Discontinuity Identification in Numerical solutions of Differential equations

Professor Harish Kumar

Department **Mathematics and Computing**
Entry Number **2018MT60798**
Author **Zuhaib Ul Zamann**

January 8, 2023

Contents

Introduction

Solving non-linear hyperbolic conservation laws often leads to solutions having discontinuities even if we start with smooth initial data.
High order numerical schemes may generate spurious oscillations and even converge to an entropy violating solution if the solution near discontinuities is not appropriately treated.
The cells near the discontinuities are referred to as troubled cells, and methods to capture these cells are required. Techniques used to capture the troubled cells are known as troubled cell indicators.

Problem Description

Consider a function $f : D \to \mathbb{R}$ where $D \subset \mathbb{R}^d, d \geq 1$ is a compact set.
Without loss of generality let D is a $d-$ dimensional rectangle given by $D = I_1 \times I_2 \times \ldots \times I_d$ where $I_i = [a_i, b_i], b_i = a_i + n_i\delta$ where $n_i \in \mathbb{N}, \delta > 0$.
Consider a uniform grid over D of $\displaystyle\prod_{i=1}^{d}(n_i + 1)$ grid points

$$S = \{(a_1 + i_1\delta, a_2 + i_2\delta, \ldots, a_d + i_d\delta), i_k \in \{0, 1, 2, \ldots, n_k\}\forall k = 1, 2, \ldots, d\}$$

Given the value of f at each of the points in S, the problem is to find the grid cells which contain points of discontinuity of f. These cells are known as *troubled cells*

Data Generation

To develop machine learning models, we need data on which the model is trained. We discuss the data generation processess for training purposes

Exact Data Generation

This section consists of the process in which we generate data of piecewise continuous functions whose points of discontinuity are known and the functional values are exact

1D Exact data generation

The CNN detector is trained on a synthetic dataset of $n = 1,000,00$ piecewise smooth function on Domain D.

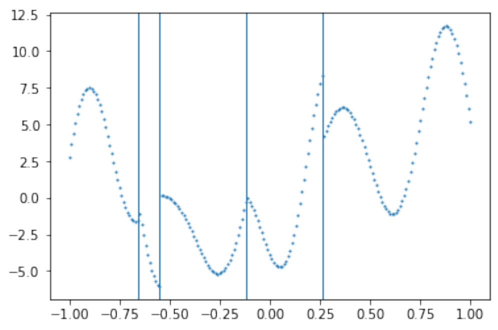
1. Randomly select an integer N_d from the set $\{0, 1, \ldots, M\}$. In the experiments of the paper $M = 3$(i.e. at most 3 discontinuities).

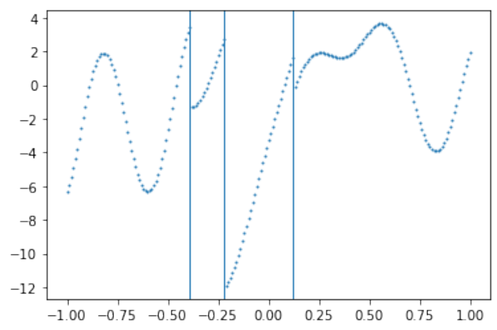
2. Using uniform distribution on D, generate N_d random numbers as the locations of the discontinuities. This partitions D into $N_d + 1$ subdomains.

3. Inside each subdomain create fourier series

$$\tilde{a}_0 + \sum_{n=1}^{N_F} \left(\tilde{a}_n \cos nx + \tilde{b}_n \sin nx\right)$$

, where $\tilde{a}_n, b_n \sim N(0, 1)$ are i.i.d Gaussian random variables and $N_F = 15$





2D Exact data generation

1. Domain D is divided into two sub-regions by a *random curve*

2. Inside each sub-region a smooth function is generated given by

$$f_i(x,y) = \sum_{m+n\leq N_p} a^{(i)}_{m,n} P_m(x) P_n(y) \; i = 1,2$$

   where $P_n's$ are the standard Legendre polynomials and $a_{m,n}$ are randomly sampled from $N(0,10)$, $N_p = 4$

3. For the random curve serving as an interface between the two sub-regions and is also the location of the discontinuity curve, following two cases are employed.

   - Line Cut: Defined by random straight line

$$\cos\theta(x-x_0) + \sin\theta(y-y_0) = 0$$

     where $\theta \sim U(0,2\pi), (x_0,y_0) \sim U(D)$

   - Circular Cut: Defined as

$$(x-x_0)^2 + (y-y_0)^2 = r$$

     where $r \sim U(0,3), (x_0,y_0) \sim U(D)$

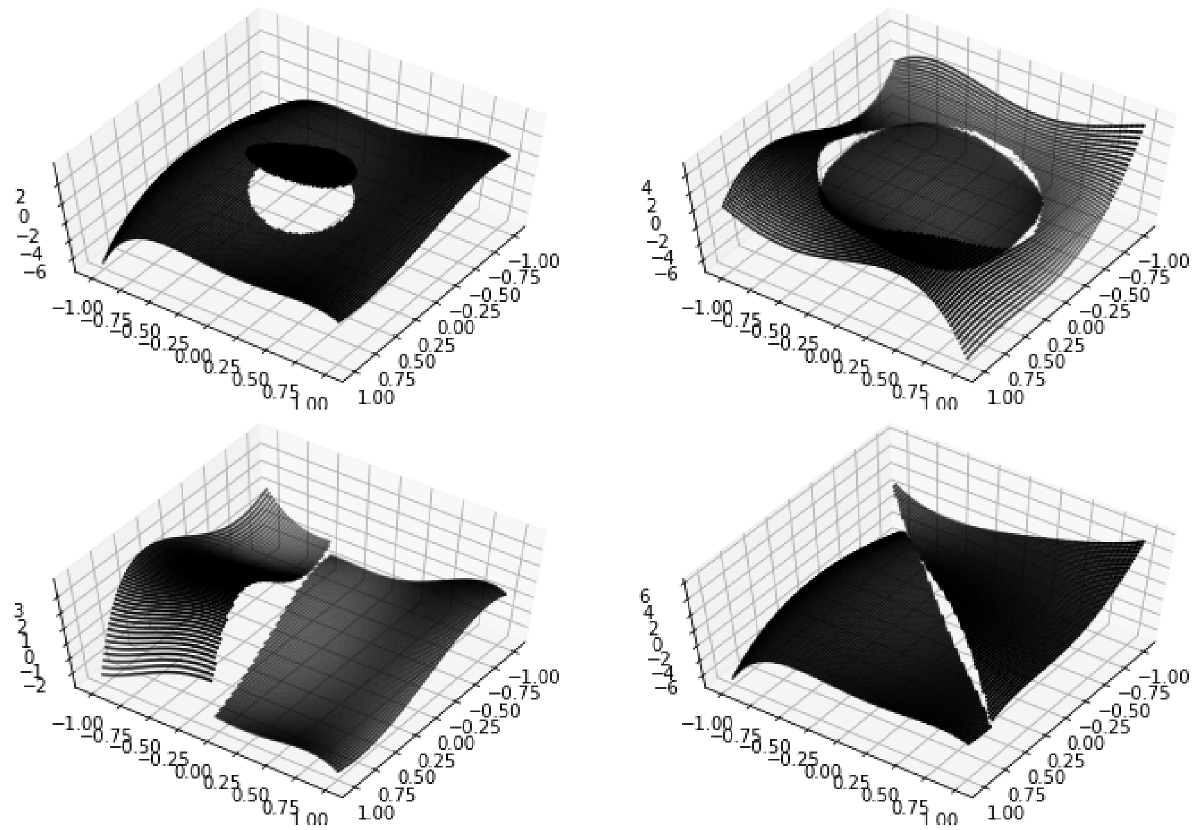


Figure 2: Examples of 2D-generated data

Numerical Data Generation

1D-Numerical Data Generation

We consider the advection differential equation $\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0$ with initial condition as
$u_0 = u(x, 0)$
It is well know that for constant $a > 0$, we have the solution is given by

$$u(x, t) = u_0(x - at) \tag{1}$$

We use this result to generate the data for our training

1. Divide the domain into M sub intervals and create function

   $$\tilde{a}_0 + \sum_{n=1}^{N_F}(\tilde{a}_n \cos nx + \tilde{b}_n \sin nx)$$

   as described in the above data generation process for 1D in each of the sub-intervals. This will form the piece-wise definition of the initial condition $u_0(x)$

