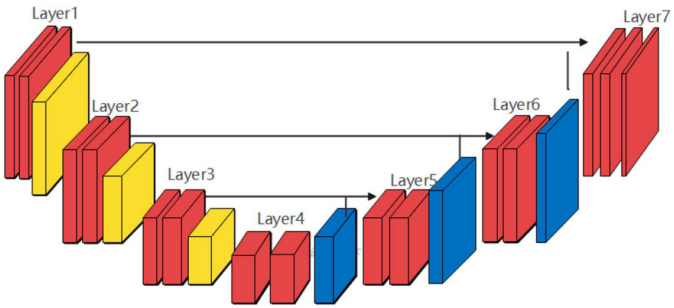
Convolution neural network

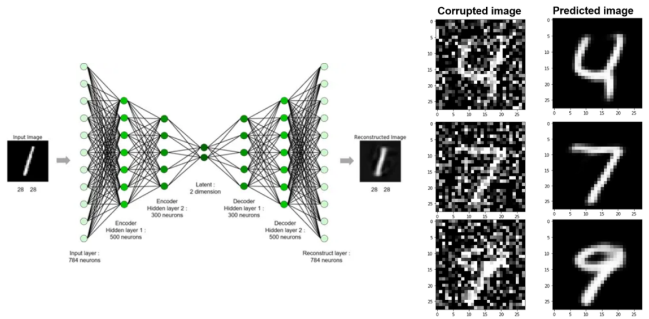
- used for image/object recognition and classification
- convolutional layer reduces the high dimensionality of images without losing its information

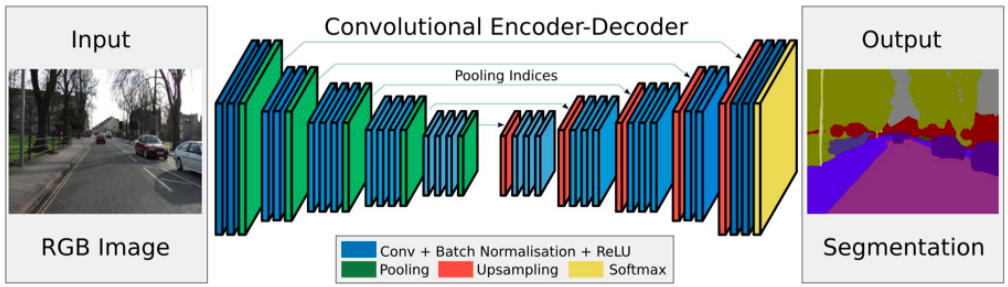


Encoder-decoder, Autoencoder and U-Net

- The output has the same character as the input
- Encoder-decoder:
  - image recognition, detection, and segmentation
- U-net:
  - is Encoder-decoder with skip connection
- Autoencoder:
  - encoder-decoder with unsupervised learning
  - is trained to copy its input to its output
  - used for image denoising, and anomaly detection







Pioneer Study

Convolution Neural Networks for CFD Problems

Pioneer study:

Guo, X., Li, W., Iorio, F. *Convolutional neural networks for steady flow approximation* (2016) Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 13-17-August-2016, pp. 481-490

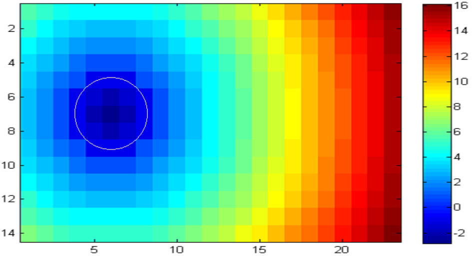
Isn't a flow field just an image?

Convolution Neural Networks for CFD Problems

Pioneer study:

Guo, X., Li, W., Iorio, F. *Convolutional neural networks for steady flow approximation* (2016) Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 13-17-August-2016, pp. 481-490

2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4
3	1	1	1	1	1	5	5	5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4
3	1	1	1	1	5	5	0	5	5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4
3	1	1	1	1	5	0	0	0	5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4
3	1	1	1	1	5	5	0	5	5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4
3	1	1	1	1	1	5	5	5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

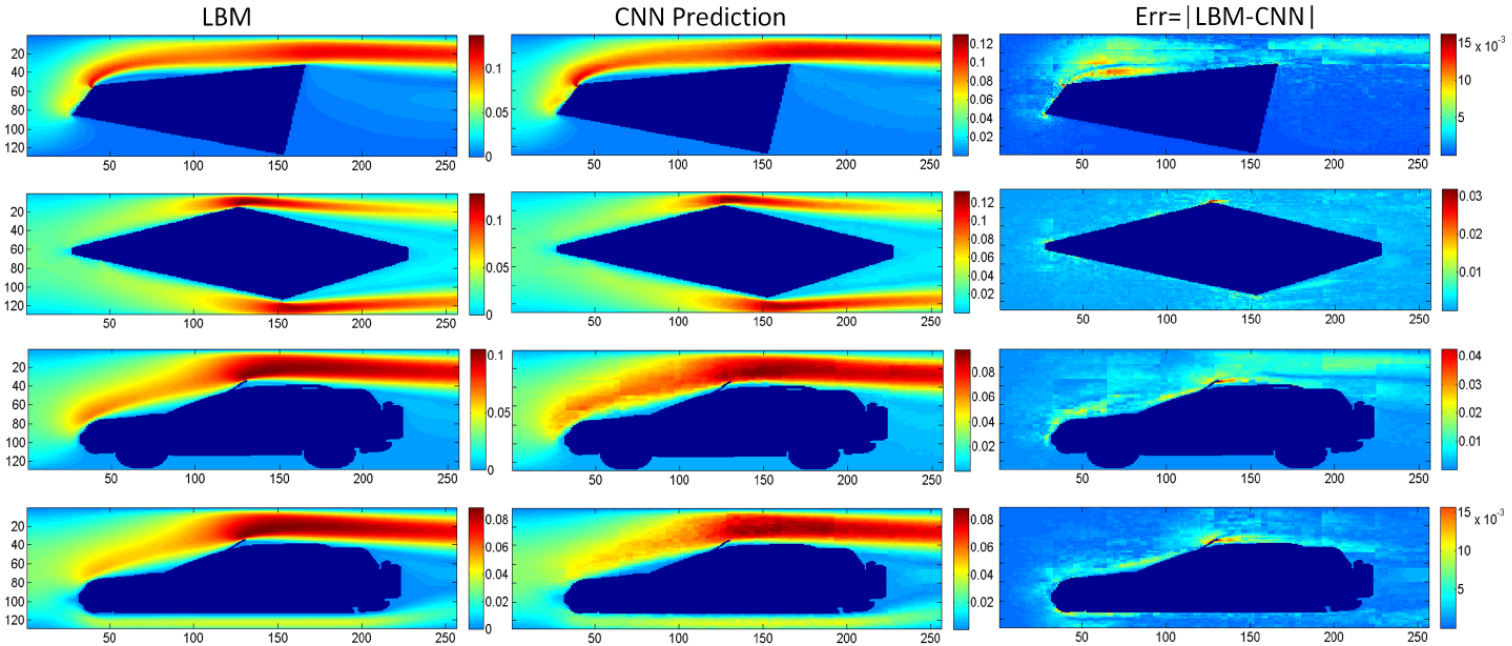


- 23x14 Cartesian grid
- Left: 2D channel fow definition for LBM (1=fuid, 2=no-slip boundary, 3=velocity boundary, 4=constant pressure boundary, 5=object boundary, 0=internal)
- Right: Signed Distance Function - Input into the CNN

Convolution Neural Networks for CFD Problems

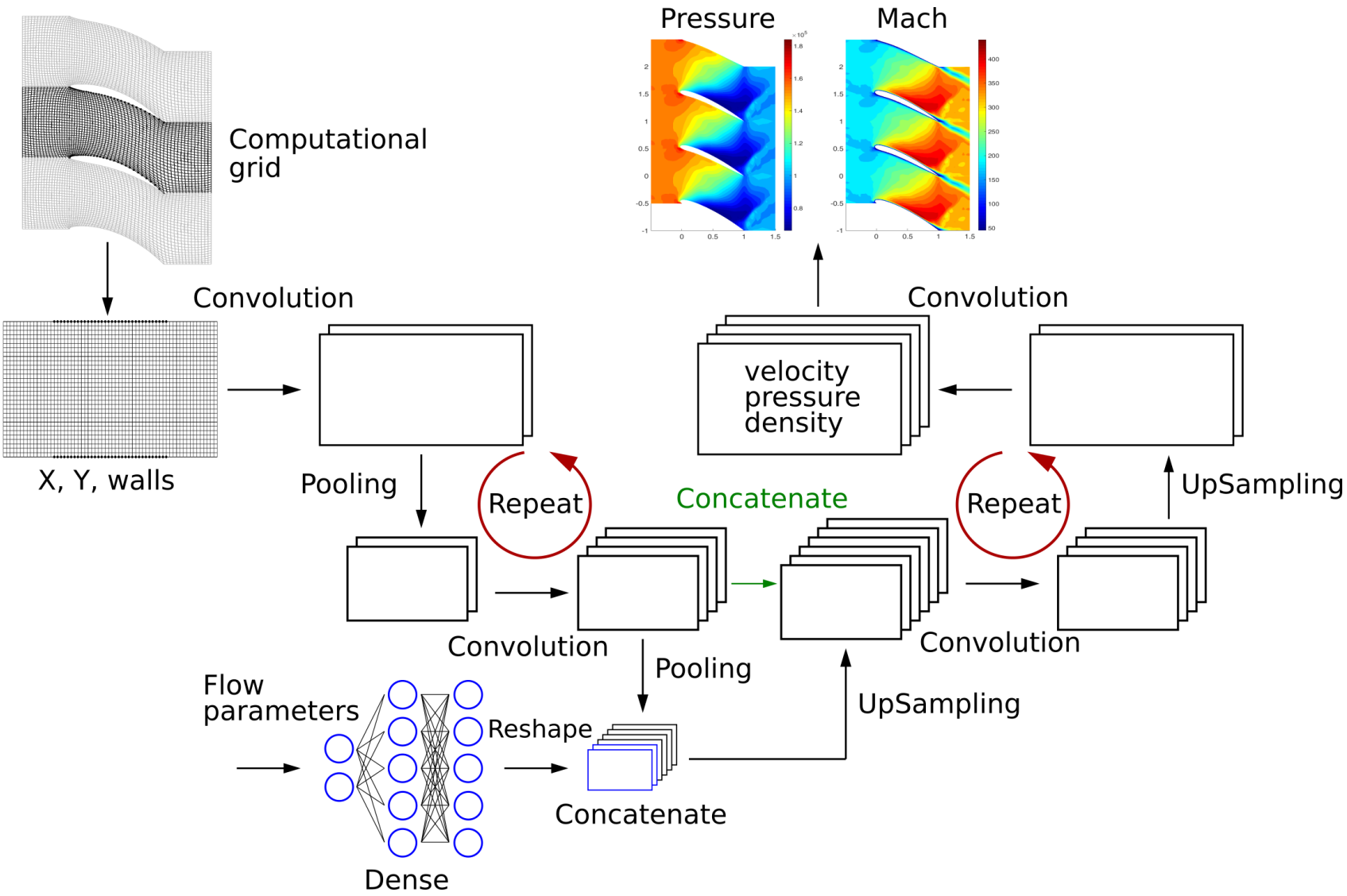
Pioneer study:

Guo, X., Li, W., Iorio, F. *Convolutional neural networks for steady flow approximation* (2016) Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 13-17-August-2016, pp. 481-490



