Lab-3 Assignment: Neural Network

Submitted By: Abhaya Shrestha
Roll No.: ACE079BCT005

Objective

Using the provided dataset `circles_binary_classification.csv` , we will reproduce and extend the workflow. The main objective of this lab is to build, train, evaluate and compare PyTorch ANNs to classify the circular data and report findings.

Theory

Neural Network

Neural networks are machine learning models that mimic the complex functions of the human brain. These models consist of interconnected nodes or neurons that process data, learn patterns and enable tasks such as pattern recognition and decision-making.

Neural networks are capable of learning and identifying patterns directly from data without pre-defined rules. These networks are built from several key components:

- **Neurons:** The basic units that receive inputs, each neuron is governed by a threshold and an activation function.
- **Connections:** Links between neurons that carry information, regulated by weights and biases.
- **Weights and Biases:** These parameters determine the strength and influence of connections.
- **Propagation Functions:** Mechanisms that help process and transfer data across layers of neurons.
- **Learning Rule:** The method that adjusts weights and biases over time to improve accuracy.

Layers in Neural Network Architecture

1. **Input Layer:** This is where the network receives its input data. Each input neuron in the layer corresponds to a feature in the input data.
2. **Hidden Layers:** These layers perform most of the computational heavy lifting. A neural network can have one or multiple hidden layers. Each layer consists of units (neurons) that transform the inputs into something that the output layer can use.
3. **Output Layer:** The final layer produces the output of the model. The format of these outputs varies depending on the specific task like classification, regression.

Tasks

Data Retreival and Collection

In [1]:
```
import pandas as pd

data = pd.read_csv('circles_binary_classification.csv')
data.head(6)
```

Out[1]:

	X1	X2	label
0	0.754246	0.231481	1
1	-0.756159	0.153259	1
2	-0.815392	0.173282	1
3	-0.393731	0.692883	1
4	0.442208	-0.896723	0
5	-0.479646	0.676435	1

Data Cleaning & Feature Design

In [2]:
```
data.describe()
```

Out[2]:

	X1	X2	label
count	1000.000000	1000.000000	1000.00000
mean	-0.000448	-0.000804	0.50000
std	0.639837	0.641156	0.50025
min	-1.059502	-1.067768	0.00000
25%	-0.619251	-0.612176	0.00000
50%	0.008762	-0.003949	0.50000
75%	0.621933	0.624822	1.00000
max	1.033712	1.036004	1.00000

In [3]:
```
# vectorizing X1 and X2 into a single tensor/nparray

X = data[['X1', 'X2']].values
y = data['label'].values
```

In [4]:
```
print(f"First 5 X features:\n{X[:5]}")
print(f"\nFirst 5 y labels:\n{y[:5]}")
```

```
First 5 X features:
[[ 0.75424625  0.23148074]
 [-0.75615888  0.15325888]
 [-0.81539193  0.17328203]
 [-0.39373073  0.69288277]
 [ 0.44220765 -0.89672343]]

First 5 y labels:
[1 1 1 1 0]
```

In [5]:
```
# Check the shapes of our features and labels
X.shape, y.shape
```

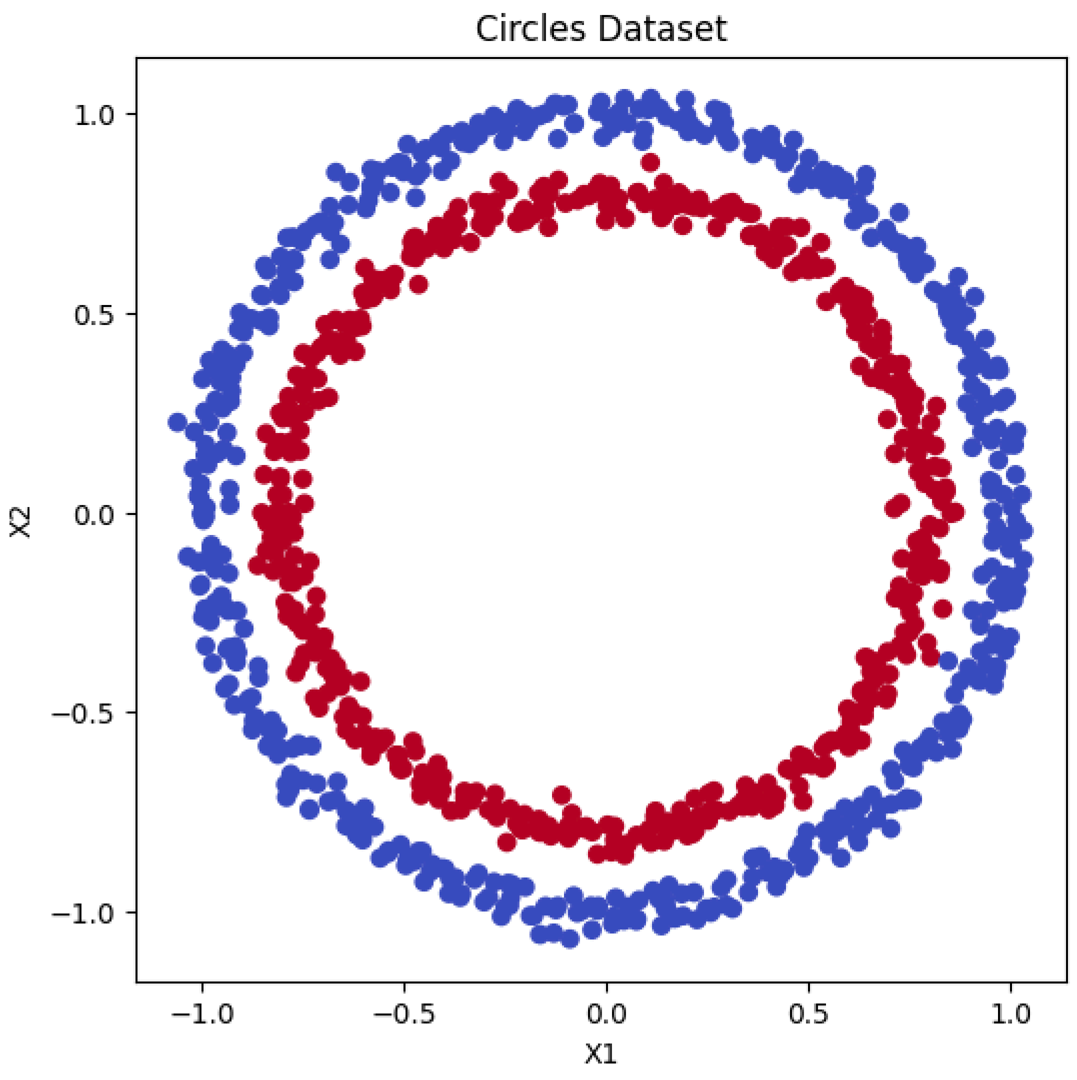
Out[5]:
```
((1000, 2), (1000,))
```

In [6]:
```
# Turn data into tensors
# Otherwise this causes issues with computations later on
import torch

X = torch.tensor(X, dtype=torch.float32)
y = torch.tensor(y, dtype=torch.float32)
y = y.unsqueeze(1)

# View the first five samples
X[:5], y[:5]
```

Out[6]:
```
(tensor([[ 0.7542,  0.2315],
         [-0.7562,  0.1533],
         [-0.8154,  0.1733],
         [-0.3937,  0.6929],
         [ 0.4422, -0.8967]]),
 tensor([[1.],
         [1.],
         [1.],
         [1.],
         [0.]]))
```

Data Visualization

In [7]:
```
# Visualize with a plot
import matplotlib.pyplot as plt
plt.figure(figsize=(6,6))
plt.scatter(X[:,0], X[:,1], c=y.squeeze(), cmap='coolwarm')
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Circles Dataset')
plt.show()
```



Train/Test Split

```
In [8]: # Split data into train and test sets
        from sklearn.model_selection import train_test_split

        X_train, X_test, y_train, y_test = train_test_split(
            X, y, test_size=0.2, random_state=42
        )
        len(X_train), len(X_test), len(y_train), len(y_test)

Out[8]: (800, 200, 800, 200)
```

Device Configuration

```
In [9]: # Standard PyTorch imports
        import torch
        from torch import nn

        # Make device agnostic code
        device = "cuda" if torch.cuda.is_available() else "cpu"
        device

Out[9]: 'cpu'
```

Model Implementation

Model 0

In [10]:
```
# 1. Construct a model class that subclasses nn.Module
class ModelV0(nn.Module):
    def __init__(self):
        super().__init__()
        # 2. Create 2 nn.Linear layers capable of handling X and y input and out
        self.layer_1 = nn.Linear(in_features=2, out_features=5) # takes in 2 fea
        self.layer_2 = nn.Linear(in_features=5, out_features=1) # takes in 5 fea

    # 3. Define a forward method containing the forward pass computation
    def forward(self, x):
        # Return the output of layer_2, a single feature, the same shape as y
        return self.layer_2(self.layer_1(x)) # computation goes through layer_1

# 4. Create an instance of the model and send it to target device
model_0 = ModelV0().to(device)
model_0
```

Out[10]:
```
ModelV0(
  (layer_1): Linear(in_features=2, out_features=5, bias=True)
  (layer_2): Linear(in_features=5, out_features=1, bias=True)
)
```

In [11]:
```
model = model_0
```

In [12]:
```
# Make predictions with the model
untrained_preds = model(X_test.to(device))
print(f"Length of predictions: {len(untrained_preds)}, Shape: {untrained_preds.s
print(f"Length of test samples: {len(y_test)}, Shape: {y_test.shape}")
print(f"\nFirst 10 predictions:\n{untrained_preds[:10]}")
print(f"\nFirst 10 test labels:\n{y_test[:10]}")
```

```
Length of predictions: 200, Shape: torch.Size([200, 1])
Length of test samples: 200, Shape: torch.Size([200, 1])

First 10 predictions:
tensor([[-0.0175],
        [ 0.1085],
        [-0.2457],
        [ 0.0468],
        [-0.0837],
        [-0.0081],
        [ 0.2183],
        [ 0.1914],
        [-0.2522],
        [ 0.1187]], grad_fn=<SliceBackward0>)

First 10 test labels:
tensor([[1.],
        [0.],
        [1.],
        [0.],
        [1.],
        [1.],
        [0.],
        [0.],
        [1.],
        [0.]])
```

Loss Function Selection

In [13]:
```
# Create a loss function
# loss_fn = nn.BCELoss() # BCELoss = no sigmoid built-in
loss_fn = nn.BCEWithLogitsLoss() # BCEWithLogitsLoss = sigmoid built-in

# Create an optimizer
optimizer = torch.optim.SGD(params=model.parameters(),
                            lr=0.1)
```

In [14]:
```
# Calculate accuracy (a classification metric)
def accuracy_fn(y_true, y_pred):
    correct = torch.eq(y_true, y_pred).sum().item() # torch.eq() calculates wher
    acc = (correct / len(y_pred)) * 100
    return acc
```

In [15]:
```
# View the frist 5 outputs of the forward pass on the test data
y_logits = model(X_test.to(device))[:5]
y_logits
```

Out[15]:
```
tensor([[-0.0175],
        [ 0.1085],
        [-0.2457],
        [ 0.0468],
        [-0.0837]], grad_fn=<SliceBackward0>)
```

In [16]:
```
# Use sigmoid on model logits
y_pred_probs = torch.sigmoid(y_logits)
y_pred_probs
```

```
Out[16]: tensor([[0.4956],
                 [0.5271],
                 [0.4389],
                 [0.5117],
                 [0.4791]], grad_fn=<SigmoidBackward0>)
```

```
In [17]: # Find the predicted labels (round the prediction probabilities)
         y_preds = torch.round(y_pred_probs)

         # In full
         y_pred_labels = torch.round(torch.sigmoid(model(X_test.to(device))[:5]))

         # Check for equality
         print(torch.eq(y_preds.squeeze(), y_pred_labels.squeeze()))

         # Get rid of extra dimension
         y_preds.squeeze()
```

```
tensor([True, True, True, True, True])
```

```
Out[17]: tensor([0., 1., 0., 1., 0.], grad_fn=<SqueezeBackward0>)
```

```
In [18]: y_test[:5]
```

```
Out[18]: tensor([[1.],
                 [0.],
                 [1.],
                 [0.],
                 [1.]])
```

Model Learning

```
In [19]: torch.manual_seed(42)

         # Set the number of epochs
         epochs = 100

         # Put data to target device
         X_train, y_train = X_train.to(device), y_train.to(device)
         X_test, y_test = X_test.to(device), y_test.to(device)

         # Ensure targets have shape (N, 1) to match model outputs
         y_train = y_train.reshape(-1, 1)
         y_test = y_test.reshape(-1, 1)

         def train_and_test_loop(
             model: nn.Module,
             epochs: int,
             X_train: torch.Tensor,
             y_train: torch.Tensor,
             X_test: torch.Tensor,
             y_test: torch.Tensor,
             loss_fn: nn.Module,
             optimizer: torch.optim.Optimizer
         ):
             # lists
             loss_list = []
             acc_list = []
             test_losses = []
```

```
    test_acc_list = []



    # Build training and evaluation loop
    for epoch in range(epochs):
        ### Training
        model.train()

        # 1. Forward pass (model outputs raw logits)
        y_logits = model(X_train) # keep shape (N,1) so it matches `y_train`
        y_pred = torch.round(torch.sigmoid(y_logits)) # turn logits -> pred prob

        # 2. Calculate loss/accuracy
        # loss = loss_fn(torch.sigmoid(y_logits), # Using nn.BCELoss you need to
        #                y_train)
        loss = loss_fn(y_logits, # Using nn.BCEWithLogitsLoss works with raw log
                       y_train)
        acc = accuracy_fn(y_true=y_train,
                          y_pred=y_pred)

        # 3. Optimizer zero grad
        optimizer.zero_grad()

        # 4. Loss backwards
        loss.backward()

        # 5. Optimizer step
        optimizer.step()

        ### Testing
        model.eval()
        with torch.inference_mode():
            # 1. Forward pass
            test_logits = model(X_test)
            test_pred = torch.round(torch.sigmoid(test_logits))
            # 2. Caculate loss/accuracy
            test_loss = loss_fn(test_logits,
                                y_test)
            test_acc = accuracy_fn(y_true=y_test,
                                   y_pred=test_pred)


        loss_list.append(loss.item())
        acc_list.append(acc)
        test_losses.append(test_loss.item())
        test_acc_list.append(test_acc)

        # Print out what's happening every 10 epochs
        if epoch % 10 == 0:
            print(f"Epoch: {epoch} | Loss: {loss:.5f}, Accuracy: {acc:.2f}% | Te

    return loss_list, acc_list, test_losses, test_acc_list
```

Model Evaluation

In [20]:
```
train_losses, acc_list, test_losses, test_acc = train_and_test_loop(
    model=model,
    epochs=epochs,
    X_train=X_train,
    y_train=y_train,
    X_test=X_test,
    y_test=y_test,
    loss_fn=loss_fn,
    optimizer=optimizer
)
```