2. Using a numerical method solve equation (1) with appropriate limiting, e.g min-mod limiting. Choose $a \sim N(0, 1)$ and solve for time $t = T$ where $T \sim N(0, 1)$. Call the obtained solution as $\tilde{u}(x, t)$

3. Label a cell as troubled cell if $u_0(x - ct)$ is discontinuous within the cell and hence obtain the label vector y.

4. The generated data point is $(\tilde{u}(x, t), y)$

CNN Discontinuity Detector

1-D Detector

Consider $D = [a, b]$ and $S = \{x_i = a + (i-1)\delta, i = 1, 2, \ldots, N+1\}$
This forms a uniform grid with N cells denoted by intervals $[x_i, x_{i+1}), i = 1, 2, \ldots, N$.
Let

$$y = (y_1, y_2, \ldots, y_n) \in \{0, 1\}^N$$

be a binary vector indicating the ground truth values for each $i = 1, 2, \ldots, N$ where $y_i = 1$ indicates that the i^{th} cell is a trouble cell and $y_i = 0$ otherwise.
Let $v_f = \{f(x) : x \in S\}$ be the set of observed function values on S.
We first standardize the observed function values and the feed them into a CNN to obtain an output vector of N real values.

$$\hat{y} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_N) = \mathcal{N}(\tilde{v}_f)$$

as an estimate of the ground truth y.
Here

$$\tilde{v}_f = \{\frac{f(x) - \mu_f}{\sigma_f} : x \in S\}$$

where μ_f, σ_f are the mean and standard-deviation of v_f respectively.
Then for a chosen threshold t, detector labels each of the i^{th} cell a trouble cell if $\hat{y}_i > t$[5]

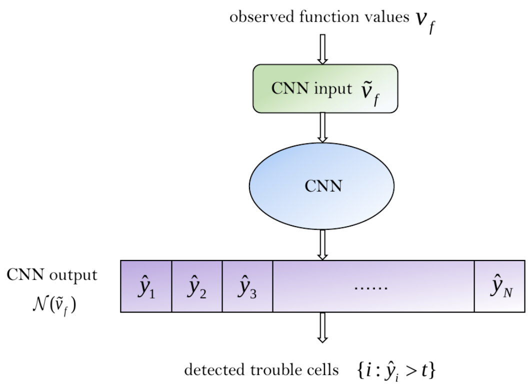

observed function values v_f
CNN input $\tilde{v}_f$
CNN
CNN output
$\mathcal{N}(\tilde{v}_f)$
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
......
$\hat{y}_N$
detected trouble cells $\{i : \hat{y}_i > t\}$

Figure 3: 1D - One Level Detector

2D Detector

Let $D = [a, b]^2$ with uniform grid points $S = \{x_{ij} : 1 \leq i, j \leq N+1\}$
where $x_{ij} = (a + (i-1)\delta, a + (j-1)\delta)$
This creates N^2 cells C_{ij}
Let $y = (y_{ij}) \in \{0, 1\}^{N \times N}$ be a binary matrix indicating the ground truth values corresponding to the cells.
Let $v_f = \{f(x) : x \in S\}$ be the set of observed function values on S.
Similar to 1D, we construct a detector that takes in normalized $\tilde{v}_f$ and outputs a binary matrix to predict y

One-Level Detection method

We standardize input as done in 1D-Detector
Prediction is done on the standardized input using CNNs.

Then for a chosen threshold t, detector labels each of the $^{\text{th}}$ cell a trouble cell if $\hat{y}_{ij} > t$

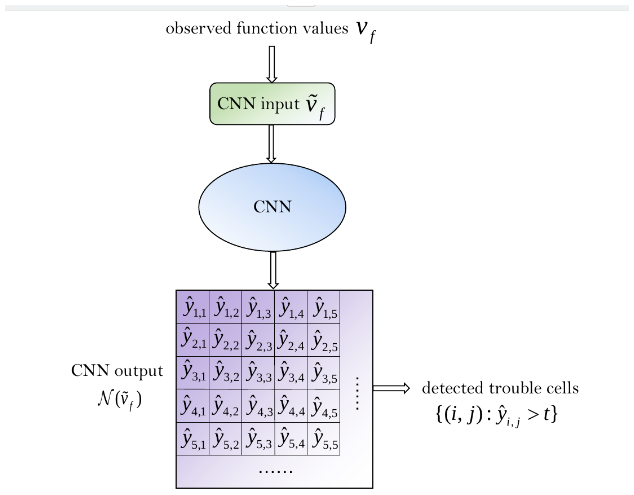

observed function values v_f
CNN input $\tilde{v}_f$
CNN
$\hat{y}_{1,1}$ $\hat{y}_{1,2}$ $\hat{y}_{1,3}$ $\hat{y}_{1,4}$ $\hat{y}_{1,5}$
$\hat{y}_{2,1}$ $\hat{y}_{2,2}$ $\hat{y}_{2,3}$ $\hat{y}_{2,4}$ $\hat{y}_{2,5}$
$\hat{y}_{3,1}$ $\hat{y}_{3,2}$ $\hat{y}_{3,3}$ $\hat{y}_{3,4}$ $\hat{y}_{3,5}$
$\hat{y}_{4,1}$ $\hat{y}_{4,2}$ $\hat{y}_{4,3}$ $\hat{y}_{4,4}$ $\hat{y}_{4,5}$
$\hat{y}_{5,1}$ $\hat{y}_{5,2}$ $\hat{y}_{5,3}$ $\hat{y}_{5,4}$ $\hat{y}_{5,5}$
……
CNN output $\mathcal{N}(\tilde{v}_f)$
detected trouble cells $\{(i,j) : \hat{y}_{i,j} > t\}$

Figure 4: 2D- One Level Detector

One-Level Detection without Dense layer

In this we generate data as described above for training. On the generated data we train a CNN without a dense layer. Intuition behind using this model is that a model which is able to detect discontinuities should be able to do that by looking at local function values around that grid cell rather than the entire grid. Hence the model should be able to work if dense layer is not used. This method reduces the model size dramatically and the predictions remain more or less similar to that of the original model.

Two-Level Detection Method

The issue with the one level detection method is the drastic increase in the number of parameters when dimension is increased by 1. To account for that, we can use two level detection method in which we first view the entire grid as a coarse grid and find the troubled cells in the coarse grid.
Now inside each cell of coarse grid is a sub-grid on which another detector is run if the grid cell was indicated as a troubled cell by level 1 detector.

