```
# Import dependencies
import numpy as np
import torch
import torchvision
from torch.utils.data.dataset import Dataset
from torchvision import datasets, transforms
from torch import nn, optim
import matplotlib.pyplot as plt
```

## Problem 4.4 Compare different network architectures

```
# ################## Part 1: Load data and create batch
##################
N_total = 600
N_train = 500
x = torch.unsqueeze(torch.linspace(0, 1, N_total), dim=1)
r = torch.randperm(N_total)
x = x[r, :]
y = 0.2 + 0.4 * torch.pow(x, 2) + 0.3 * x * torch.sin(15 * x) + 0.05 *
torch.cos(50 * x)

class CustomDataset(Dataset):
    def __init__(self, x, y):
        self.y = y
        self.x = x

    def __len__(self):
        return len(self.y)

    def __getitem__(self, idx):
        y1 = self.y[idx]
        x1 = self.x[idx]
        return (x1, y1)

# Change batch_size here to test different values
batch_size = 32  # Experiment with different batch sizes: 32, 64, 128
trainset = CustomDataset(x[0:N_train, :], y[0:N_train, :])
testset = CustomDataset(x[N_train:N_total, :], y[N_train:N_total, :])
train_loader = torch.utils.data.DataLoader(trainset,
batch_size=batch_size)
test_loader = torch.utils.data.DataLoader(testset,
batch_size=batch_size)

# ################## Part 2: Define Different Network Architectures
##################

# Architecture 1: One hidden layer with 16 neurons
model1 = nn.Sequential(nn.Linear(1, 16),
                       nn.ReLU(),
                       nn.Linear(16, 1))
```

```
# Architecture 2: One hidden layer with 32 neurons
model2 = nn.Sequential(nn.Linear(1, 32),
                       nn.ReLU(),
                       nn.Linear(32, 1))

# Architecture 3: One hidden layer with 64 neurons
model3 = nn.Sequential(nn.Linear(1, 64),
                       nn.ReLU(),
                       nn.Linear(64, 1))

# Function to initialize weights
def init_weights(m):
    if isinstance(m, nn.Linear):
        m.weight.data.uniform_(-1, 1)
        m.bias.data.uniform_(-1, 1)

# Initialize weights for all models
model1.apply(init_weights)
model2.apply(init_weights)
model3.apply(init_weights)
```

```
Sequential(
  (0): Linear(in_features=1, out_features=64, bias=True)
  (1): ReLU()