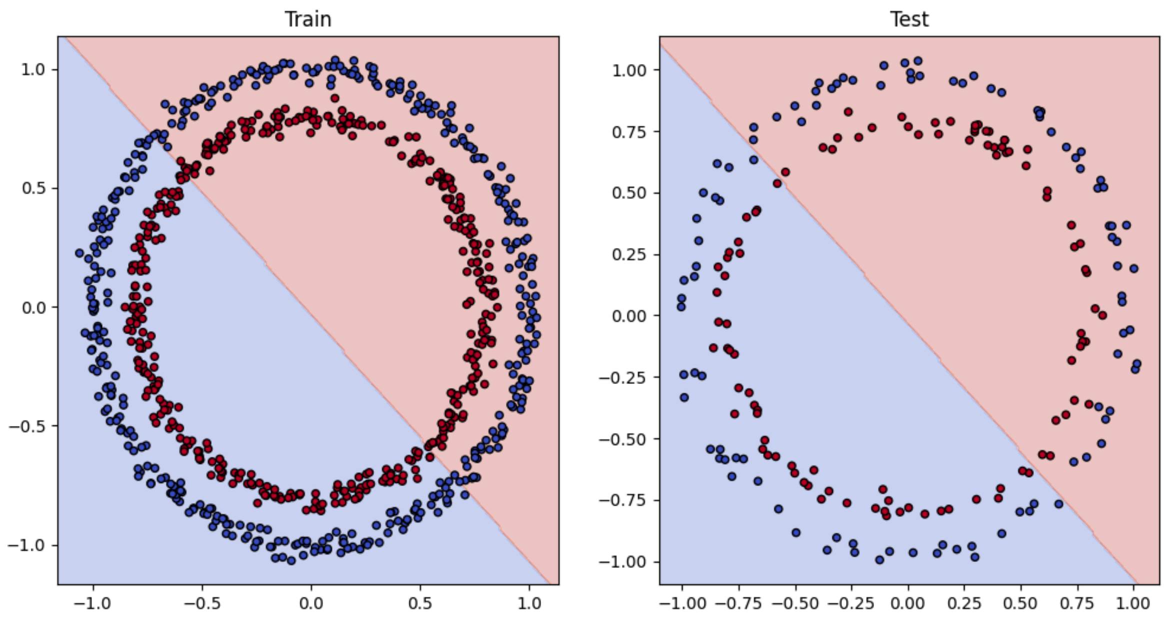
```
Epoch: 0 | Loss: 0.69563, Accuracy: 50.00% | Test loss: 0.70248, Test acc: 47.00%
Epoch: 10 | Loss: 0.69500, Accuracy: 50.25% | Test loss: 0.70147, Test acc: 48.0
0%
Epoch: 20 | Loss: 0.69462, Accuracy: 50.50% | Test loss: 0.70072, Test acc: 47.5
0%
Epoch: 30 | Loss: 0.69435, Accuracy: 50.88% | Test loss: 0.70010, Test acc: 47.5
0%
Epoch: 40 | Loss: 0.69413, Accuracy: 50.75% | Test loss: 0.69957, Test acc: 47.5
0%
Epoch: 50 | Loss: 0.69396, Accuracy: 50.62% | Test loss: 0.69911, Test acc: 47.5
0%
Epoch: 60 | Loss: 0.69381, Accuracy: 50.88% | Test loss: 0.69870, Test acc: 47.0
0%
Epoch: 70 | Loss: 0.69369, Accuracy: 51.00% | Test loss: 0.69834, Test acc: 47.0
0%
Epoch: 80 | Loss: 0.69358, Accuracy: 50.88% | Test loss: 0.69801, Test acc: 47.5
0%
Epoch: 90 | Loss: 0.69350, Accuracy: 51.00% | Test loss: 0.69772, Test acc: 47.0
0%
```

In [21]:

```
import numpy as np
import torch
import matplotlib.pyplot as plt


def plot_decision_boundary(model, X, y, device=None, cmap='coolwarm'):

    model_device = next(model.parameters()).device if any(p.requires_grad for p
    use_device = torch.device(device) if device is not None else model_device

    # Convert inputs to numpy
    if isinstance(X, torch.Tensor):
        X_np = X.detach().cpu().numpy()
    else:
        X_np = np.asarray(X)

    if isinstance(y, torch.Tensor):
        y_np = y.detach().cpu().squeeze().numpy()
    else:
        y_np = np.asarray(y).squeeze()

    x_min, x_max = X_np[:, 0].min() - 0.1, X_np[:, 0].max() + 0.1
    y_min, y_max = X_np[:, 1].min() - 0.1, X_np[:, 1].max() + 0.1

    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 200), np.linspace(y_min, y_ma
    grid = np.c_[xx.ravel(), yy.ravel()]

    # Run model predictions on CPU for plotting
    model_cpu = model.to('cpu')
    model_cpu.eval()

    grid_t = torch.tensor(grid, dtype=torch.float32)
    with torch.inference_mode():
        logits = model_cpu(grid_t).squeeze()
        probs = torch.sigmoid(logits).numpy()

    Z = (probs > 0.5).astype(int).reshape(xx.shape)

    plt.contourf(xx, yy, Z, alpha=0.3, cmap=cmap)
    plt.scatter(X_np[:, 0], X_np[:, 1], c=y_np, s=20, cmap=cmap, edgecolor='k')
    plt.xlim(x_min, x_max)
    plt.ylim(y_min, y_max)
```

In [22]:

```
# Plot decision boundaries for training and test sets
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.title("Train")
plot_decision_boundary(model, X_train, y_train)
plt.subplot(1, 2, 2)
plt.title("Test")
plot_decision_boundary(model, X_test, y_test)
```



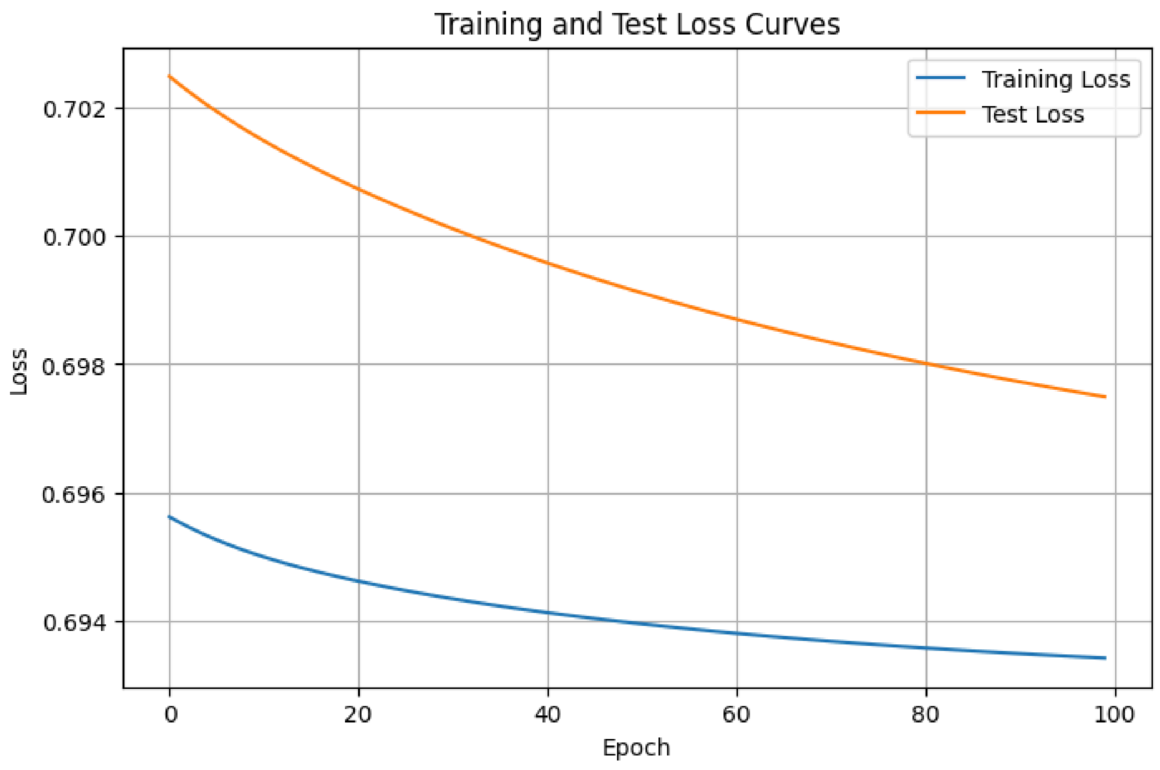
In machine learning terms, our model is `underfitting`, meaning it's not learning predictive patterns from the data.

Let's also plot the loss graph:

In [23]:

```
import matplotlib.pyplot as plt

def plot_loss_curves(train_losses, test_losses):

    epochs = range(len(train_losses))

    plt.figure(figsize=(8, 5))
    plt.plot(epochs, train_losses, label="Training Loss")
    plt.plot(epochs, test_losses, label="Test Loss")

    plt.xlabel("Epoch")
    plt.ylabel("Loss")
    plt.title("Training and Test Loss Curves")
    plt.legend()
    plt.grid(True)
    plt.show()
```

In [24]:
```
plot_loss_curves(train_losses, test_losses)
```



Improving the Model

To increase the model’s learning capacity, we’ll make the following changes:

- Add an extra hidden layer
- Increase the number of hidden units from **5** → **15**
- Train for longer, increasing epochs from **100** → **1000**

We’ll follow the same modeling steps as before, but with these updated hyperparameters, and then evaluate whether these changes help reduce underfitting and improve performance.

Model 1

In [25]:
```
class ModelV1(nn.Module):
    def __init__(self):
        super().__init__()
        self.layer_1 = nn.Linear(in_features=2, out_features=15)
        self.layer_2 = nn.Linear(in_features=15, out_features=15)
        self.layer_3 = nn.Linear(in_features=15, out_features=1)

    def forward(self, x):
        return self.layer_3(self.layer_2(self.layer_1(x)))

model_1 = ModelV1().to(device)
model_1
```

Out[25]:
```
ModelV1(
  (layer_1): Linear(in_features=2, out_features=15, bias=True)
  (layer_2): Linear(in_features=15, out_features=15, bias=True)
  (layer_3): Linear(in_features=15, out_features=1, bias=True)
)
```

In [26]:
```
model = model_1
epochs = 1000

# Create a loss function
loss_fn = nn.BCEWithLogitsLoss() # BCEWithLogitsLoss = sigmoid built-in

# Create an optimizer
optimizer = torch.optim.SGD(params=model.parameters(),
                            lr=0.1)
```

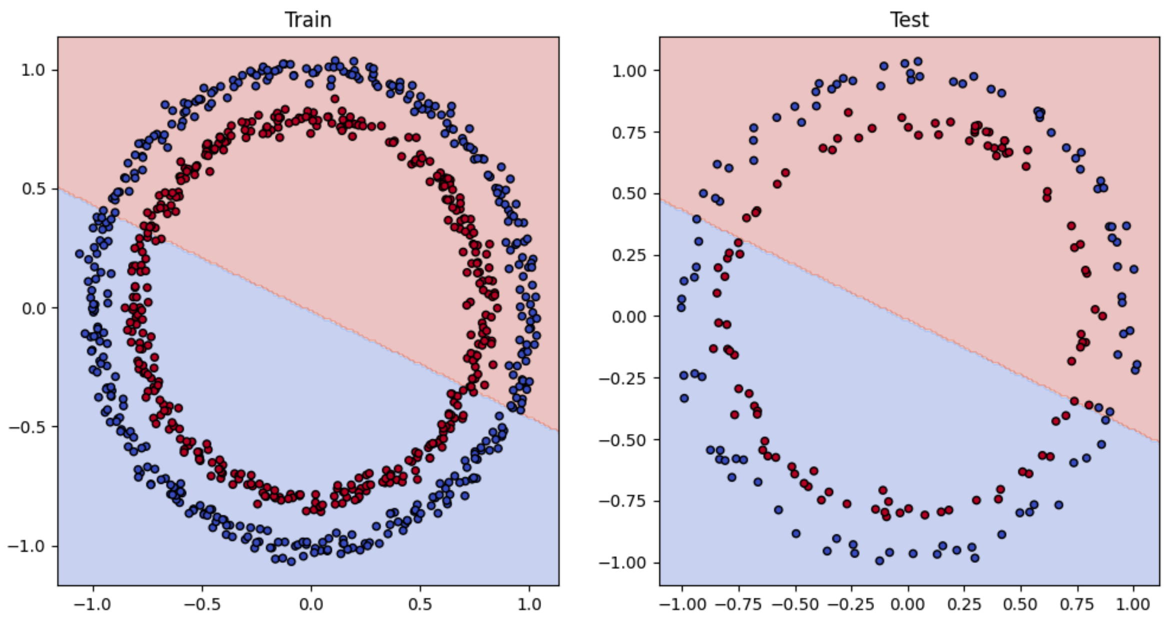
In [27]:
```
train_losses, acc_list, test_losses, test_acc = train_and_test_loop(
    model=model,
    epochs=epochs,
    X_train=X_train,
    y_train=y_train,
    X_test=X_test,
    y_test=y_test,
    loss_fn=loss_fn,
    optimizer=optimizer
)

# Plot decision boundaries for training and test sets
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.title("Train")
plot_decision_boundary(model, X_train, y_train)
plt.subplot(1, 2, 2)
plt.title("Test")
plot_decision_boundary(model, X_test, y_test)
```

```
Epoch: 0 | Loss: 0.69588, Accuracy: 50.00% | Test loss: 0.69540, Test acc: 50.00%
Epoch: 10 | Loss: 0.69357, Accuracy: 50.00% | Test loss: 0.69407, Test acc: 50.0
0%
Epoch: 20 | Loss: 0.69313, Accuracy: 55.38% | Test loss: 0.69402, Test acc: 51.5
0%
Epoch: 30 | Loss: 0.69304, Accuracy: 52.00% | Test loss: 0.69413, Test acc: 49.0
0%
Epoch: 40 | Loss: 0.69301, Accuracy: 51.38% | Test loss: 0.69424, Test acc: 48.0
0%
Epoch: 50 | Loss: 0.69300, Accuracy: 51.12% | Test loss: 0.69432, Test acc: 47.0
0%
Epoch: 60 | Loss: 0.69299, Accuracy: 50.88% | Test loss: 0.69439, Test acc: 47.0
0%
Epoch: 70 | Loss: 0.69299, Accuracy: 51.00% | Test loss: 0.69444, Test acc: 47.0
0%
Epoch: 80 | Loss: 0.69299, Accuracy: 51.12% | Test loss: 0.69449, Test acc: 47.0
0%
Epoch: 90 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69452, Test acc: 47.0
0%
Epoch: 100 | Loss: 0.69298, Accuracy: 51.25% | Test loss: 0.69455, Test acc: 46.5
0%
Epoch: 110 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69458, Test acc: 46.5
0%
Epoch: 120 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69460, Test acc: 46.0
0%
Epoch: 130 | Loss: 0.69298, Accuracy: 51.25% | Test loss: 0.69461, Test acc: 46.0
0%
Epoch: 140 | Loss: 0.69298, Accuracy: 51.38% | Test loss: 0.69462, Test acc: 46.0
0%
Epoch: 150 | Loss: 0.69298, Accuracy: 51.38% | Test loss: 0.69463, Test acc: 46.0
0%
Epoch: 160 | Loss: 0.69298, Accuracy: 51.50% | Test loss: 0.69464, Test acc: 46.0
0%
Epoch: 170 | Loss: 0.69298, Accuracy: 51.38% | Test loss: 0.69465, Test acc: 45.5
0%
Epoch: 180 | Loss: 0.69298, Accuracy: 51.38% | Test loss: 0.69466, Test acc: 44.5
0%
Epoch: 190 | Loss: 0.69298, Accuracy: 51.50% | Test loss: 0.69466, Test acc: 44.5
0%
Epoch: 200 | Loss: 0.69298, Accuracy: 51.62% | Test loss: 0.69466, Test acc: 45.0
0%
Epoch: 210 | Loss: 0.69298, Accuracy: 51.38% | Test loss: 0.69467, Test acc: 45.0
0%
Epoch: 220 | Loss: 0.69298, Accuracy: 51.38% | Test loss: 0.69467, Test acc: 45.0
0%
Epoch: 230 | Loss: 0.69298, Accuracy: 51.38% | Test loss: 0.69467, Test acc: 45.0
0%
Epoch: 240 | Loss: 0.69298, Accuracy: 51.25% | Test loss: 0.69467, Test acc: 45.0
0%
Epoch: 250 | Loss: 0.69298, Accuracy: 51.25% | Test loss: 0.69467, Test acc: 45.0
0%
Epoch: 260 | Loss: 0.69298, Accuracy: 51.12% | Test loss: 0.69468, Test acc: 45.5
0%
Epoch: 270 | Loss: 0.69298, Accuracy: 51.12% | Test loss: 0.69468, Test acc: 45.5
0%
Epoch: 280 | Loss: 0.69298, Accuracy: 51.12% | Test loss: 0.69468, Test acc: 45.5
0%
Epoch: 290 | Loss: 0.69298, Accuracy: 51.12% | Test loss: 0.69468, Test acc: 45.5
0%
Epoch: 300 | Loss: 0.69298, Accuracy: 51.12% | Test loss: 0.69468, Test acc: 46.0
```