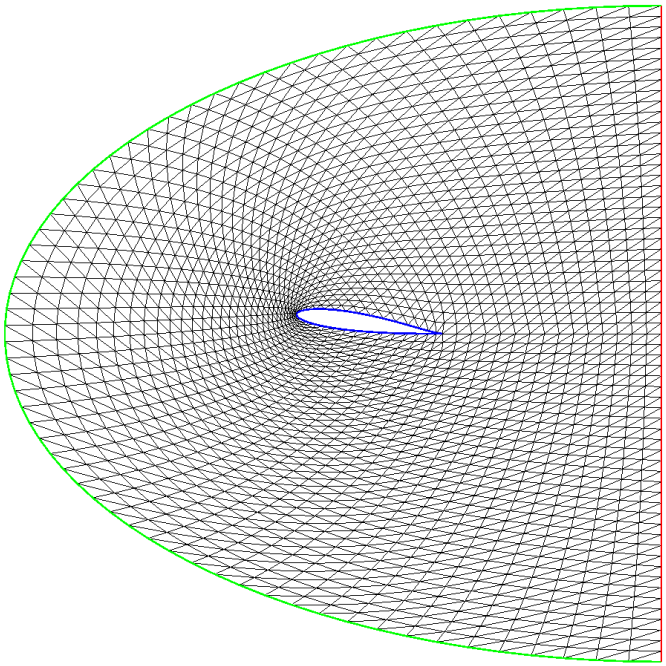
Prediction of Steady Flow Fields

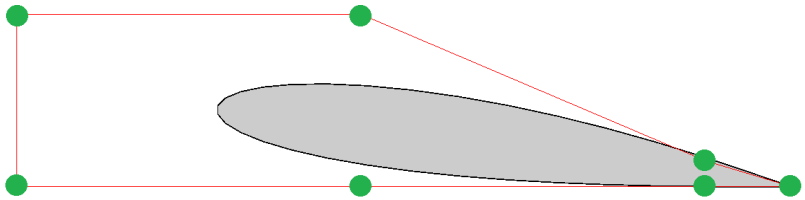
Neural Network Architecture - U-net



Stead flow around aerofoils

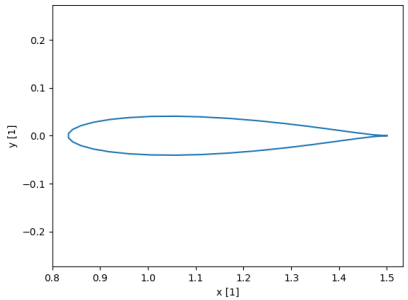
- Inviscid fluid flow around airfoil, angle of attack $\alpha = 0$, Mach number $M_\infty = 0.4$
- Aerofoil shape is described using the Bezier curve with 8 control points
- The first and the last points are fixed on the aerofoil trailing edge
- Set of 1866 aerofoils for various control point positions were generated
- **CNN input**: C-mesh with 64 x 32 points
- **CNN output**: flow field (ϱ, p, u_x, u_y)



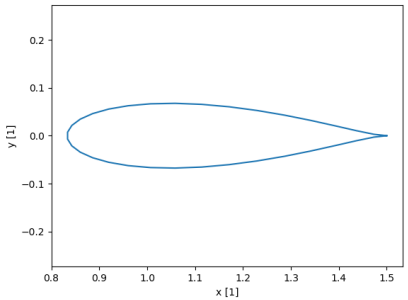


Prediction of Steady Flow Fields - Testing NACA Aerofoils

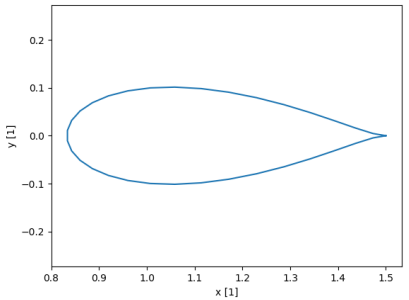
0012



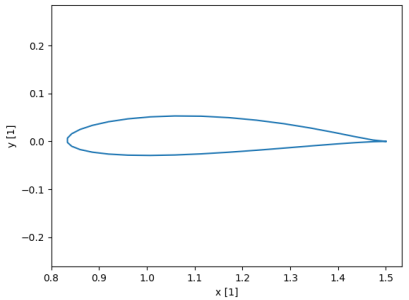
0020



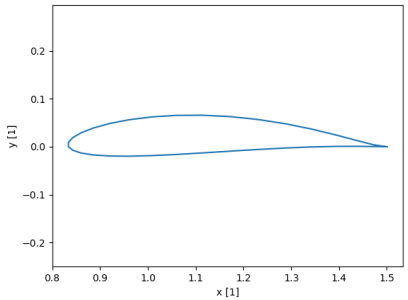
0030



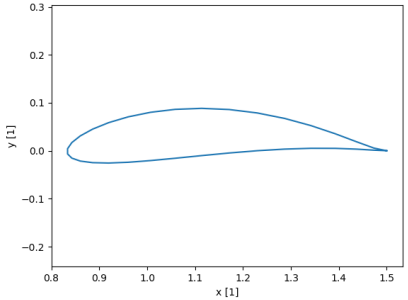
2412



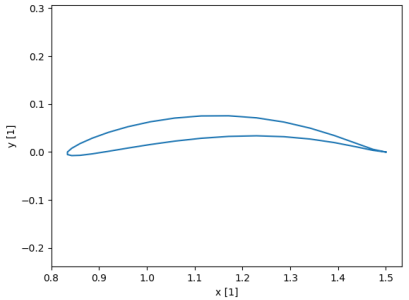
4412



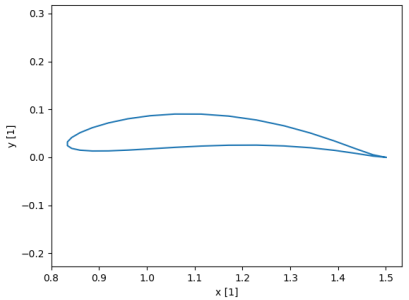
6615



8607

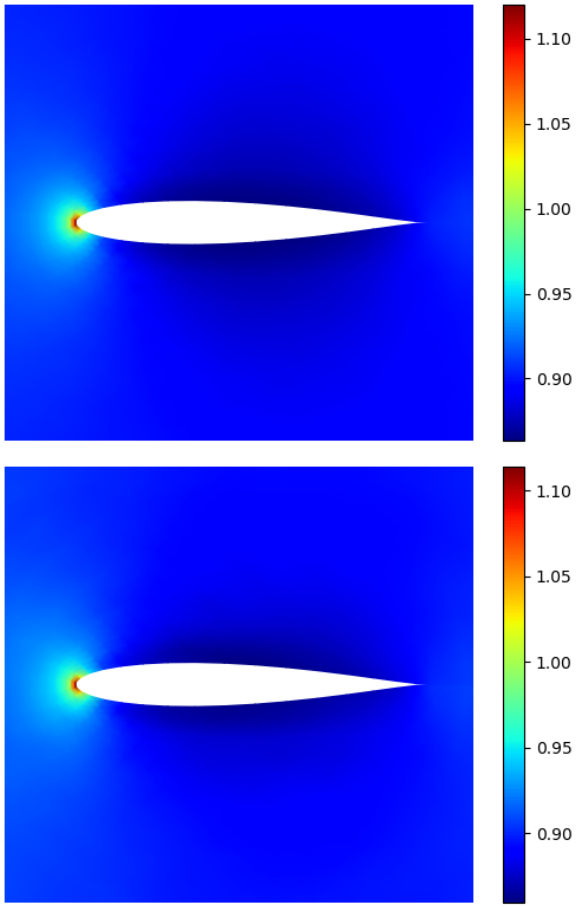


9210

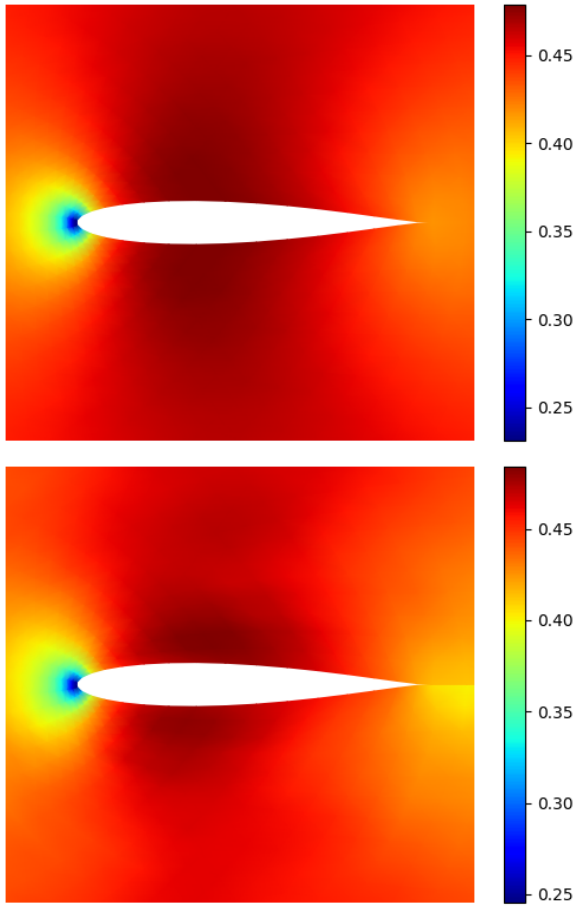


Prediction of Steady Flow Fields - Flow Field (top CNN, bottom CFD)