  (2): Linear(in_features=64, out_features=1, bias=True)
)
```

```
# ################## Part 3: Define Loss and Optimizer
##################
criterion = torch.nn.MSELoss()

# You can adjust learning rates here
optimizer1 = torch.optim.Adam(model1.parameters(), lr=0.001)
optimizer2 = torch.optim.Adam(model2.parameters(), lr=0.001)
optimizer3 = torch.optim.Adam(model3.parameters(), lr=0.001)

# ################## Part 4: Train and Test Function
##################
def train_NN(model, optimizer):
    model.train()
    for images, labels in train_loader:
        out = model(images)
        loss = criterion(out, labels)
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
    return loss.item()

def test_NN(model, loader):
    model.eval()
```

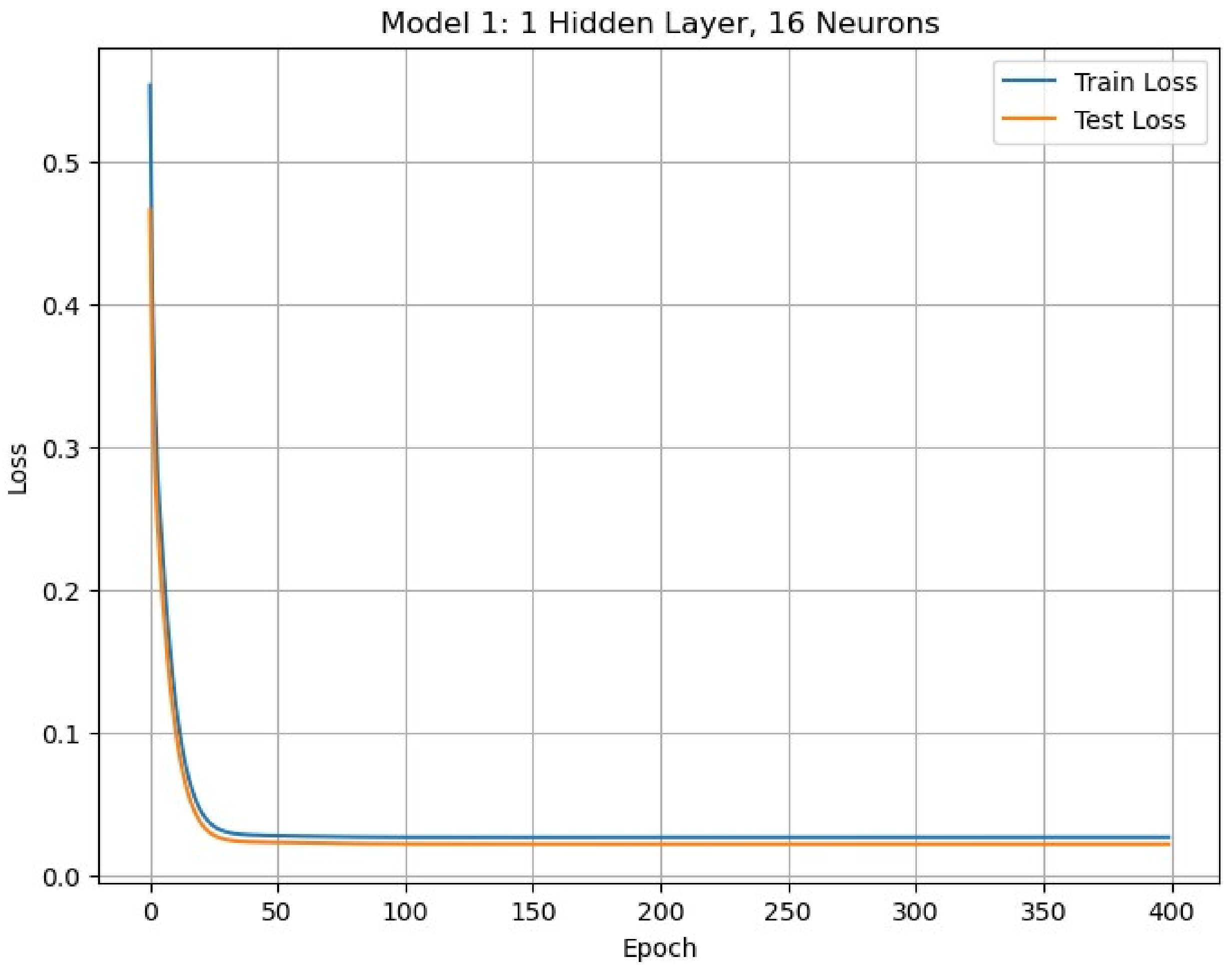
```python
    loss = 0
    with torch.no_grad():
        for images, labels in loader:
            out = model(images)
            loss += criterion(out, labels).item()
    return loss / len(loader)

# ################## Part 5: Train Models and Track Loss
##################
N_epoch = 400

def run_training(model, optimizer):