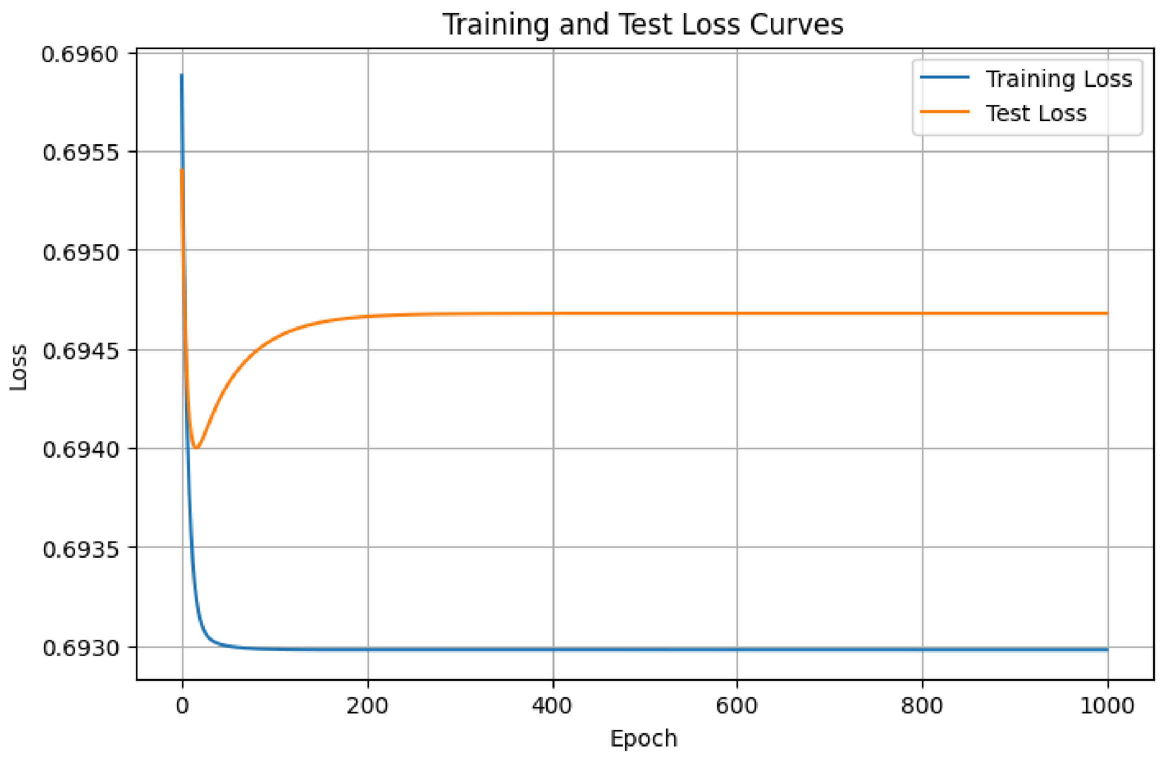
```
0%
Epoch: 310 | Loss: 0.69298, Accuracy: 51.12% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 320 | Loss: 0.69298, Accuracy: 51.12% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 330 | Loss: 0.69298, Accuracy: 51.12% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 340 | Loss: 0.69298, Accuracy: 51.12% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 350 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 360 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 370 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 380 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 390 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 400 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 410 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 420 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 430 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 440 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 450 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 460 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 470 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 480 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 490 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 500 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 510 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 520 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 530 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 540 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 550 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 560 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 570 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 580 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 590 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 600 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
```

```
0%
Epoch: 610 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 620 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 630 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 640 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 650 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 660 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 670 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 680 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 690 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 700 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 710 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 720 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 730 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 740 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 750 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 760 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 770 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 780 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 790 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 800 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 810 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 820 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 830 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 840 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 850 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 860 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 870 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 880 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 890 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 900 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
```

```
0%
Epoch: 910 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 920 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 930 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 940 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 950 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 960 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 970 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 980 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
Epoch: 990 | Loss: 0.69298, Accuracy: 51.00% | Test loss: 0.69468, Test acc: 46.0
0%
```



```
In [28]: plot_loss_curves(train_losses, test_losses)
```



Let's try adding activation functions:

Model 2

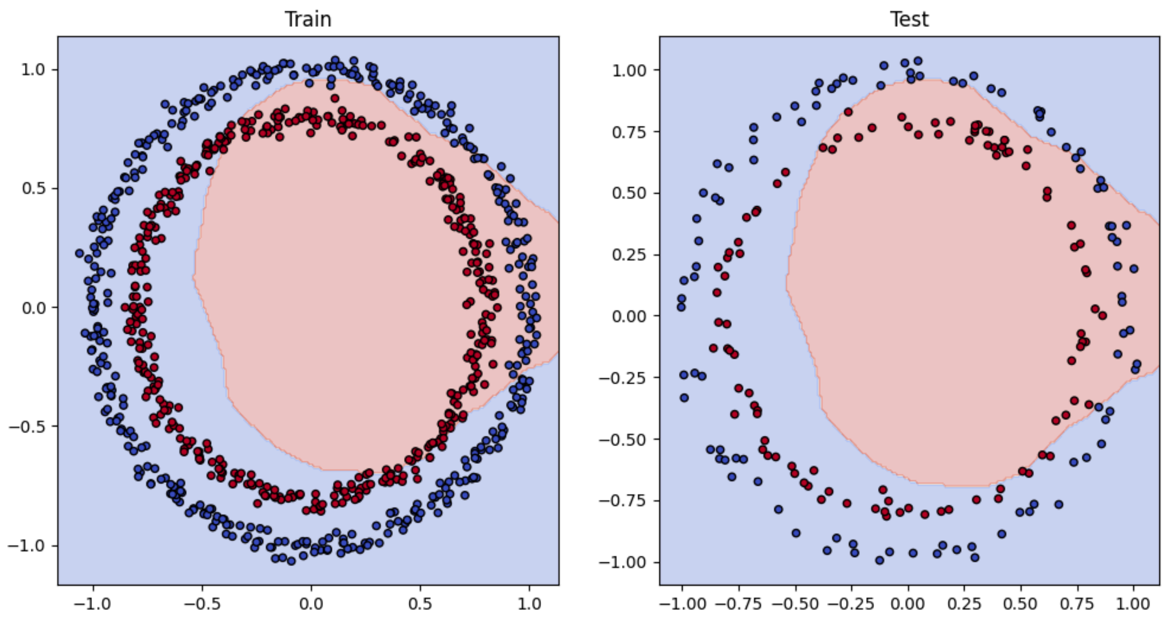
In [29]:
```python
# Build model with non-linear activation function
from torch import nn
class ModelV2(nn.Module):
    def __init__(self):
        super().__init__()
        self.layer_1 = nn.Linear(in_features=2, out_features=64)
        self.layer_2 = nn.Linear(in_features=64, out_features=64)
        self.layer_3 = nn.Linear(in_features=64, out_features=10)
        self.layer_4=nn.Linear(in_features=10,out_features=1)
        self.relu = nn.ReLU() # <- add in ReLU activation function
        # Can also put sigmoid in the model
        # This would mean you don't need to use it on the predictions
        # self.sigmoid = nn.Sigmoid()

    def forward(self, x):
      # Intersperse the ReLU activation function between layers
       return self.layer_4(self.relu(self.layer_3(self.relu(self.layer_2(self.re

model_2 = ModelV2().to(device)
print(model_2)
```

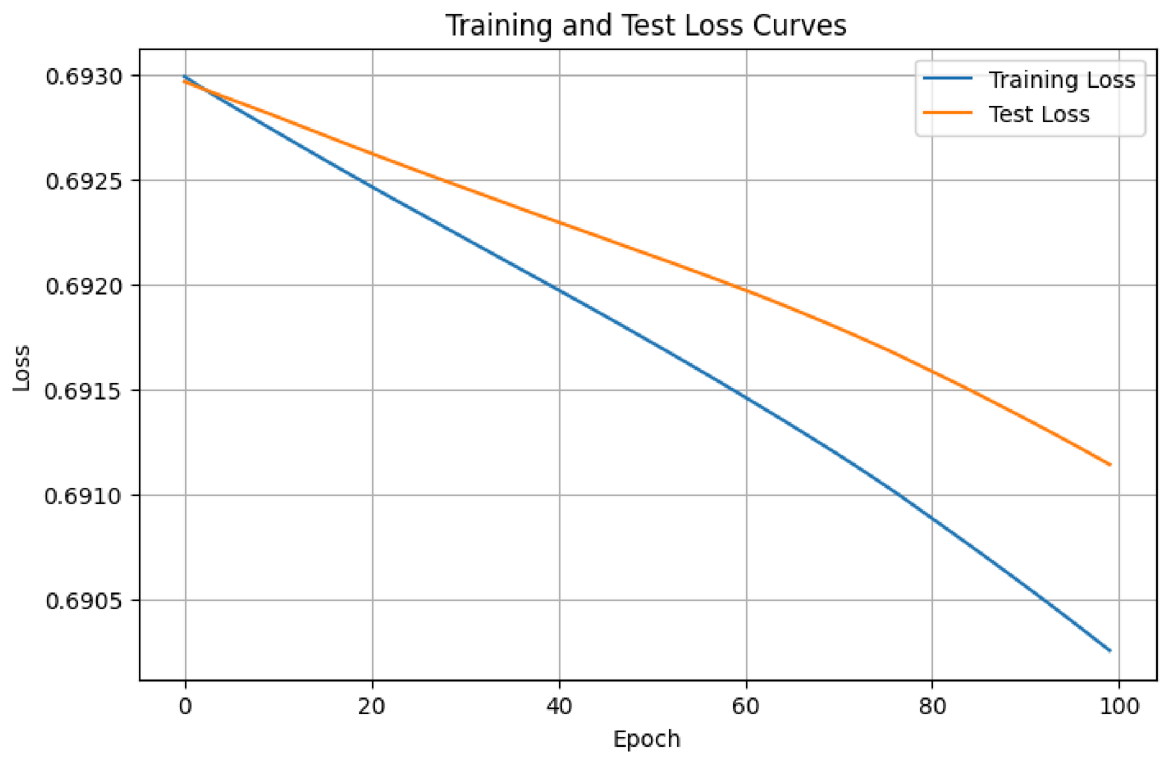
```
ModelV2(
  (layer_1): Linear(in_features=2, out_features=64, bias=True)
  (layer_2): Linear(in_features=64, out_features=64, bias=True)
  (layer_3): Linear(in_features=64, out_features=10, bias=True)
  (layer_4): Linear(in_features=10, out_features=1, bias=True)
  (relu): ReLU()
)
```

In [30]:
```python
model = model_2
epochs = 100
```

```python
# Create a loss function
loss_fn = nn.BCEWithLogitsLoss() # BCEWithLogitsLoss = sigmoid built-in

# Create an optimizer
optimizer = torch.optim.SGD(params=model.parameters(),
                            lr=0.1)

train_losses, acc_list, test_losses, test_acc = train_and_test_loop(
    model=model,
    epochs=epochs,
    X_train=X_train,
    y_train=y_train,
    X_test=X_test,
    y_test=y_test,
    loss_fn=loss_fn,
    optimizer=optimizer
)

# Plot decision boundaries for training and test sets
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.title("Train")
plot_decision_boundary(model, X_train, y_train)
plt.subplot(1, 2, 2)
plt.title("Test")
plot_decision_boundary(model, X_test, y_test)
plot_loss_curves(train_losses, test_losses)
```

```
Epoch: 0 | Loss: 0.69299, Accuracy: 47.75% | Test loss: 0.69297, Test acc: 47.00%
Epoch: 10 | Loss: 0.69273, Accuracy: 49.62% | Test loss: 0.69280, Test acc: 44.5
0%
Epoch: 20 | Loss: 0.69247, Accuracy: 52.25% | Test loss: 0.69263, Test acc: 48.5
0%
Epoch: 30 | Loss: 0.69222, Accuracy: 56.38% | Test loss: 0.69246, Test acc: 56.5
0%
Epoch: 40 | Loss: 0.69198, Accuracy: 61.38% | Test loss: 0.69230, Test acc: 60.0
0%
Epoch: 50 | Loss: 0.69173, Accuracy: 63.50% | Test loss: 0.69214, Test acc: 61.5
0%
Epoch: 60 | Loss: 0.69147, Accuracy: 63.75% | Test loss: 0.69198, Test acc: 62.0
0%
Epoch: 70 | Loss: 0.69119, Accuracy: 64.25% | Test loss: 0.69179, Test acc: 62.0
0%
Epoch: 80 | Loss: 0.69089, Accuracy: 64.38% | Test loss: 0.69159, Test acc: 62.0
0%
Epoch: 90 | Loss: 0.69057, Accuracy: 65.50% | Test loss: 0.69136, Test acc: 63.0
0%
```





Now, we train the model again but with a higher epochs i.e. train for a longer time.

```
In [31]: epochs = 1000
model = ModelV2().to(device) # reset model
optimizer = torch.optim.SGD(params=model.parameters(),
                            lr=0.1)

train_losses, acc_list, test_losses, test_acc = train_and_test_loop(
    model=model,
    epochs=epochs,
    X_train=X_train,
    y_train=y_train,
    X_test=X_test,
    y_test=y_test,
    loss_fn=loss_fn,
    optimizer=optimizer
)
```

