AE 622

Computing of High Speed Flows

(Stage 3)

Keywords: CNN, diamond-shaped airfoils, supersonic flows, U-net

FOILNET: A CONVOLUTION BASED NEURAL NETWORK FOR PREDICTION OF PRESSURE FIELD AROUND OSCILLATING AIRFOILS

HARSH ARORA, RANJEET JHA, AND PRADEEP KUMAR

Sanskar Maddi (190010041)
Venkatakrishnan V K (19D180032)

Abstract

- A deep learning model that can predict the pressure contours around airfoils is presented.
- The model is trained with data generated by performing CFD simulations using **Reynolds Stress Model** by ANSYS Fluent for airfoils for a range of angle of attack (α).
- A Convolutional Neural Network (CNN) based U-Net Encoder-Decoder architecture is developed and named as 'foilNET'. The foilNET is found to yield accurate predictions on the airfoils from training set. While, the performance is found to be satisfactory on testing set.
- Furthermore, the deep learning model is found to be way faster as compared to CFD and also accurate in prediction of pressure field around the airfoil.
- The paper's emphasis is on subsonic airfoils, but in this project, we implement the same algorithms on supersonic airfoils.

Introduction

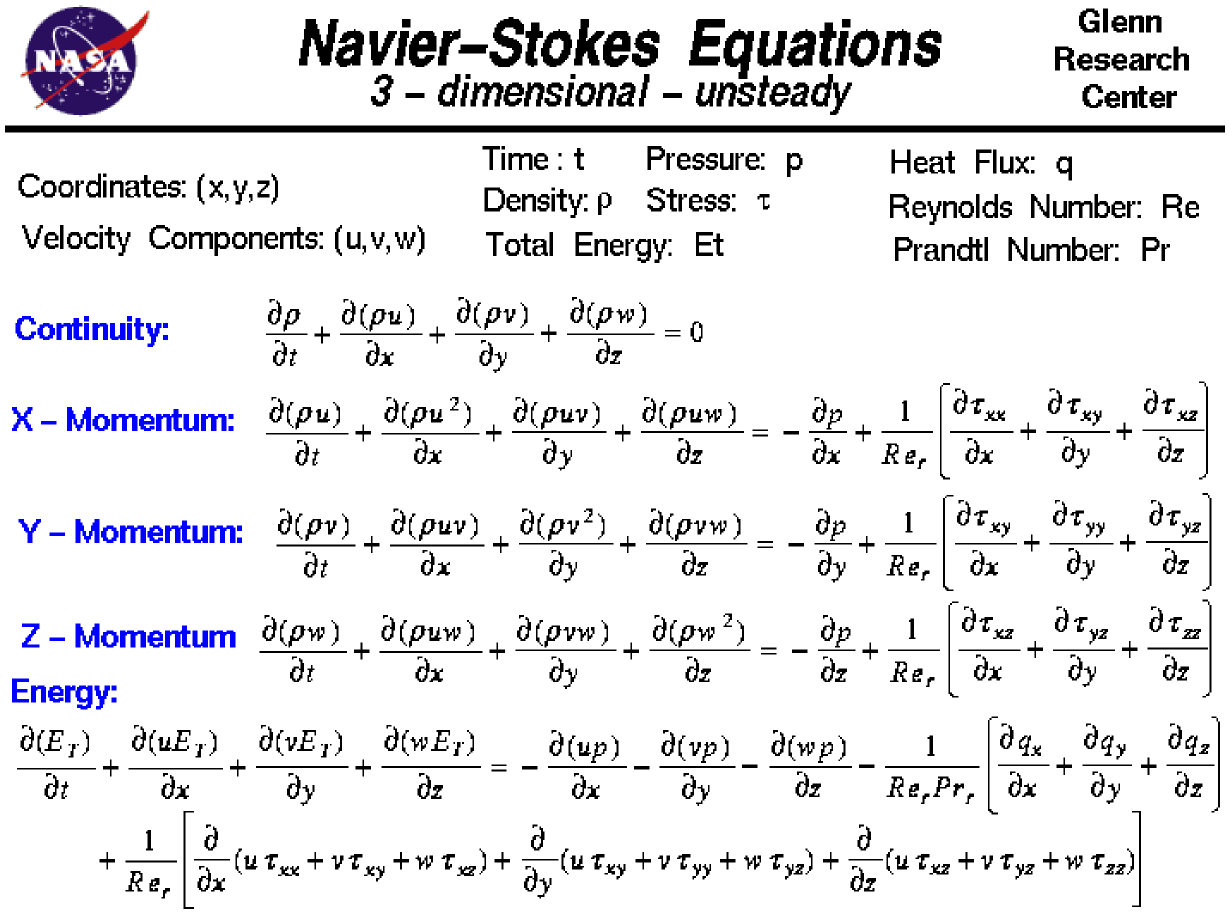
- In order to estimate the flow field around an airfoil, numerical experiments utilizing CFD are used to aid in the design and optimization of airfoil designs. To get the desired performance for a case, many simulations are run while changing the airfoil shape and angle of attack.
- This requires a significant amount of time and resources. On the other hand, deep learning is now a powerful tool for learning and predicting high dimensional data because of the improvements in the field and the use of Convolutional Neural Networks (CNN).
- Therefore, deep learning can assist with not only making effective use of the existing CFD data but also with quick physical quantity prediction that may be many orders of magnitude faster than the traditional CFD techniques.

Objective

- This objective of the study is to explore the ability of deep learning to learn the pressure field from the shown data and predict pressure field around the airfoil between 0^o to 21^o angle of attack
- The overall approach consists of four parts –
  - Preparing the model
  - Generating data for training
  - Training the model
  - Defining performance parameters to evaluate the performance of the model.

CFD Simulation

Governing equations



Navier–Stokes Equations

3 – dimensional – unsteady

Glenn
Research
Center

Coordinates: (x,y,z)

Velocity Components: (u,v,w)

Time : t   Pressure: p

Density: ρ   Stress: τ

Total Energy: Et

Heat Flux: q

Reynolds Number: Re

Prandtl Number: Pr

Continuity:
$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial(\rho w)}{\partial z} = 0$$

X – Momentum:
$$\frac{\partial(\rho u)}{\partial t} + \frac{\partial(\rho u^2)}{\partial x} + \frac{\partial(\rho uv)}{\partial y} + \frac{\partial(\rho uw)}{\partial z} = -\frac{\partial p}{\partial x} + \frac{1}{Re_r}\left[\frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \tau_{xz}}{\partial z}\right]$$

Y – Momentum:
$$\frac{\partial(\rho v)}{\partial t} + \frac{\partial(\rho uv)}{\partial x} + \frac{\partial(\rho v^2)}{\partial y} + \frac{\partial(\rho vw)}{\partial z} = -\frac{\partial p}{\partial y} + \frac{1}{Re_r}\left[\frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{yz}}{\partial z}\right]$$

Z – Momentum
$$\frac{\partial(\rho w)}{\partial t} + \frac{\partial(\rho uw)}{\partial x} + \frac{\partial(\rho vw)}{\partial y} + \frac{\partial(\rho w^2)}{\partial z} = -\frac{\partial p}{\partial z} + \frac{1}{Re_r}\left[\frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z}\right]$$

Energy:
$$\frac{\partial(E_T)}{\partial t} + \frac{\partial(uE_T)}{\partial x} + \frac{\partial(vE_T)}{\partial y} + \frac{\partial(wE_T)}{\partial z} = -\frac{\partial(up)}{\partial x} - \frac{\partial(vp)}{\partial y} - \frac{\partial(wp)}{\partial z} - \frac{1}{Re_r Pr_r}\left[\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} + \frac{\partial q_z}{\partial z}\right]$$
$$+ \frac{1}{Re_r}\left[\frac{\partial}{\partial x}(u\,\tau_{xx} + v\,\tau_{xy} + w\,\tau_{xz}) + \frac{\partial}{\partial y}(u\,\tau_{xy} + v\,\tau_{yy} + w\,\tau_{yz}) + \frac{\partial}{\partial z}(u\,\tau_{xz} + v\,\tau_{yz} + w\,\tau_{zz})\right]$$

Image Source: https://www.grc.nasa.gov/www/k-12/airplane/nseqs.html

CFD Simulation

Reynold’s Averaged Navier-Stokes (RANS)

- Reynolds-Averaged Navier Stokes Equations
  - The **RANS equations** are time-averaged equations of motion for fluid flow. They are primarily used while dealing with turbulent flows. These equations can be used with approximations based on knowledge of the properties of flow turbulence to give approximate averaged solutions to the Navier–Stokes equations.

$$\rho \frac{\partial \bar{u}_j \bar{u}_i}{\partial x_j} = \rho \bar{f}_i + \frac{\partial}{\partial x_j}\left[-\bar{p}\delta_{ij} + \mu\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right) - \rho\overline{u_i' u_j'}\right].$$

  - The final term is **Reynolds Stresses** which is evaluated with additional equations in our turbulence models.

