GitHub Link: https://github.com/anushkachaubey/Anushka-EA-Assignment3

For both questions just run the single notebook

Question 1:

1. **Model Architecture:**
   I defined a feedforward neural network (`FeedforwardNN`) with 3 hidden layers. Each hidden layer has 64 neurons and uses ReLU activation. The input layer takes 2 features, and the output layer predicts a single continuous value.

2. **Data Preparation:**
   The input data `X` and target values `y` were converted into PyTorch tensors to be used for training.

3. **Model 1 (MSE Loss):**
   I trained the first model using mean squared error (MSE) loss and the Adam optimizer. The training ran for 200 epochs. During training, the model learns to minimize the squared difference between its predictions and the true target values.

4. **Model 2 Eikonal Loss (PINN):**
   I defined a custom function `compute_eikonal_loss` that calculates the residual of the Eikonal equation using automatic differentiation to compute gradients of the model output w.r.t input coordinates. This enforces the physical constraint that the gradient norm times a velocity function should equal 1.
   A second model (`model2`) with the same architecture was trained using this Eikonal loss for 1000 epochs. This model acts like a Physics-Informed Neural Network (PINN).

5. **Model 3 Hybrid Loss PINNs**

   I implemented a third model (`model3`) using the same neural network architecture as before.
   This model is trained with a **hybrid loss** function combining two terms:
   1. **MSE loss** (data loss): Measures how close the predictions are to the true values.
   2. **Eikonal residual loss** (physics loss): Enforces the physical constraint by penalizing deviations from the Eikonal equation using gradients computed via automatic differentiation.

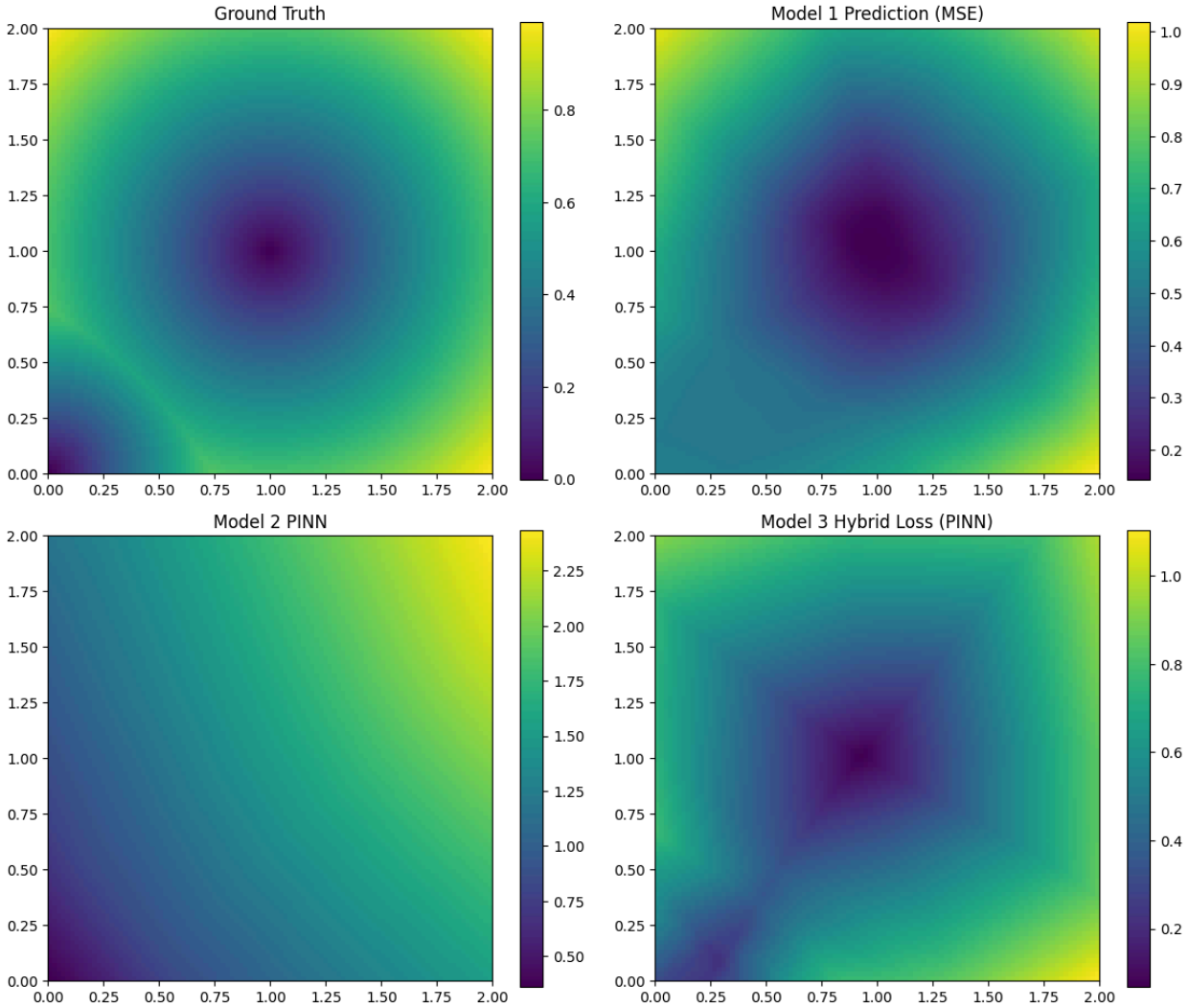
This approach blends data-driven learning and physics-based regularization, known as a Physics-Informed Neural Network (PINN) with hybrid loss.