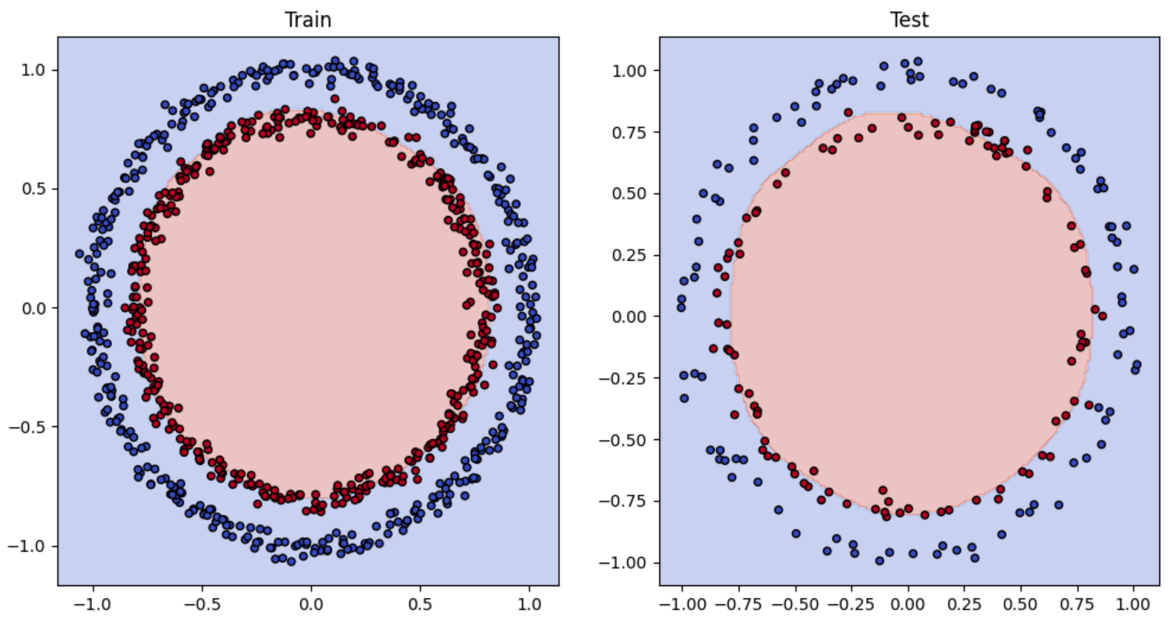
```python

# Plot decision boundaries for training and test sets
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.title("Train")
plot_decision_boundary(model, X_train, y_train)
plt.subplot(1, 2, 2)
plt.title("Test")
plot_decision_boundary(model, X_test, y_test)
plot_loss_curves(train_losses, test_losses)
```

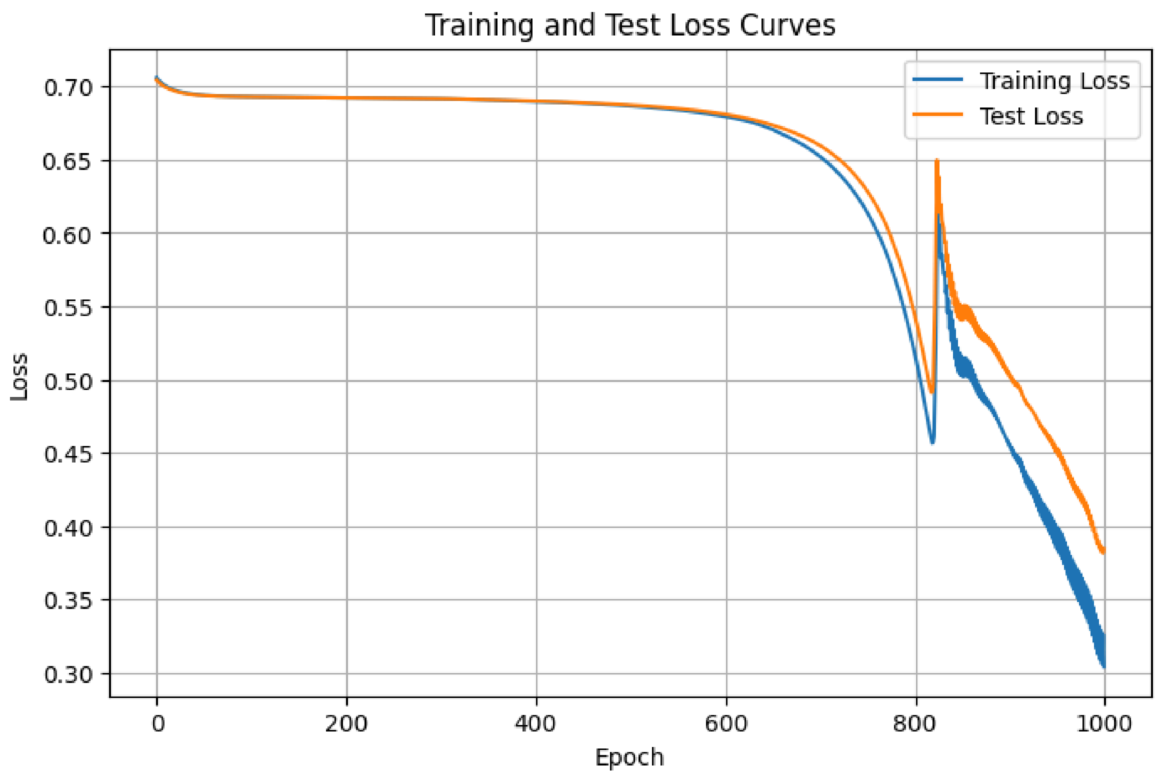
```
Epoch: 0 | Loss: 0.70581, Accuracy: 50.00% | Test loss: 0.70432, Test acc: 50.00%
Epoch: 10 | Loss: 0.70012, Accuracy: 50.00% | Test loss: 0.69923, Test acc: 50.0
0%
Epoch: 20 | Loss: 0.69718, Accuracy: 50.00% | Test loss: 0.69663, Test acc: 50.0
0%
Epoch: 30 | Loss: 0.69549, Accuracy: 50.00% | Test loss: 0.69509, Test acc: 50.0
0%
Epoch: 40 | Loss: 0.69450, Accuracy: 50.00% | Test loss: 0.69421, Test acc: 50.0
0%
Epoch: 50 | Loss: 0.69386, Accuracy: 50.00% | Test loss: 0.69364, Test acc: 50.0
0%
Epoch: 60 | Loss: 0.69346, Accuracy: 50.00% | Test loss: 0.69332, Test acc: 50.0
0%
Epoch: 70 | Loss: 0.69321, Accuracy: 50.00% | Test loss: 0.69312, Test acc: 50.0
0%
Epoch: 80 | Loss: 0.69305, Accuracy: 50.00% | Test loss: 0.69298, Test acc: 50.0
0%
Epoch: 90 | Loss: 0.69294, Accuracy: 50.00% | Test loss: 0.69289, Test acc: 50.0
0%
Epoch: 100 | Loss: 0.69286, Accuracy: 50.00% | Test loss: 0.69281, Test acc: 50.0
0%
Epoch: 110 | Loss: 0.69280, Accuracy: 51.00% | Test loss: 0.69275, Test acc: 51.5
0%
Epoch: 120 | Loss: 0.69274, Accuracy: 56.62% | Test loss: 0.69270, Test acc: 57.5
0%
Epoch: 130 | Loss: 0.69269, Accuracy: 58.13% | Test loss: 0.69265, Test acc: 60.0
0%
Epoch: 140 | Loss: 0.69264, Accuracy: 56.12% | Test loss: 0.69259, Test acc: 55.0
0%
Epoch: 150 | Loss: 0.69258, Accuracy: 54.62% | Test loss: 0.69254, Test acc: 53.5
0%
Epoch: 160 | Loss: 0.69253, Accuracy: 54.50% | Test loss: 0.69249, Test acc: 52.5
0%
Epoch: 170 | Loss: 0.69248, Accuracy: 54.00% | Test loss: 0.69243, Test acc: 54.0
0%
Epoch: 180 | Loss: 0.69240, Accuracy: 54.50% | Test loss: 0.69237, Test acc: 55.0
0%
Epoch: 190 | Loss: 0.69235, Accuracy: 54.62% | Test loss: 0.69232, Test acc: 53.5
0%
Epoch: 200 | Loss: 0.69230, Accuracy: 54.50% | Test loss: 0.69228, Test acc: 54.0
0%
Epoch: 210 | Loss: 0.69224, Accuracy: 54.62% | Test loss: 0.69223, Test acc: 54.0
0%
Epoch: 220 | Loss: 0.69218, Accuracy: 54.75% | Test loss: 0.69218, Test acc: 55.0
0%
Epoch: 230 | Loss: 0.69213, Accuracy: 54.62% | Test loss: 0.69213, Test acc: 55.0
0%
Epoch: 240 | Loss: 0.69206, Accuracy: 54.37% | Test loss: 0.69207, Test acc: 55.0
0%
Epoch: 250 | Loss: 0.69200, Accuracy: 54.25% | Test loss: 0.69202, Test acc: 55.0
0%
Epoch: 260 | Loss: 0.69192, Accuracy: 54.25% | Test loss: 0.69195, Test acc: 55.0
0%
Epoch: 270 | Loss: 0.69184, Accuracy: 54.25% | Test loss: 0.69187, Test acc: 55.0
0%
Epoch: 280 | Loss: 0.69175, Accuracy: 54.37% | Test loss: 0.69179, Test acc: 55.0
0%
Epoch: 290 | Loss: 0.69166, Accuracy: 56.50% | Test loss: 0.69170, Test acc: 57.5
0%
Epoch: 300 | Loss: 0.69154, Accuracy: 60.88% | Test loss: 0.69160, Test acc: 59.0
```

```
0%
Epoch: 310 | Loss: 0.69142, Accuracy: 59.38% | Test loss: 0.69149, Test acc: 57.5
0%
Epoch: 320 | Loss: 0.69126, Accuracy: 59.00% | Test loss: 0.69133, Test acc: 59.0
0%
Epoch: 330 | Loss: 0.69105, Accuracy: 59.50% | Test loss: 0.69113, Test acc: 57.0
0%
Epoch: 340 | Loss: 0.69088, Accuracy: 61.00% | Test loss: 0.69098, Test acc: 59.5
0%
Epoch: 350 | Loss: 0.69072, Accuracy: 61.50% | Test loss: 0.69083, Test acc: 59.5
0%
Epoch: 360 | Loss: 0.69055, Accuracy: 62.25% | Test loss: 0.69068, Test acc: 61.0
0%
Epoch: 370 | Loss: 0.69037, Accuracy: 63.75% | Test loss: 0.69052, Test acc: 61.5
0%
Epoch: 380 | Loss: 0.69018, Accuracy: 66.00% | Test loss: 0.69035, Test acc: 65.5
0%
Epoch: 390 | Loss: 0.68998, Accuracy: 68.38% | Test loss: 0.69016, Test acc: 67.0
0%
Epoch: 400 | Loss: 0.68977, Accuracy: 70.00% | Test loss: 0.68998, Test acc: 68.0
0%
Epoch: 410 | Loss: 0.68954, Accuracy: 70.88% | Test loss: 0.68979, Test acc: 68.5
0%
Epoch: 420 | Loss: 0.68930, Accuracy: 71.12% | Test loss: 0.68958, Test acc: 68.5
0%
Epoch: 430 | Loss: 0.68904, Accuracy: 71.25% | Test loss: 0.68935, Test acc: 68.5
0%
Epoch: 440 | Loss: 0.68876, Accuracy: 71.25% | Test loss: 0.68910, Test acc: 68.5
0%
Epoch: 450 | Loss: 0.68845, Accuracy: 71.38% | Test loss: 0.68882, Test acc: 69.5
0%
Epoch: 460 | Loss: 0.68813, Accuracy: 71.75% | Test loss: 0.68854, Test acc: 69.5
0%
Epoch: 470 | Loss: 0.68778, Accuracy: 72.12% | Test loss: 0.68824, Test acc: 69.0
0%
Epoch: 480 | Loss: 0.68741, Accuracy: 71.88% | Test loss: 0.68793, Test acc: 69.0
0%
Epoch: 490 | Loss: 0.68701, Accuracy: 72.38% | Test loss: 0.68759, Test acc: 70.0
0%
Epoch: 500 | Loss: 0.68657, Accuracy: 72.25% | Test loss: 0.68723, Test acc: 70.5
0%
Epoch: 510 | Loss: 0.68610, Accuracy: 72.25% | Test loss: 0.68683, Test acc: 70.0
0%
Epoch: 520 | Loss: 0.68559, Accuracy: 72.38% | Test loss: 0.68640, Test acc: 69.5
0%
Epoch: 530 | Loss: 0.68504, Accuracy: 72.38% | Test loss: 0.68593, Test acc: 69.0
0%
Epoch: 540 | Loss: 0.68444, Accuracy: 72.50% | Test loss: 0.68542, Test acc: 69.0
0%
Epoch: 550 | Loss: 0.68378, Accuracy: 72.88% | Test loss: 0.68486, Test acc: 69.5
0%
Epoch: 560 | Loss: 0.68307, Accuracy: 73.12% | Test loss: 0.68424, Test acc: 70.0
0%
Epoch: 570 | Loss: 0.68228, Accuracy: 73.38% | Test loss: 0.68353, Test acc: 69.5
0%
Epoch: 580 | Loss: 0.68139, Accuracy: 73.75% | Test loss: 0.68272, Test acc: 70.5
0%
Epoch: 590 | Loss: 0.68040, Accuracy: 74.62% | Test loss: 0.68184, Test acc: 71.0
0%
Epoch: 600 | Loss: 0.67931, Accuracy: 74.75% | Test loss: 0.68087, Test acc: 71.5
```

```
0%
Epoch: 610 | Loss: 0.67810, Accuracy: 75.25% | Test loss: 0.67977, Test acc: 71.5
0%
Epoch: 620 | Loss: 0.67673, Accuracy: 76.38% | Test loss: 0.67852, Test acc: 74.0
0%
Epoch: 630 | Loss: 0.67509, Accuracy: 78.25% | Test loss: 0.67707, Test acc: 74.5
0%
Epoch: 640 | Loss: 0.67294, Accuracy: 78.38% | Test loss: 0.67530, Test acc: 73.5
0%
Epoch: 650 | Loss: 0.67046, Accuracy: 77.00% | Test loss: 0.67347, Test acc: 70.0
0%
Epoch: 660 | Loss: 0.66734, Accuracy: 75.25% | Test loss: 0.67131, Test acc: 70.5
0%
Epoch: 670 | Loss: 0.66423, Accuracy: 75.12% | Test loss: 0.66896, Test acc: 70.0
0%
Epoch: 680 | Loss: 0.66070, Accuracy: 75.00% | Test loss: 0.66625, Test acc: 71.0
0%
Epoch: 690 | Loss: 0.65664, Accuracy: 76.62% | Test loss: 0.66313, Test acc: 72.0
0%
Epoch: 700 | Loss: 0.65194, Accuracy: 77.25% | Test loss: 0.65947, Test acc: 72.0
0%
Epoch: 710 | Loss: 0.64647, Accuracy: 78.38% | Test loss: 0.65514, Test acc: 73.5
0%
Epoch: 720 | Loss: 0.64006, Accuracy: 80.38% | Test loss: 0.65001, Test acc: 74.0
0%
Epoch: 730 | Loss: 0.63260, Accuracy: 82.50% | Test loss: 0.64399, Test acc: 76.0
0%
Epoch: 740 | Loss: 0.62380, Accuracy: 83.75% | Test loss: 0.63675, Test acc: 78.0
0%
Epoch: 750 | Loss: 0.61332, Accuracy: 86.12% | Test loss: 0.62806, Test acc: 80.0
0%
Epoch: 760 | Loss: 0.60075, Accuracy: 88.38% | Test loss: 0.61748, Test acc: 82.5
0%
Epoch: 770 | Loss: 0.58551, Accuracy: 91.00% | Test loss: 0.60437, Test acc: 86.0
0%
Epoch: 780 | Loss: 0.56639, Accuracy: 93.88% | Test loss: 0.58756, Test acc: 89.0
0%
Epoch: 790 | Loss: 0.54371, Accuracy: 96.00% | Test loss: 0.56746, Test acc: 91.0
0%
Epoch: 800 | Loss: 0.51640, Accuracy: 96.62% | Test loss: 0.54271, Test acc: 95.0
0%
Epoch: 810 | Loss: 0.48392, Accuracy: 98.75% | Test loss: 0.51281, Test acc: 97.0
0%
Epoch: 820 | Loss: 0.46151, Accuracy: 94.50% | Test loss: 0.52162, Test acc: 66.5
0%
Epoch: 830 | Loss: 0.57186, Accuracy: 57.38% | Test loss: 0.60317, Test acc: 54.5
0%
Epoch: 840 | Loss: 0.52045, Accuracy: 62.62% | Test loss: 0.56616, Test acc: 58.0
0%
Epoch: 850 | Loss: 0.50199, Accuracy: 63.88% | Test loss: 0.54996, Test acc: 60.5
0%
Epoch: 860 | Loss: 0.49849, Accuracy: 63.62% | Test loss: 0.54517, Test acc: 60.5
0%
Epoch: 870 | Loss: 0.48636, Accuracy: 65.12% | Test loss: 0.53282, Test acc: 62.5
0%
Epoch: 880 | Loss: 0.47930, Accuracy: 65.62% | Test loss: 0.52609, Test acc: 63.0
0%
Epoch: 890 | Loss: 0.46784, Accuracy: 66.25% | Test loss: 0.51496, Test acc: 63.0
0%
Epoch: 900 | Loss: 0.45571, Accuracy: 67.75% | Test loss: 0.50360, Test acc: 63.5
```

```
0%
Epoch: 910 | Loss: 0.44613, Accuracy: 68.88% | Test loss: 0.49487, Test acc: 64.5
0%
Epoch: 920 | Loss: 0.43235, Accuracy: 70.12% | Test loss: 0.48215, Test acc: 65.5
0%
Epoch: 930 | Loss: 0.42125, Accuracy: 71.25% | Test loss: 0.47065, Test acc: 66.5
0%
Epoch: 940 | Loss: 0.41033, Accuracy: 71.88% | Test loss: 0.46006, Test acc: 66.5
0%
Epoch: 950 | Loss: 0.39874, Accuracy: 72.62% | Test loss: 0.44902, Test acc: 67.5
0%
Epoch: 960 | Loss: 0.38544, Accuracy: 74.88% | Test loss: 0.43583, Test acc: 70.5
0%
Epoch: 970 | Loss: 0.37122, Accuracy: 77.38% | Test loss: 0.42258, Test acc: 70.5
0%
Epoch: 980 | Loss: 0.35914, Accuracy: 78.62% | Test loss: 0.41108, Test acc: 70.5
0%
Epoch: 990 | Loss: 0.34009, Accuracy: 81.25% | Test loss: 0.39227, Test acc: 72.5
0%
```