CFD Simulation

Reynold’s Averaged Navier-Stokes (RANS)

- Reynold’s Averaging –

$$u_{mean} = \bar{u} = \lim_{T\to\infty} \frac{1}{T}\int_0^T u(x,y,z,t)dt$$

- Then we can model u as –

$$u = \bar{u} + u'$$

Where u' are the fluctuations with mean of zero $(\overline{u'} = 0)$

- For turbulent modelling, we can use the averaged values in the N-S Equations to finally obtain **Reynold’s Averaged Navier Stokes** equations, also called **RANS equations**.
- 3-D RANS Equations (Tensor Form) has 9 equations –

$$\boxed{\frac{\partial \overline{u_i' u_j'}}{\partial t} + \overline{u_k}\frac{\partial \overline{u_i' u_j'}}{\partial x_k} = -\overline{u_i u_k}\frac{\partial \overline{u_j}}{\partial x_k} - \overline{u_j u_k}\frac{\partial \overline{u_i}}{\partial x_k} + \overline{\frac{p'}{\rho}\left(\frac{\partial u_i'}{\partial x_j} + \frac{\partial u_j'}{\partial x_i}\right)} - \frac{\partial}{\partial x_k}\left(\overline{u_i' u_j' u_k'} + \frac{\overline{p' u_i'}}{\rho}\delta_{jk} + \frac{\overline{p' u_j'}}{\rho}\delta_{ik} - \nu\frac{\partial \overline{u_i' u_j'}}{\partial x_k}\right) - 2\nu\,\frac{\overline{\partial u_i' \partial u_j'}}{\partial^2 x_k}}$$

3-D RANS equations source: https://resources.system-analysis.cadence.com/blog/msa2021-the-reynolds-averaged-navier-stokes-rans-equations-and-models

CFD Simulation

Turbulence model equations

- Reynold‘s Stress model

The Reynolds stress model involves calculation of the individual Reynolds stresses $\rho\,\overline{u_i{}'u_j{}'}$ using differential transport equations. The individual Reynolds stresses are then used to obtain closure of the Reynolds-averaged momentum equation.

$$\underbrace{\frac{\partial}{\partial t}(\rho\,\overline{u_i'u_j'})}_{\text{Local Time Derivative}} + \underbrace{\frac{\partial}{\partial x_k}(\rho u_k\overline{u_i'u_j'})}_{C_{ij}\equiv\text{Convection}} = \underbrace{-\frac{\partial}{\partial x_k}\left[\rho\,\overline{u_i'u_j'u_k'}+\overline{p\left(\delta_{kj}u_i'+\delta_{ik}u_j'\right)}\right]}_{D_{T,ij}\equiv\text{Turbulent Diffusion}}$$

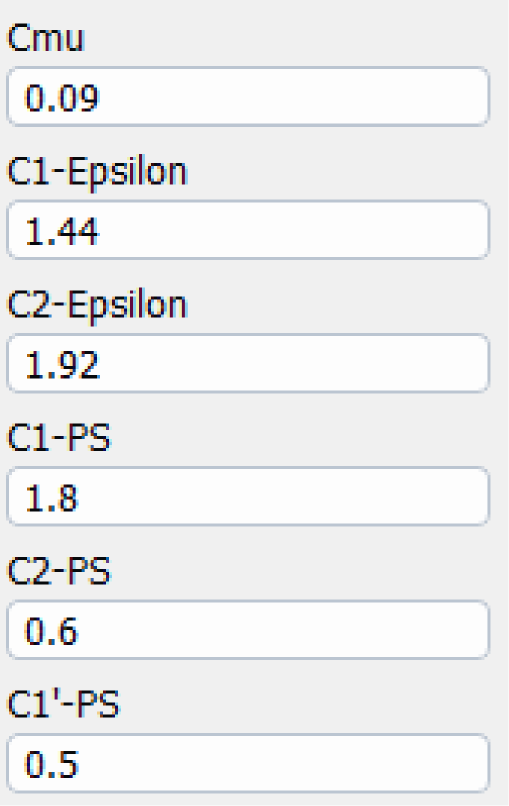
$$+\underbrace{\frac{\partial}{\partial x_k}\left[\mu\frac{\partial}{\partial x_k}(\overline{u_i'u_j'})\right]}_{D_{L,ij}\equiv\text{Molecular Diffusion}} \underbrace{-\rho\left(\overline{u_i'u_k'}\frac{\partial u_j}{\partial x_k}+\overline{u_j'u_k'}\frac{\partial u_i}{\partial x_k}\right)}_{P_{ij}\equiv\text{Stress Production}} \underbrace{-\rho\beta(g_i\overline{u_j'\theta}+g_j\overline{u_i'\theta})}_{G_{ij}\equiv\text{Buoyancy Production}}$$

$$+\underbrace{\overline{p\left(\frac{\partial u_i'}{\partial x_j}+\frac{\partial u_j'}{\partial x_i}\right)}}_{\phi_{ij}\equiv\text{Pressure Strain}} - \underbrace{2\mu\overline{\frac{\partial u_i'}{\partial x_k}\frac{\partial u_j'}{\partial x_k}}}_{\epsilon_{ij}\equiv\text{Dissipation}} \underbrace{-2\rho\Omega_k\left(\overline{u_j'u_m'}\epsilon_{ikm}+\overline{u_i'u_m'}\epsilon_{jkm}\right)}_{F_{ij}\equiv\text{Production by System Rotation}} + \underbrace{S_{\text{user}}}_{\text{User-Defined Source Term}}$$

- https://www.afs.enea.it/project/neptunius/docs/fluent/html/th/node77.htm

CFD Simulation

Reynold's Stress model parameters



Cmu
0.09

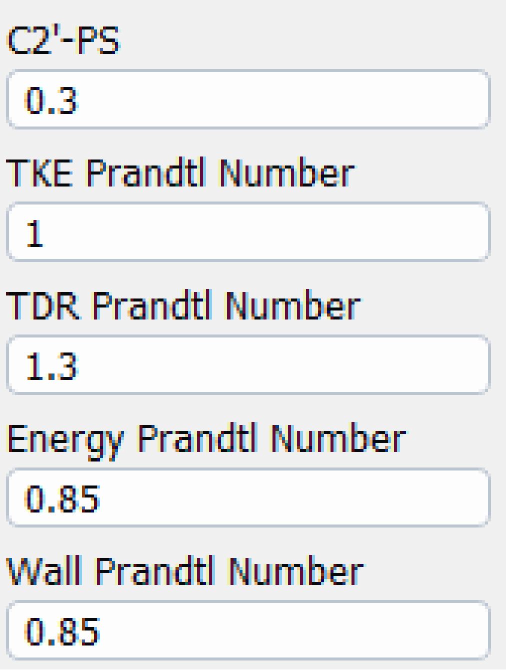
C1-Epsilon
1.44

C2-Epsilon
1.92

C1-PS
1.8

C2-PS
0.6

C1'-PS
0.5



C2'-PS
0.3

TKE Prandtl Number
1

TDR Prandtl Number
1.3

Energy Prandtl Number
0.85

Wall Prandtl Number
0.85

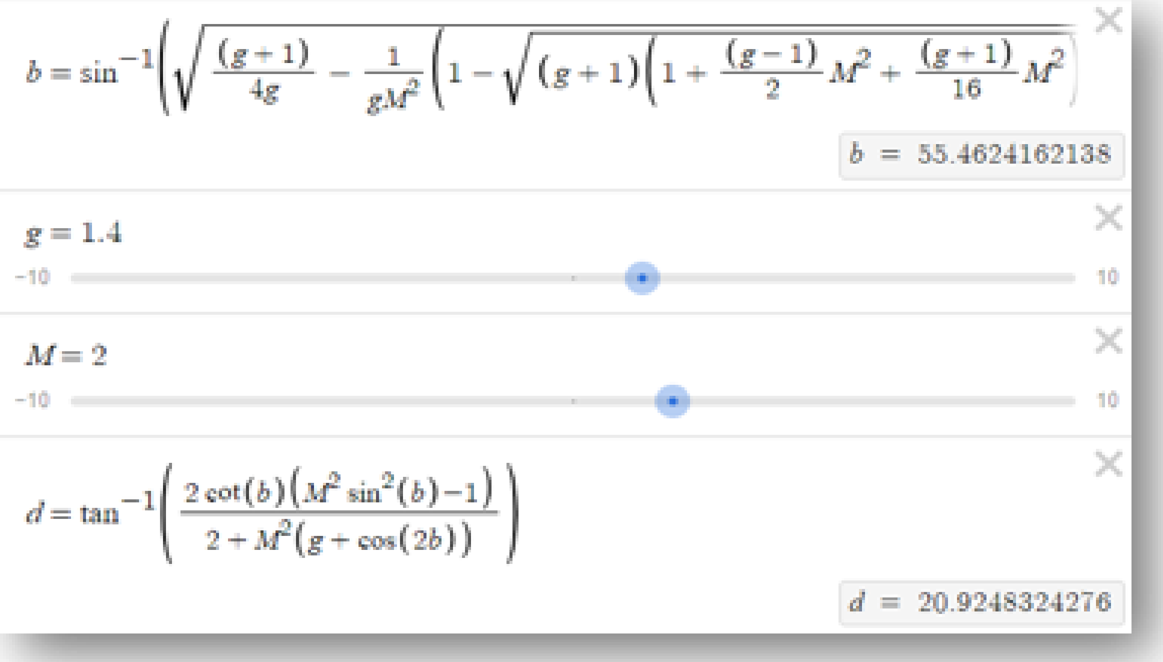
CFD Simulation

Test Cases

- We choose to vary the **half angle (HA)** and the **flow angle of attack (AoA)** for our training and test data, with Mach no. = 2

- We choose them such that:

HA + AoA = Total Deflection ≤ 21°



$$b = \sin^{-1}\left(\sqrt{\frac{(g+1)}{4g} - \frac{1}{gM^2}\left(1 - \sqrt{(g+1)\left(1 + \frac{(g-1)}{2}M^2 + \frac{(g+1)}{16}M^2\right.}\right.}\right.$$

$b = 55.4624162138$

$g = 1.4$

$M = 2$

$$d = \tan^{-1}\left(\frac{2\cot(b)\left(M^2\sin^2(b)-1\right)}{2+M^2\left(g+\cos(2b)\right)}\right)$$