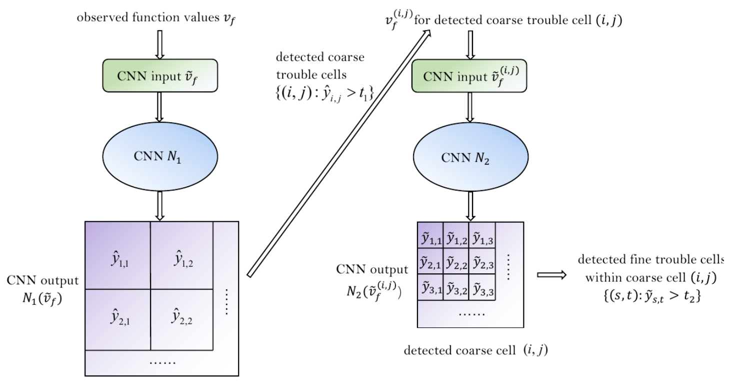
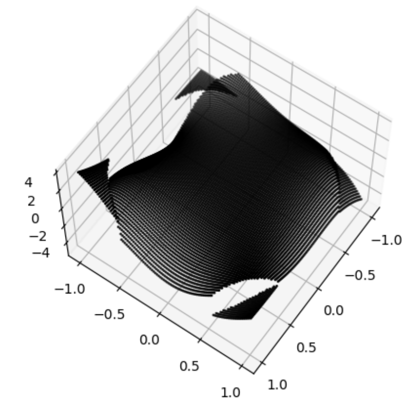
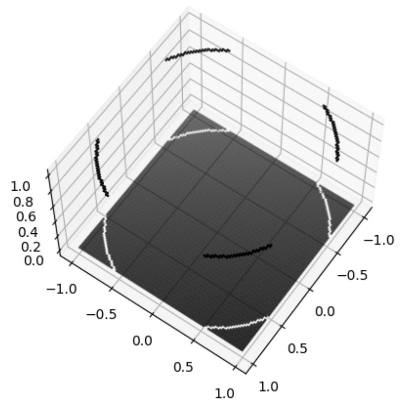

observed function values v_f
CNN input $\tilde{v}_f$
CNN N_1
$\hat{y}_{1,1}$ $\hat{y}_{1,2}$
$\hat{y}_{2,1}$ $\hat{y}_{2,2}$
……
CNN output $N_1(\tilde{v}_f)$
detected coarse trouble cells $\{(i,j) : \hat{y}_{i,j} > t_1\}$
$v_f^{(i,j)}$ for detected coarse trouble cell (i,j)
CNN input $\tilde{v}_f^{(i,j)}$
CNN N_2
$\tilde{y}_{1,1}$ $\tilde{y}_{1,2}$ $\tilde{y}_{1,3}$
$\tilde{y}_{2,1}$ $\tilde{y}_{2,2}$ $\tilde{y}_{2,3}$
$\tilde{y}_{3,1}$ $\tilde{y}_{3,2}$ $\tilde{y}_{3,3}$
……
CNN output $N_2(\tilde{v}_f^{(i,j)})$
detected coarse cell (i,j)
detected fine trouble cells within coarse cell (i,j) $\{(s,t) : \tilde{y}_{s,t} > t_2\}$

Figure 5: 2D- Two Level Detector

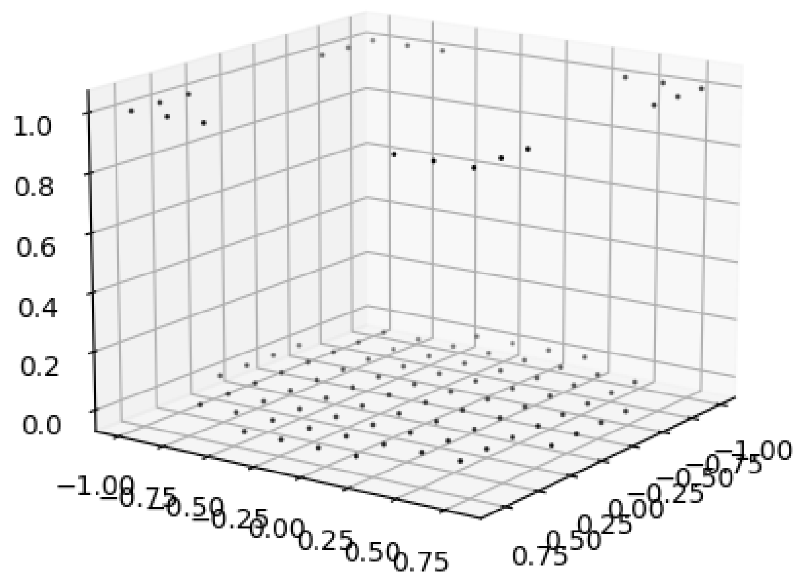
We implement this model on a grid of size 101×101. Each coarse cell contains 10×10 fine cells. We mark a coarse cell as a troubled cell if any of the fine cells within the coarse cell is a troubled cell.



(a) Function Data



(b) Fine cell labels



(c) Coarse cell labels

The second level detector is then applied on coarse cells to do the final prediction.
The model architechture dimensions for the coarse and fine model chosen are described as follows

Coarse Model Architecture		
Layer type	Output Shape	Param #
Conv2D	(None, 98, 98, 32)	544
Conv2D	(None, 48, 48, 32)	4128
Conv2D	(None, 47, 47, 32)	4128
Conv2D	(None, 46, 46, 32)	4128
Dropout	(None, 46, 46, 32)	0
Flatten	(None, 67712)	0
Dense	(None, 100)	**6771300**

Fine Model Architecture		
Layer type	Output Shape	Param #
Conv2D	(None, 10, 10, 32)	160
Conv2D	(None, 9, 9, 32)	4128
Conv2D	(None, 8, 8, 32)	4128
Conv2D	(None, 46, 46, 32)	4128
Flatten	(None, 2048)	0
Dropout	(None, 2048)	0
Dense	(None, 100)	**204900**

The two models are coupled to get the final predictions

U-net Architecture

This is one of the most promising models for the task of image segmentation. This model architecture looks like a ”U” and hence the name. The architecture consists mainly of two major parts. The left part is called contracting path and is general convolutional process where as the right portion is the upsampling path. The motivation behind using the architecture for the task of discontinuity identification is that the task of discontinuity identification can be transformed into the task of image segmentation where each segment is formed by the continuous portion of the function. The discontinuous cells can the be identified by finding the contours of segments which can be implemented using a basic contour identification algorithm