5. **Evaluation:**

   - Model 1: MSE loss.
   - Model 2 : physics-based Eikonal equation.
   - Model 3 : Hybrid Loss: MSE + Physics loss

Model	Loss	RSME
Model 1	MSE Loss	0.0262
Model 2	Physics-based Eikonal equation.	1.0339
Model 3	Hybrid Loss: MSE + Physics loss	0.0792

6. **Visualization:**
   Activation Maps



Question 2

1. Simple Neural Network (SimpleNN):

- A baseline fully connected feedforward neural network with one hidden layer.
- Takes 2-dimensional input features, processes them through a hidden layer with ReLU activation, and outputs a single value activated by a sigmoid to produce a probability for binary classification.

2. Neural ODE Block (ODEFunc):

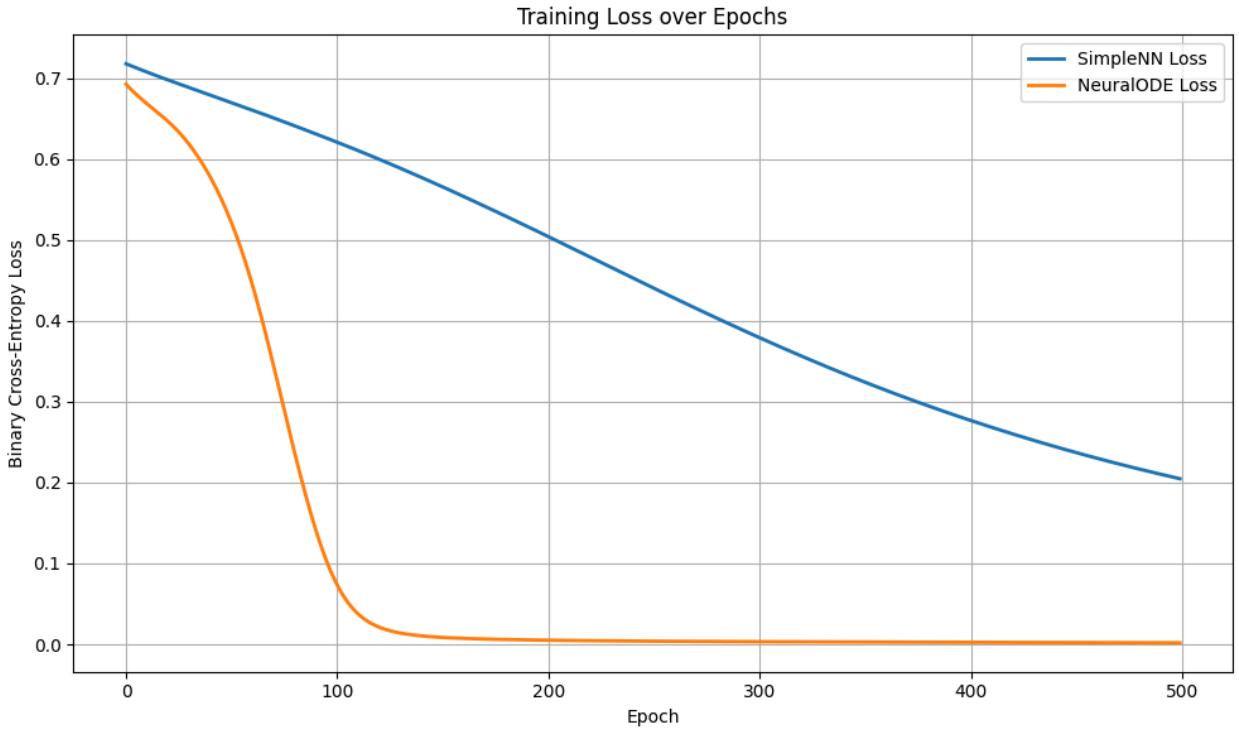
- Implements a small neural network defining the dynamics of the hidden state in continuous time, i.e., the vector field of an ODE.
- Initialized with small weights close to zero to promote stable training of the ODE solver.