In [32]:
```
model = ModelV2().to(device) # reset model
epochs = 1500
optimizer = torch.optim.SGD(params=model.parameters(),
                            lr=0.1)

train_losses, acc_list, test_losses, test_acc = train_and_test_loop(
    model=model,
    epochs=epochs,
    X_train=X_train,
    y_train=y_train,
    X_test=X_test,
    y_test=y_test,
    loss_fn=loss_fn,
    optimizer=optimizer
)

# Plot decision boundaries for training and test sets
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.title("Train")
plot_decision_boundary(model, X_train, y_train)
plt.subplot(1, 2, 2)
plt.title("Test")
plot_decision_boundary(model, X_test, y_test)
plot_loss_curves(train_losses, test_losses)
```

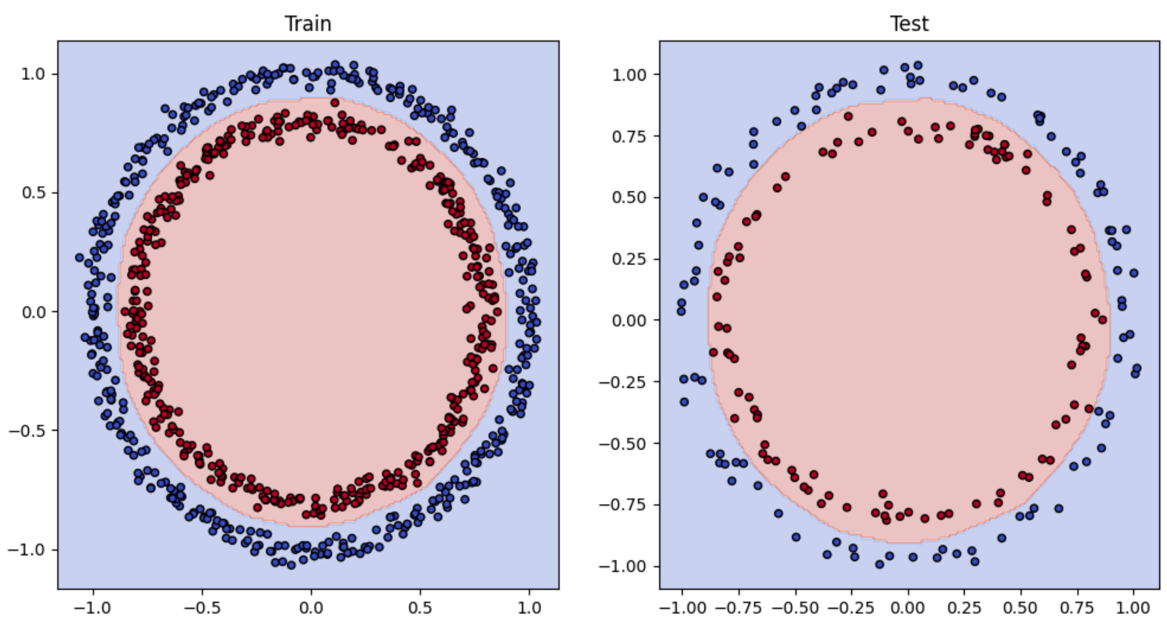
```
Epoch: 0 | Loss: 0.69662, Accuracy: 50.00% | Test loss: 0.69565, Test acc: 50.00%
Epoch: 10 | Loss: 0.69422, Accuracy: 50.00% | Test loss: 0.69378, Test acc: 50.0
0%
Epoch: 20 | Loss: 0.69299, Accuracy: 50.00% | Test loss: 0.69290, Test acc: 50.0
0%
Epoch: 30 | Loss: 0.69213, Accuracy: 66.62% | Test loss: 0.69229, Test acc: 67.0
0%
Epoch: 40 | Loss: 0.69141, Accuracy: 52.75% | Test loss: 0.69184, Test acc: 50.0
0%
Epoch: 50 | Loss: 0.69069, Accuracy: 53.50% | Test loss: 0.69131, Test acc: 48.5
0%
Epoch: 60 | Loss: 0.69016, Accuracy: 53.50% | Test loss: 0.69096, Test acc: 51.5
0%
Epoch: 70 | Loss: 0.68968, Accuracy: 54.00% | Test loss: 0.69065, Test acc: 52.0
0%
Epoch: 80 | Loss: 0.68922, Accuracy: 56.12% | Test loss: 0.69032, Test acc: 54.5
0%
Epoch: 90 | Loss: 0.68874, Accuracy: 58.00% | Test loss: 0.68996, Test acc: 55.0
0%
Epoch: 100 | Loss: 0.68815, Accuracy: 62.75% | Test loss: 0.68949, Test acc: 58.0
0%
Epoch: 110 | Loss: 0.68755, Accuracy: 67.25% | Test loss: 0.68912, Test acc: 60.0
0%
Epoch: 120 | Loss: 0.68690, Accuracy: 70.62% | Test loss: 0.68871, Test acc: 64.0
0%
Epoch: 130 | Loss: 0.68627, Accuracy: 70.50% | Test loss: 0.68830, Test acc: 66.0
0%
Epoch: 140 | Loss: 0.68562, Accuracy: 70.38% | Test loss: 0.68787, Test acc: 68.5
0%
Epoch: 150 | Loss: 0.68495, Accuracy: 72.38% | Test loss: 0.68740, Test acc: 72.0
0%
Epoch: 160 | Loss: 0.68423, Accuracy: 74.62% | Test loss: 0.68690, Test acc: 73.5
0%
Epoch: 170 | Loss: 0.68347, Accuracy: 75.25% | Test loss: 0.68634, Test acc: 74.0
0%
Epoch: 180 | Loss: 0.68264, Accuracy: 75.38% | Test loss: 0.68574, Test acc: 73.5
0%
Epoch: 190 | Loss: 0.68173, Accuracy: 75.25% | Test loss: 0.68506, Test acc: 74.0
0%
Epoch: 200 | Loss: 0.68075, Accuracy: 75.12% | Test loss: 0.68431, Test acc: 74.0
0%
Epoch: 210 | Loss: 0.67968, Accuracy: 75.25% | Test loss: 0.68351, Test acc: 73.0
0%
Epoch: 220 | Loss: 0.67851, Accuracy: 75.25% | Test loss: 0.68264, Test acc: 72.5
0%
Epoch: 230 | Loss: 0.67722, Accuracy: 75.25% | Test loss: 0.68167, Test acc: 72.5
0%
Epoch: 240 | Loss: 0.67577, Accuracy: 75.50% | Test loss: 0.68057, Test acc: 71.0
0%
Epoch: 250 | Loss: 0.67414, Accuracy: 75.50% | Test loss: 0.67929, Test acc: 71.0
0%
Epoch: 260 | Loss: 0.67234, Accuracy: 76.38% | Test loss: 0.67788, Test acc: 71.5
0%
Epoch: 270 | Loss: 0.67039, Accuracy: 76.88% | Test loss: 0.67630, Test acc: 72.0
0%
Epoch: 280 | Loss: 0.66820, Accuracy: 78.38% | Test loss: 0.67447, Test acc: 73.5
0%
Epoch: 290 | Loss: 0.66574, Accuracy: 79.62% | Test loss: 0.67240, Test acc: 74.5
0%
Epoch: 300 | Loss: 0.66295, Accuracy: 81.12% | Test loss: 0.67001, Test acc: 76.5
```