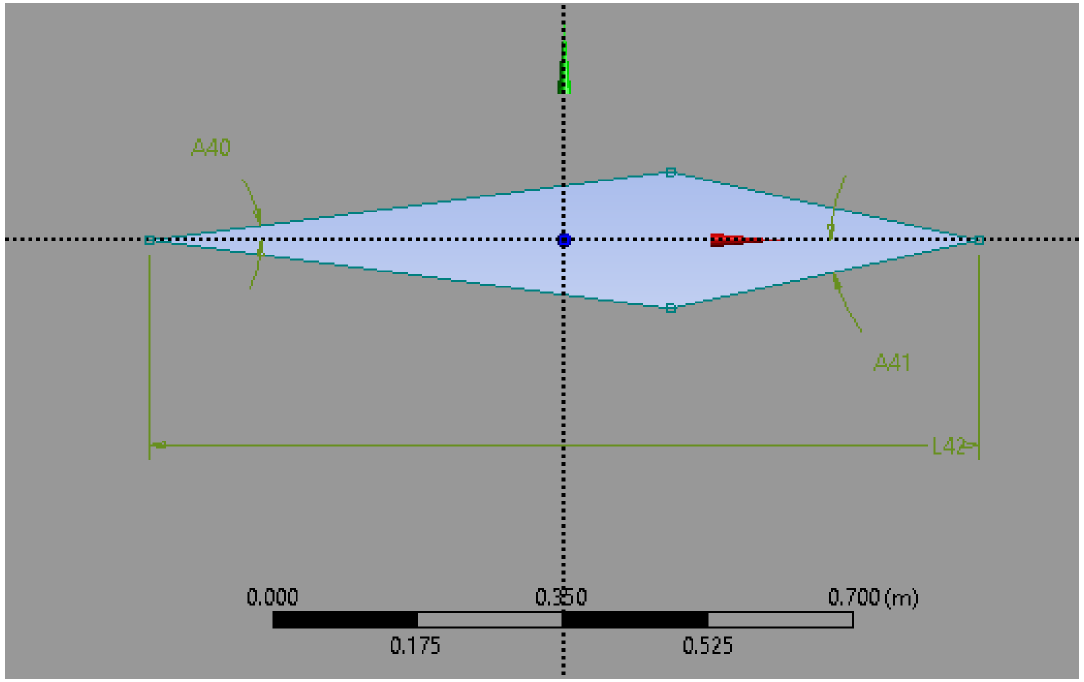
$d = 20.9248324276$

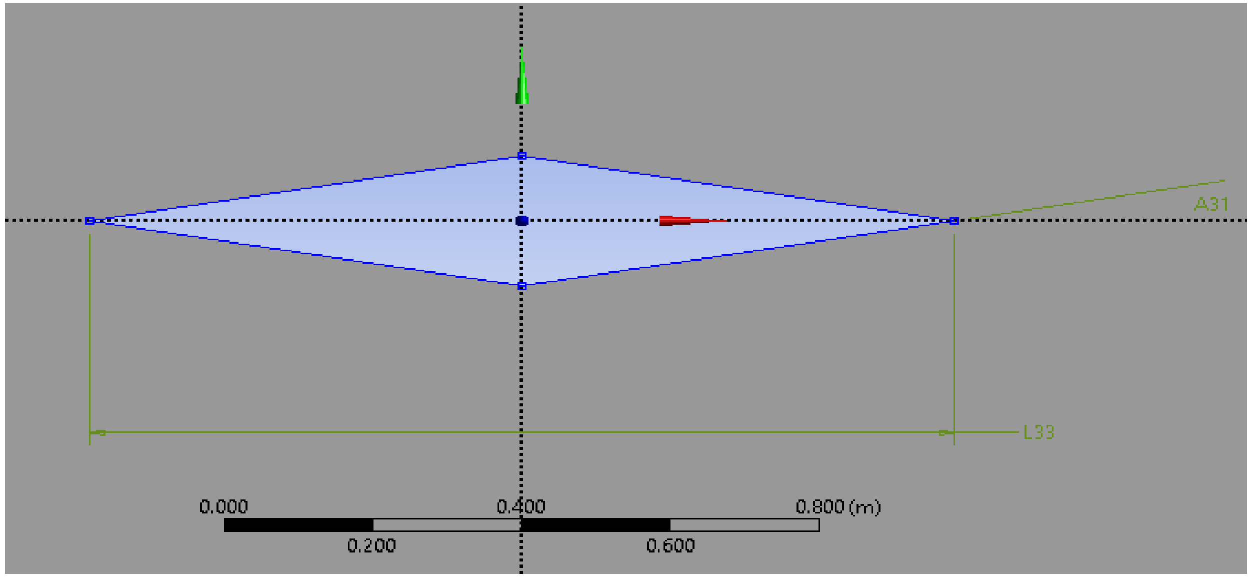
CFD Simulation

Test Cases

We have started with 5 different airfoils

- 3 symmetric with half angle 5, 7.5, 10 deg
- 2 asymmetric with leading edge half angle 7.5, 10 deg & trailing edge half angle being 5 deg more than it

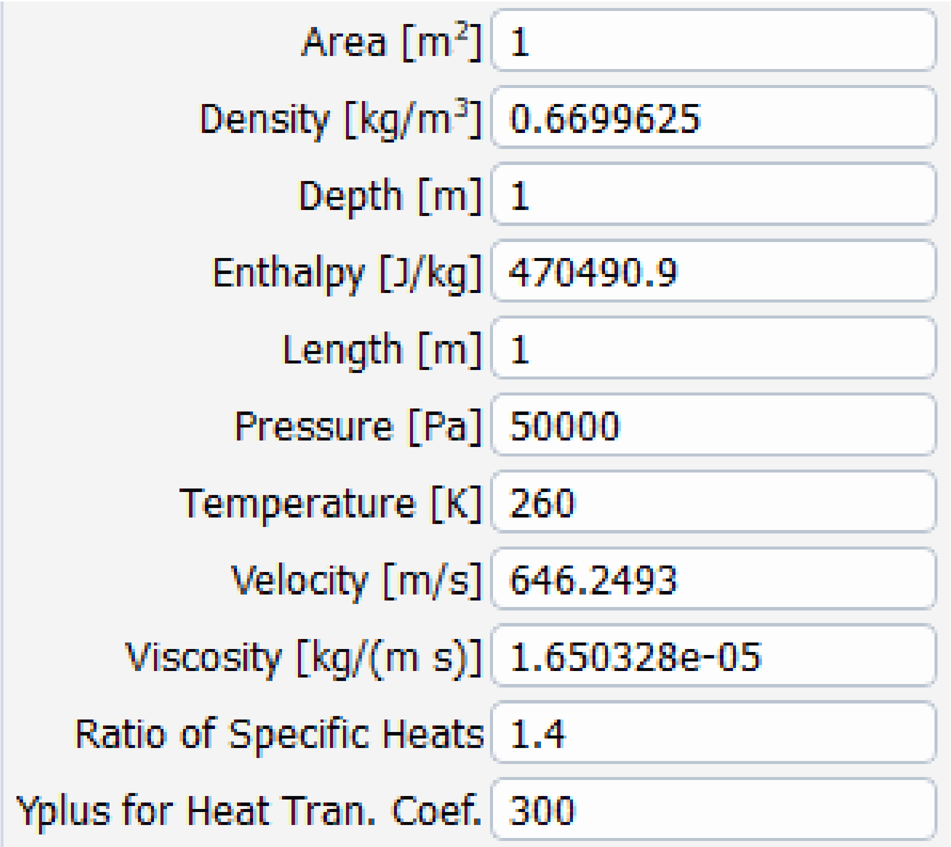




CFD Simulation

Boundary conditions

- Airfoil – No slip wall BC
  - $u = v = 0$
  - $u' = v' = 0$
- Far field
  - Mach no = 2
  - Gauge pressure = 0 Pa
- Reference values

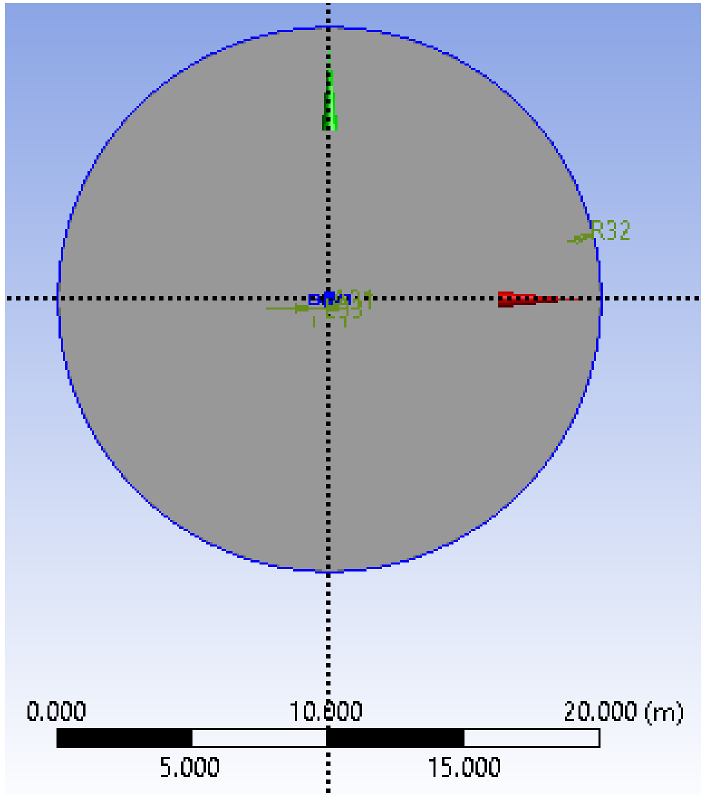


Area [m^2]	1
Density [kg/m^3]	0.6699625
Depth [m]	1
Enthalpy [J/kg]	470490.9
Length [m]	1
Pressure [Pa]	50000
Temperature [K]	260
Velocity [m/s]	646.2493
Viscosity [kg/(m s)]	1.650328e-05
Ratio of Specific Heats	1.4
Yplus for Heat Tran. Coef.	300

CFD Simulation

Flow domain

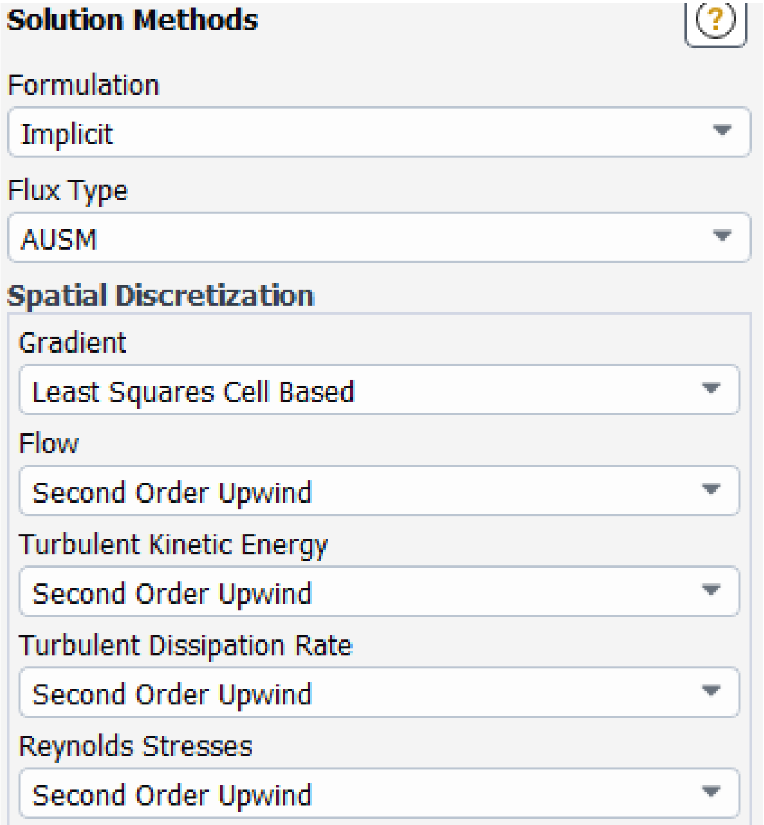
- Flow domain – Circle of radius 20x airfoil chord (20m in our case)