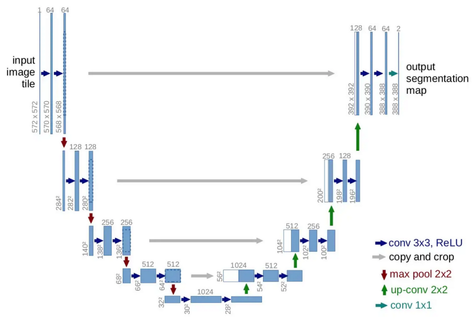


Figure 7: U-Net Architecture

Polynomial Annihilation Detection

The goal of polynomial annihilation is to construct a function $L_m f(x), m \in \mathbb{N}$, such that for x away from discontinuity points of $f(\cdot), L_m f(x) \approx 0$.
Hence the detection of discontinuities is based on $|L_m f(x)| > t$ where t is some threshold.
Suppose that f is known only on the discrete set S. Let Π_m be the space of all polynomials of degree $\leq m-1$

in d variables.
The value of $L_m f(x)$ at $x \in D$ is determined by the function values of f on a local set $S_x \subset S$ of $m_d = \binom{m+d}{d}$ points around x.
Let $S_x = x_1, x_2, \ldots, x_{m_d}$, be the set of m_d nearest points to x.
For polynomial annihilation up to degree $m-1$, one solves the linear system for coefficients $\{c_j(x) : j = 1, 2, \ldots, m_d\}$

$$\sum_{x_j \in S_x} c_j(x) p_i(x_j) = \sum_{|\alpha|_1 = m} p_i^{\alpha}(x)$$

where $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_d), \alpha_i \in \mathbb{N} \bigcup \{0\}, p_1, p_2, \ldots, p_{m_d}$ is a basis of Π_m
Since the solution of the above system exists and is unique, one can define

$$L_m f(x) = \frac{1}{q_{m,d}(x)} \sum_{x_j \in S_x} c_j(x) f(x_j)$$

where $q_{m,d}(x)$ is a normalization factor[1].

1D-Polynomial annihilation detector

We consider the basis as $\{1, x, x^2, \ldots, x^m\}$.
The grid is uniformly constructed as $[-1, -1+h], [-1+h, -1+2h], \ldots, [-1+(n-1)h, 1]$ and on this grid we define S_x as the set of nearest $m+1$ points in the grid to x.
We define $S_x^+ = \{i \in \mathbb{N} | x_i \in S_x, x_i \geq x\}$ and choose the normalization factor as

$$q_m(x) = \sum_{i \in S_x^+} c_i(x)$$

The system of equations to be solved for $L_m f(x)$ in this case becomes

$$\begin{bmatrix} 1 & 1 & 1 & 1\ldots & 1 \\ x_1 & x_2 & x_3 & \ldots & x_{m+1} \\ x_1^2 & x_2^2 & x_3^2 & \ldots & x_{m+1}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^m & x_2^m & x_3^m & \ldots & x_{m+1}^m \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_m \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ m! \end{bmatrix}$$

In the implementation we take $m = 5$. Better results can be obtained by taking larger values of m.

2D-Polynomial annihilation detector

We consider the basis $\{x^i y^j | i + j \leq m\}$.
The entire space is divided into triangular elements.
Let $S = \{T_1, T_2, \ldots, T_N\}$ be the discretization of the space into triangular elements.
For an element T_j of S define $\bar{x_j} = \left(\frac{x_{j1} + x_{j2} + x_{3j}}{3}, \frac{y_{j1} + y_{j2} + y_{j3}}{3}\right)$ where the vertices of T_j are $(x_{j1}, y_{j1}), (x_{j2}, y_{j2}), (x_{j3}, y_{j3}$
For $x \in \Omega$, define $S_x = T_j \cup S_{T_j}$ where $x \in T_j$ and S_{T_j} is the collection of nearest $m_2 - 3$ points to x in $S \backslash T_j$
The detection algorithm can be described in the following 3-steps
Step 1: For element T_j we consider $S_{\bar{x_j}}$ and order $S_{\bar{x_j}} = \{x_1, x_2, \ldots, x_{m_2}\}$ in order such that $f(x_1) \leq f(x_2) \leq f(x_3) \ldots \leq f(x_{m_2})$.
Step 2: Solve the system

$$\sum_{x_i \in S_{\bar{x_j}}} c_i x_{i1}^{\alpha_1} x_{i2}^{\alpha_2} = \begin{cases} 0, \text{ if } \alpha_1 + \alpha_2 < m \\ \alpha_1! \alpha_2!, \text{ if } \alpha_1 + \alpha_2 = m \end{cases}$$

Step 3: Calculate $q_m(\bar{x_j}) = \sum_{x_i \in \mathcal{P}_{\bar{x_j}}} c_i,$

where $\mathcal{P}_x = \{x_1, x_2, x_3, \ldots, x_r\}$, and $|f(x_{r+1} - f(x_r)| = \max_{i=1,2,\ldots,m_2-1} |f(x_{i+1}) - f(x_i)|$

Step 4: Calculate $L_m f(\bar{x_j}) = \frac{1}{q_m(\bar{x_j})} \sum_{x_i \in S_{\bar{x_j}}} c_i f(x_i)$

Detect discontinuity if $|L_m f(\bar{x_j})| \geq t$ for some suitably chosen threshold t

Polynomial Annihilation Coupled with MinMod

We obtain far more superior results if we couple polynomial annihilation method with minmod limitter. The procedure is as follows
Define the set $\mathcal{M} = \{1, 2, \ldots, M\}$
Define $L_{\mathcal{M}} f = \{L_m f : \mathbb{R} \to \mathbb{R} | m \in \mathcal{M}\}$
The *minmod* function is hence given by

$$MM(L_{\mathcal{M}} f(x) = \begin{cases} \min\limits_{m \in \mathcal{M}} L_m f(x) \text{ if } L_m f(x) > 0 \text{ for all } m \in \mathcal{M} \\ \max\limits_{m \in \mathcal{M}} L_m f(x) \text{ if } L_m f(x) < 0 \text{ for all } m \in \mathcal{M} \\ 0 \text{ Otherwise} \end{cases}$$

Delaunay Triangulation

Polynomial annihilation method works best when we use delaunay triangulation for finding the simplices containing some point.
The constraints of delaunay triangulation are

1. No point lies in the interior of circumcircle of any triangular element.
2. Triangles are as equiangular as possible, i.e. the minimum interior angle of all triangles is maximized, and the maximum minimized.

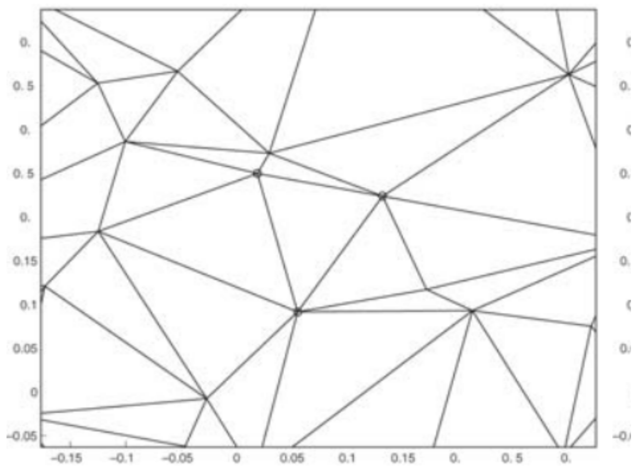


Figure 8: An example of delaunay triangulation

We use inbuilt delaunay function from scipy.spatial module to implement delaunay triangulation of our mesh.

CNN Model + Polynomial Annihilation model

In this model, we use the coarse CNN model to predict coarse troubled cells and then use polynomial annihilation model on the finer cells instead of CNN model.

Multi layer perceptron detector

In this we use a neural network as limiter. The aim is to develop a troubled cell indicator, by using the local values such as cell averages and neighbouring cell averages as inputs to the neural network and predict if the cell is a troubled cell or not[4][2].