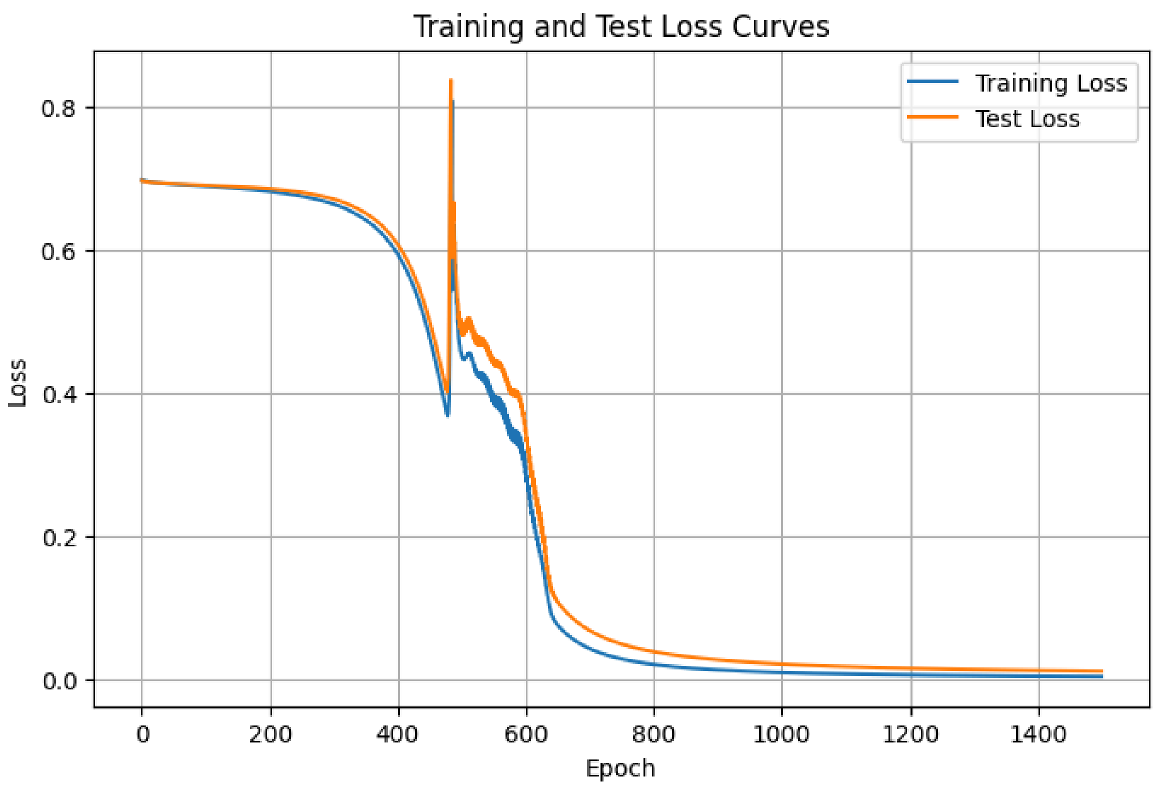
```
0%
Epoch: 310 | Loss: 0.65971, Accuracy: 83.38% | Test loss: 0.66716, Test acc: 77.5
0%
Epoch: 320 | Loss: 0.65599, Accuracy: 85.00% | Test loss: 0.66379, Test acc: 81.5
0%
Epoch: 330 | Loss: 0.65136, Accuracy: 86.50% | Test loss: 0.65967, Test acc: 83.0
0%
Epoch: 340 | Loss: 0.64632, Accuracy: 88.12% | Test loss: 0.65526, Test acc: 85.0
0%
Epoch: 350 | Loss: 0.64060, Accuracy: 89.75% | Test loss: 0.65012, Test acc: 88.0
0%
Epoch: 360 | Loss: 0.63394, Accuracy: 92.62% | Test loss: 0.64407, Test acc: 89.5
0%
Epoch: 370 | Loss: 0.62609, Accuracy: 95.25% | Test loss: 0.63694, Test acc: 92.5
0%
Epoch: 380 | Loss: 0.61682, Accuracy: 97.00% | Test loss: 0.62850, Test acc: 94.0
0%
Epoch: 390 | Loss: 0.60581, Accuracy: 98.12% | Test loss: 0.61844, Test acc: 95.0
0%
Epoch: 400 | Loss: 0.59264, Accuracy: 99.00% | Test loss: 0.60646, Test acc: 97.5
0%
Epoch: 410 | Loss: 0.57684, Accuracy: 99.25% | Test loss: 0.59205, Test acc: 98.0
0%
Epoch: 420 | Loss: 0.55782, Accuracy: 99.75% | Test loss: 0.57453, Test acc: 99.0
0%
Epoch: 430 | Loss: 0.53490, Accuracy: 99.88% | Test loss: 0.55323, Test acc: 99.5
0%
Epoch: 440 | Loss: 0.50733, Accuracy: 100.00% | Test loss: 0.52778, Test acc: 99.
50%
Epoch: 450 | Loss: 0.47520, Accuracy: 100.00% | Test loss: 0.49810, Test acc: 10
0.00%
Epoch: 460 | Loss: 0.43828, Accuracy: 100.00% | Test loss: 0.46382, Test acc: 10
0.00%
Epoch: 470 | Loss: 0.39713, Accuracy: 100.00% | Test loss: 0.42563, Test acc: 10
0.00%
Epoch: 480 | Loss: 0.39226, Accuracy: 93.62% | Test loss: 0.50776, Test acc: 60.5
0%
Epoch: 490 | Loss: 0.57811, Accuracy: 51.75% | Test loss: 0.57183, Test acc: 52.0
0%
Epoch: 500 | Loss: 0.44929, Accuracy: 66.50% | Test loss: 0.49666, Test acc: 63.0
0%
Epoch: 510 | Loss: 0.45431, Accuracy: 63.38% | Test loss: 0.50378, Test acc: 60.5
0%
Epoch: 520 | Loss: 0.43670, Accuracy: 67.38% | Test loss: 0.48358, Test acc: 65.0
0%
Epoch: 530 | Loss: 0.42707, Accuracy: 68.62% | Test loss: 0.47600, Test acc: 66.5
0%
Epoch: 540 | Loss: 0.41690, Accuracy: 70.50% | Test loss: 0.46440, Test acc: 68.0
0%
Epoch: 550 | Loss: 0.39640, Accuracy: 74.38% | Test loss: 0.44648, Test acc: 69.0
0%
Epoch: 560 | Loss: 0.38993, Accuracy: 75.00% | Test loss: 0.44077, Test acc: 69.5
0%
Epoch: 570 | Loss: 0.36874, Accuracy: 78.25% | Test loss: 0.41993, Test acc: 71.5
0%
Epoch: 580 | Loss: 0.34803, Accuracy: 81.12% | Test loss: 0.40313, Test acc: 73.5
0%
Epoch: 590 | Loss: 0.34024, Accuracy: 82.00% | Test loss: 0.39574, Test acc: 74.5
0%
Epoch: 600 | Loss: 0.29526, Accuracy: 88.12% | Test loss: 0.34876, Test acc: 80.0
```

```
0%
Epoch: 610 | Loss: 0.23458, Accuracy: 93.50% | Test loss: 0.28939, Test acc: 87.0
0%
Epoch: 620 | Loss: 0.18795, Accuracy: 97.12% | Test loss: 0.24218, Test acc: 93.0
0%
Epoch: 630 | Loss: 0.13910, Accuracy: 99.00% | Test loss: 0.18455, Test acc: 97.5
0%
Epoch: 640 | Loss: 0.09017, Accuracy: 99.88% | Test loss: 0.12660, Test acc: 99.5
0%
Epoch: 650 | Loss: 0.07460, Accuracy: 100.00% | Test loss: 0.10739, Test acc: 10
0.00%
Epoch: 660 | Loss: 0.06534, Accuracy: 100.00% | Test loss: 0.09603, Test acc: 10
0.00%
Epoch: 670 | Loss: 0.05783, Accuracy: 100.00% | Test loss: 0.08694, Test acc: 10
0.00%
Epoch: 680 | Loss: 0.05164, Accuracy: 100.00% | Test loss: 0.07934, Test acc: 10
0.00%
Epoch: 690 | Loss: 0.04644, Accuracy: 100.00% | Test loss: 0.07286, Test acc: 10
0.00%
Epoch: 700 | Loss: 0.04203, Accuracy: 100.00% | Test loss: 0.06734, Test acc: 10
0.00%
Epoch: 710 | Loss: 0.03827, Accuracy: 100.00% | Test loss: 0.06254, Test acc: 10
0.00%
Epoch: 720 | Loss: 0.03500, Accuracy: 100.00% | Test loss: 0.05830, Test acc: 10
0.00%
Epoch: 730 | Loss: 0.03219, Accuracy: 100.00% | Test loss: 0.05458, Test acc: 10
0.00%
Epoch: 740 | Loss: 0.02974, Accuracy: 100.00% | Test loss: 0.05129, Test acc: 10
0.00%
Epoch: 750 | Loss: 0.02760, Accuracy: 100.00% | Test loss: 0.04838, Test acc: 10
0.00%
Epoch: 760 | Loss: 0.02573, Accuracy: 100.00% | Test loss: 0.04581, Test acc: 10
0.00%
Epoch: 770 | Loss: 0.02407, Accuracy: 100.00% | Test loss: 0.04351, Test acc: 10
0.00%
Epoch: 780 | Loss: 0.02259, Accuracy: 100.00% | Test loss: 0.04144, Test acc: 10
0.00%
Epoch: 790 | Loss: 0.02126, Accuracy: 100.00% | Test loss: 0.03957, Test acc: 10
0.00%
Epoch: 800 | Loss: 0.02007, Accuracy: 100.00% | Test loss: 0.03784, Test acc: 10
0.00%
Epoch: 810 | Loss: 0.01898, Accuracy: 100.00% | Test loss: 0.03626, Test acc: 10
0.00%
Epoch: 820 | Loss: 0.01800, Accuracy: 100.00% | Test loss: 0.03480, Test acc: 10
0.00%
Epoch: 830 | Loss: 0.01710, Accuracy: 100.00% | Test loss: 0.03347, Test acc: 10
0.00%
Epoch: 840 | Loss: 0.01628, Accuracy: 100.00% | Test loss: 0.03224, Test acc: 10
0.00%
Epoch: 850 | Loss: 0.01553, Accuracy: 100.00% | Test loss: 0.03111, Test acc: 10
0.00%
Epoch: 860 | Loss: 0.01484, Accuracy: 100.00% | Test loss: 0.03006, Test acc: 10
0.00%
Epoch: 870 | Loss: 0.01420, Accuracy: 100.00% | Test loss: 0.02908, Test acc: 10
0.00%
Epoch: 880 | Loss: 0.01361, Accuracy: 100.00% | Test loss: 0.02817, Test acc: 10
0.00%
Epoch: 890 | Loss: 0.01306, Accuracy: 100.00% | Test loss: 0.02732, Test acc: 10
0.00%
Epoch: 900 | Loss: 0.01255, Accuracy: 100.00% | Test loss: 0.02651, Test acc: 10
```

```
0.00%
Epoch: 910 | Loss: 0.01208, Accuracy: 100.00% | Test loss: 0.02575, Test acc: 10
0.00%
Epoch: 920 | Loss: 0.01163, Accuracy: 100.00% | Test loss: 0.02503, Test acc: 10
0.00%
Epoch: 930 | Loss: 0.01121, Accuracy: 100.00% | Test loss: 0.02435, Test acc: 10
0.00%
Epoch: 940 | Loss: 0.01082, Accuracy: 100.00% | Test loss: 0.02373, Test acc: 10
0.00%
Epoch: 950 | Loss: 0.01046, Accuracy: 100.00% | Test loss: 0.02314, Test acc: 10
0.00%
Epoch: 960 | Loss: 0.01011, Accuracy: 100.00% | Test loss: 0.02258, Test acc: 10
0.00%
Epoch: 970 | Loss: 0.00979, Accuracy: 100.00% | Test loss: 0.02205, Test acc: 10
0.00%
Epoch: 980 | Loss: 0.00948, Accuracy: 100.00% | Test loss: 0.02155, Test acc: 10
0.00%
Epoch: 990 | Loss: 0.00920, Accuracy: 100.00% | Test loss: 0.02107, Test acc: 10
0.00%
Epoch: 1000 | Loss: 0.00892, Accuracy: 100.00% | Test loss: 0.02062, Test acc: 10
0.00%
Epoch: 1010 | Loss: 0.00867, Accuracy: 100.00% | Test loss: 0.02019, Test acc: 10
0.00%
Epoch: 1020 | Loss: 0.00842, Accuracy: 100.00% | Test loss: 0.01978, Test acc: 10
0.00%
Epoch: 1030 | Loss: 0.00819, Accuracy: 100.00% | Test loss: 0.01939, Test acc: 10
0.00%
Epoch: 1040 | Loss: 0.00797, Accuracy: 100.00% | Test loss: 0.01902, Test acc: 10
0.00%
Epoch: 1050 | Loss: 0.00776, Accuracy: 100.00% | Test loss: 0.01866, Test acc: 10
0.00%
Epoch: 1060 | Loss: 0.00756, Accuracy: 100.00% | Test loss: 0.01833, Test acc: 10
0.00%
Epoch: 1070 | Loss: 0.00737, Accuracy: 100.00% | Test loss: 0.01802, Test acc: 10
0.00%
Epoch: 1080 | Loss: 0.00719, Accuracy: 100.00% | Test loss: 0.01771, Test acc: 10
0.00%
Epoch: 1090 | Loss: 0.00702, Accuracy: 100.00% | Test loss: 0.01742, Test acc: 10
0.00%
Epoch: 1100 | Loss: 0.00686, Accuracy: 100.00% | Test loss: 0.01714, Test acc: 10
0.00%
Epoch: 1110 | Loss: 0.00670, Accuracy: 100.00% | Test loss: 0.01687, Test acc: 10
0.00%
Epoch: 1120 | Loss: 0.00655, Accuracy: 100.00% | Test loss: 0.01660, Test acc: 10
0.00%
Epoch: 1130 | Loss: 0.00641, Accuracy: 100.00% | Test loss: 0.01635, Test acc: 10
0.00%
Epoch: 1140 | Loss: 0.00627, Accuracy: 100.00% | Test loss: 0.01610, Test acc: 10
0.00%
Epoch: 1150 | Loss: 0.00614, Accuracy: 100.00% | Test loss: 0.01587, Test acc: 10
0.00%
Epoch: 1160 | Loss: 0.00601, Accuracy: 100.00% | Test loss: 0.01564, Test acc: 10
0.00%
Epoch: 1170 | Loss: 0.00589, Accuracy: 100.00% | Test loss: 0.01541, Test acc: 10
0.00%
Epoch: 1180 | Loss: 0.00577, Accuracy: 100.00% | Test loss: 0.01520, Test acc: 10
0.00%
Epoch: 1190 | Loss: 0.00566, Accuracy: 100.00% | Test loss: 0.01499, Test acc: 10
0.00%
Epoch: 1200 | Loss: 0.00555, Accuracy: 100.00% | Test loss: 0.01479, Test acc: 10
```

```
0.00%
Epoch: 1210 | Loss: 0.00544, Accuracy: 100.00% | Test loss: 0.01460, Test acc: 10
0.00%
Epoch: 1220 | Loss: 0.00534, Accuracy: 100.00% | Test loss: 0.01441, Test acc: 10
0.00%
Epoch: 1230 | Loss: 0.00524, Accuracy: 100.00% | Test loss: 0.01423, Test acc: 10
0.00%
Epoch: 1240 | Loss: 0.00515, Accuracy: 100.00% | Test loss: 0.01405, Test acc: 10
0.00%
Epoch: 1250 | Loss: 0.00506, Accuracy: 100.00% | Test loss: 0.01387, Test acc: 10
0.00%
Epoch: 1260 | Loss: 0.00497, Accuracy: 100.00% | Test loss: 0.01370, Test acc: 10
0.00%
Epoch: 1270 | Loss: 0.00488, Accuracy: 100.00% | Test loss: 0.01354, Test acc: 10
0.00%
Epoch: 1280 | Loss: 0.00480, Accuracy: 100.00% | Test loss: 0.01338, Test acc: 10
0.00%
Epoch: 1290 | Loss: 0.00472, Accuracy: 100.00% | Test loss: 0.01322, Test acc: 10
0.00%
Epoch: 1300 | Loss: 0.00464, Accuracy: 100.00% | Test loss: 0.01307, Test acc: 10
0.00%
Epoch: 1310 | Loss: 0.00457, Accuracy: 100.00% | Test loss: 0.01293, Test acc: 10
0.00%
Epoch: 1320 | Loss: 0.00449, Accuracy: 100.00% | Test loss: 0.01278, Test acc: 10
0.00%
Epoch: 1330 | Loss: 0.00442, Accuracy: 100.00% | Test loss: 0.01264, Test acc: 10
0.00%
Epoch: 1340 | Loss: 0.00435, Accuracy: 100.00% | Test loss: 0.01250, Test acc: 10
0.00%
Epoch: 1350 | Loss: 0.00428, Accuracy: 100.00% | Test loss: 0.01236, Test acc: 10
0.00%
Epoch: 1360 | Loss: 0.00421, Accuracy: 100.00% | Test loss: 0.01223, Test acc: 10
0.00%
Epoch: 1370 | Loss: 0.00415, Accuracy: 100.00% | Test loss: 0.01212, Test acc: 10
0.00%
Epoch: 1380 | Loss: 0.00409, Accuracy: 100.00% | Test loss: 0.01200, Test acc: 10
0.00%
Epoch: 1390 | Loss: 0.00403, Accuracy: 100.00% | Test loss: 0.01188, Test acc: 10
0.00%
Epoch: 1400 | Loss: 0.00397, Accuracy: 100.00% | Test loss: 0.01177, Test acc: 10
0.00%
Epoch: 1410 | Loss: 0.00391, Accuracy: 100.00% | Test loss: 0.01166, Test acc: 10
0.00%
Epoch: 1420 | Loss: 0.00386, Accuracy: 100.00% | Test loss: 0.01155, Test acc: 10
0.00%
Epoch: 1430 | Loss: 0.00381, Accuracy: 100.00% | Test loss: 0.01144, Test acc: 10
0.00%
Epoch: 1440 | Loss: 0.00375, Accuracy: 100.00% | Test loss: 0.01134, Test acc: 10
0.00%
Epoch: 1450 | Loss: 0.00370, Accuracy: 100.00% | Test loss: 0.01124, Test acc: 10
0.00%
Epoch: 1460 | Loss: 0.00365, Accuracy: 100.00% | Test loss: 0.01114, Test acc: 10
0.00%
Epoch: 1470 | Loss: 0.00360, Accuracy: 100.00% | Test loss: 0.01104, Test acc: 10
0.00%
Epoch: 1480 | Loss: 0.00356, Accuracy: 100.00% | Test loss: 0.01094, Test acc: 10
0.00%
Epoch: 1490 | Loss: 0.00351, Accuracy: 100.00% | Test loss: 0.01085, Test acc: 10
0.00%
```





Optimizer Comparison: Adam vs SGD

We'll train the same architecture (`ModelV2`) using two different optimizers (SGD and Adam) to see how they affect learning. We'll compare training/test loss curves and decision boundaries for each optimizer.