CFD Simulation

Numerical Method

- Implicit scheme gives us freedom for Δt, as this method is stable for all time steps.
- The inviscid flux vector is evaluated by a standard upwind, flux-difference splitting. This approach acknowledges that the flux vector contains characteristic information propagating through the domain with speed and direction according to the eigenvalues of the system. The AUSM scheme has several desirable properties:
  - Provides exact resolution of contact and shock discontinuities
  - Preserves positivity of scalar quantities
  - Free of oscillations at stationary and moving shocks



Solution Methods

Formulation: Implicit

Flux Type: AUSM

Spatial Discretization

Gradient: Least Squares Cell Based

Flow: Second Order Upwind

Turbulent Kinetic Energy: Second Order Upwind

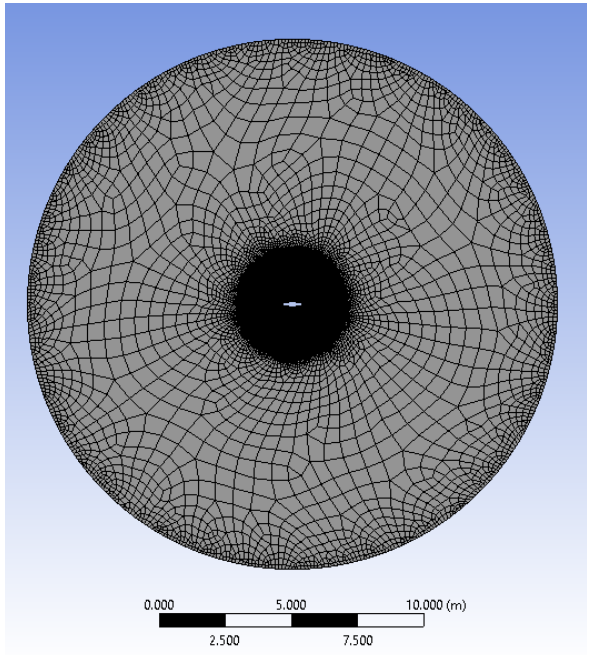
Turbulent Dissipation Rate: Second Order Upwind

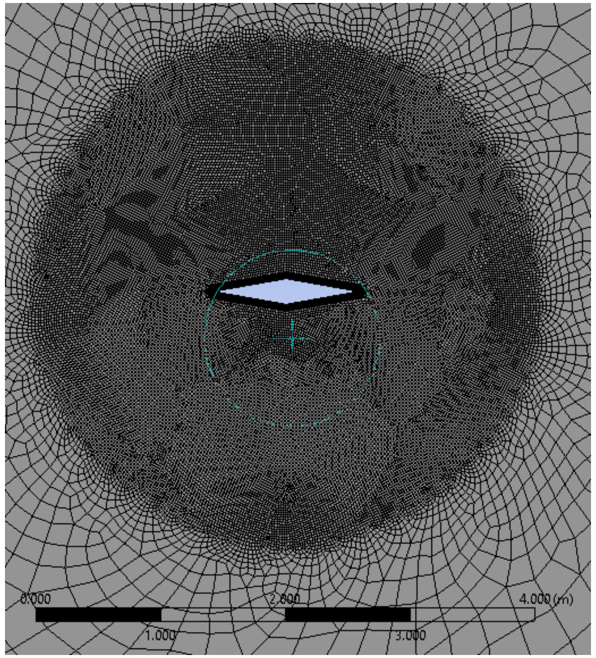
Reynolds Stresses: Second Order Upwind

1. https://www.afs.enea.it/project/neptunius/docs/fluent/html/th/node379.htm
2. Phongthanapanich, Sutthisak & Takayama, Kazuyoshi. (2019). A Comparison of the Roe's FDS, HLLC, AUFS, and AUSMDV+ Schemes on Triangular Grids. Applied Science and Engineering Progress. 12. 10.14416/j.asep.2019.07.003.

CFD Simulation

Computational grid

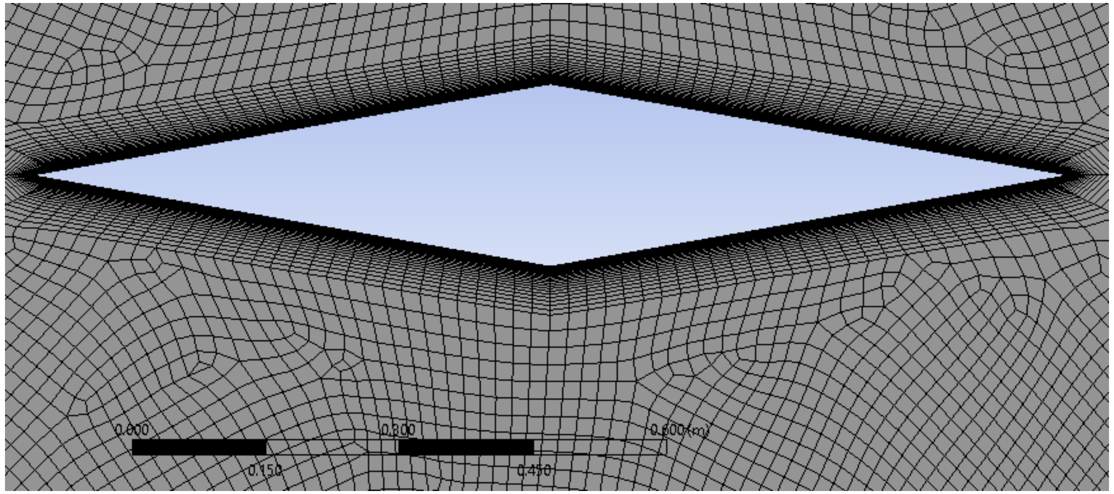


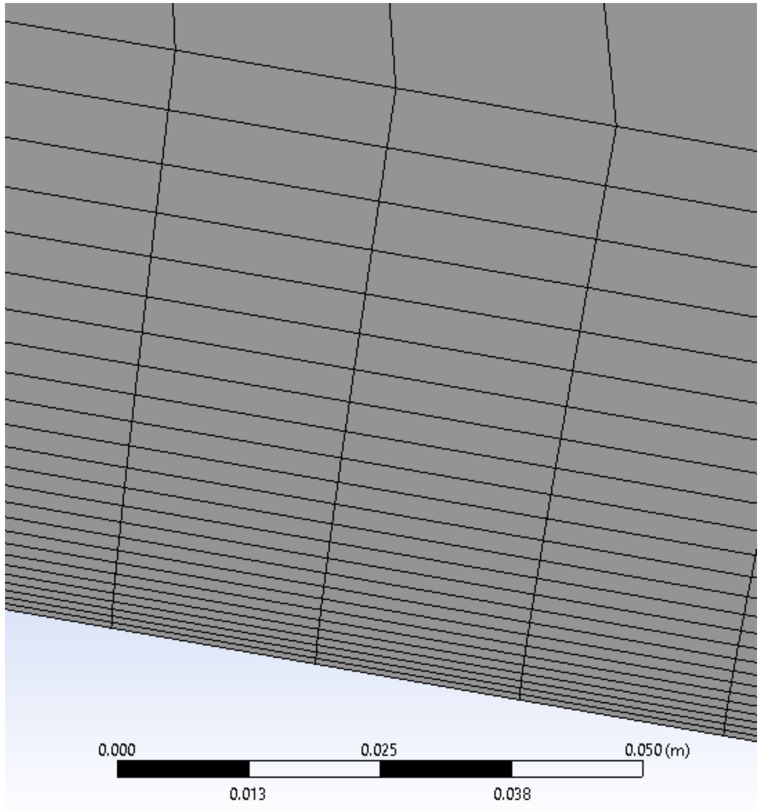


- A O-grid mesh is used for the crop zone to capture the flow near to the airfoil surface with comparatively lesser number of elements in total.
- Outside that, we use unstructured meshing with good orthogonality to capture the aggregate flow without increasing computational load.

CFD Simulation

Computational grid





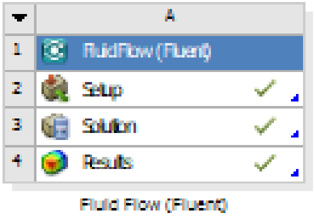
- A inflation zone of 25 layers with growth ratio 1.1 is added in the crop zone on the boundary of the airfoil to capture gradient effects which are predominant in viscous flow such as wall stress.

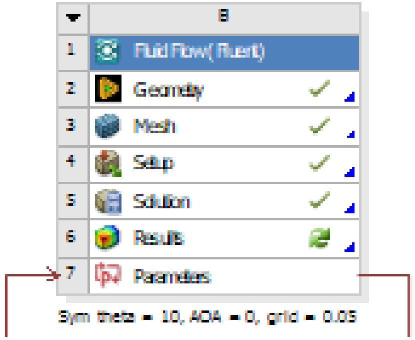
CFD Simulation

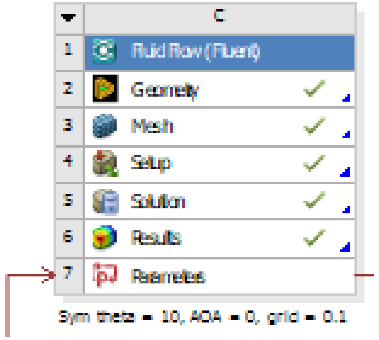
Grid convergence

- After varying grid sizes, from very coarse to a point where we attain grid convergence at size = 2cm.