    train_loss = []
    test_loss = []
    for epoch in range(N_epoch):
        train_l = train_NN(model, optimizer)
        test_l = test_NN(model, test_loader)
        train_loss.append(train_l)
        test_loss.append(test_l)
        if epoch % 50 == 0:
            print(f'Epoch {epoch}, Train Loss: {train_l}, Test Loss:
{test_l}')
    return train_loss, test_loss

# Train all models
train_loss1, test_loss1 = run_training(model1, optimizer1)
train_loss2, test_loss2 = run_training(model2, optimizer2)
train_loss3, test_loss3 = run_training(model3, optimizer3)
```

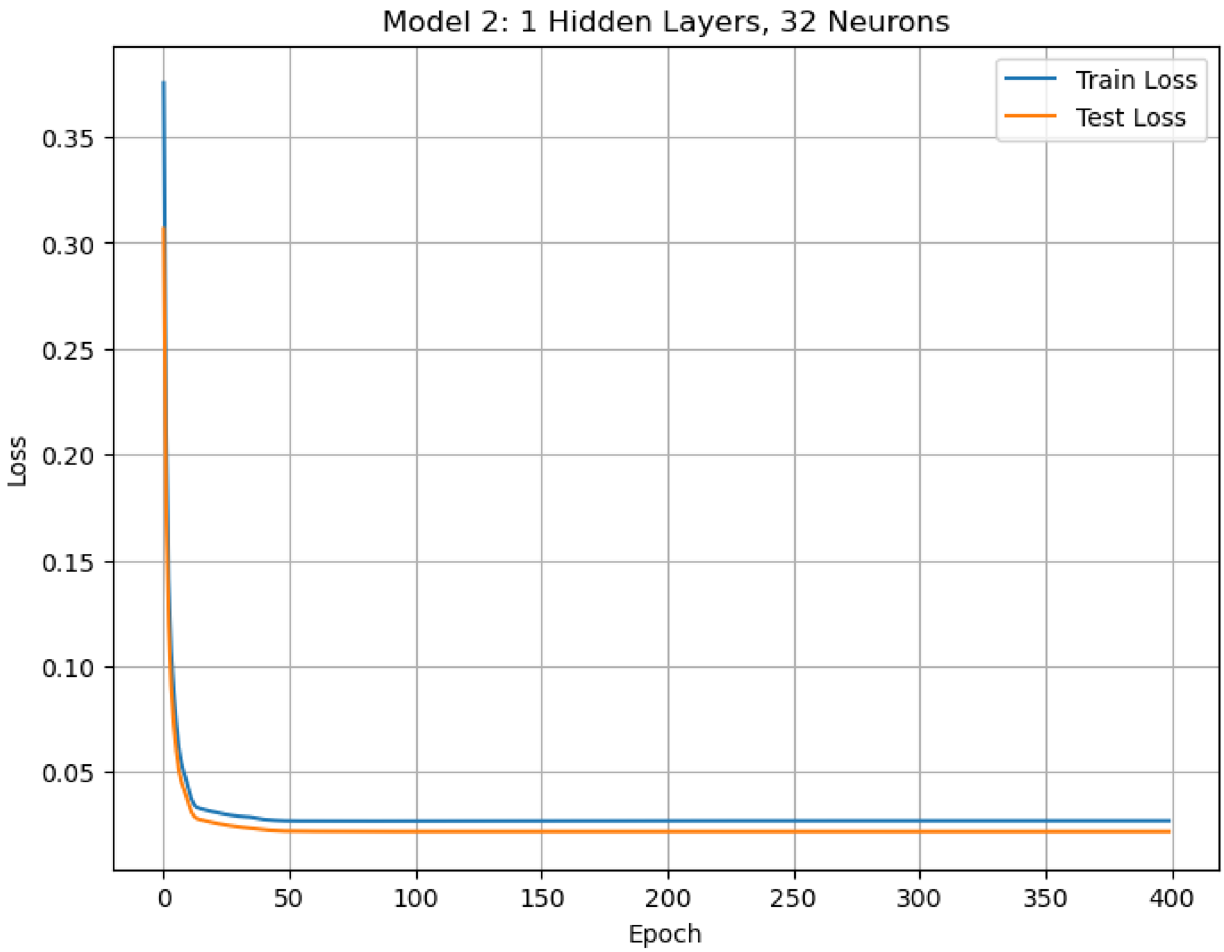
```
Epoch 0, Train Loss: 0.5534979701042175, Test Loss:
0.46619994193315506
Epoch 50, Train Loss: 0.027902286499738693, Test Loss:
0.023273158818483353
Epoch 100, Train Loss: 0.026946479454636574, Test Loss:
0.022135890321806073
Epoch 150, Train Loss: 0.026902684941887856, Test Loss:
0.022042084252461 79
Epoch 200, Train Loss: 0.026918213814496994, Test Loss:
0.022009468637406826
Epoch 250, Train Loss: 0.02693040296435356, Test Loss:
0.021983390441164374
Epoch 300, Train Loss: 0.02694222331047058, Test Loss:
0.021972597111016512
Epoch 350, Train Loss: 0.026946749538183212, Test Loss:
0.021967295557260513
Epoch 0, Train Loss: 0.3755437433719635, Test Loss:
0.30665967613458633
Epoch 50, Train Loss: 0.026898186653852463, Test Loss:
0.0221025357022881 5
Epoch 100, Train Loss: 0.026842549443244934, Test Loss:
```

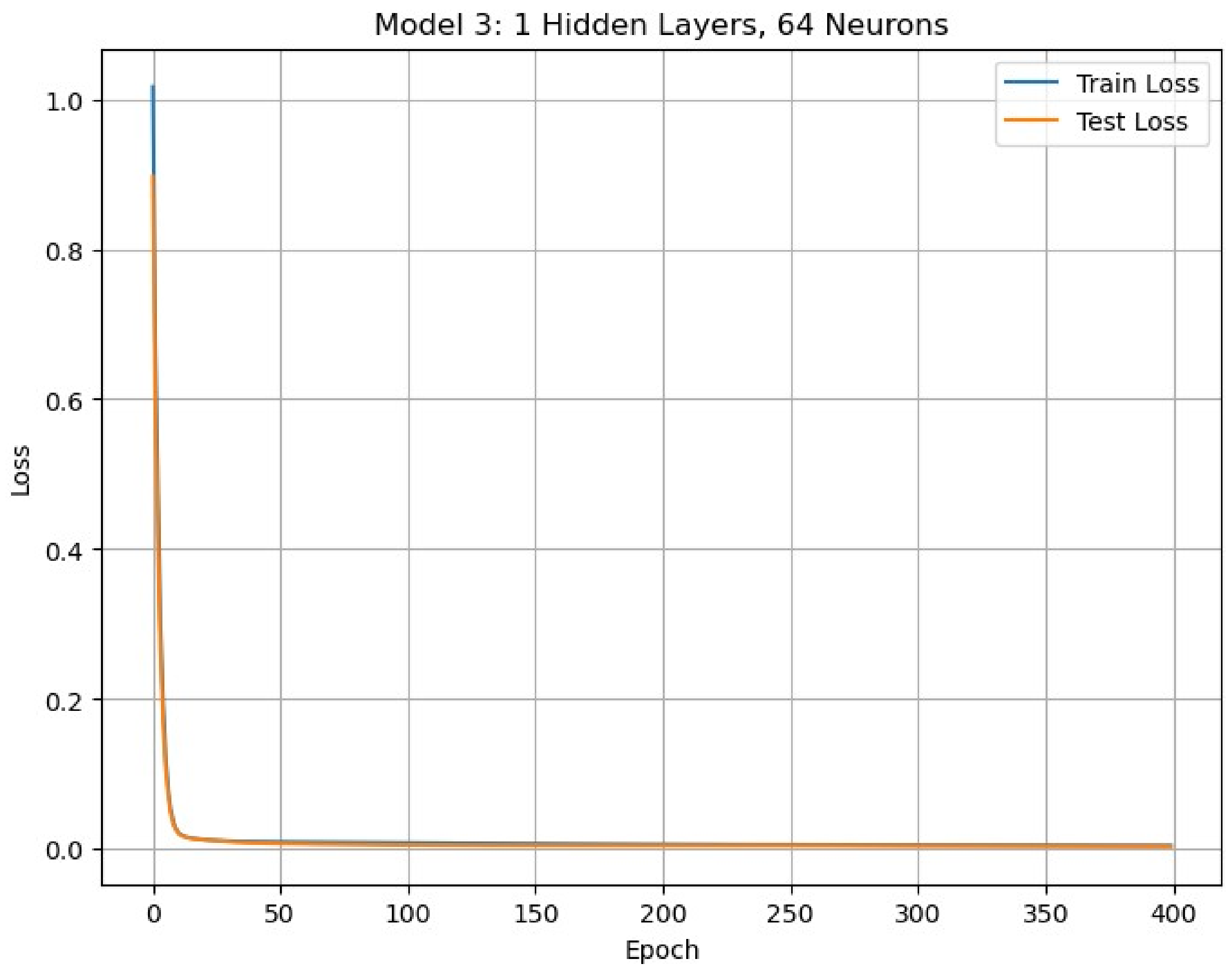
```
0.021872150478884578
Epoch 150, Train Loss: 0.026902323588728905, Test Loss:
0.021868218434974555
Epoch 200, Train Loss: 0.026947304606437683, Test Loss:
0.021860392997041345
Epoch 250, Train Loss: 0.02696821093559265, Test Loss:
0.021859925240278244
Epoch 300, Train Loss: 0.026974279433488846, Test Loss:
0.021858484717085958
Epoch 350, Train Loss: 0.026981288567185402, Test Loss:
0.021858786698430777
Epoch 0, Train Loss: 1.0164848566055298, Test Loss: 0.8969292938709259
Epoch 50, Train Loss: 0.009616399183869362, Test Loss:
0.007627377170138061
Epoch 100, Train Loss: 0.008318718522787094, Test Loss:
0.005719899199903011
Epoch 150, Train Loss: 0.007492950651794672, Test Loss:
0.005141673900652677
Epoch 200, Train Loss: 0.006459412164986134, Test Loss:
0.0051222327165305614
Epoch 250, Train Loss: 0.005756204016506672, Test Loss:
0.00505948078352958
Epoch 300, Train Loss: 0.005308591760694981, Test Loss:
0.004682371974922717
Epoch 350, Train Loss: 0.004956355784088373, Test Loss:
0.004287457326427102

# ################## Part 6: Plot Results ##################

# Plot for Model 1
plt.figure(figsize=(8, 6))
plt.plot(train_loss1, label='Train Loss')
plt.plot(test_loss1, label='Test Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.title('Model 1: 1 Hidden Layer, 16 Neurons')
plt.grid(True)
plt.show()

# Plot for Model 2
plt.figure(figsize=(8, 6))
plt.plot(train_loss2, label='Train Loss')
plt.plot(test_loss2, label='Test Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.title('Model 2: 1 Hidden Layers, 32 Neurons')
plt.grid(True)
plt.show()
```

```
# Plot for Model 3
plt.figure(figsize=(8, 6))
plt.plot(train_loss3, label='Train Loss')
plt.plot(test_loss3, label='Test Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.title('Model 3: 1 Hidden Layers, 64 Neurons')
plt.grid(True)
plt.show()

# ################# Part 7: Plot Predictions #################

# Generate test data for prediction (true data and predictions)
x_test = torch.unsqueeze(torch.linspace(0, 1, 1999), dim=1)

# Predictions for Model 1
y_pred1 = model1(x_test).detach().numpy()

# Predictions for Model 2
y_pred2 = model2(x_test).detach().numpy()

# Predictions for Model 3
y_pred3 = model3(x_test).detach().numpy()

# True data (based on original function f(x))
y_true = 0.2 + 0.4 * torch.pow(x_test, 2) + 0.3 * x_test *
torch.sin(15 * x_test) + 0.05 * torch.cos(50 * x_test)

# Plot for Model 1 predictions vs true data
plt.figure(figsize=(8, 6))
plt.plot(x_test, y_true, 'bo', label='True Data')
plt.plot(x_test, y_pred1, 'r', label='Model 1 Prediction')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.title('True Data vs Model 1 Prediction')
plt.grid(True)
plt.show()

# Plot for Model 2 predictions vs true data
plt.figure(figsize=(8, 6))
plt.plot(x_test, y_true, 'bo', label='True Data')
plt.plot(x_test, y_pred2, 'r', label='Model 2 Prediction')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.title('True Data vs Model 2 Prediction')
plt.grid(True)