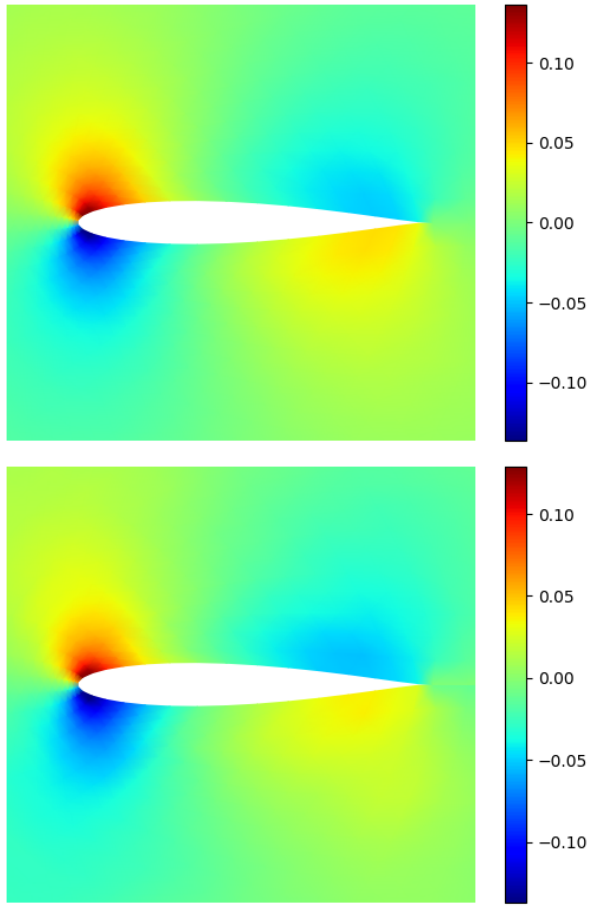
pressure



x-velocity

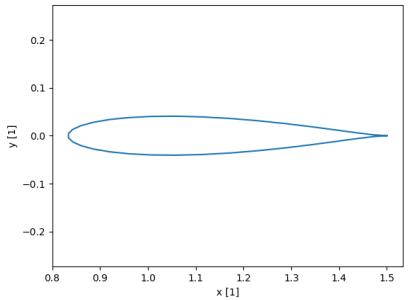


y-velocity

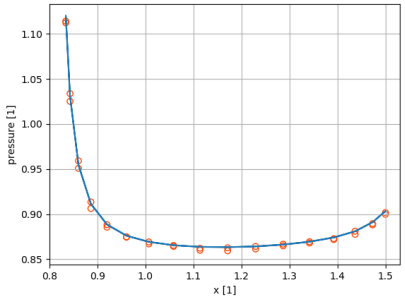


Prediction of Steady Flow Fields - Pressure Distribution

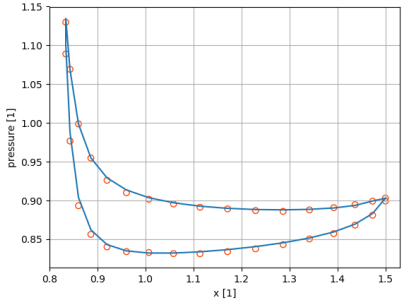
0012



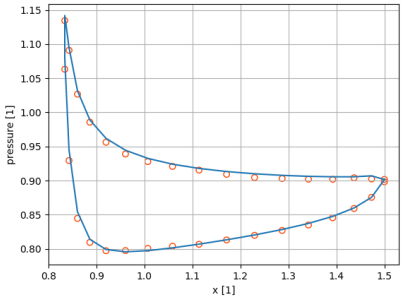
$\alpha = 0°$



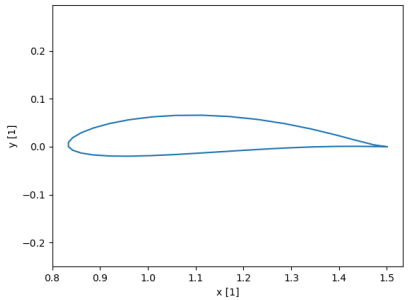
$\alpha = 5°$



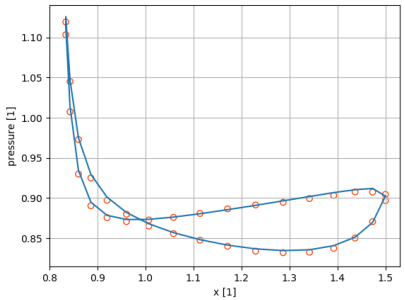
$\alpha = 10°$



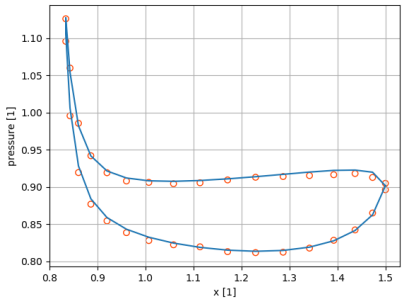
4412



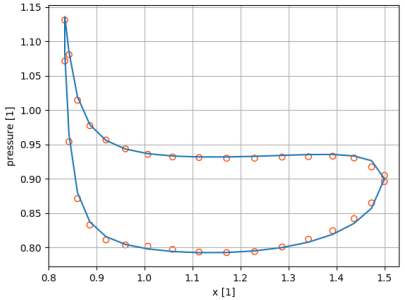
$\alpha = 0°$



$\alpha = 5°$



$\alpha = 10°$



Prediction of Steady Flow Fields - CNN Error

Absolute error $\left|c_L^{CFD} - c_L^{CNN}\right| \text{x}10^3$

α\airfoil	0012	0020	0030	2412	4412	6615	8607	9210
0	0.88	1.52	0.87	0.67	0.34	2.38	1.99	2.26
5	0.93	0.85	0.69	1.57	0.13	3.50	0.96	1.39
10	1.97	0.57	2.68	2.08	2.02	5.37	0.59	0.31
20	4.82	2.93	3.82	4.67	3.98	4.25	4.44	1.72

Relative error $\frac{\left|c_L^{CFD} - c_L^{CNN}\right|}{\left|c_L^{CFD}\right|} \text{x}100$

α\airfoil	0012	0020	0030	2412	4412	6615	8607	9210
0	-	-	-	**7.00**	1.78	**9.71**	**5.91**	**5.08**
5	2.57	2.63	2.60	3.45	0.24	**6.12**	2.07	4.76
10	2.89	0.93	**5.30**	2.71	2.39	**6.29**	0.75	0.56
20	4.51	3.00	4.54	4.11	3.32	3.52	3.88	1.92

Prediction of Steady Flow Fields - Summary

- Structured mesh with 64 x 32 = 2048 cells
- Training data set consisting of 1866 aerofoils

CFD solution:

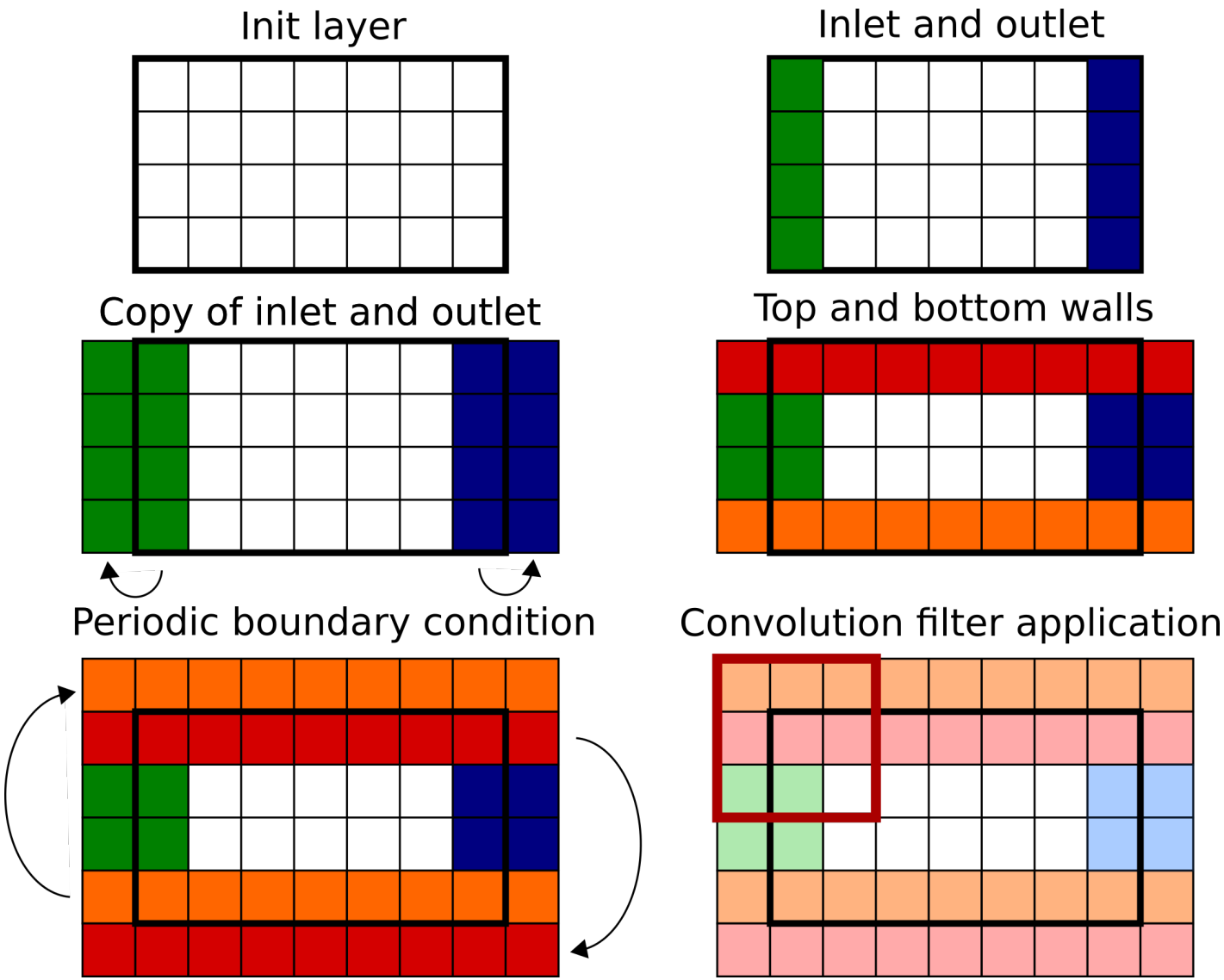
- DG method (FlowPro)
- First order of spatial accuracy
- Total CPU time to simulate flow for 1866 airfoils: **4.5hour**
- Average CPU time of a simulation for one aerofoil: **8.7s**
- Average CPU time of a simulation with second order of accuracy for one aerofoil: 26s

CNN solution:

- Total CPU time of 1866 airfoils: **10.7s**
- CPU time of one solution: **0.0057s**

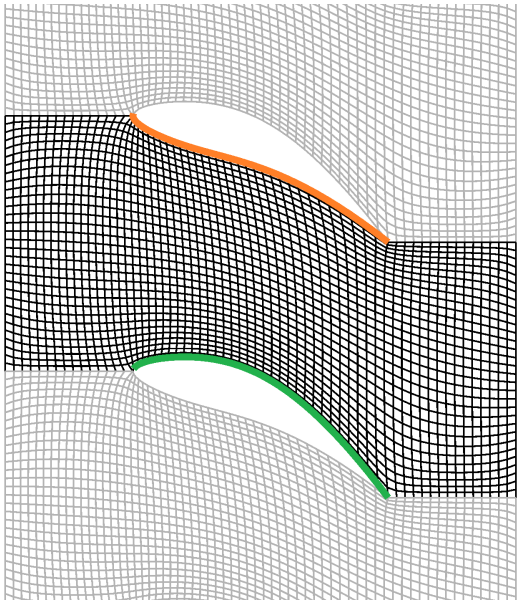
- The convolution neural network gives solution **1500 times faster** than a classical CFD solver.

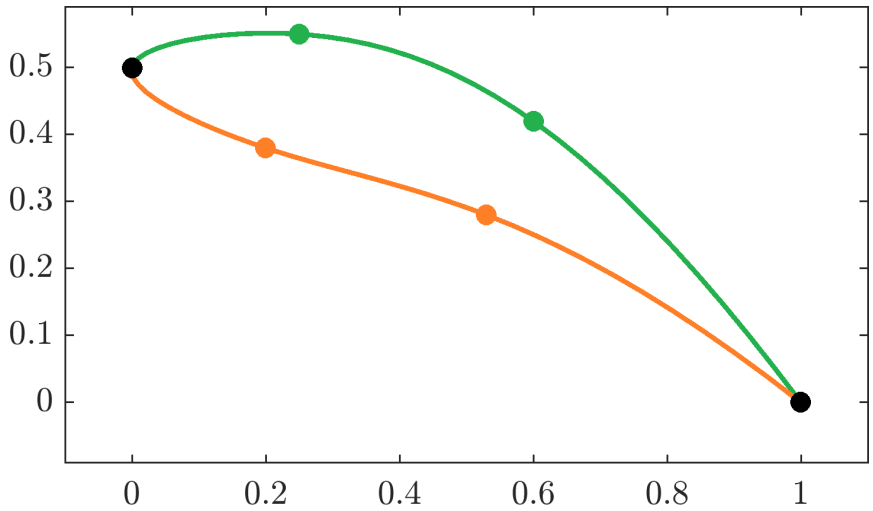
Main Neural Network Architecture - Convolution With Periodic Padding



Problem Setup

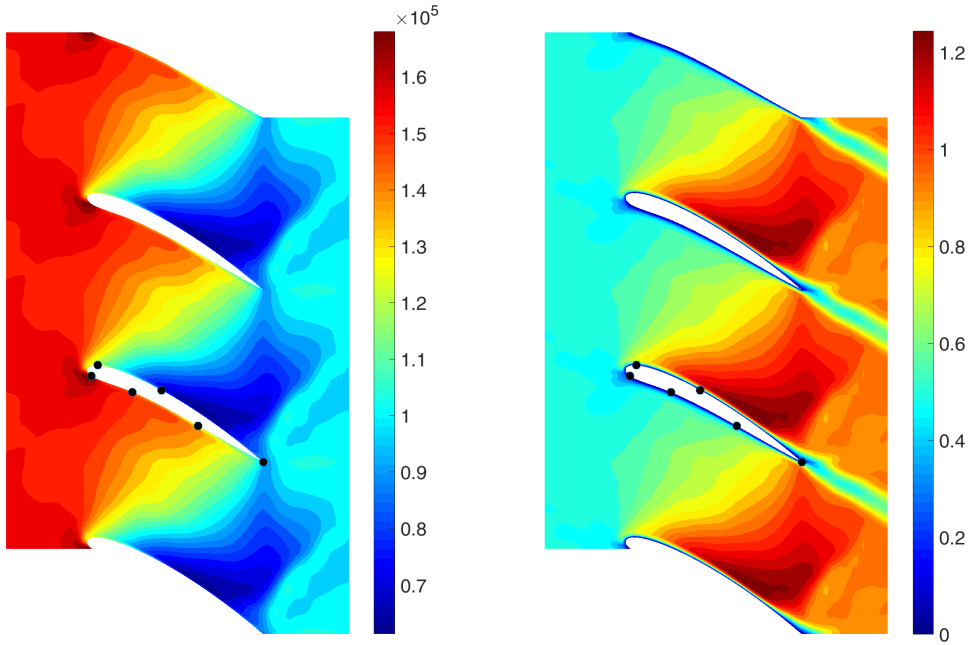
- Laminar fluid flow in a blade cascade
- Angle of attack $\alpha = 10°$
- Mach numbers Ma $= 0.9$
- Reynolds numbers Re $= 10000$
- Structured grid with 64 x 32 points, generated by elliptic mesh generator
- Periodic boundary condition
- Blade shape is described by cubic spline with 6 control points