RKDG Scheme

Consider the problem

$$\begin{cases} \partial_t u + \partial_x f(u) = 0, (x,t) \in \Omega \times [0,\infty] \\ u(x,0) = u_0(x) \\ u_{\partial\Omega} = g \end{cases}$$

Let $\Omega \in \mathbb{R}$ be a regular domain discretized by N elements $K_p = \left[x_{p-1/2}, x_{p+1/2}\right]$ for $p = 1, 2, \ldots, N$.
Consider the local space $\mathcal{V}$ given by the set $\{\phi_i\}_{i=0}^{n}$ of one-dimensional Legendre polynomials of degree at most n in x.
The numerical solution for any element K is written as

$$u^K(x,t) = \sum_{i=0}^{n} \hat{u}_i^K(t)\phi_i(x)$$

, where the coefficient $u_i^K(t)$ is chosen to minimize L^2 error between the approximation and the exact solution, i.e. an L^2 projection
The scheme then reads as

$$\frac{d\hat{u}_i^K}{dt} + [\hat{f}(u^K(x,t))\phi_i(x)]_{x_{p-1/2}}^{x_{p+1/2}} - \int_K f(u^K(x,t))\partial_x\phi_i(x)dx = 0, i = 0, 1, 2, \ldots, n$$

After this, discretization in space is obtained by using Strong Stability preserving Runge-Kutta method.
To avoid the problem of getting oscillatory solutions, due to strong discontinuities, the method is coupled with a slope limitter.

$$\tilde{u}_1^K = \frac{1}{\sqrt{3}}\text{minmod}\left(\sqrt{3}\hat{u}_1^K, \frac{1}{2}(\hat{u}_0^K - \hat{u}_0^{K_l}), \frac{1}{2}(\hat{u}_0^{K_r} - \hat{u}_0^K)\right)$$

where K_l, K_r are the left and the right cells corresponding to a given cell and

$$\text{minmod}(a,b,c) = \begin{cases} s\min(|a|,|b|,|c|), s = sgn(a) = sgn(b) = sgn(c) \\ 0, \text{ Otherwise} \end{cases}$$

The numerical solution, then becomes

$$\tilde{u}^K = \hat{u}_0^K + \tilde{u}_1^K\phi_1^K$$

if the limited weights are not same as the unlimited weights(i.e. $u_1^K = \tilde{u}_1^K$), otherwise the solution is taken as the unlimited one.

MLP Coupling with Numerical method

Conceptually a limiter can be considered as a procedure with two following sub steps:

1. A shock indicator function $\mathcal{C}$, that identifies a troubled cell
2. A reconstruction procedure Π that modifies the solution polynomial into a reconstructed polynomial, which is less oscillatory

The aim is to use locally defined values in a cell to identify the cell as a troubled cell or not. This mapping is learned using a neural network. The main aspect to this process is the process of data generation.
We choose to solve the linear advection problem using modal RKDG scheme and use min-mod limiter as an indicator variable. To circumvent over-determination of min-mod limiter a cell is labelled as a troubled cell if the modification by min-mod limiter leads to more than 1% change in the final solution
The neural network trained on the data is the coupled with an existing CFD code and new limiting procedure is henceforth obtained. The algorithm can be described as
Algorithm Coupling of neural network with existing CFD code
Data Solution at cell i, u_i
Output Stabilised solution $\tilde{u}_i$

```
1. $X = \text{Features}(u_i)$
2. label $= \mathcal{C}(X)$
3. **if** label $= 1$ **then**
4. $\tilde{u}_i = \Pi(u_i)$
5. **else**
6. $\tilde{u}_i = u_i$
```

Input for MLP

Features for 1D MLP		
ID	Feature Name	Description
1	h	Cell width
2	$\bar{u_i}$	Average value of solution at cell i
3	$\bar{u_{i+1}}$	Average of solution at cell $i+1$
4	$\bar{u_{i-1}}$	Average value of solution at cell $i-1$
5	$u^{+}_{i-1/2}$	Value of solution at interface $i-1/2$ reconstructed in cell i
6	$u^{+}_{i+1/2}$	Value of solution at interface $i+1/2$ reconstructed in cell i
7	$u^{-}_{i-1/2}$	Value of solution at interface $i-1/2$ reconstructed at cell $i-1$
8	$u^{+}_{i+1/2}$	Value of solution at interface $i+1/2$ reconstructed at cell $i+1$
9	du_{i+1}	Undivided difference between $\bar{u_i}$ and $\bar{u_{i+1}}$
10	du_{i-1}	Undivided difference between $\bar{u_{i-1}}$ and $\bar{u_i}$
11	du_i	Undivided difference between $\bar{u_{i-1}}$ and $\bar{u_{i+1}}$, divided by 2

Results

Comparison of Model accuracies

1D Models		
Model Name	Number of parameters	Accuracy
CNN Model	949,400	99.95%
Polynomial Annihilation($m = 5$)	0	94.42%
Polynomial Annihilation($m = 7$)	0	92.91%
Polynomial + MinMod($m = 5$)	0	99.01%
Polynomial + MinMod($m = 7$)	0	99%

2D Models		
Model Name	Number of parameters	Accuracy
1 Level CNN Model	721,111,033	99.45%
2 Level CNN Model	6,784,228 + 213,316 = 6,997,544	99.82%
U-net architecture	1,179,121	99.75%
No Dense CNN Model	242,657	98.47%

Accuracy of 2D polynomial annihilation models was not calulated as they take approximately $1 - 2$ minutes per example for calculation.

1. 1D CNN model is able to detect discontinuity with high accuracy on synthetically generated data.
2. Polynomial annihilation method labels cells which are near troubled cells also as troubled cells.
3. For the function
$$f(x) = \begin{cases} \sin(x), -1 \leq x < -0.9 \\ 2\sin(x), -0.9 \leq x < -0.8 \\ 3\sin(x), -0.8 \leq x < -0.7 \\ 4\sin(x), -0.7 \leq x < -0.6 \\ 5\sin(x), -0.6 \leq x < -0.5 \\ 7\sin(x), -0.5 \leq x < -0.4 \\ 8\sin(x), -0.4 \leq x \leq 0.4 \\ 6\sin(x), 0.4 \leq x \leq 1 \end{cases}$$
   1D CNN is able to predict correctly, but polynomial annihilation fails to predict all the discontinuity points
4. Polynomial annihilation coupled with Minmod performs better than polynomial annihilation model generally. Minmod polynomial is able to find all the troubled cells in above case.
5. Predictions of polynomial annihilation model are highly sensitive to threshold value. The same is not true for CNN-model
6. Polynomial annihilation model works better for discontinuities which are very close to each other, where as 1D-CNN model works better for cases in which discontinuities are lesser in number. This can be improved by training the model on a different dataset in which the number of discontinuities is more.
7. It can be seen that polynomial annihilation model predictions get better and better as we increase m. However this model is extremely slow.

8. 2D One Level CNN with and without Dense layer make accurate predictions for circular discontinuities, but for line discontinuities the predictions spreads to nearby cells.
9. On taking standard triangular discretization of the element space, polynomial annihilation method produces singular matrices and hence the system cannot be solved for unique solution.
10. Two level CNN model performs better than single level model in terms of accuracy and speed.