	Grid A	Grid B	Grid C (final)
O- grid element size	10 cm	5 cm	2 cm
Nodes	15510	30556	117082
Elements	5014	10020	38860
Y+ (wall function)	241.963	114.7599	41.67461
Cd (parameter used for grid convergence)	0.0**53**428269	0.044051888	0.043782653







https://forum.ansys.com/forums/topic/checking-y-value/

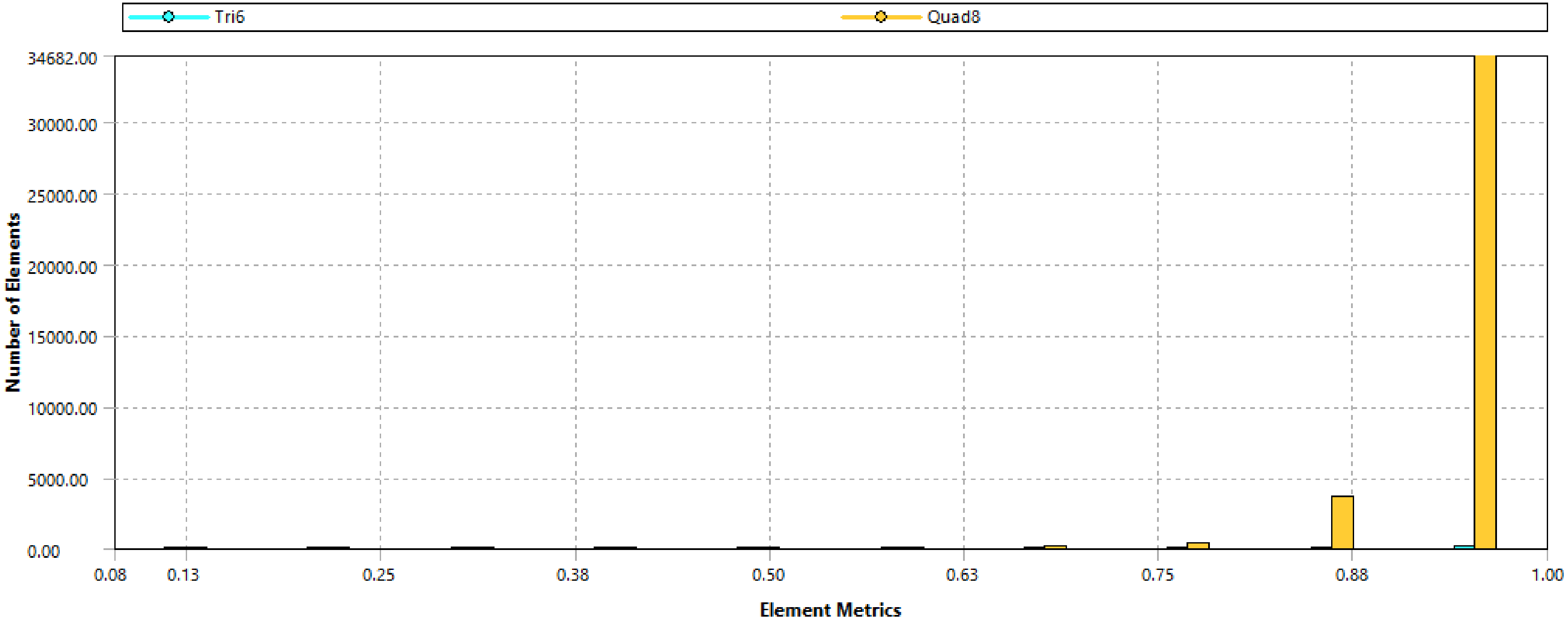
CFD Simulation

Computational grid

- Mesh statistics

Nodes	117082
Elements	38860

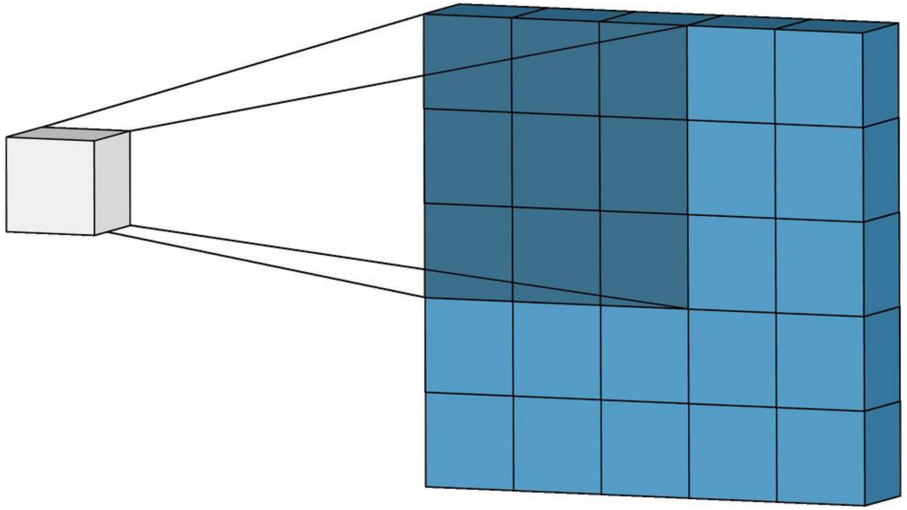
- Orthogonality



Deep Learning

Architecture Details

Convolutional Neural Network (CNNs)



<u>General 1D convolution</u>

$$(f * h)(t) = \int_{-\infty}^{\infty} f(\tau)\, h(t-\tau)\, d\tau$$

<u>2D convolution</u>

$$(I * K)(x, y) = \iint_{-\infty}^{\infty} K(\epsilon, \tau)\, I(x-\epsilon, y-\tau)\, d\epsilon\, d\tau$$

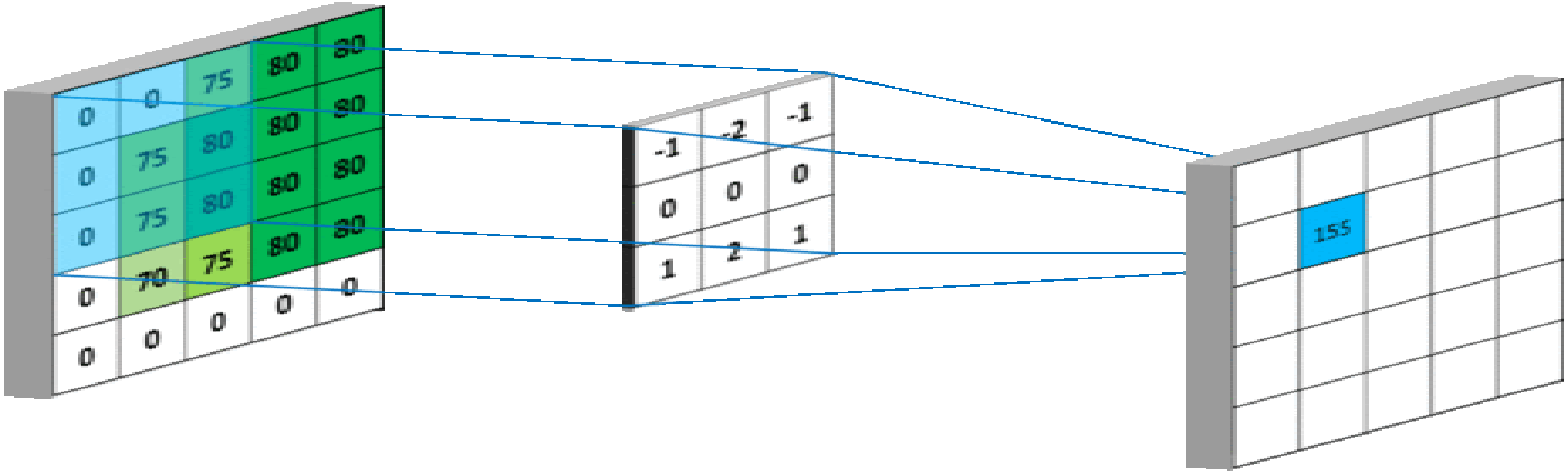
Where -

1. I = Image
2. K = Kernel

Image Source – https://miro.medium.com/max/1400/1*Fw-ehcNBR9byHtho-Rxbtw.gif

Architecture Details

Convolutional Neural Network (CNNs)