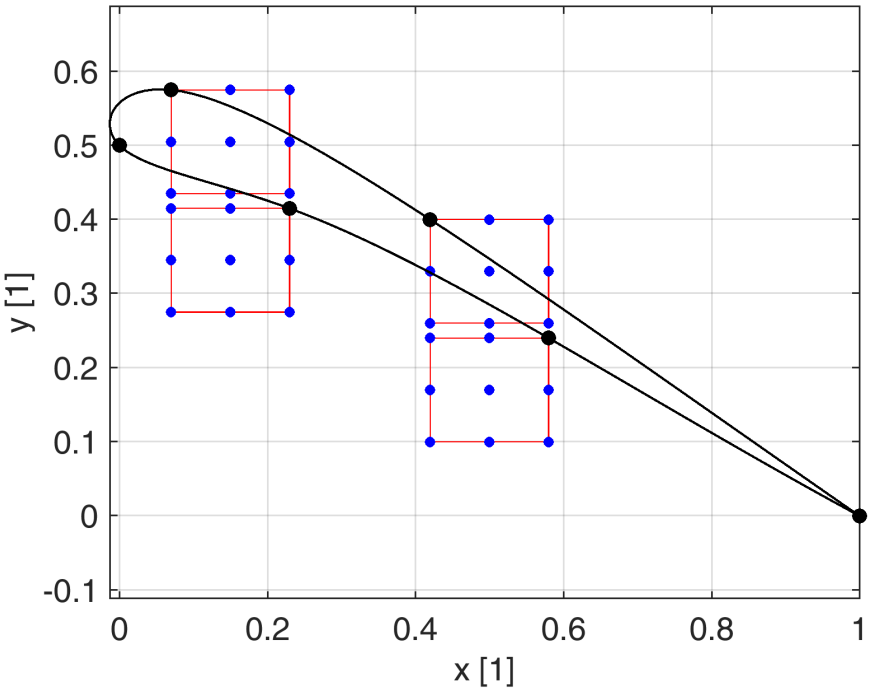
Neural Network - Summary

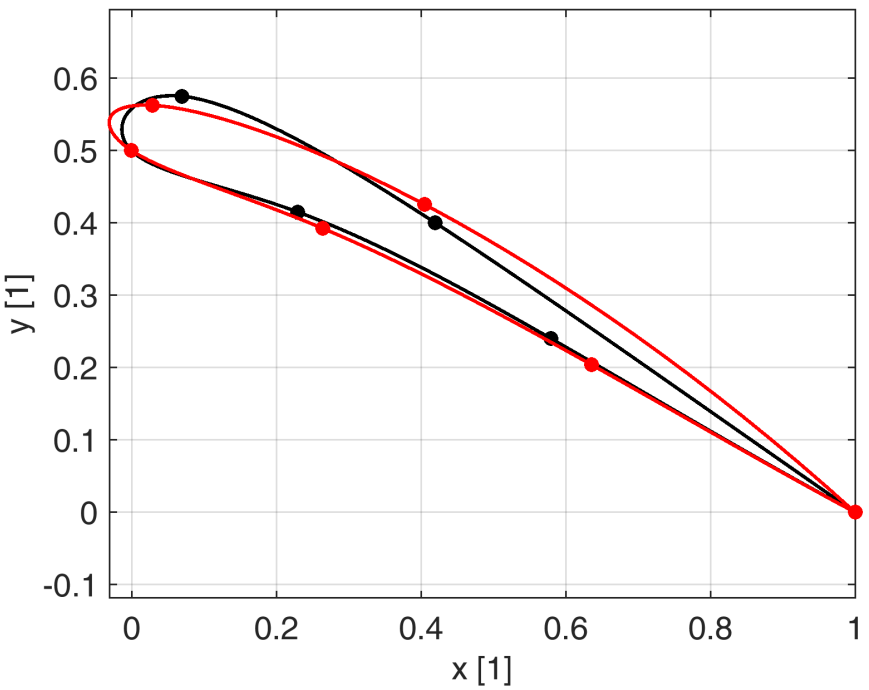
- **Input tensor** $[n_{spec}, n_1, n_2, 3]$, $(X, Y, walls)$ grid coordinates and wall markers ($n_1 = 64$ x $32 = n_2$ points)
- **Output tensor** $[n_{spec}, n_1, n_2, 4]$: flow field $(u_x,\ u_y,\ p,\ \varrho)$
- 402 928 trainable parameters
- Trained on the set of 136 random aerofoils
- Keras and TensorFlow libraries, Python interface



Application - Blade Optimization

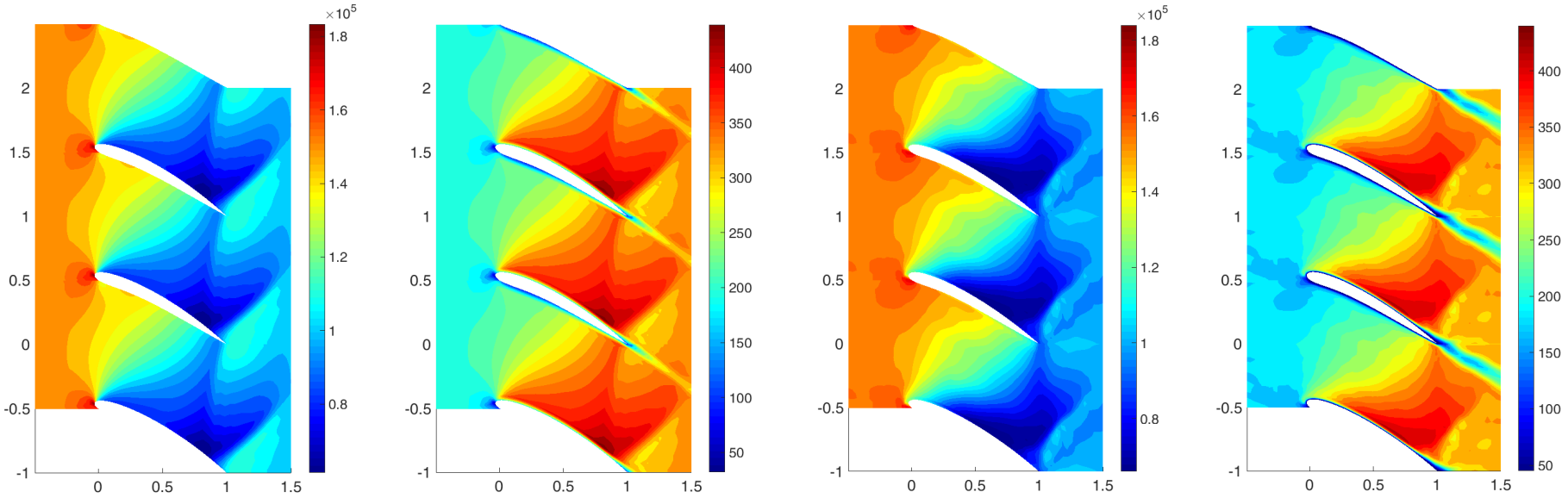
- Blade profile optimization for the inlet Mach number $M = 0.95$
- Target functional: $\max(f(\boldsymbol{x})), f(\boldsymbol{x}) = \frac{c_L(x)}{1+c_D(x)}, \quad c_L = \oint_\Gamma p\, n_y, \quad c_D = \oint_\Gamma p\, n_x$
- Algorithm of optimization:
  - First step roughly search the state space - $9^4 = 6561$ combinations of control points - **13.3s of CPU time**
  - Second step perform 100 steps of gradient descent method - **32s of CPU time**





Application - Blade Optimization

Comparison of flow fields for optimal blade



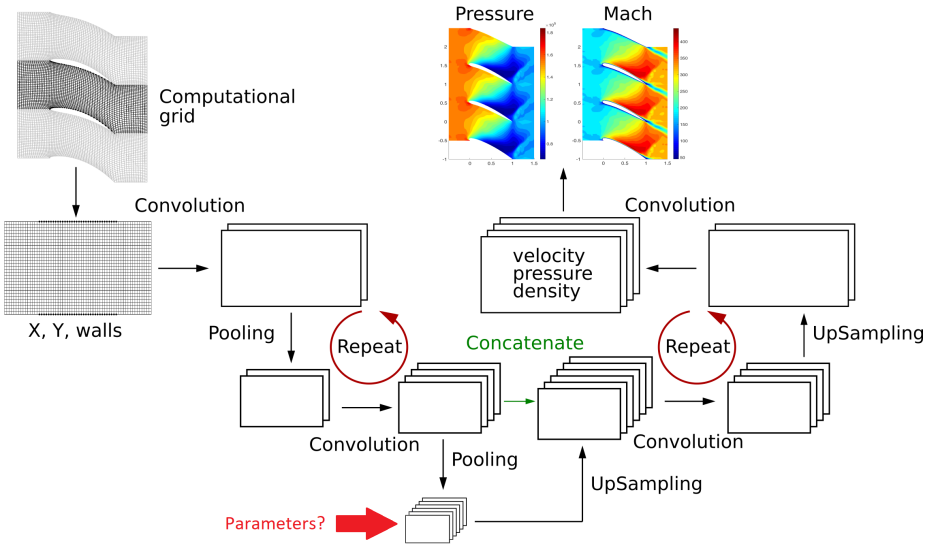
FlowPro (CFD software)