3. Neural ODE Model (NeuralODE):

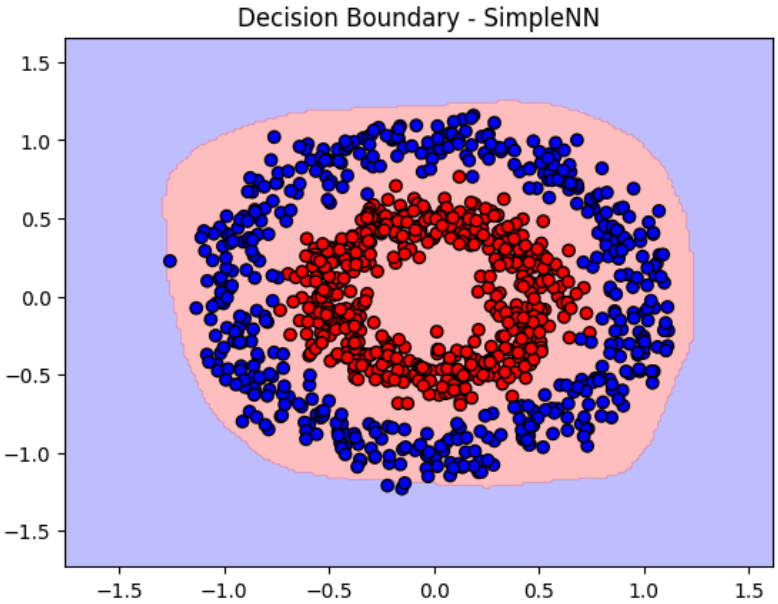
- Combines an input linear layer to encode input features into a latent space, the ODEFunc defining the continuous evolution of the hidden state, and an output linear layer to map back to prediction space.
- Uses odeint from the torchdiffeq package to solve the neural ODE from initial time t=0 to t=1 with the Runge-Kutta 4 (RK4) method.
  The output is passed through a sigmoid for binary classification probability.

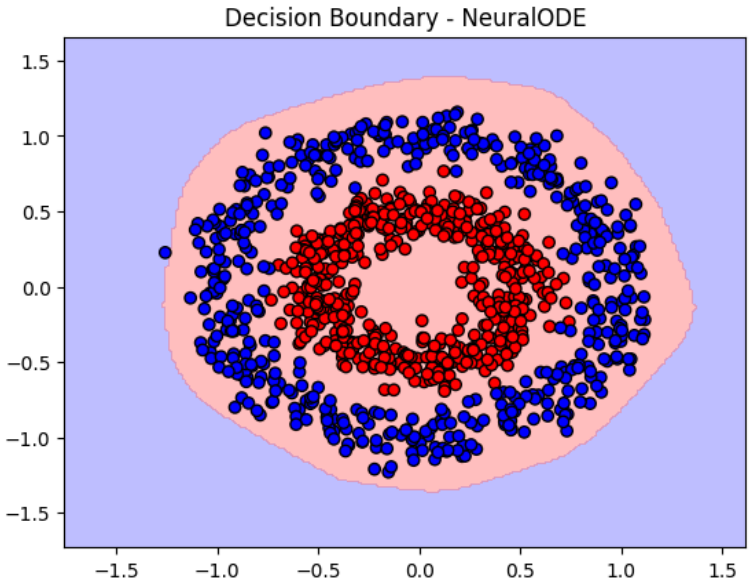
5. Final Accuracy Reporting:

Model	Training Accuracy	Testing Accuracy
Simple NN	0.9962	0.965
Neural ODE	1	0.98



6. Decision Boundary Visualization:





Q2 Part C attached below

__/__/____

Assignment 3

2. Part C:

b] NN: Discrete learning i.e $h_{i+1} = f(h_i, \theta_i)$
The transformation happens in discrete steps

Neural ODE: Continuous transformation.
It replaces layer-by-layer with a continuous differential $\frac{dh(t)}{dt} = f(h(t), t, \theta)$

Due to this change, instead of jumping from one layer to another, the model smoothly flows through a transformation path.

c] Residual connection: $h_{i+1} = h_i + f(h_i)$ — (1)

Euler's method: $\frac{d(h(t))}{dt} = f(ht)$

if we discritize time with step Δt

$$h(t + \Delta t) \approx h(t) + \Delta t \cdot f(h(t))$$

Now let $\Delta t = 1$:

$$\Rightarrow h(t+1) \approx h(t) + f(h(t)) \quad \text{— (2)}$$

Compare (1): $h_{i+1} = h_i + f(h_i)$
(2): $h(t+1) = h(t) + f(h(t))$

∴ (1) (2) are of identical forms

So residual connection is a type of euler ODE where $\Delta t = 1$.