0	0	75	80	80
0	75	80	80	80
0	75	80	80	80
0	70	75	80	80
0	0	0	0	0

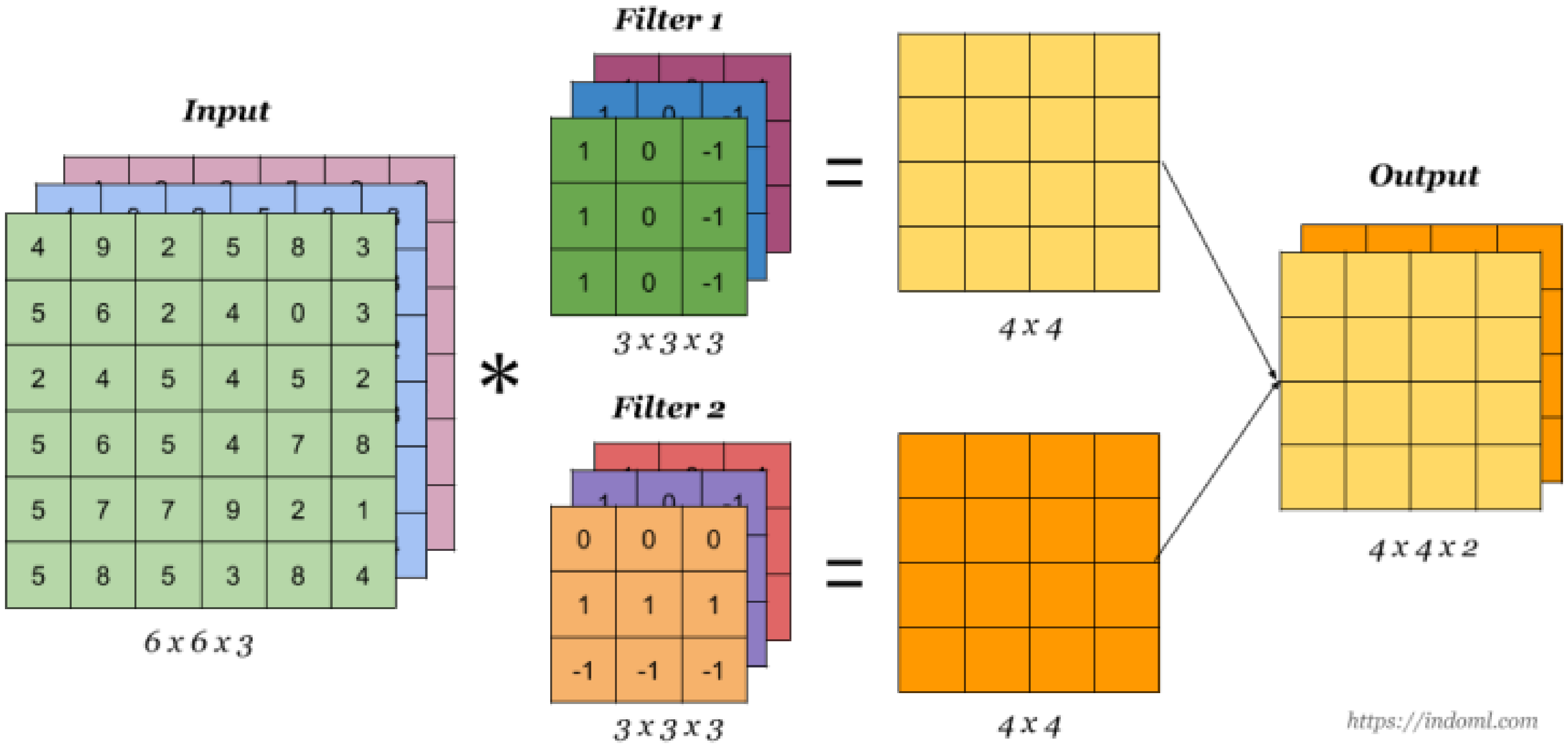
-1	-2	-1
0	0	0
1	2	1

Image Source – https://149695847.v2.pressablecdn.com/wp-content/uploads/2018/01/conv-full-layer.gif

Deep Learning

Architecture Details

Convolutional Neural Network (CNNs)



Input

4	9	2	5	8	3
5	6	2	4	0	3
2	4	5	4	5	2
5	6	5	4	7	8
5	7	7	9	2	1
5	8	5	3	8	4

Filter 1

1	0	-1
1	0	-1
1	0	-1

Filter 2

0	0	0
1	1	1
-1	-1	-1

Main points –

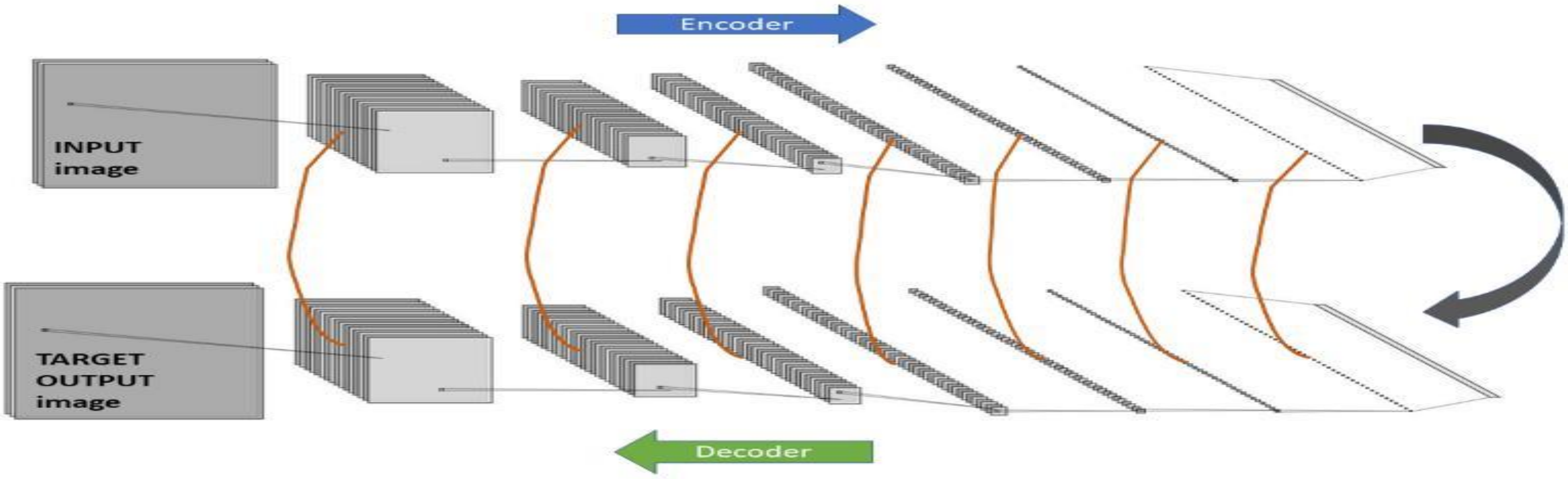
1. Filter depth = Image depth
2. Filter weights are trainable

Image Source – https://indoml.files.wordpress.com/2018/03/convolution-with-multiple-filters2.png?w=736