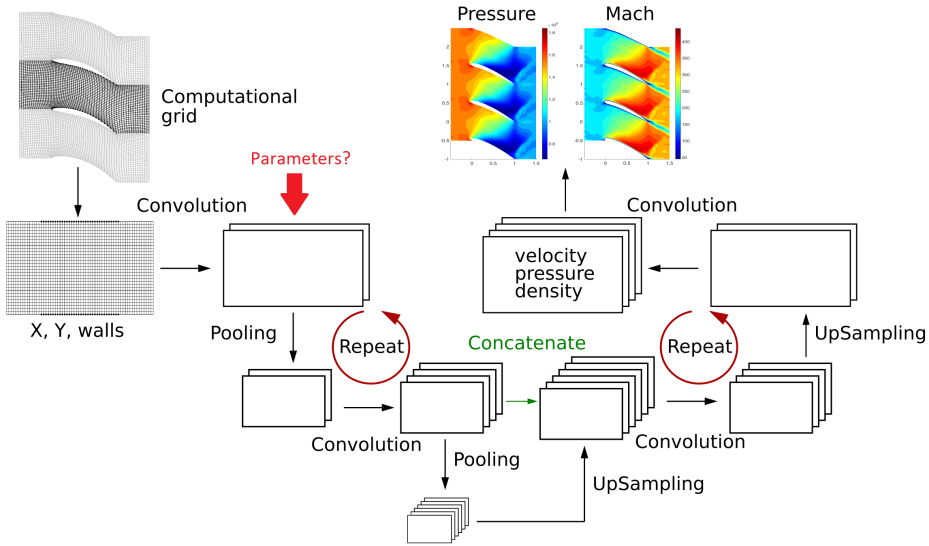
Neural network prediction

Parametrization

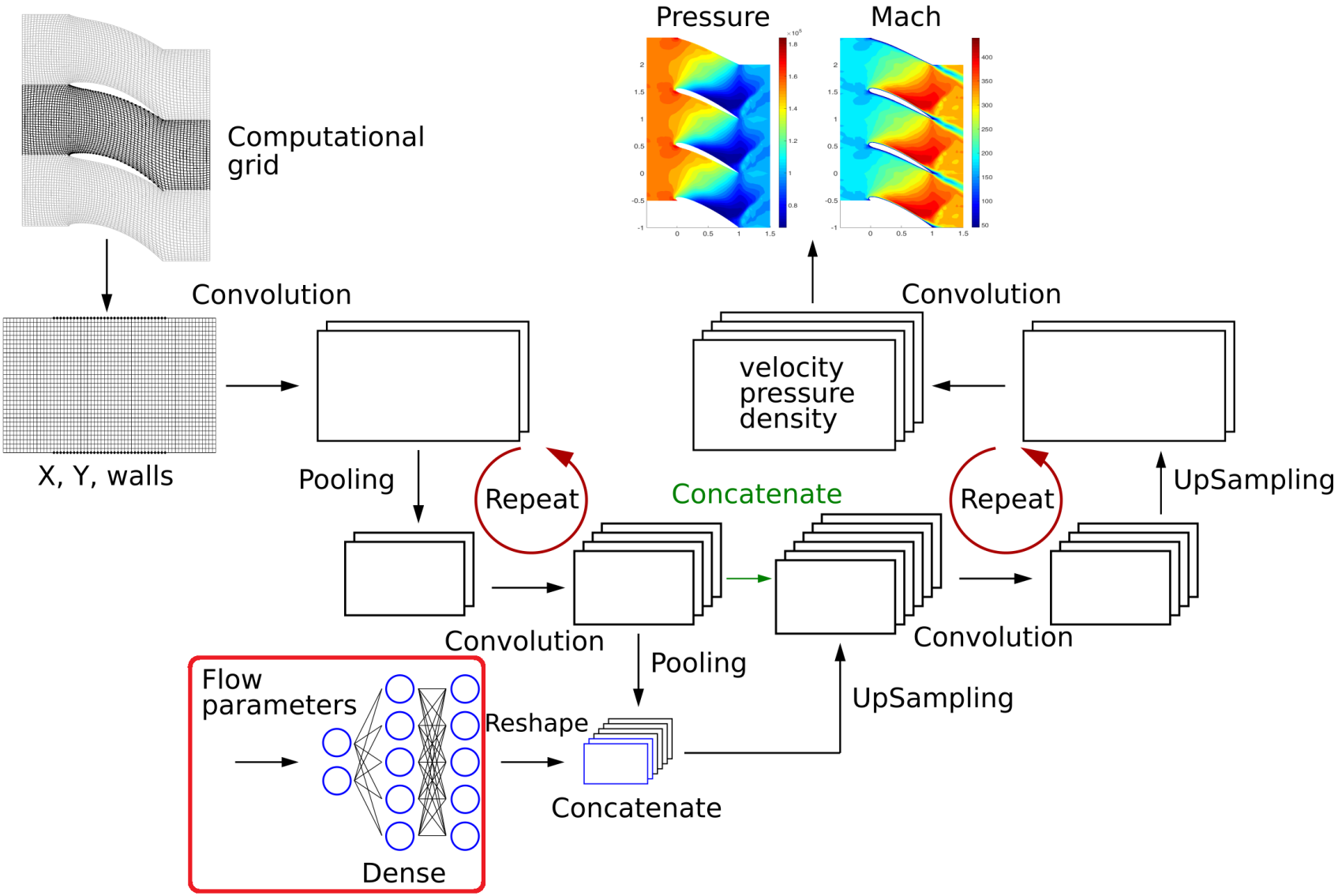
Parametrization

- How to include parameters in the neural network?
- In general, the sooner is the better



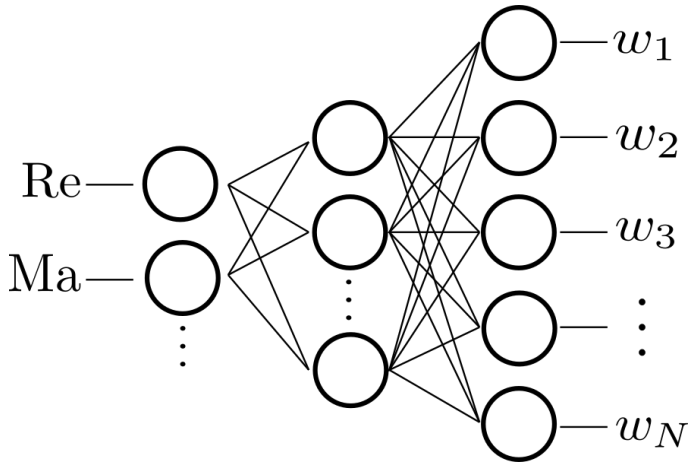


Main Neural Network Architecture - U-net

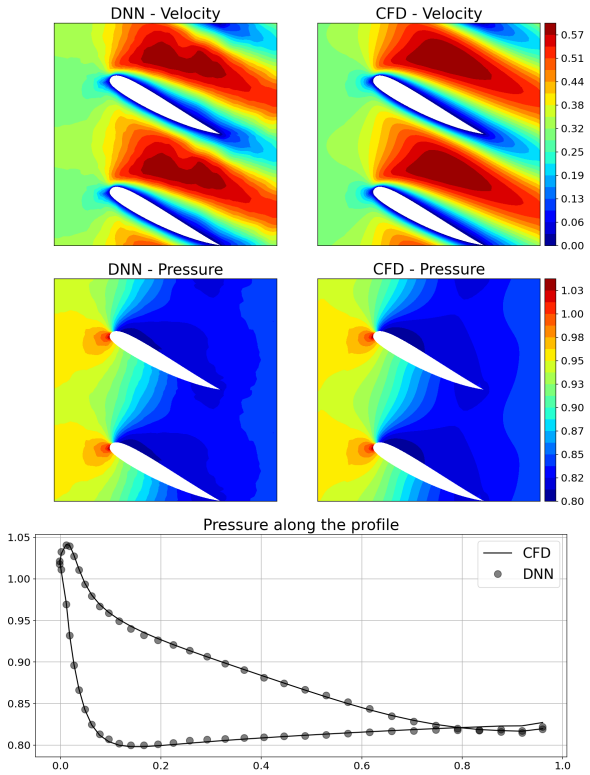


Hyper Neural Network

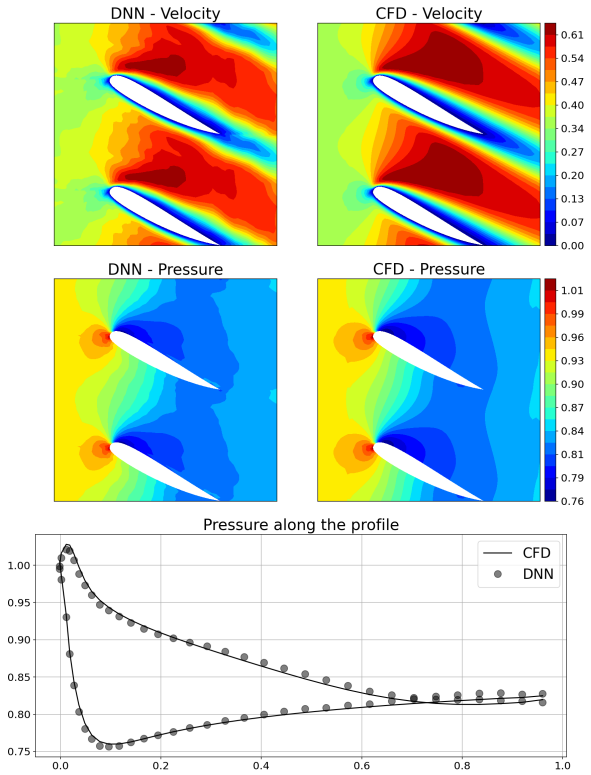
- Used for parametrization of a main network
- Main network is trained for all combinations of flow parameters and resulting weights are stored
- Map flow parameters into main neural network weights
- Dense neural network - one hidden layer



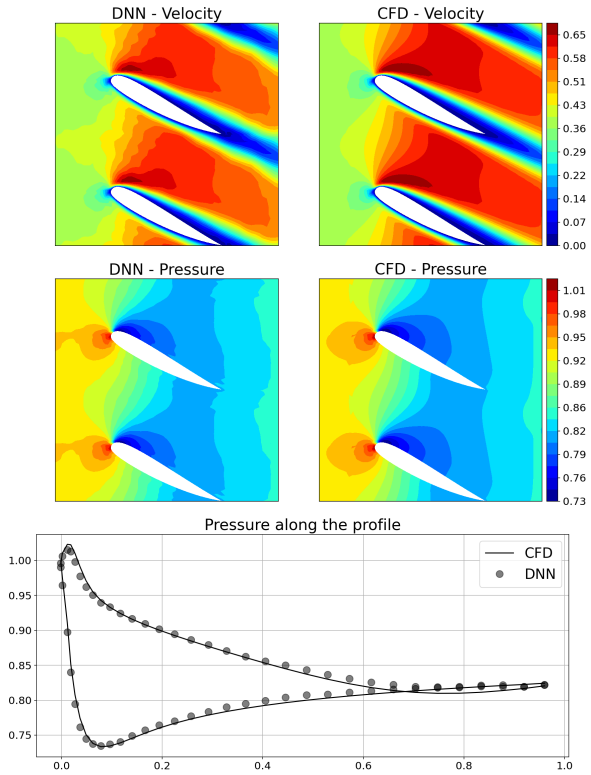
Hyper Neural Network - Single Parameter Re



$Re = 100$



$Re = 250$



$Re = 500$

Hyper Neural Network - Single Parameter Re

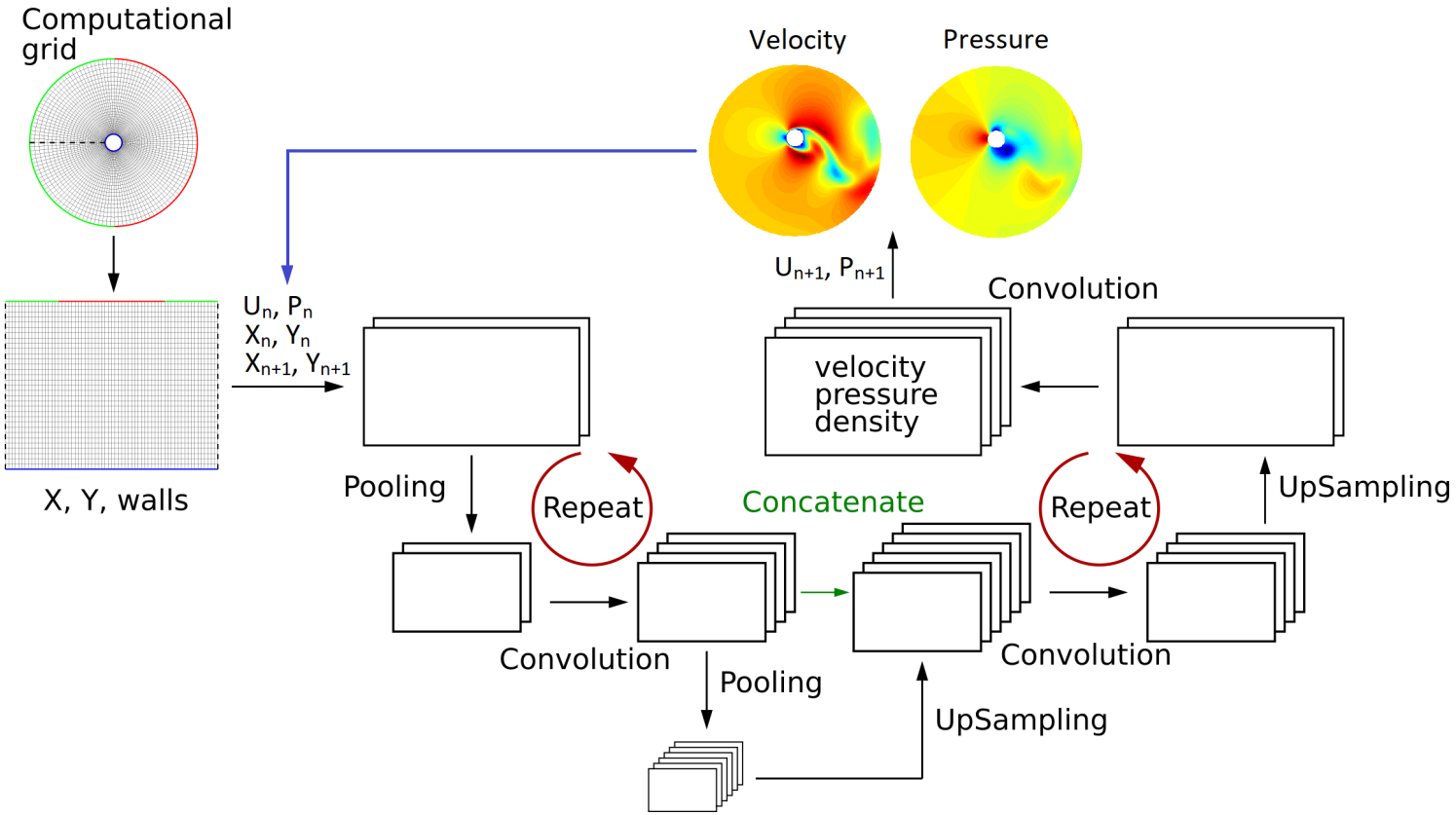
name	symbol	value
heat capacity ratio	κ	1.4
Training Reynolds numbers	Re	$100, 500, 1000$
Prandtl number	Pr	0.72
pressure ratio	$p_{\text{out}}/p_{\text{in}_0}$	0.843
angle of attack	α	15°