In [33]:
```
# train with SGD and record metrics
m_sgd = ModelV2().to(device)
opt_sgd = torch.optim.SGD(m_sgd.parameters(), lr=0.1)
tr_losses_sgd, tr_acc_sgd, te_losses_sgd, te_acc_sgd = train_and_test_loop(
    model=m_sgd,
    epochs=1000,
    X_train=X_train,
    y_train=y_train,
    X_test=X_test,
    y_test=y_test,
    loss_fn=nn.BCEWithLogitsLoss(),
    optimizer=opt_sgd
)
print(f"SGD final test acc: {te_acc_sgd[-1]:.2f}%")
```

```
Epoch: 0 | Loss: 0.69615, Accuracy: 50.00% | Test loss: 0.69582, Test acc: 50.00%
Epoch: 10 | Loss: 0.69468, Accuracy: 50.00% | Test loss: 0.69452, Test acc: 50.0
0%
Epoch: 20 | Loss: 0.69367, Accuracy: 50.00% | Test loss: 0.69361, Test acc: 50.0
0%
Epoch: 30 | Loss: 0.69307, Accuracy: 50.00% | Test loss: 0.69307, Test acc: 50.0
0%
Epoch: 40 | Loss: 0.69273, Accuracy: 50.00% | Test loss: 0.69277, Test acc: 50.0
0%
Epoch: 50 | Loss: 0.69248, Accuracy: 50.00% | Test loss: 0.69254, Test acc: 50.0
0%
Epoch: 60 | Loss: 0.69227, Accuracy: 50.00% | Test loss: 0.69234, Test acc: 50.0
0%
Epoch: 70 | Loss: 0.69208, Accuracy: 50.00% | Test loss: 0.69216, Test acc: 50.0
0%
Epoch: 80 | Loss: 0.69190, Accuracy: 50.50% | Test loss: 0.69200, Test acc: 50.0
0%
Epoch: 90 | Loss: 0.69173, Accuracy: 54.37% | Test loss: 0.69184, Test acc: 54.0
0%
Epoch: 100 | Loss: 0.69155, Accuracy: 56.75% | Test loss: 0.69168, Test acc: 57.0
0%
Epoch: 110 | Loss: 0.69137, Accuracy: 63.38% | Test loss: 0.69151, Test acc: 63.0
0%
Epoch: 120 | Loss: 0.69118, Accuracy: 65.12% | Test loss: 0.69134, Test acc: 64.5
0%
Epoch: 130 | Loss: 0.69098, Accuracy: 63.88% | Test loss: 0.69116, Test acc: 66.5
0%
Epoch: 140 | Loss: 0.69078, Accuracy: 65.12% | Test loss: 0.69097, Test acc: 66.5
0%
Epoch: 150 | Loss: 0.69057, Accuracy: 65.75% | Test loss: 0.69078, Test acc: 68.5
0%
Epoch: 160 | Loss: 0.69034, Accuracy: 67.25% | Test loss: 0.69057, Test acc: 68.5
0%
Epoch: 170 | Loss: 0.69010, Accuracy: 67.75% | Test loss: 0.69034, Test acc: 71.5
0%
Epoch: 180 | Loss: 0.68984, Accuracy: 69.38% | Test loss: 0.69010, Test acc: 72.5
0%
Epoch: 190 | Loss: 0.68956, Accuracy: 70.75% | Test loss: 0.68984, Test acc: 72.0
0%
Epoch: 200 | Loss: 0.68926, Accuracy: 72.38% | Test loss: 0.68956, Test acc: 73.0
0%
Epoch: 210 | Loss: 0.68894, Accuracy: 72.75% | Test loss: 0.68925, Test acc: 73.5
0%
Epoch: 220 | Loss: 0.68858, Accuracy: 74.00% | Test loss: 0.68893, Test acc: 73.0
0%
Epoch: 230 | Loss: 0.68820, Accuracy: 74.75% | Test loss: 0.68857, Test acc: 74.5
0%
Epoch: 240 | Loss: 0.68778, Accuracy: 75.75% | Test loss: 0.68817, Test acc: 75.0
0%
Epoch: 250 | Loss: 0.68731, Accuracy: 76.62% | Test loss: 0.68774, Test acc: 75.5
0%
Epoch: 260 | Loss: 0.68682, Accuracy: 78.75% | Test loss: 0.68728, Test acc: 76.5
0%
Epoch: 270 | Loss: 0.68629, Accuracy: 79.75% | Test loss: 0.68678, Test acc: 78.5
0%
Epoch: 280 | Loss: 0.68570, Accuracy: 80.25% | Test loss: 0.68624, Test acc: 78.5
0%
Epoch: 290 | Loss: 0.68505, Accuracy: 81.50% | Test loss: 0.68563, Test acc: 80.0
0%
Epoch: 300 | Loss: 0.68435, Accuracy: 82.50% | Test loss: 0.68497, Test acc: 81.0
```

```
0%
Epoch: 310 | Loss: 0.68356, Accuracy: 83.25% | Test loss: 0.68423, Test acc: 82.0
0%
Epoch: 320 | Loss: 0.68269, Accuracy: 83.62% | Test loss: 0.68341, Test acc: 82.5
0%
Epoch: 330 | Loss: 0.68171, Accuracy: 84.50% | Test loss: 0.68248, Test acc: 83.5
0%
Epoch: 340 | Loss: 0.68062, Accuracy: 85.62% | Test loss: 0.68144, Test acc: 83.5
0%
Epoch: 350 | Loss: 0.67936, Accuracy: 85.88% | Test loss: 0.68028, Test acc: 84.5
0%
Epoch: 360 | Loss: 0.67792, Accuracy: 86.75% | Test loss: 0.67896, Test acc: 86.5
0%
Epoch: 370 | Loss: 0.67626, Accuracy: 87.88% | Test loss: 0.67745, Test acc: 88.5
0%
Epoch: 380 | Loss: 0.67438, Accuracy: 89.50% | Test loss: 0.67574, Test acc: 90.5
0%
Epoch: 390 | Loss: 0.67224, Accuracy: 90.75% | Test loss: 0.67379, Test acc: 92.0
0%
Epoch: 400 | Loss: 0.66974, Accuracy: 92.75% | Test loss: 0.67153, Test acc: 93.5
0%
Epoch: 410 | Loss: 0.66682, Accuracy: 94.50% | Test loss: 0.66891, Test acc: 94.0
0%
Epoch: 420 | Loss: 0.66347, Accuracy: 96.00% | Test loss: 0.66592, Test acc: 94.0
0%
Epoch: 430 | Loss: 0.65957, Accuracy: 96.88% | Test loss: 0.66243, Test acc: 94.5
0%
Epoch: 440 | Loss: 0.65497, Accuracy: 97.62% | Test loss: 0.65832, Test acc: 95.5
0%
Epoch: 450 | Loss: 0.64934, Accuracy: 98.12% | Test loss: 0.65329, Test acc: 95.5
0%
Epoch: 460 | Loss: 0.64265, Accuracy: 98.62% | Test loss: 0.64733, Test acc: 95.0
0%
Epoch: 470 | Loss: 0.63475, Accuracy: 98.62% | Test loss: 0.64015, Test acc: 96.0
0%
Epoch: 480 | Loss: 0.62526, Accuracy: 98.88% | Test loss: 0.63148, Test acc: 96.5
0%
Epoch: 490 | Loss: 0.61377, Accuracy: 98.88% | Test loss: 0.62088, Test acc: 97.0
0%
Epoch: 500 | Loss: 0.59976, Accuracy: 98.88% | Test loss: 0.60789, Test acc: 98.5
0%
Epoch: 510 | Loss: 0.58255, Accuracy: 99.25% | Test loss: 0.59198, Test acc: 99.0
0%
Epoch: 520 | Loss: 0.56161, Accuracy: 99.62% | Test loss: 0.57235, Test acc: 99.5
0%
Epoch: 530 | Loss: 0.53584, Accuracy: 99.62% | Test loss: 0.54814, Test acc: 99.5
0%
Epoch: 540 | Loss: 0.50483, Accuracy: 99.88% | Test loss: 0.51935, Test acc: 100.
00%
Epoch: 550 | Loss: 0.47042, Accuracy: 99.62% | Test loss: 0.49255, Test acc: 95.0
0%
Epoch: 560 | Loss: 0.57217, Accuracy: 52.12% | Test loss: 0.61419, Test acc: 50.0
0%
Epoch: 570 | Loss: 0.49277, Accuracy: 65.12% | Test loss: 0.53472, Test acc: 57.0
0%
Epoch: 580 | Loss: 0.53734, Accuracy: 53.62% | Test loss: 0.59120, Test acc: 51.5
0%
Epoch: 590 | Loss: 0.48452, Accuracy: 61.38% | Test loss: 0.53019, Test acc: 57.0
0%
Epoch: 600 | Loss: 0.50935, Accuracy: 55.50% | Test loss: 0.55853, Test acc: 52.5
```

```
0%
Epoch: 610 | Loss: 0.47435, Accuracy: 61.50% | Test loss: 0.52166, Test acc: 58.0
0%
Epoch: 620 | Loss: 0.47935, Accuracy: 60.62% | Test loss: 0.52600, Test acc: 57.5
0%
Epoch: 630 | Loss: 0.46050, Accuracy: 63.00% | Test loss: 0.50688, Test acc: 61.0
0%
Epoch: 640 | Loss: 0.45395, Accuracy: 63.62% | Test loss: 0.49984, Test acc: 62.0
0%
Epoch: 650 | Loss: 0.44131, Accuracy: 66.50% | Test loss: 0.48611, Test acc: 64.5
0%
Epoch: 660 | Loss: 0.43134, Accuracy: 68.00% | Test loss: 0.47549, Test acc: 66.0
0%
Epoch: 670 | Loss: 0.41980, Accuracy: 69.38% | Test loss: 0.46335, Test acc: 67.0
0%
Epoch: 680 | Loss: 0.40577, Accuracy: 72.25% | Test loss: 0.44987, Test acc: 67.5
0%
Epoch: 690 | Loss: 0.39193, Accuracy: 74.12% | Test loss: 0.43702, Test acc: 69.0
0%
Epoch: 700 | Loss: 0.37727, Accuracy: 75.88% | Test loss: 0.42337, Test acc: 70.0
0%
Epoch: 710 | Loss: 0.36187, Accuracy: 78.38% | Test loss: 0.40883, Test acc: 71.5
0%
Epoch: 720 | Loss: 0.34094, Accuracy: 81.50% | Test loss: 0.38964, Test acc: 73.5
0%
Epoch: 730 | Loss: 0.32017, Accuracy: 84.50% | Test loss: 0.37119, Test acc: 75.5
0%
Epoch: 740 | Loss: 0.28969, Accuracy: 87.88% | Test loss: 0.34009, Test acc: 79.5
0%
Epoch: 750 | Loss: 0.20354, Accuracy: 94.75% | Test loss: 0.24211, Test acc: 93.0
0%
Epoch: 760 | Loss: 0.08880, Accuracy: 100.00% | Test loss: 0.12239, Test acc: 99.
50%
Epoch: 770 | Loss: 0.07248, Accuracy: 100.00% | Test loss: 0.10361, Test acc: 10
0.00%
Epoch: 780 | Loss: 0.06251, Accuracy: 100.00% | Test loss: 0.09184, Test acc: 10
0.00%
Epoch: 790 | Loss: 0.05462, Accuracy: 100.00% | Test loss: 0.08246, Test acc: 10
0.00%
Epoch: 800 | Loss: 0.04824, Accuracy: 100.00% | Test loss: 0.07472, Test acc: 10
0.00%
Epoch: 810 | Loss: 0.04302, Accuracy: 100.00% | Test loss: 0.06824, Test acc: 10
0.00%
Epoch: 820 | Loss: 0.03868, Accuracy: 100.00% | Test loss: 0.06277, Test acc: 10
0.00%
Epoch: 830 | Loss: 0.03505, Accuracy: 100.00% | Test loss: 0.05812, Test acc: 10
0.00%
Epoch: 840 | Loss: 0.03197, Accuracy: 100.00% | Test loss: 0.05411, Test acc: 10
0.00%
Epoch: 850 | Loss: 0.02932, Accuracy: 100.00% | Test loss: 0.05063, Test acc: 10
0.00%
Epoch: 860 | Loss: 0.02704, Accuracy: 100.00% | Test loss: 0.04757, Test acc: 10
0.00%
Epoch: 870 | Loss: 0.02506, Accuracy: 100.00% | Test loss: 0.04488, Test acc: 10
0.00%
Epoch: 880 | Loss: 0.02331, Accuracy: 100.00% | Test loss: 0.04250, Test acc: 10
0.00%
Epoch: 890 | Loss: 0.02178, Accuracy: 100.00% | Test loss: 0.04036, Test acc: 10
0.00%
Epoch: 900 | Loss: 0.02041, Accuracy: 100.00% | Test loss: 0.03845, Test acc: 10
```

```
0.00%
Epoch: 910 | Loss: 0.01919, Accuracy: 100.00% | Test loss: 0.03671, Test acc: 10
0.00%
Epoch: 920 | Loss: 0.01810, Accuracy: 100.00% | Test loss: 0.03514, Test acc: 10
0.00%
Epoch: 930 | Loss: 0.01711, Accuracy: 100.00% | Test loss: 0.03370, Test acc: 10
0.00%
Epoch: 940 | Loss: 0.01622, Accuracy: 100.00% | Test loss: 0.03238, Test acc: 10
0.00%
Epoch: 950 | Loss: 0.01541, Accuracy: 100.00% | Test loss: 0.03117, Test acc: 10
0.00%
Epoch: 960 | Loss: 0.01467, Accuracy: 100.00% | Test loss: 0.03006, Test acc: 10
0.00%
Epoch: 970 | Loss: 0.01399, Accuracy: 100.00% | Test loss: 0.02903, Test acc: 10
0.00%
Epoch: 980 | Loss: 0.01337, Accuracy: 100.00% | Test loss: 0.02808, Test acc: 10
0.00%
Epoch: 990 | Loss: 0.01280, Accuracy: 100.00% | Test loss: 0.02719, Test acc: 10
0.00%
SGD final test acc: 100.00%
```

In [34]:

```python
# train with Adam optimizer
m_adam = ModelV2().to(device)
opt_adam = torch.optim.Adam(m_adam.parameters(), lr=0.1)
tr_losses_adam, tr_acc_adam, te_losses_adam, te_acc_adam = train_and_test_loop(
    model=m_adam,
    epochs=1000,
    X_train=X_train,
    y_train=y_train,
    X_test=X_test,
    y_test=y_test,
    loss_fn=nn.BCEWithLogitsLoss(),
    optimizer=opt_adam
)
print(f"Adam final test acc: {te_acc_adam[-1]:.2f}%")
```