Deep Learning

Architecture Details

U-Net



Trainable Parameters = 363,435,907

Image Source – Foilnet Paper

Architecture Details

Dense Blocks

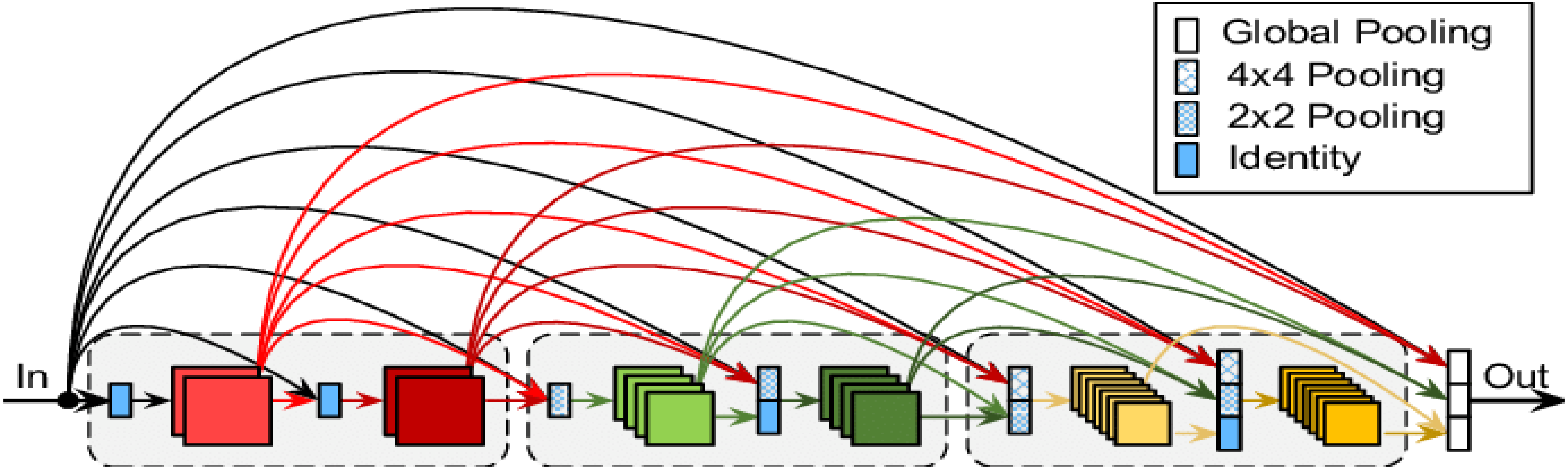


Image Source – Foilnet Paper

Deep Learning

Architecture Details

Dense U-Net

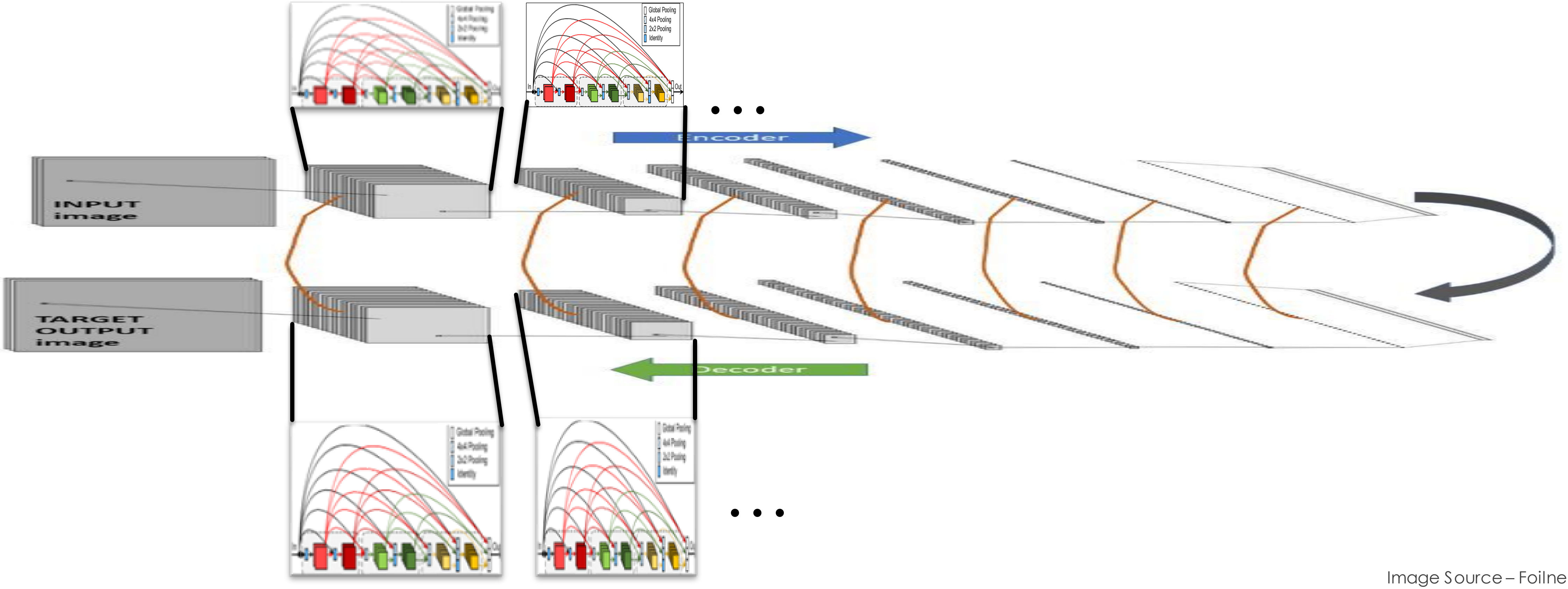
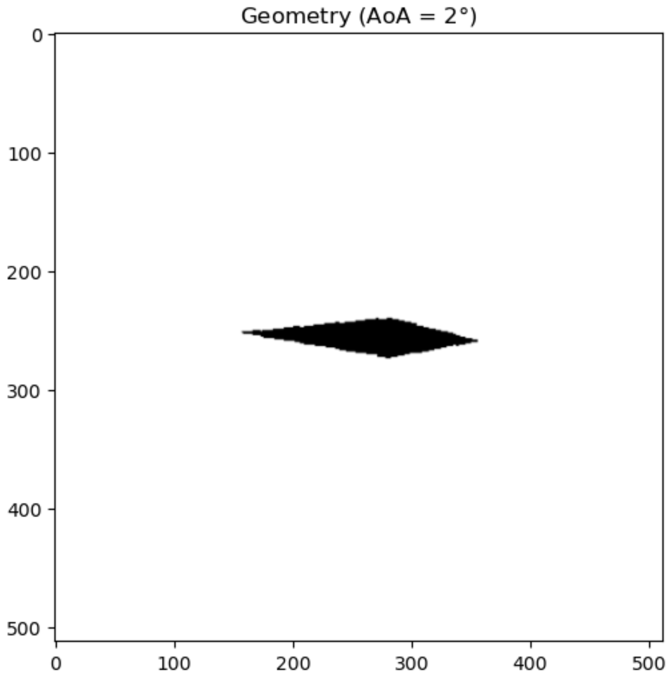
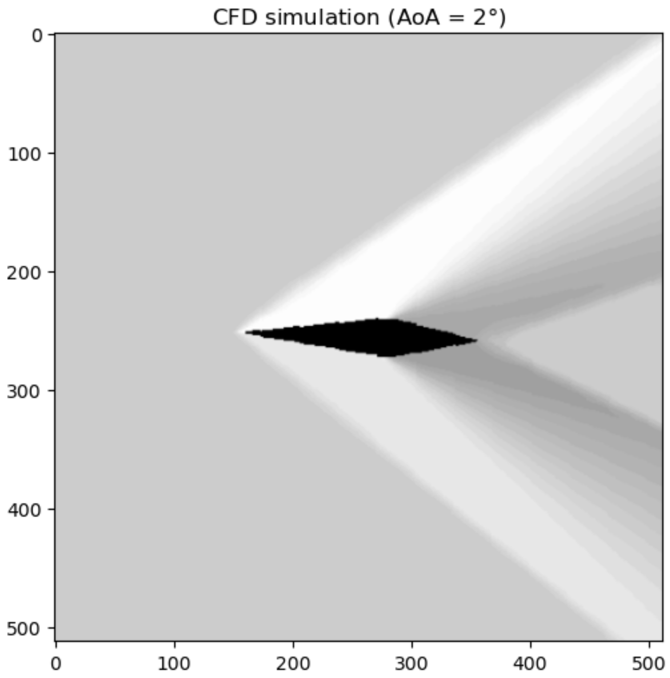


Image Source – Foilnet Paper

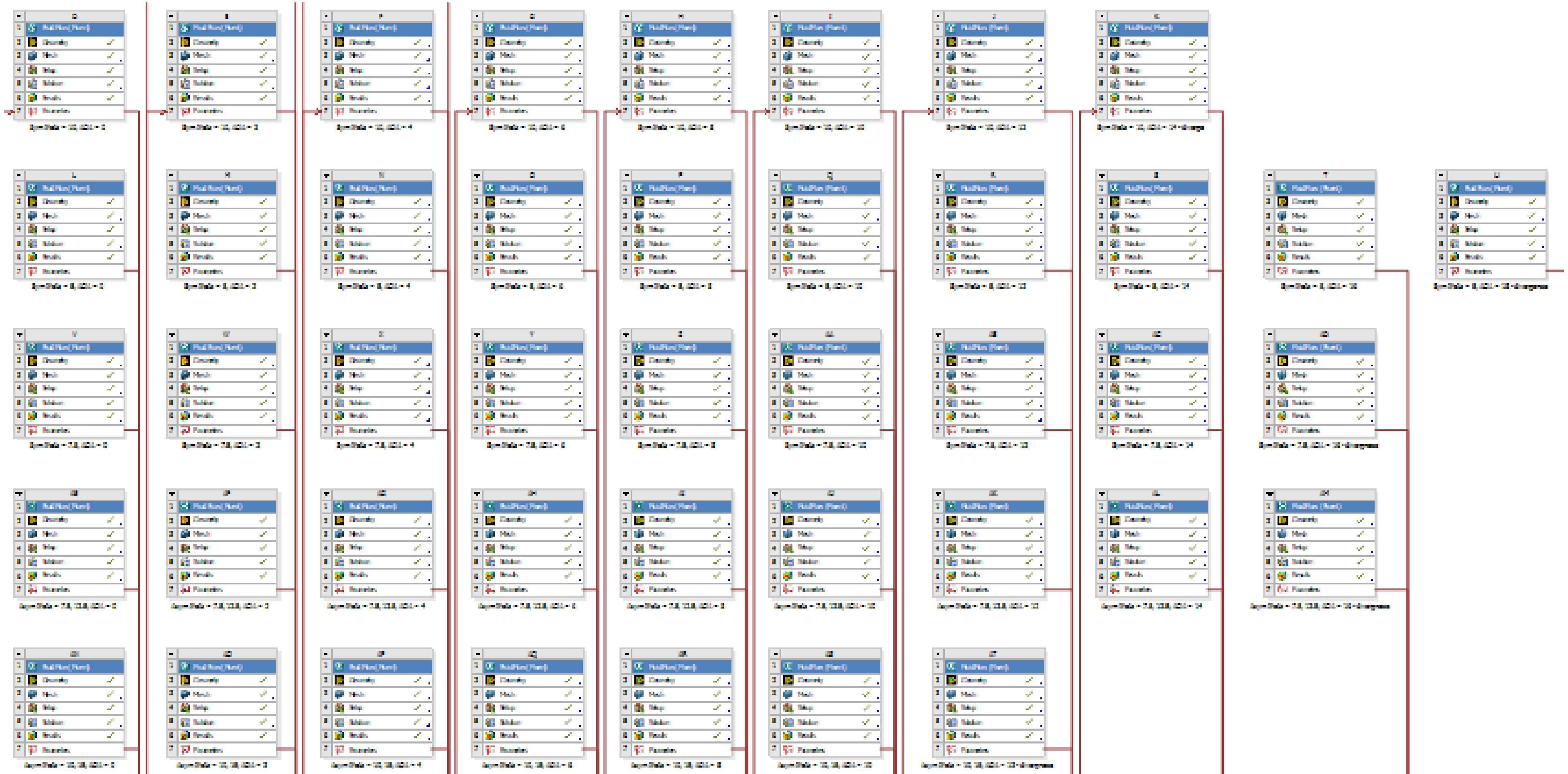
Data Preparation

- After the circular crop, data pre processing is done to obtain the images on the right as square images to help train our model
- Left image – Geometry of airfoil
- Right image – Pressure contour at cropzone
- We have set a universal range of cP to vary from **–1 to 2 in all test cases**, to ensure uniformity with our data.
- Greyscale conversion is done to reduce the image size from 3 layers (RGB) to 1 layer, hence reducing the training time by 3 times.


Geometry (AoA = 2°)


CFD simulation (AoA = 2°)

Data Preparation



Data Augmentation

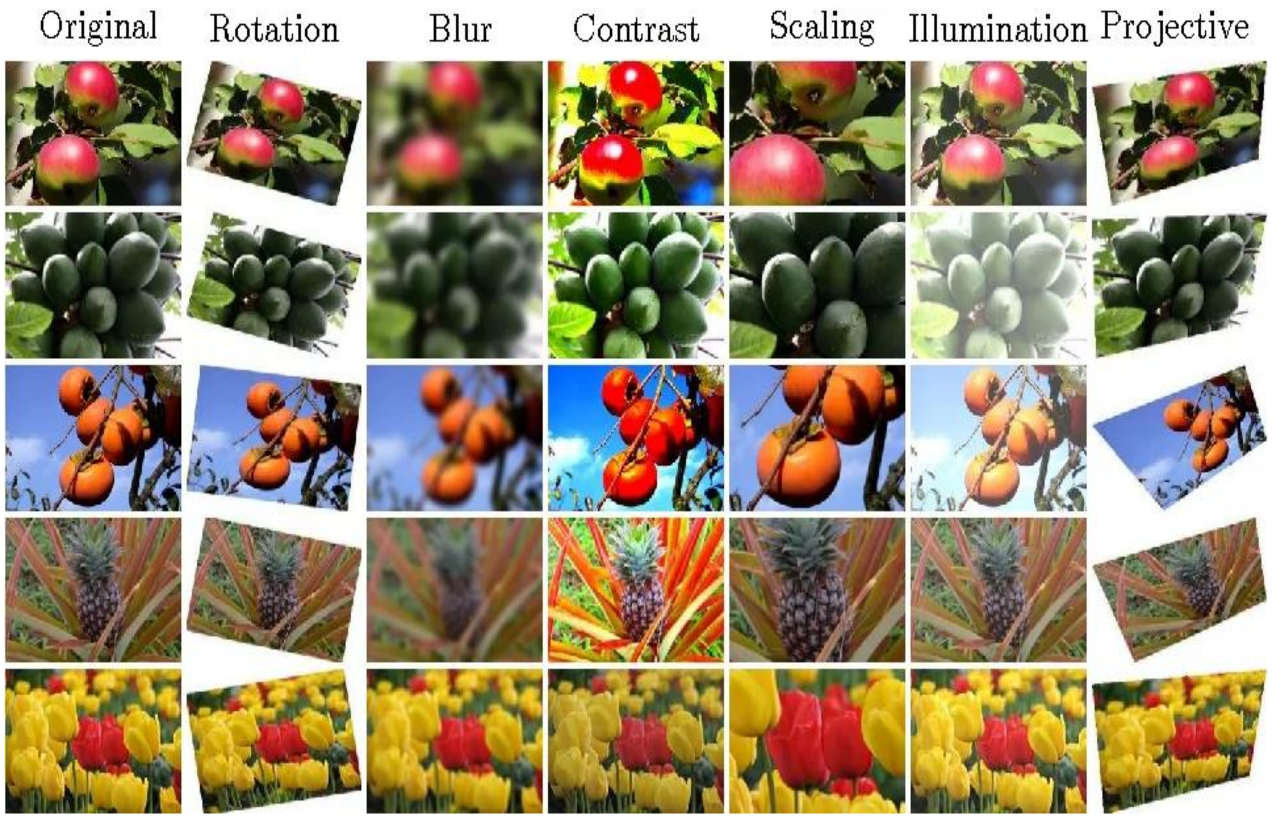
Data Augmentation includes techniques to –