Re	Drag		Lift	
	average err [%]	SD	average err [%]	SD
100	1.9	1.2	1.1	0.5
250	**3.6**	**3.7**	**4.1**	**1.3**
500	3.9	1.4	2.4	1.4
750	**3.3**	**2.3**	**2.7**	**2.0**
1000	2.9	1.7	3.0	2.3

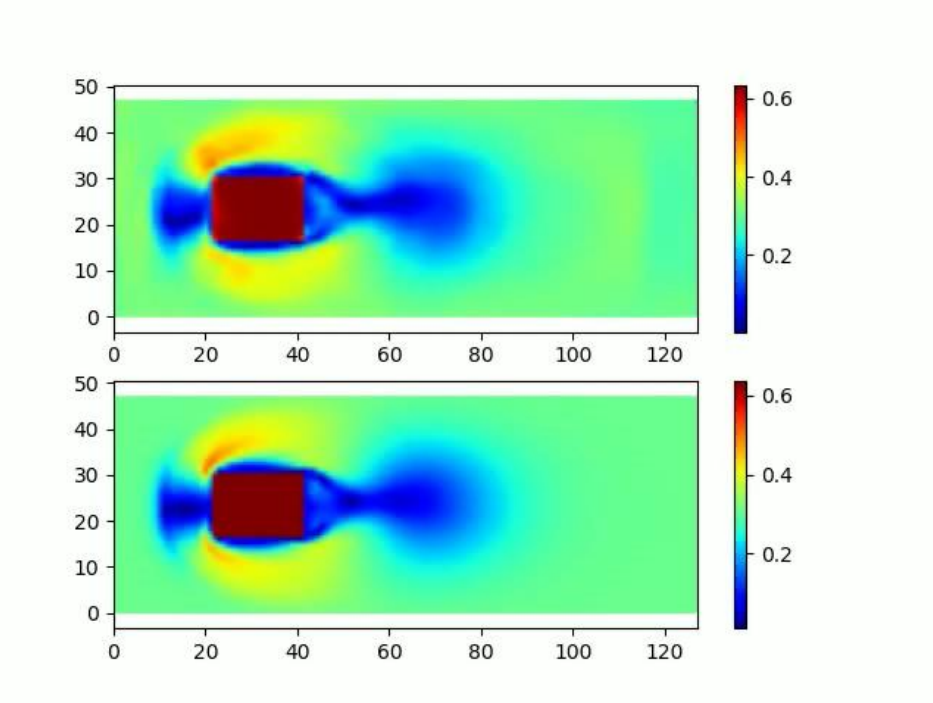
Prediction of unsteady flow field

Neural network architecture

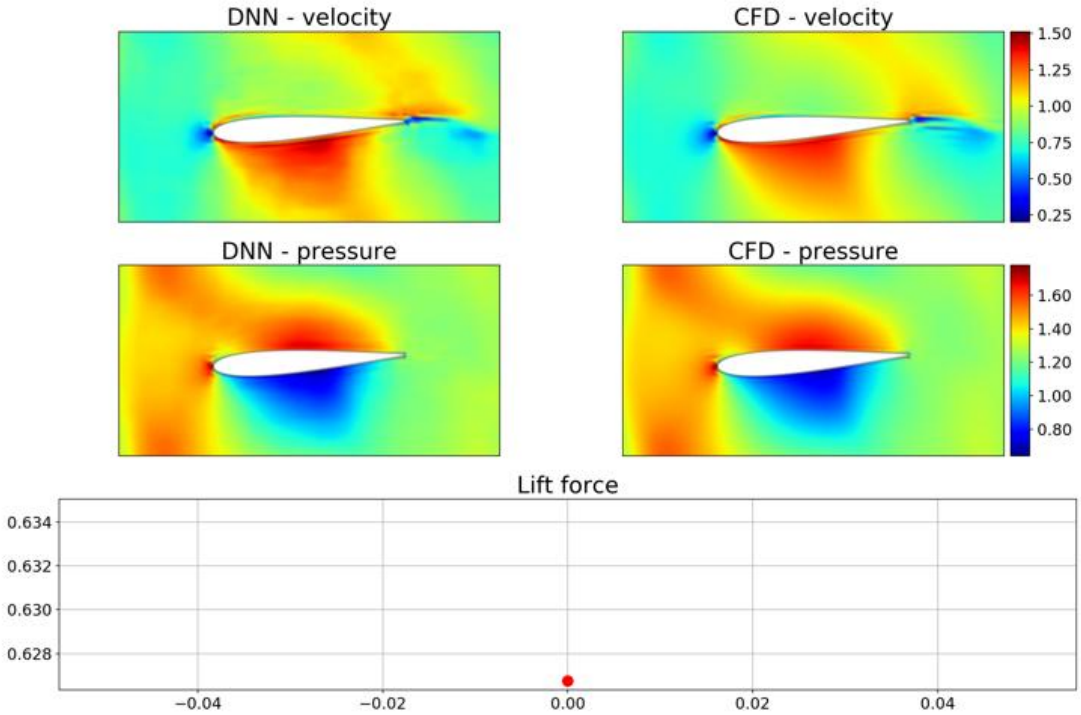
- The architecture is same as in the case of prediction steady flow field
- The solution of n time level is added as another input
- If the mesh is moving, the points coordinates in $n+1$ time level are also added as another input
- The outputs is the solution at $n+1$ time level



Unsteady flow field prediction



Flow field prediction with moving mesh



Fluid Structure Interaction

Vortex induced vibrations

- Structure equation of motion:

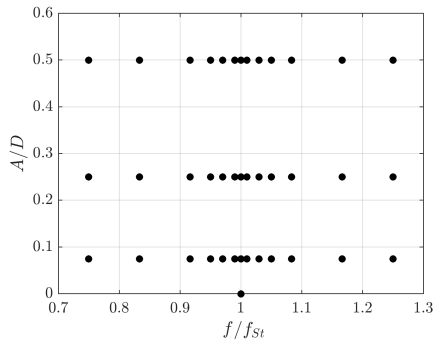
$$\ddot{y} + 2\,\zeta\,\omega_n\,\dot{y} + \omega_n^2\,y = \frac{L}{m}$$

- Lift force:

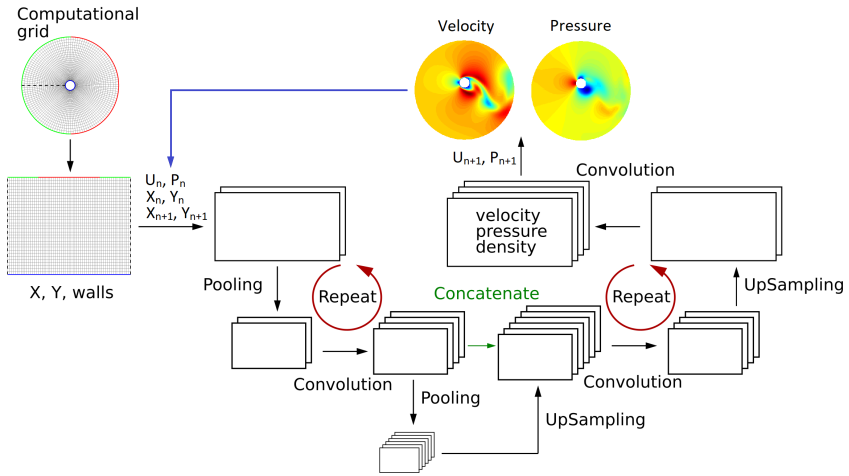
$$L = \oint_\Gamma \left(\sigma_{xx}\,n_x + \sigma_{yx}\,n_y\right)\,dS$$

- Parameters:
  - Damping ratio: $\zeta = \frac{c}{2m\omega_n}$
  - Stiffness: $k = m\,\omega_n^2$
  - Mass: $m = 10$
  - Damping: $c = 0.25$
  - Natural frequency: $f_n = \frac{\omega_n}{2\pi} = f_{St}$
  - Strouhal frequency: $f_{St} = \frac{\mu}{\varrho_\infty L^2} 0.212(\mathrm{Re} - 21.2)$

- Convolution neural network predict unsteady flow-field with moving boundary
- Training frequencies and amplitudes:

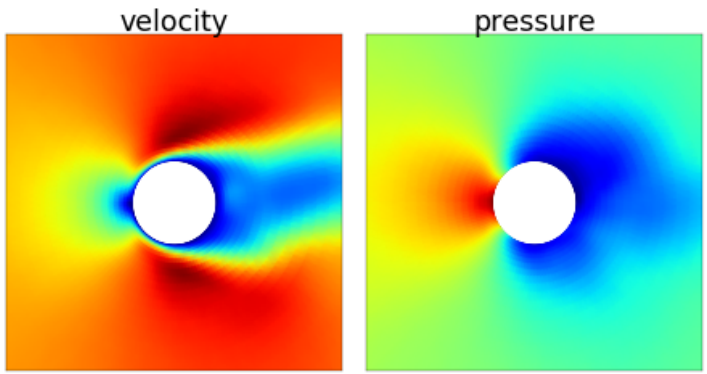


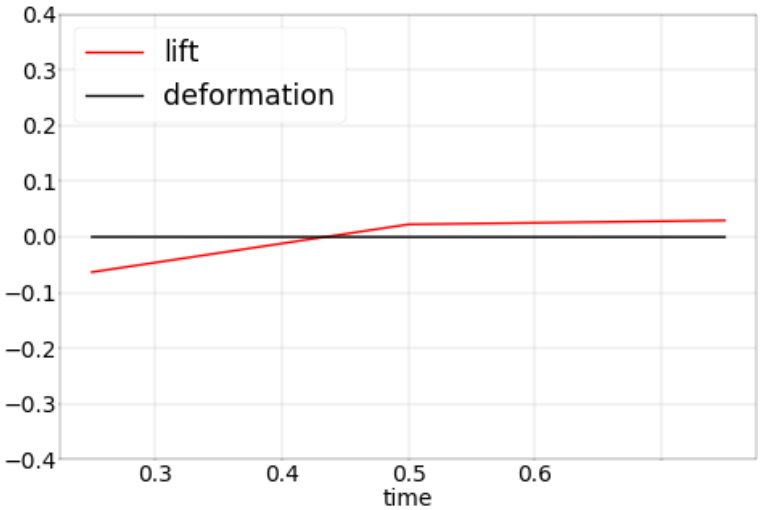
- Convolution neural network architecture:

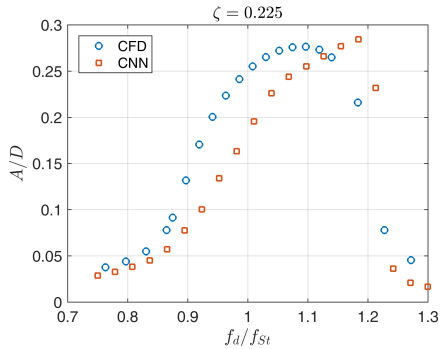


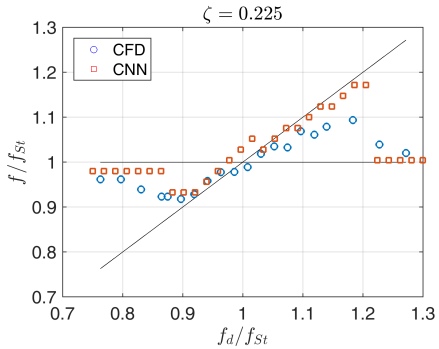
Vortex induced vibration

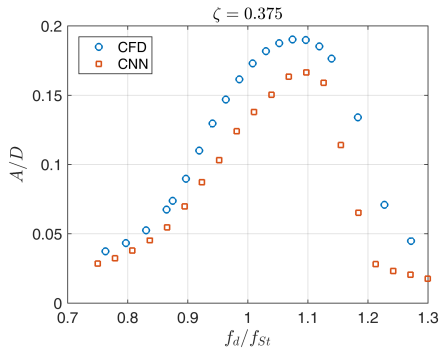
CNN predicted unsteady flow field

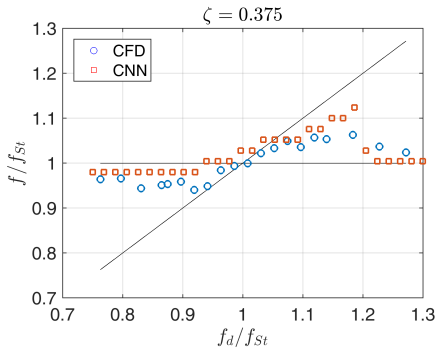


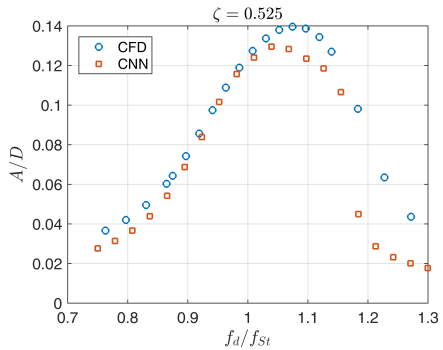


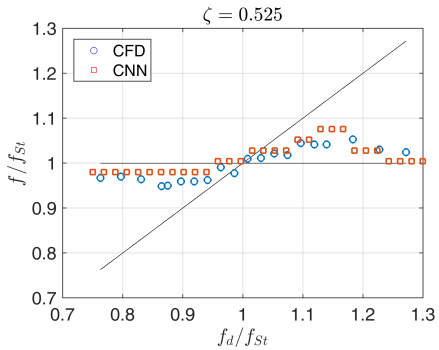








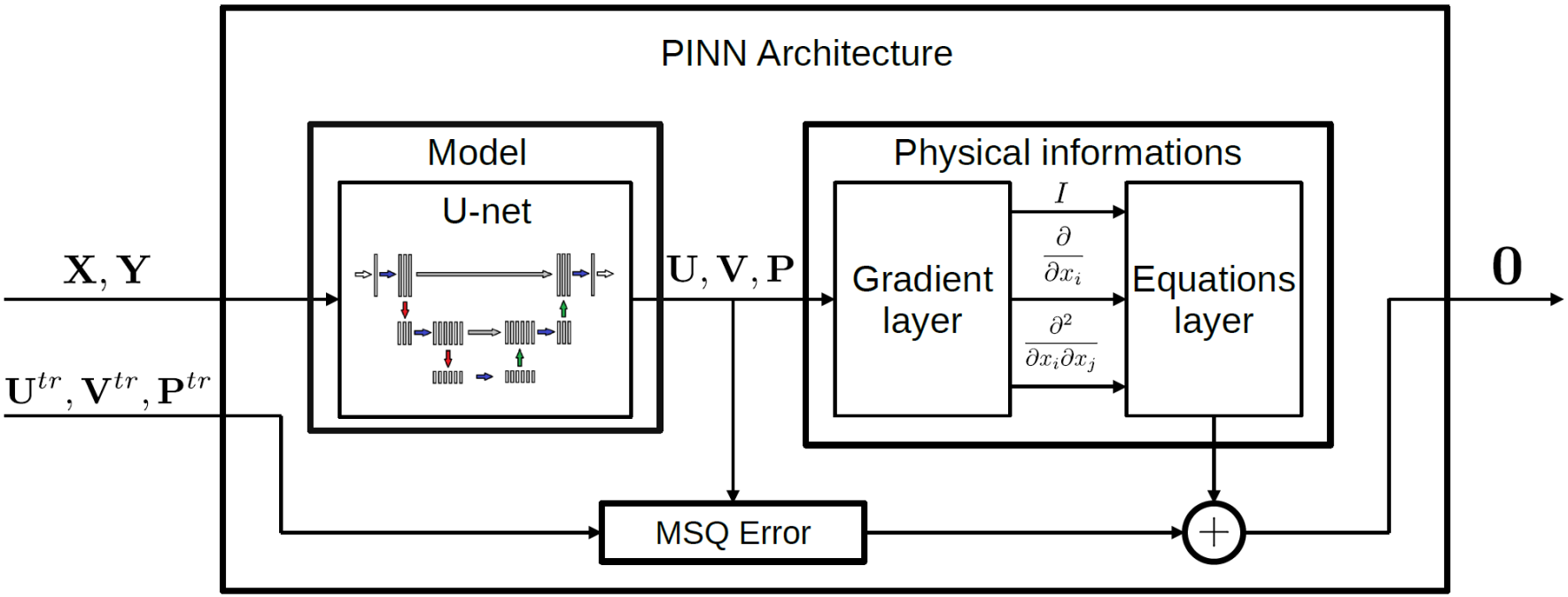




Physic Informed Neural Network

PINN architecture with convolution neural network

- implements the PDE of the problem into the loss function
- the model could be trained without training samples



Gradient layer

- compute first and second derivatives of the flow field variables u, v and p
- use Lagrangian interpolation

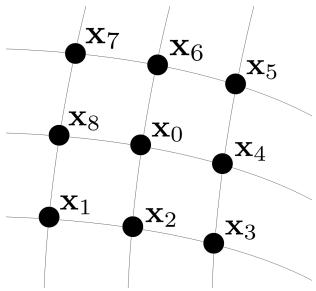
$$\mathcal{L}_j(\mathbf{x}_i) = a_0 + a_1 x_i + a_2 y_i + a_3 x_i y_i + a_4 x_i^2 + a_5 y_i^2 + a_6 x_i^2 y_i + a_7 x_i y_i^2$$

$$\mathcal{L}_j(\mathbf{x}_i) = \delta_{ji}, \quad \forall \quad i, j = 1, 2, \ldots, 8$$

$$\frac{\partial \mathcal{L}_j}{\partial x}(\mathbf{x}_i) = a_1 + a_3 y_i + 2a_4 x_i + 2a_6 x_i y_i + a_7 y_i^2$$

$$\frac{\partial \mathcal{L}_j}{\partial y}(\mathbf{x}_i) = a_2 + a_3 x_i + 2a_5 y_i + a_6 x_i^2 + 2a_7 x_i y_i$$

$$\frac{\partial^2 \mathcal{L}_j}{\partial x^2}(\mathbf{x}_i) = 2a_4 + 2a_6 y_i, \quad \frac{\partial^2 \mathcal{L}_j}{\partial y^2}(\mathbf{x}_i) = 2a_5 + 2a_7 x_i$$



- coefficients a_i are pre-calculated and stored for each internal point of the training computation network
- resulting formulas for the first and second derivatives

$$\left.\frac{\partial u}{\partial x}\right|_{x_0} = \sum_{j=1}^{n} u_j \frac{\partial \mathcal{L}_j}{\partial x}(x_0), \quad \left.\frac{\partial u}{\partial y}\right|_{x_0} = \sum_{j=1}^{n} u_j \frac{\partial \mathcal{L}_j}{\partial y}(x_0),$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_0} = \sum_{j=1}^{n} u_j \frac{\partial \mathcal{L}_j^2}{\partial x^2}(x_0), \quad \left.\frac{\partial^2 u}{\partial xy}\right|_{x_0} = \sum_{j=1}^{n} u_j \frac{\partial \mathcal{L}_j^2}{\partial xy}(x_0), \quad \left.\frac{\partial^2 u}{\partial y^2}\right|_{x_0} = \sum_{j=1}^{n} u_j \frac{\partial \mathcal{L}_j^2}{\partial y^2}(x_0).$$

Equation layer

- use derivatives computed in gradient layer
- evaluate loss function according to problem PDE

$$\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = 0$$

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} = \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

$$u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} = \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)$$

- loss function

$$\mathcal{C}(u,v) = \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = 0$$

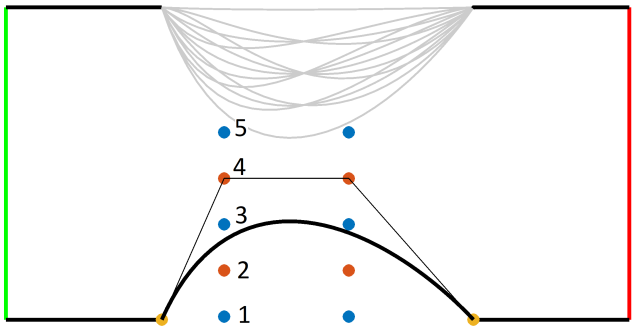
$$\mathcal{M}_x(u,v,p) = u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0$$

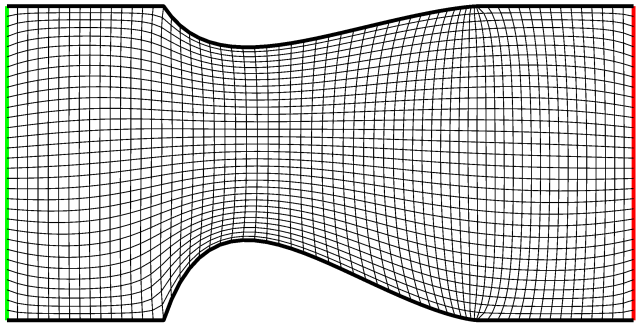
$$\mathcal{M}_y(u,v,p) = u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) = 0$$

$$Loss_{eq} = \lambda_C \left\|\mathcal{C}(u,v)\right\| + \lambda_M \left\|\mathcal{M}_x(u,v,p)\right\| + \lambda_M \left\|\mathcal{M}_y(u,v,p)\right\|$$

Problem description

- incompressible fluid flow in the channel
- the walls are formed by Bezier curves
- 5 admissible positions of each control point





- geometrical variants

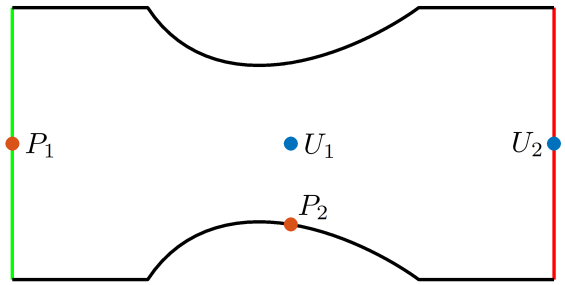
geometry set	control points set	number of samples
G_{135}	$\{1, 3, 5\}$	81
G_{15}	$\{1, 5\}$	16
G_{24}	$\{2, 4\}$	16

Problem description

- analyzed variants

variant	geometry set	CFD data set	PINN
A	G_{135}	T_{135}	false
B	G_{15}	T_{15}	false
C	G_{135}	0	true
D	G_{135}	T_{15}	true