1D-CNN-Model Predictions

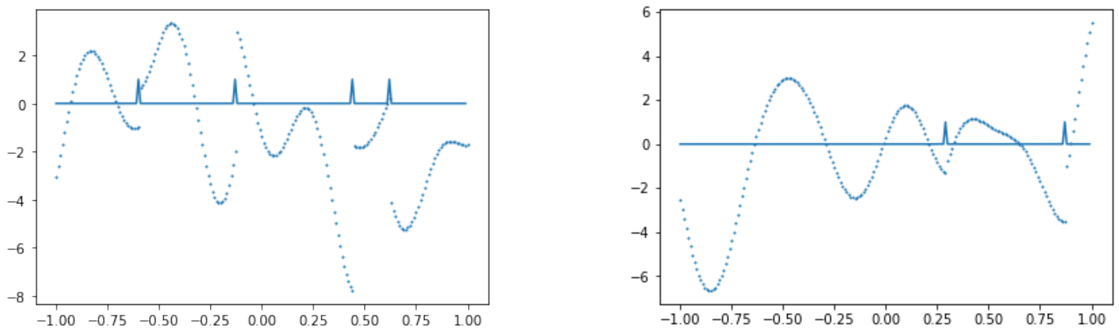


Figure 9: 1D-CNN Model predictions

1D Polynomial Annihilation predictions

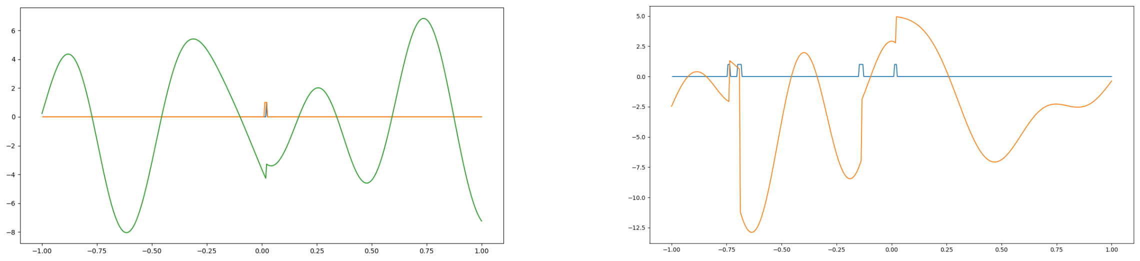


Figure 10: Polynomial Annihilation Predictions

CNN vs Polynomial annihilation for $f(x)$ and a general function

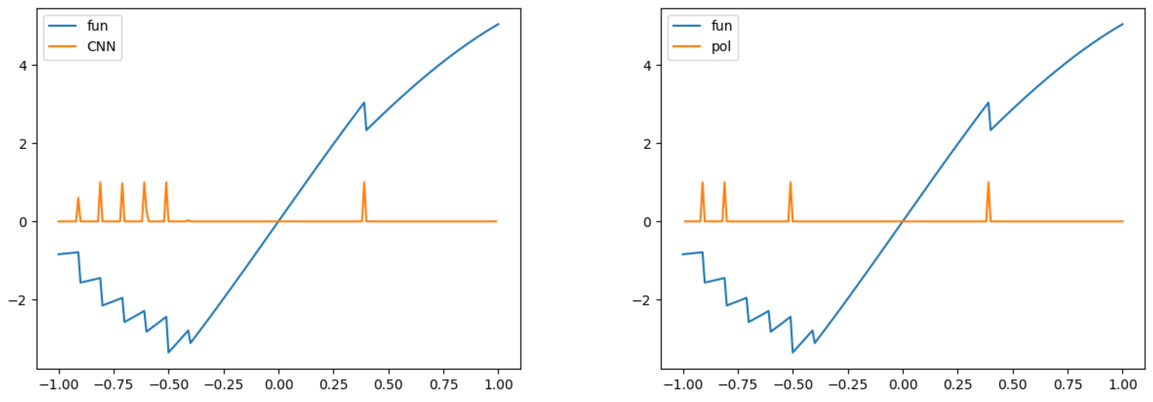


Figure 11: CNN(left) vs polynomial Annihilation(right) for $f(x)$

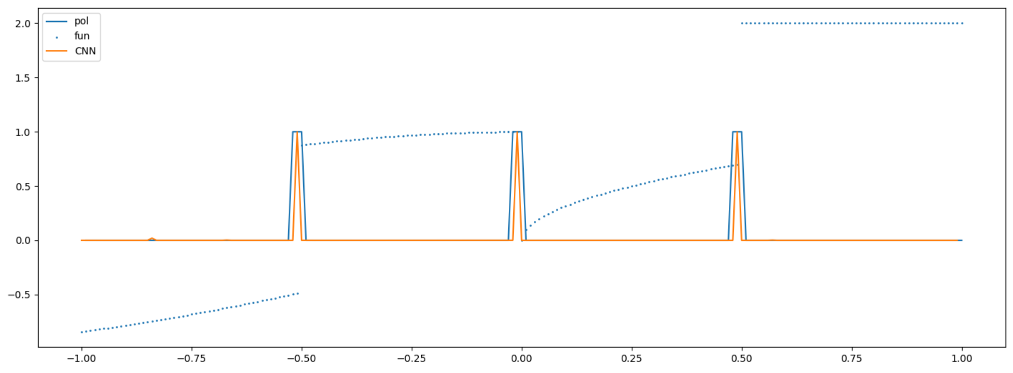


Figure 12: Comparison of CNN and Polynomial annihilation for $[x]$(left)

Polynomial annihilation coupled with Minmod for $f(x)$ and a general function

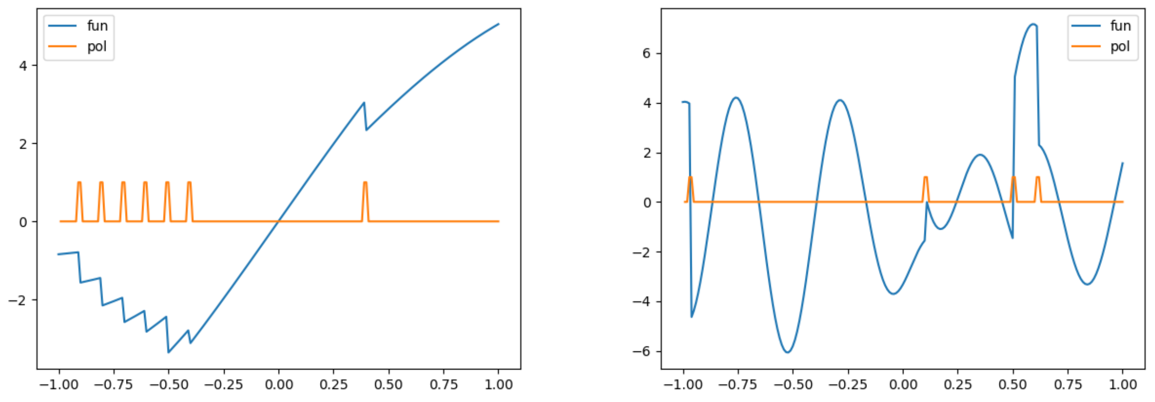


Figure 13: Minmod Polynomial detection for $f(x)$(left) and general function (right)