```

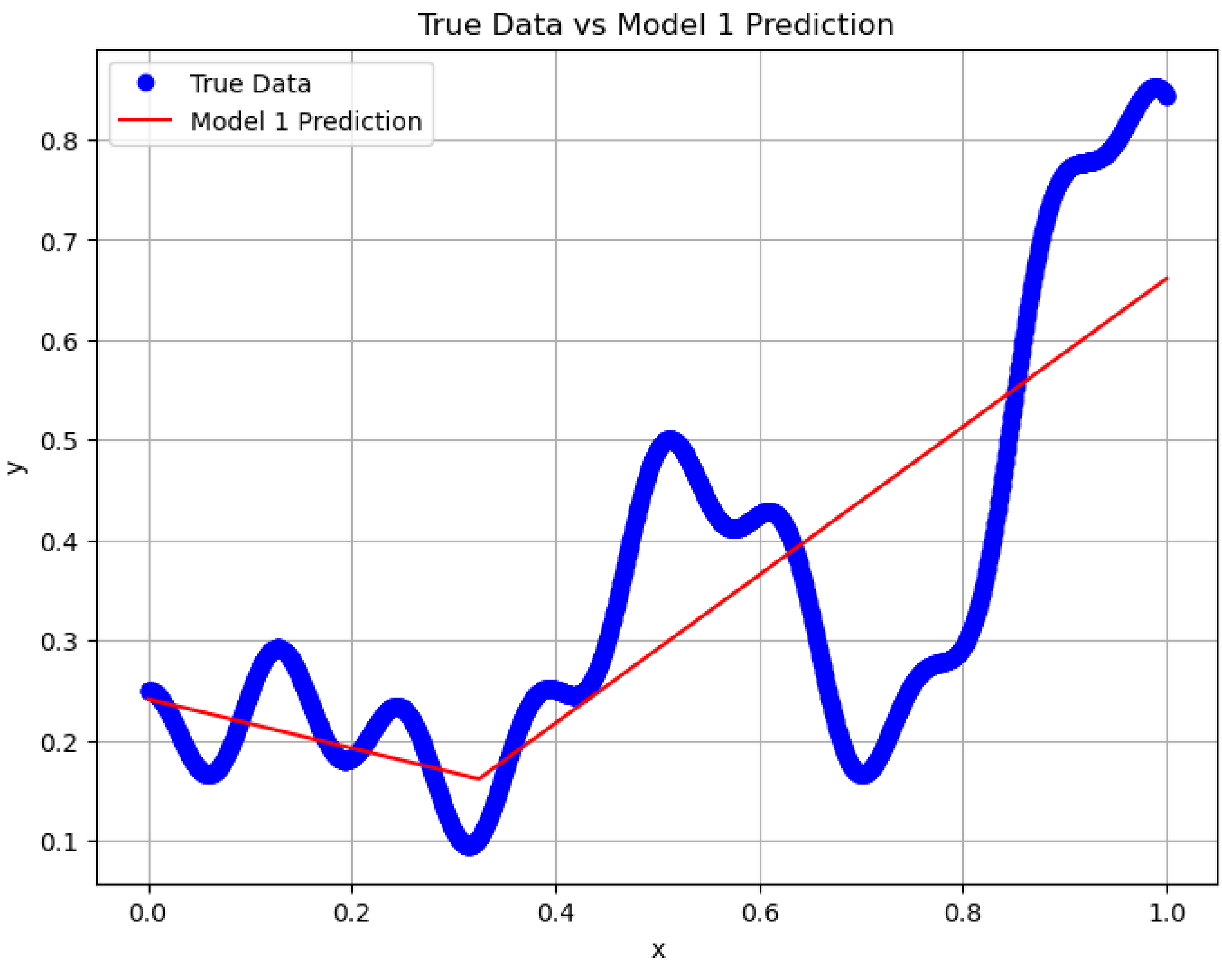
```
plt.show()

# Plot for Model 3 predictions vs true data
plt.figure(figsize=(8, 6))
plt.plot(x_test, y_true, 'bo', label='True Data')
plt.plot(x_test, y_pred3, 'r', label='Model 3 Prediction')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.title('True Data vs Model 3 Prediction')
plt.grid(True)
plt.show()
```

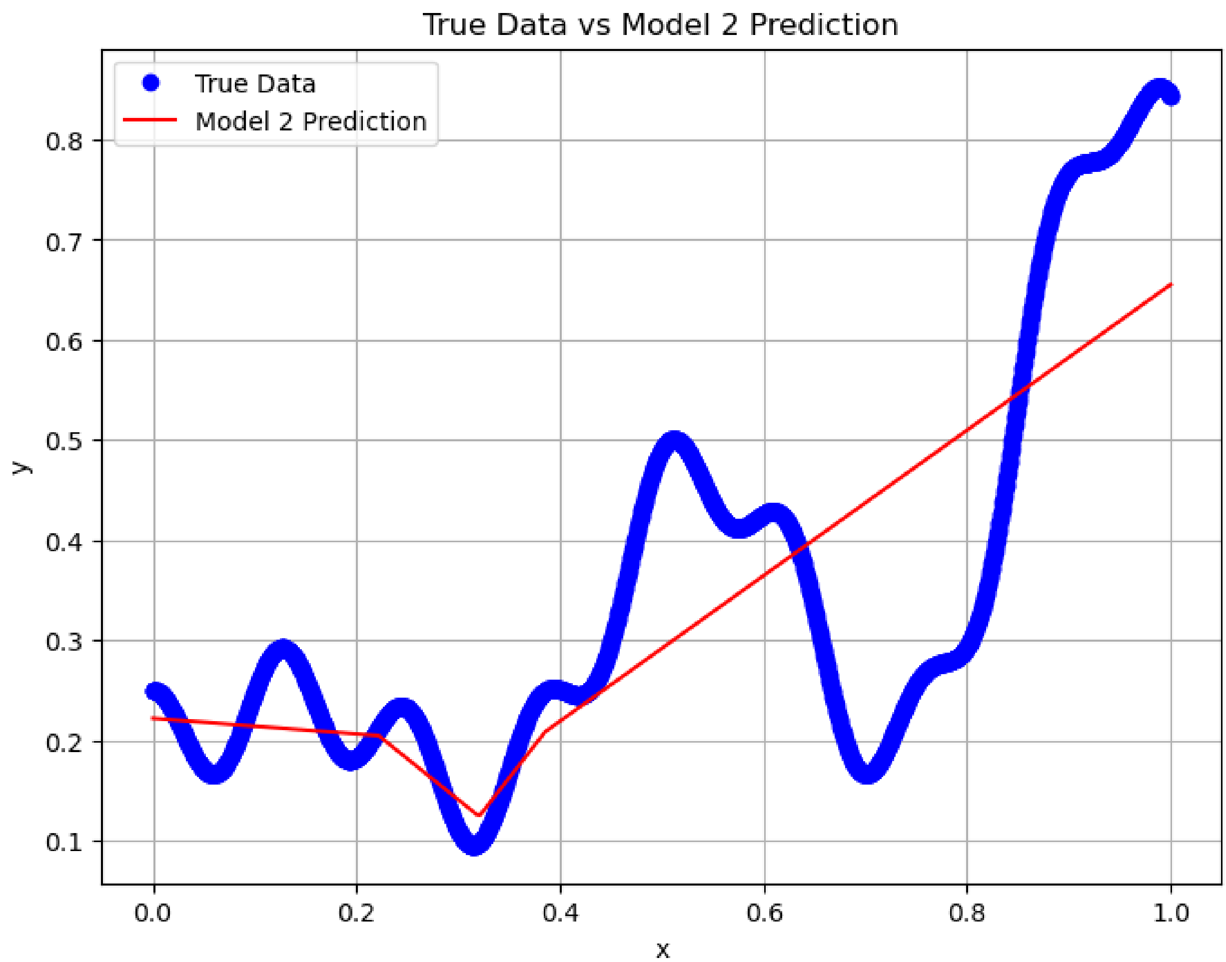


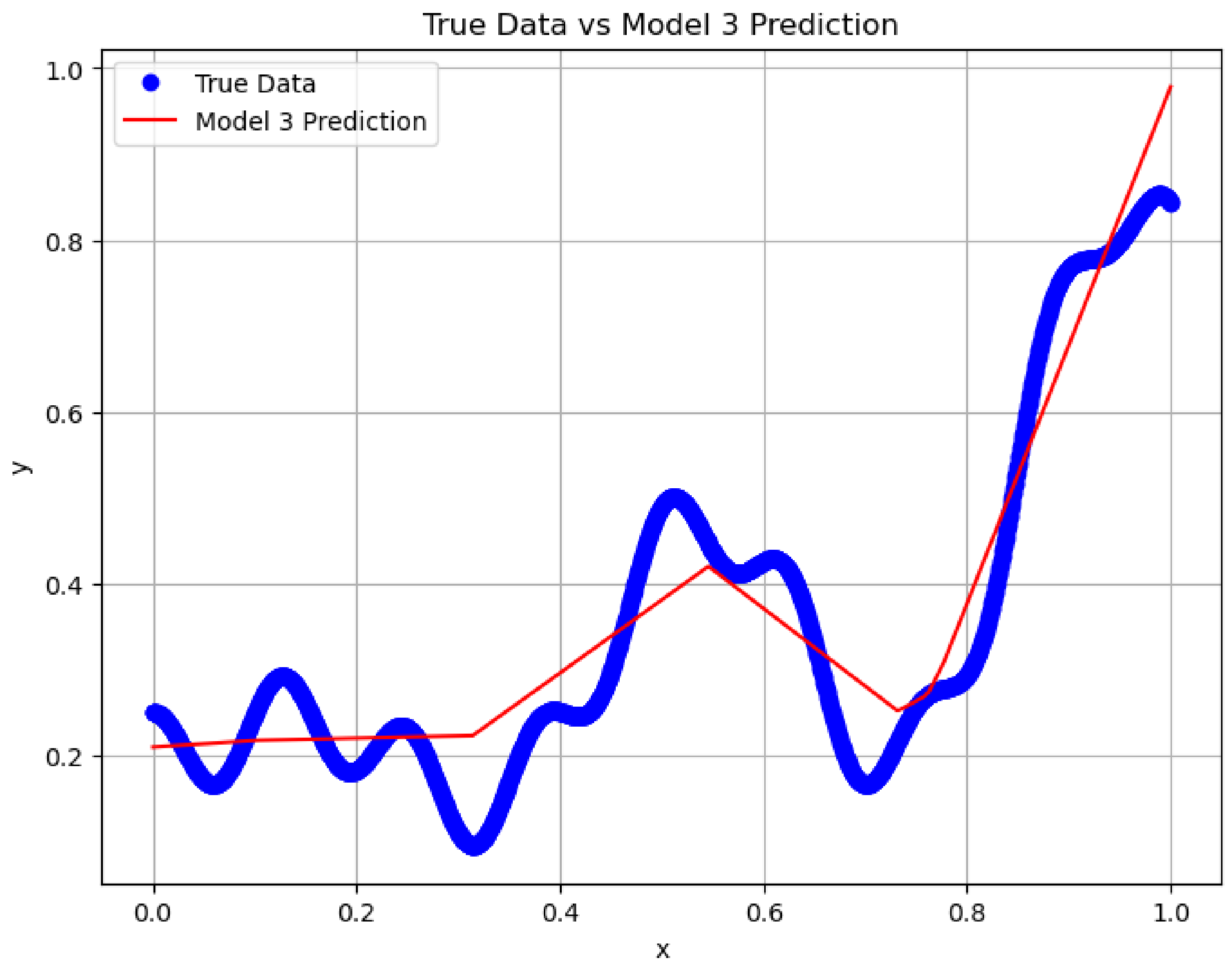

Model 2: 1 Hidden Layers, 32 Neurons
Train Loss
Test Loss
0.35
0.30
0.25
0.20
0.15
0.10
0.05
Loss
0
50
100
150
200
250
300
350
400
Epoch



Model 3: 1 Hidden Layers, 64 Neurons
Train Loss
Test Loss
Loss
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
300
350
400
Epoch


True Data vs Model 1 Prediction
True Data
Model 1 Prediction
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
y
0.0
0.2
0.4
0.6
0.8
1.0
x


True Data vs Model 2 Prediction
True Data
Model 2 Prediction
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
y
0.0
0.2
0.4
0.6
0.8
1.0
x


True Data vs Model 3 Prediction
True Data
Model 3 Prediction
y
x
0.0
0.2
0.4
0.6
0.8
1.0