- selected points for error measurement

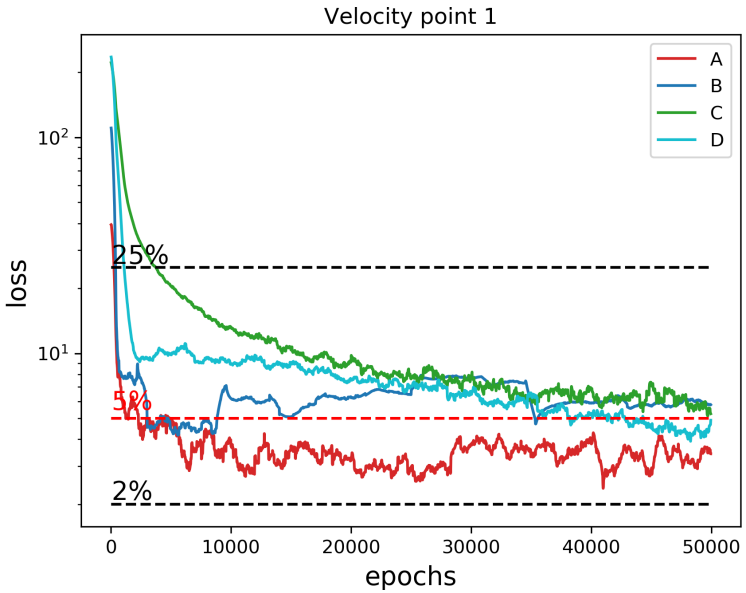


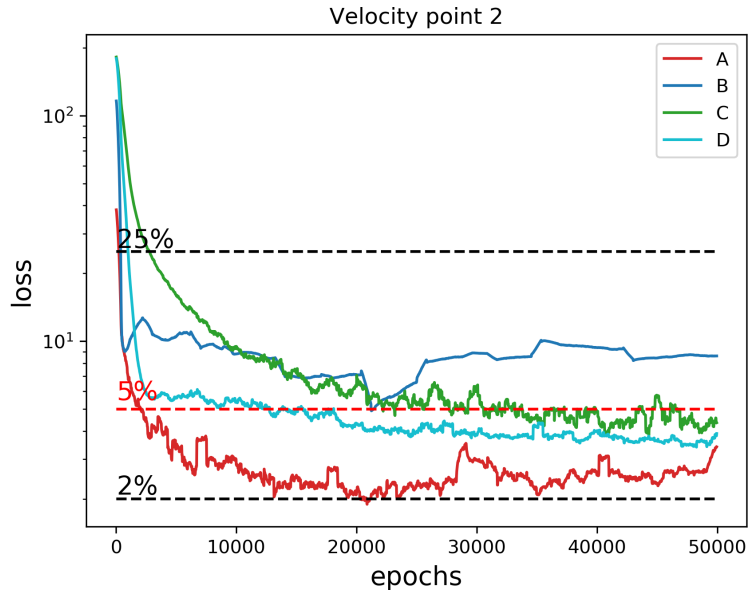
- error definition

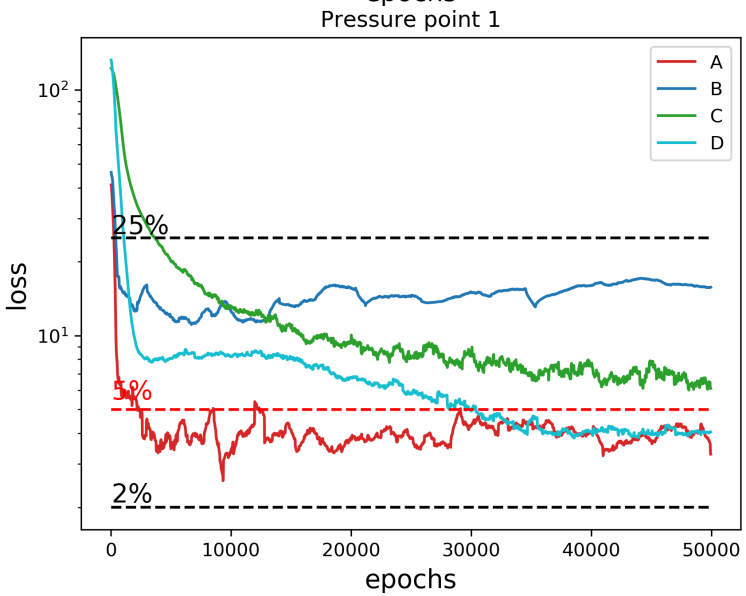
$$U_i^{err} = \frac{1}{n(G_{24})} \sum_{g \in G_{24}} \frac{U_i(g) - U_{ref}}{U_{ref}}, \quad i = 1, 2$$

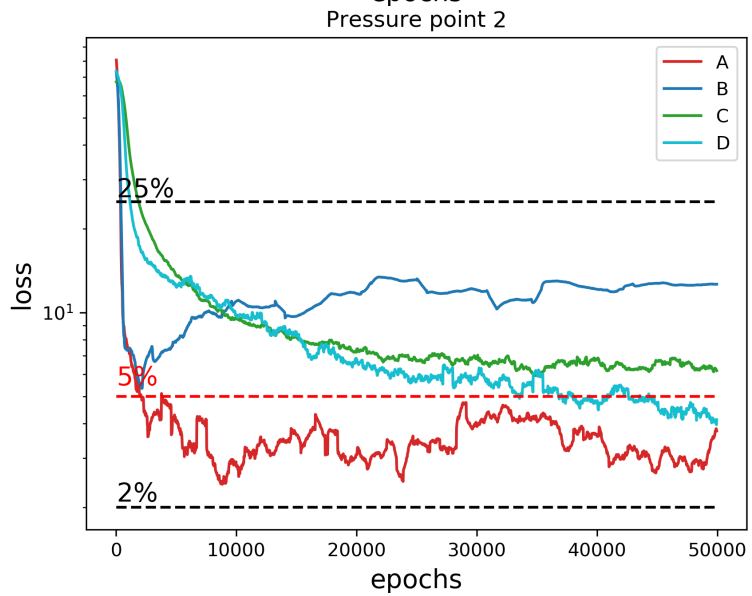
$$P_i^{err} = \frac{1}{n(G_{24})} \sum_{g \in G_{24}} \frac{P_i(g) - P_{ref}}{P_{ref}}, \quad i = 1, 2$$

Results

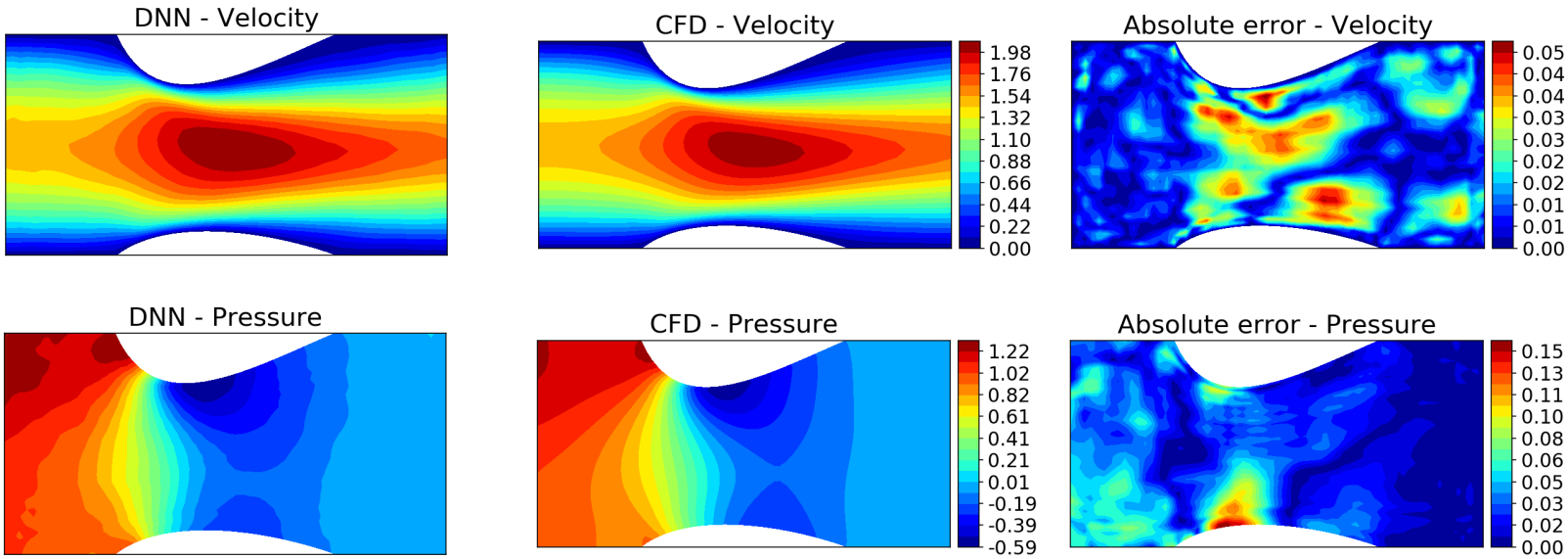

Velocity point 1
loss
epochs
A
B
C
D
25%
5%
2%

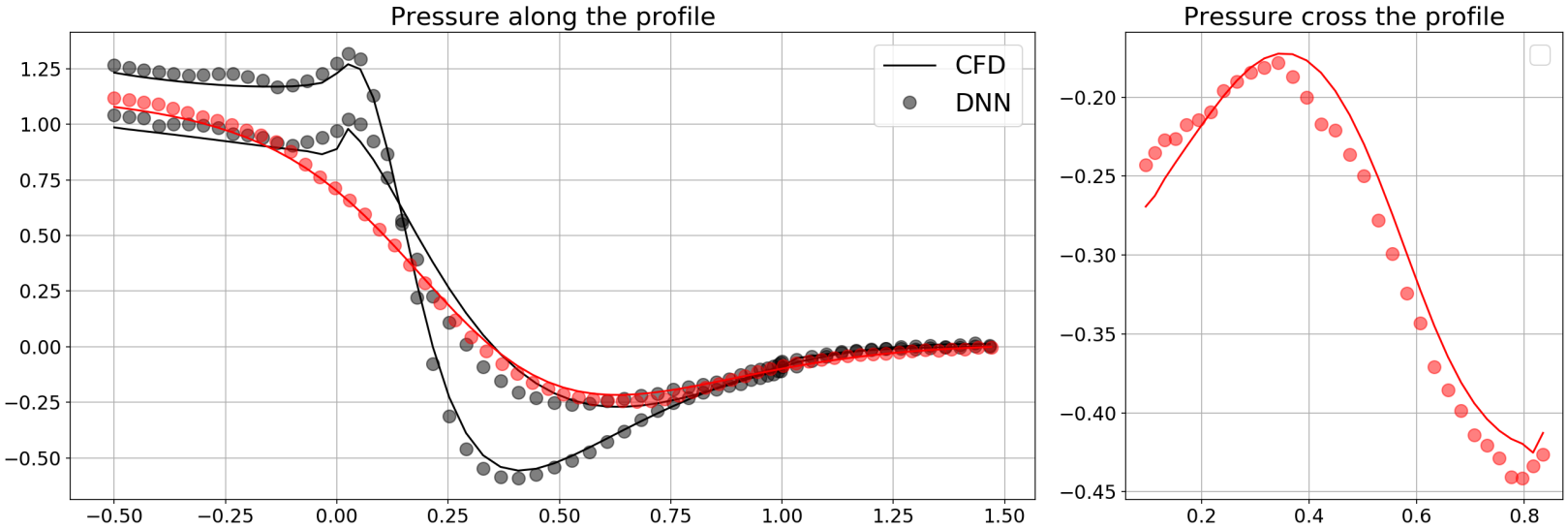

Velocity point 2
loss
epochs
A
B
C
D
25%
5%
2%


Pressure point 1
loss
epochs
A
B
C
D
25%
5%
2%


Pressure point 2
loss
epochs
A
B
C
D
25%
5%
2%

Results - test geometry: 4, variant: A





Results - test geometry: 4, variant: C