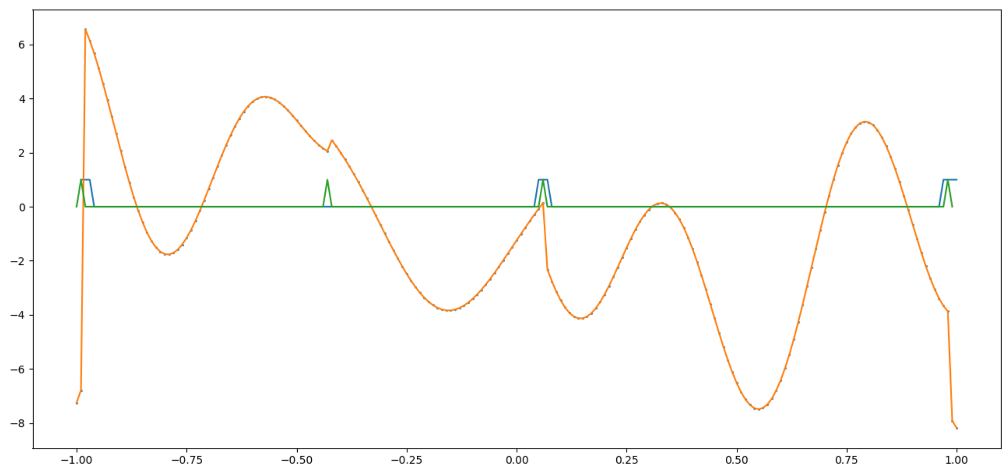


Figure 14: Comparison between CNN and Polynomial annihilation on functions on which CNN is trained

2D-Model predictions

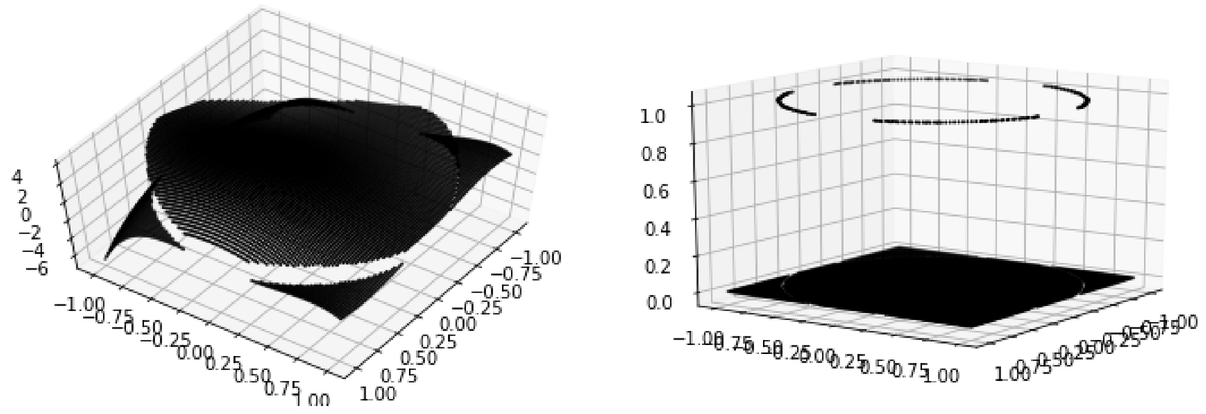
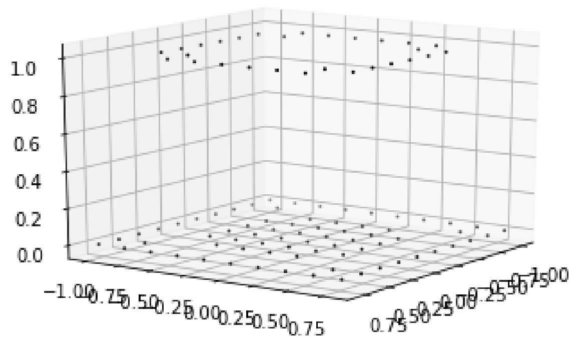
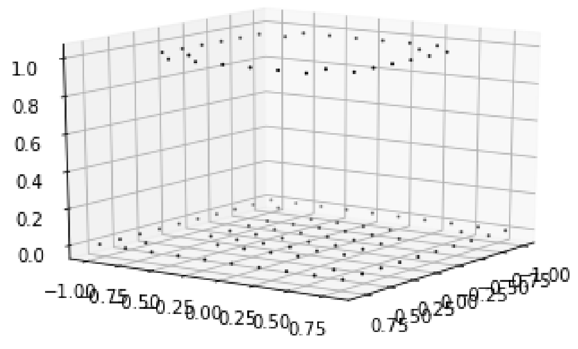