- Artificially increase the amount of data
- Generating new data points from existing data
- Includes making small changes to data

Data Augmentation

For data augmentation, making simple alterations on visual data is popular. Classic image processing activities are:

- Padding
- **Rotating**
- **Re-scaling**
- **Vertical flipping**
- Horizontal flipping
- Translation (along X, Y direction)
- **Cropping/zooming**
- Adding noise
- Random erasing
- Darkening & brightening/color modification,
- **Gray-scaling**



https://research.aimultiple.com/data-augmentation/

Data Augmentation

Benefits of data augmentation include:

- Improving model prediction accuracy
  - Adding more training data
  - Preventing data scarcity
  - Reducing data overfitting
  - Creating variability in data
  - Increasing generalization ability (Better test scores)
  - Helping resolve class imbalance issues in classification
- Reducing costs of collecting and labeling more data
- Enables rare event prediction

https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/

https://research.aimultiple.com/data-augmentation/

Loss Function

Loss Function

- Pixel-wise Mean Square Error (MSE) Loss

$$MSE = \frac{1}{b(n_x - 1)(n_y - 1)} \sum_{m=1}^{b} \left(\sum_{i=1}^{n_x - 1} \sum_{j=1}^{n_y - 1} \left(y_{ij}^{b} - \hat{y}_{ij}^{b}\right)^2 \right)$$

Where –

- b = Batch size
- n_x = No. of pixels in *horizontal* direction (Width of image)
- n_y = No. of pixels in *vertical* direction (Height of image)
- y_{ij}^{b} = True value of pixel at (i, j) of batch *b*
- $\hat{y}_{ij}^{b}$ = Predicted value of pixel at (i, j) of batch *b*

Optimizer

<u>Stochastic Gradient Descent</u>

- *Optimization –*

$$v_{t,d\theta} = \beta v_{t-1,d\theta} + \frac{d(\mathrm{Loss})}{d\theta}$$

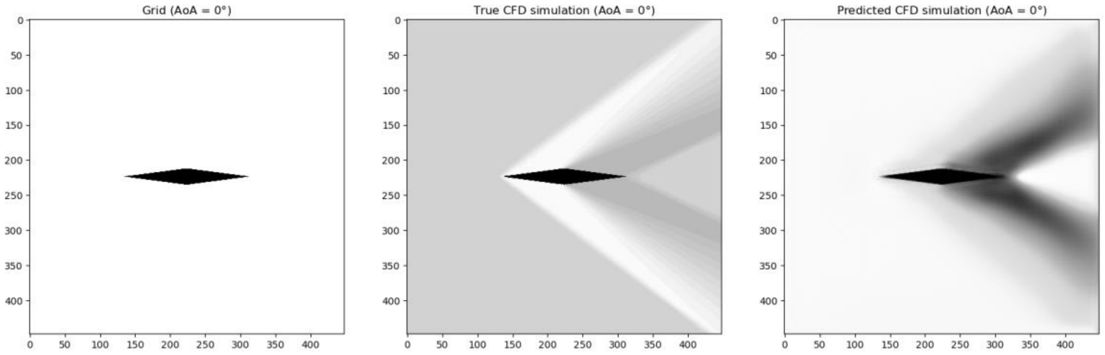
$$\theta_t = \theta_{t-1} + \alpha v_{d\theta}$$

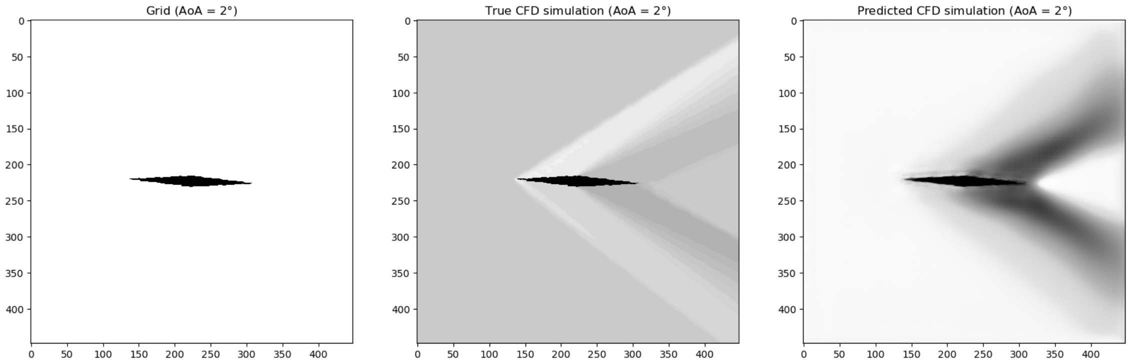
Where –

- **θ** = Parameters
- $\boldsymbol{\alpha}$ = Learning Rate = **0.1**
- $\boldsymbol{\beta}$ = Momentum = **0.9**

Results

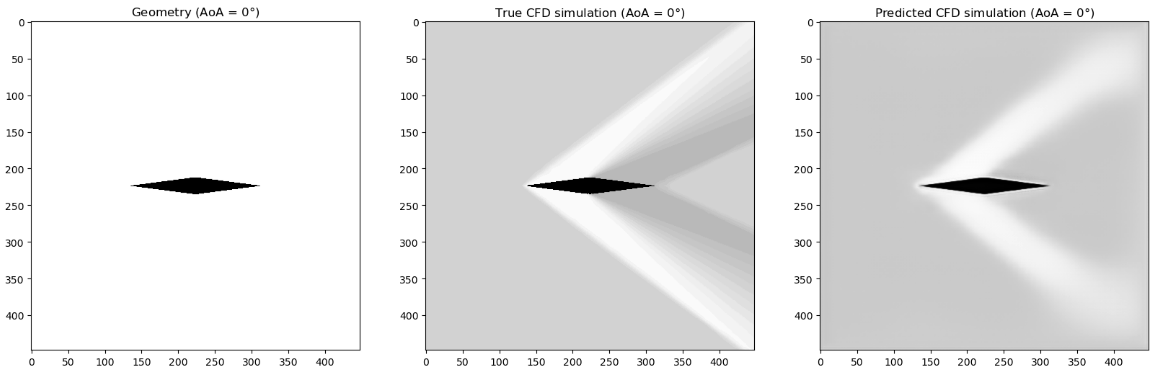
- Training Data Prediction
- Test Data Prediction





Results

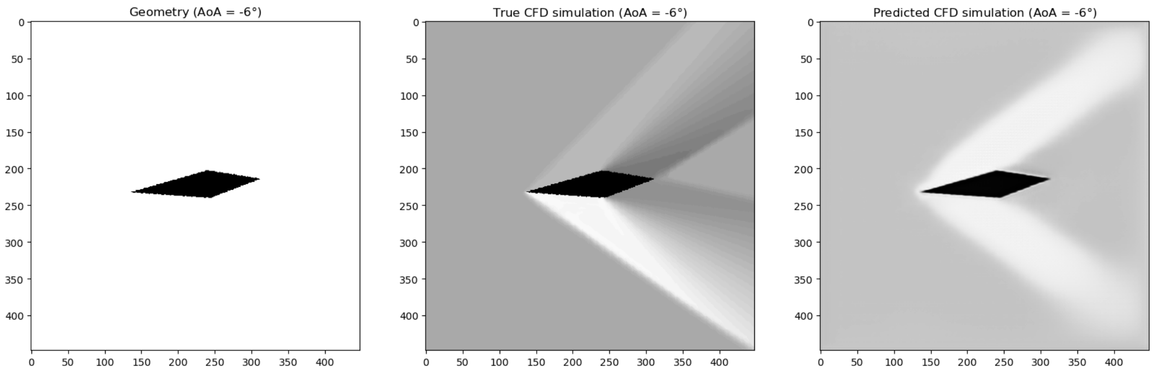
- Training Data Prediction



Pixel Loss = 0.3428

Prediction time = 1.8s

- Test Data Prediction



Pixel Loss = 0.2547

Prediction time = 1.7s

Similarity Parameters

- Reynolds Number

$$Re = \frac{\mu \rho v}{L}$$

- Pressure Coefficient

$$C_p = \frac{p - p_\infty}{0.5 \times \rho v^2} = \frac{2}{\gamma M^2}\left(\frac{p}{p_\infty} - 1\right)$$

- Mach Number

$$M = \frac{v}{\sqrt{\gamma R T}}$$

Future Scope of Improvement

- Check with Physics Informed Neural Networks (PINNs)
- Have more data
- Extend to more Mach numbers

Thank you!