Figure 15: One Level CNN model predictions

2D Coarse Cell Model

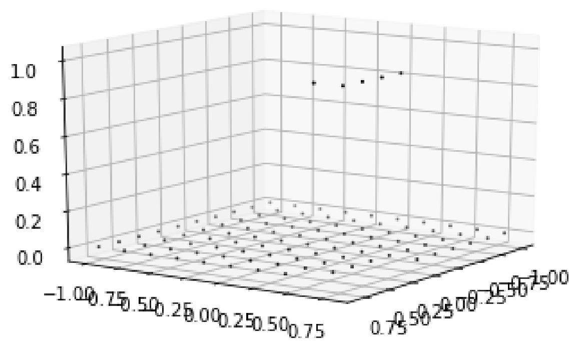


(a) Coarse cell Labels

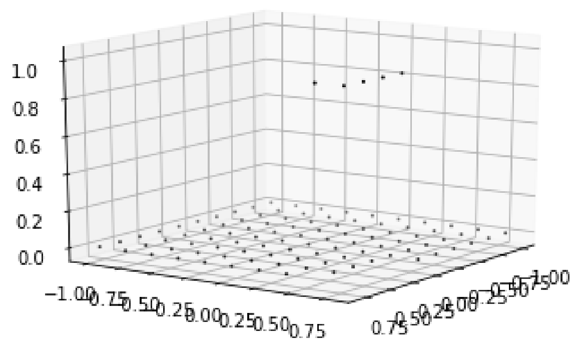


(b) Coarse Model Prediction

2D Fine Cell Model

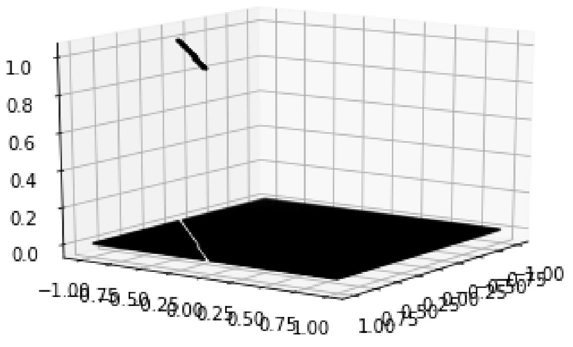


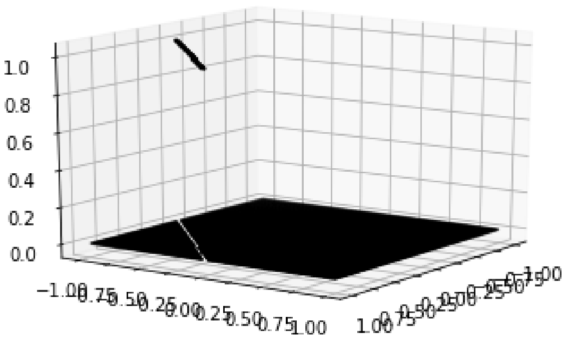
(a) Ground Truth



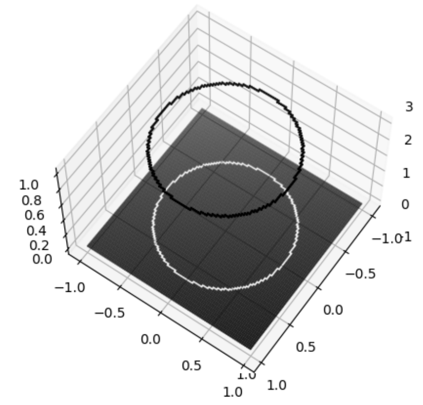
(b) Fine Model prediction

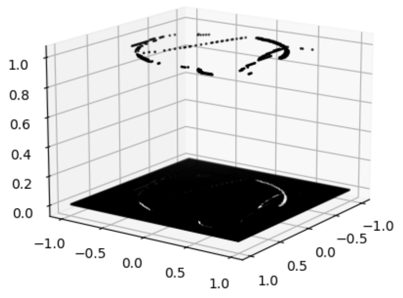
Two level CNN Model Predictions

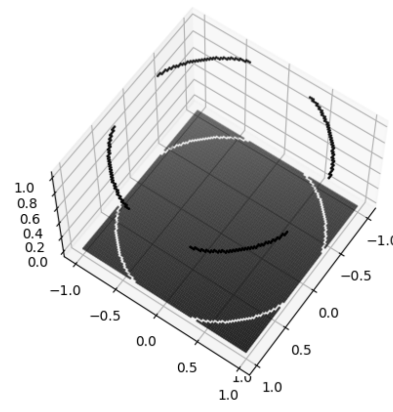

(a) Ground Truth

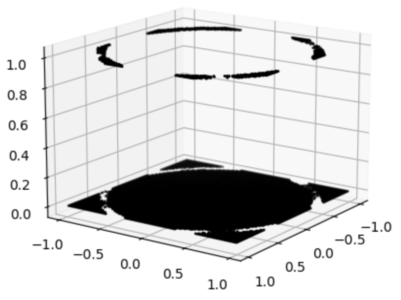

(b) Coupled Model prediction

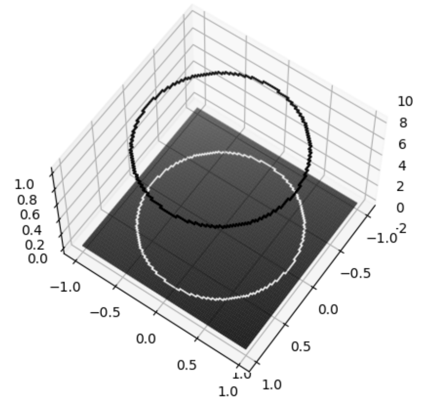
2D Polynomial Annihilation

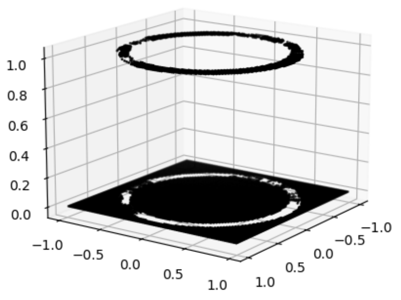

(a) Ground Truth


(b) Model prediction with $m = 2$

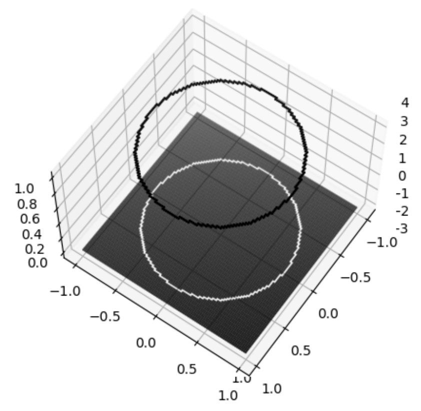

(a) Ground Truth

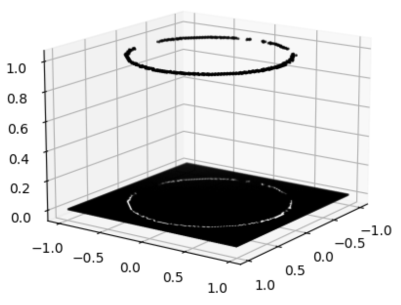

(b) Model prediction with $m = 5$


(a) Ground Truth

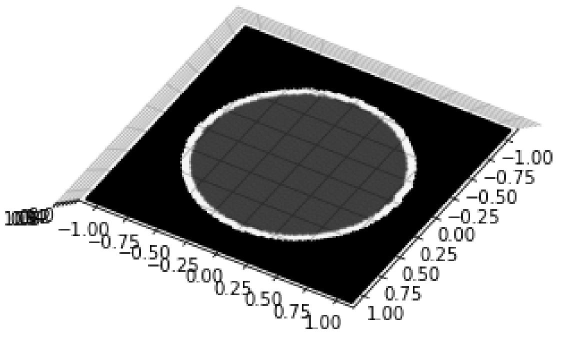

(b) Model prediction with $m = 7$

2D Polynomial minmod coupled predictions

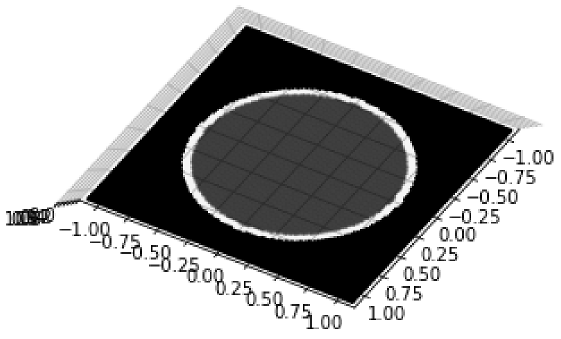

(a) Ground Truth


(b) Model prediction with $m = 7$

U-net Architecture Predictions



(a) Ground Truth



(b) U-net model prediction with 10 epochs

Future works

1. The trained data is synthetically generated and does not incorporate for errors that are present because of applying numerical methods. Training the model on data generated using numerical methods can make the model more robust[3].

2. Data used for training 2D models takes a lot of space and hence training the model needs high performance computing if number of epochs is large. We aim to use GPU for speeding the training process and develop models trained for larger number of epochs and large size of data.

3. Develop a numerical scheme with the models made as troubled indicators along with a reconstruction scheme and compare the results.

4. Develop methods that can be implemented on unstructured grids.

Link to the project implementations

Discontinuity Identification in numerical solutions of Differential equations

References

[1] Rick Archibald, Anne Gelb, and Jungho Yoon. Polynomial fitting for edge detection in irregularly sampled signals and images. *SIAM J. Numerical Analysis*, 43:259–279, 01 2005.

[2] Niccolò Discacciati, Jan S. Hesthaven, and Deep Ray. Controlling oscillations in high-order discontinuous galerkin schemes using artificial viscosity tuned by neural networks. *Journal of Computational Physics*, 409:109304, 2020.

[3] Zheng Sun. Convolution neural network shock detector for numerical solution of conservation laws. *Communications in Computational Physics*, 2020.

[4] Maria Han Veiga and Rémi Abgrall. Neural network based limiter with transfer learning, 2019.

[5] Shuyi Wang, Zixu Zhou, Lo-Bin Chang, and Dongbin Xiu. Construction of discontinuity detectors using convolutional neural networks. *J. Sci. Comput.*, 91(2), may 2022.