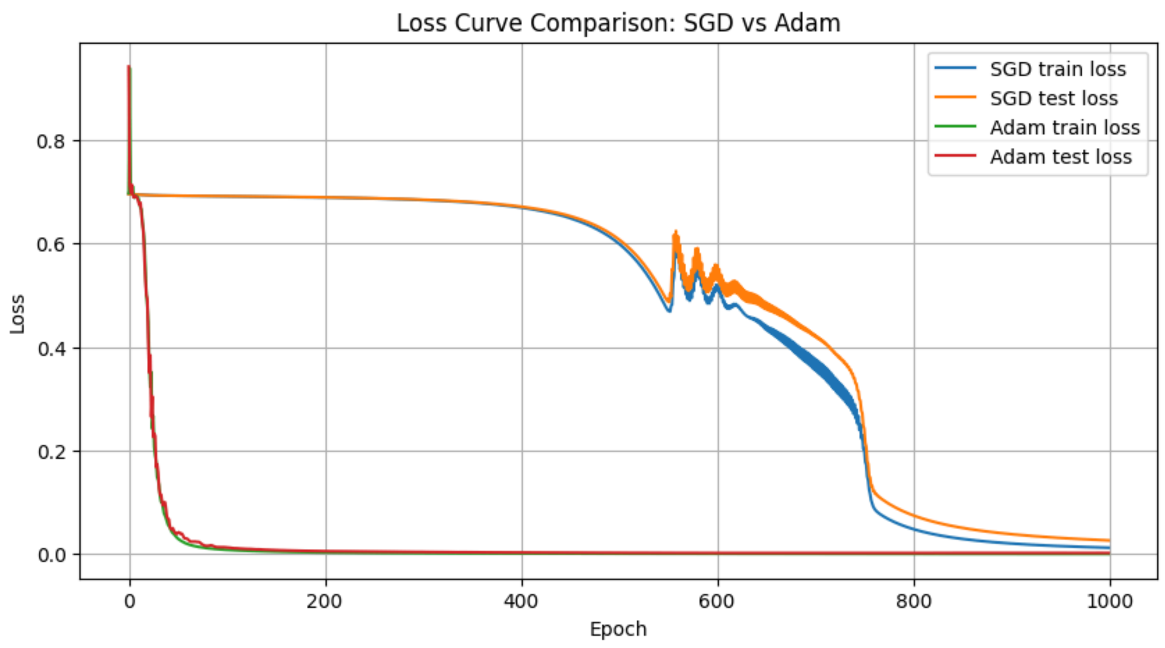
```
Epoch: 0 | Loss: 0.69779, Accuracy: 50.00% | Test loss: 0.94223, Test acc: 50.00%
Epoch: 10 | Loss: 0.68689, Accuracy: 57.00% | Test loss: 0.68008, Test acc: 54.0
0%
Epoch: 20 | Loss: 0.44838, Accuracy: 70.88% | Test loss: 0.40923, Test acc: 94.5
0%
Epoch: 30 | Loss: 0.14387, Accuracy: 99.25% | Test loss: 0.15797, Test acc: 99.0
0%
Epoch: 40 | Loss: 0.05702, Accuracy: 100.00% | Test loss: 0.06856, Test acc: 99.5
0%
Epoch: 50 | Loss: 0.03010, Accuracy: 100.00% | Test loss: 0.04079, Test acc: 99.5
0%
Epoch: 60 | Loss: 0.01944, Accuracy: 100.00% | Test loss: 0.02918, Test acc: 99.5
0%
Epoch: 70 | Loss: 0.01428, Accuracy: 100.00% | Test loss: 0.02341, Test acc: 99.5
0%
Epoch: 80 | Loss: 0.01141, Accuracy: 100.00% | Test loss: 0.01581, Test acc: 99.5
0%
Epoch: 90 | Loss: 0.00954, Accuracy: 100.00% | Test loss: 0.01346, Test acc: 100.
00%
Epoch: 100 | Loss: 0.00822, Accuracy: 100.00% | Test loss: 0.01256, Test acc: 99.
50%
Epoch: 110 | Loss: 0.00721, Accuracy: 100.00% | Test loss: 0.01123, Test acc: 10
0.00%
Epoch: 120 | Loss: 0.00641, Accuracy: 100.00% | Test loss: 0.00996, Test acc: 10
0.00%
Epoch: 130 | Loss: 0.00575, Accuracy: 100.00% | Test loss: 0.00908, Test acc: 10
0.00%
Epoch: 140 | Loss: 0.00521, Accuracy: 100.00% | Test loss: 0.00825, Test acc: 10
0.00%
Epoch: 150 | Loss: 0.00474, Accuracy: 100.00% | Test loss: 0.00765, Test acc: 10
0.00%
Epoch: 160 | Loss: 0.00434, Accuracy: 100.00% | Test loss: 0.00703, Test acc: 10
0.00%
Epoch: 170 | Loss: 0.00399, Accuracy: 100.00% | Test loss: 0.00667, Test acc: 10
0.00%
Epoch: 180 | Loss: 0.00369, Accuracy: 100.00% | Test loss: 0.00618, Test acc: 10
0.00%
Epoch: 190 | Loss: 0.00342, Accuracy: 100.00% | Test loss: 0.00596, Test acc: 10
0.00%
Epoch: 200 | Loss: 0.00319, Accuracy: 100.00% | Test loss: 0.00557, Test acc: 10
0.00%
Epoch: 210 | Loss: 0.00298, Accuracy: 100.00% | Test loss: 0.00544, Test acc: 10
0.00%
Epoch: 220 | Loss: 0.00279, Accuracy: 100.00% | Test loss: 0.00525, Test acc: 10
0.00%
Epoch: 230 | Loss: 0.00262, Accuracy: 100.00% | Test loss: 0.00505, Test acc: 10
0.00%
Epoch: 240 | Loss: 0.00247, Accuracy: 100.00% | Test loss: 0.00484, Test acc: 10
0.00%
Epoch: 250 | Loss: 0.00233, Accuracy: 100.00% | Test loss: 0.00467, Test acc: 10
0.00%
Epoch: 260 | Loss: 0.00221, Accuracy: 100.00% | Test loss: 0.00459, Test acc: 10
0.00%
Epoch: 270 | Loss: 0.00209, Accuracy: 100.00% | Test loss: 0.00441, Test acc: 10
0.00%
Epoch: 280 | Loss: 0.00199, Accuracy: 100.00% | Test loss: 0.00427, Test acc: 10
0.00%
Epoch: 290 | Loss: 0.00189, Accuracy: 100.00% | Test loss: 0.00418, Test acc: 10
0.00%
Epoch: 300 | Loss: 0.00180, Accuracy: 100.00% | Test loss: 0.00406, Test acc: 10
```

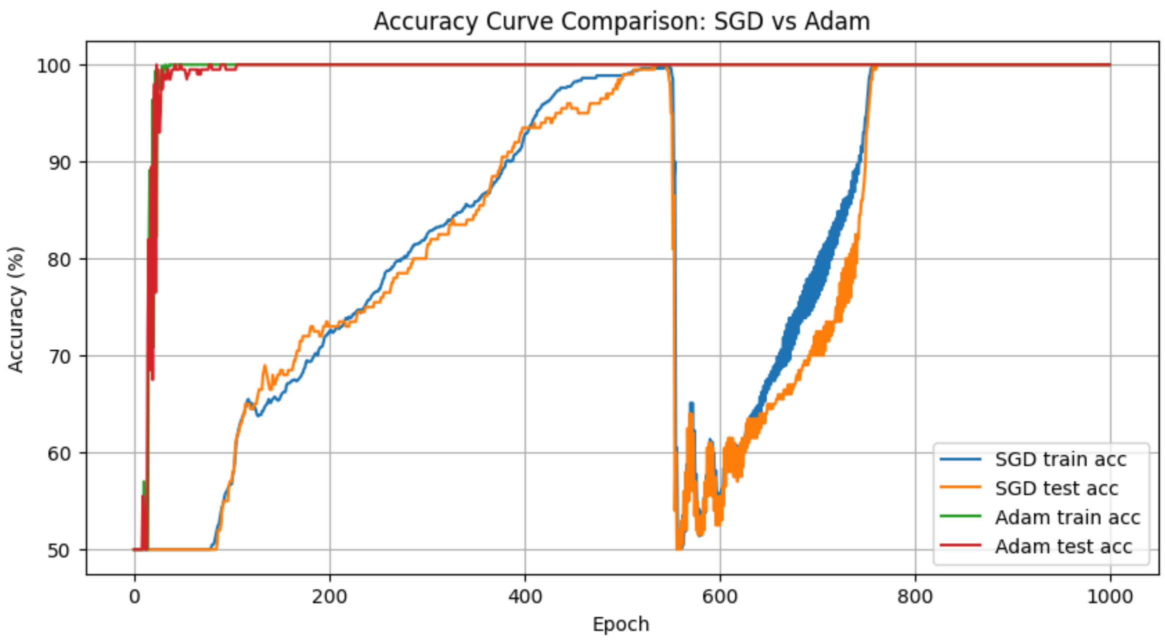
```
0.00%
Epoch: 310 | Loss: 0.00171, Accuracy: 100.00% | Test loss: 0.00396, Test acc: 10
0.00%
Epoch: 320 | Loss: 0.00164, Accuracy: 100.00% | Test loss: 0.00384, Test acc: 10
0.00%
Epoch: 330 | Loss: 0.00157, Accuracy: 100.00% | Test loss: 0.00377, Test acc: 10
0.00%
Epoch: 340 | Loss: 0.00150, Accuracy: 100.00% | Test loss: 0.00365, Test acc: 10
0.00%
Epoch: 350 | Loss: 0.00144, Accuracy: 100.00% | Test loss: 0.00347, Test acc: 10
0.00%
Epoch: 360 | Loss: 0.00138, Accuracy: 100.00% | Test loss: 0.00347, Test acc: 10
0.00%
Epoch: 370 | Loss: 0.00132, Accuracy: 100.00% | Test loss: 0.00328, Test acc: 10
0.00%
Epoch: 380 | Loss: 0.00127, Accuracy: 100.00% | Test loss: 0.00323, Test acc: 10
0.00%
Epoch: 390 | Loss: 0.00122, Accuracy: 100.00% | Test loss: 0.00315, Test acc: 10
0.00%
Epoch: 400 | Loss: 0.00118, Accuracy: 100.00% | Test loss: 0.00305, Test acc: 10
0.00%
Epoch: 410 | Loss: 0.00114, Accuracy: 100.00% | Test loss: 0.00300, Test acc: 10
0.00%
Epoch: 420 | Loss: 0.00110, Accuracy: 100.00% | Test loss: 0.00289, Test acc: 10
0.00%
Epoch: 430 | Loss: 0.00106, Accuracy: 100.00% | Test loss: 0.00285, Test acc: 10
0.00%
Epoch: 440 | Loss: 0.00102, Accuracy: 100.00% | Test loss: 0.00276, Test acc: 10
0.00%
Epoch: 450 | Loss: 0.00099, Accuracy: 100.00% | Test loss: 0.00271, Test acc: 10
0.00%
Epoch: 460 | Loss: 0.00096, Accuracy: 100.00% | Test loss: 0.00265, Test acc: 10
0.00%
Epoch: 470 | Loss: 0.00093, Accuracy: 100.00% | Test loss: 0.00260, Test acc: 10
0.00%
Epoch: 480 | Loss: 0.00090, Accuracy: 100.00% | Test loss: 0.00256, Test acc: 10
0.00%
Epoch: 490 | Loss: 0.00087, Accuracy: 100.00% | Test loss: 0.00252, Test acc: 10
0.00%
Epoch: 500 | Loss: 0.00084, Accuracy: 100.00% | Test loss: 0.00244, Test acc: 10
0.00%
Epoch: 510 | Loss: 0.00082, Accuracy: 100.00% | Test loss: 0.00241, Test acc: 10
0.00%
Epoch: 520 | Loss: 0.00080, Accuracy: 100.00% | Test loss: 0.00237, Test acc: 10
0.00%
Epoch: 530 | Loss: 0.00077, Accuracy: 100.00% | Test loss: 0.00236, Test acc: 10
0.00%
Epoch: 540 | Loss: 0.00075, Accuracy: 100.00% | Test loss: 0.00229, Test acc: 10
0.00%
Epoch: 550 | Loss: 0.00073, Accuracy: 100.00% | Test loss: 0.00229, Test acc: 10
0.00%
Epoch: 560 | Loss: 0.00071, Accuracy: 100.00% | Test loss: 0.00227, Test acc: 10
0.00%
Epoch: 570 | Loss: 0.00069, Accuracy: 100.00% | Test loss: 0.00220, Test acc: 10
0.00%
Epoch: 580 | Loss: 0.00067, Accuracy: 100.00% | Test loss: 0.00220, Test acc: 10
0.00%
Epoch: 590 | Loss: 0.00066, Accuracy: 100.00% | Test loss: 0.00215, Test acc: 10
0.00%
Epoch: 600 | Loss: 0.00064, Accuracy: 100.00% | Test loss: 0.00211, Test acc: 10
```

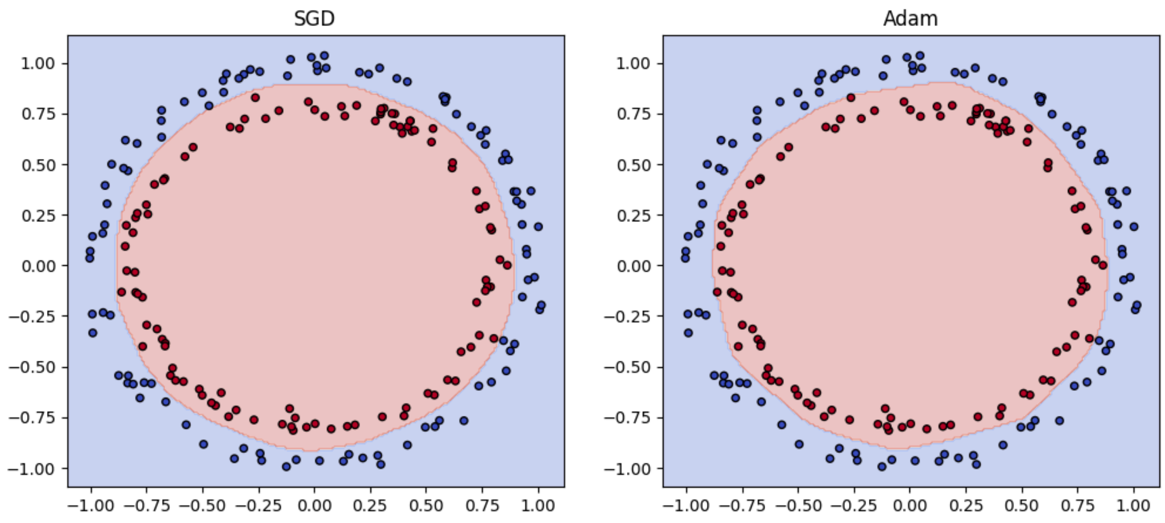
```
0.00%
Epoch: 610 | Loss: 0.00062, Accuracy: 100.00% | Test loss: 0.00214, Test acc: 10
0.00%
Epoch: 620 | Loss: 0.00061, Accuracy: 100.00% | Test loss: 0.00213, Test acc: 10
0.00%
Epoch: 630 | Loss: 0.00059, Accuracy: 100.00% | Test loss: 0.00209, Test acc: 10
0.00%
Epoch: 640 | Loss: 0.00058, Accuracy: 100.00% | Test loss: 0.00213, Test acc: 10
0.00%
Epoch: 650 | Loss: 0.00056, Accuracy: 100.00% | Test loss: 0.00210, Test acc: 10
0.00%
Epoch: 660 | Loss: 0.00055, Accuracy: 100.00% | Test loss: 0.00208, Test acc: 10
0.00%
Epoch: 670 | Loss: 0.00054, Accuracy: 100.00% | Test loss: 0.00212, Test acc: 10
0.00%
Epoch: 680 | Loss: 0.00053, Accuracy: 100.00% | Test loss: 0.00207, Test acc: 10
0.00%
Epoch: 690 | Loss: 0.00051, Accuracy: 100.00% | Test loss: 0.00209, Test acc: 10
0.00%
Epoch: 700 | Loss: 0.00050, Accuracy: 100.00% | Test loss: 0.00209, Test acc: 10
0.00%
Epoch: 710 | Loss: 0.00049, Accuracy: 100.00% | Test loss: 0.00212, Test acc: 10
0.00%
Epoch: 720 | Loss: 0.00048, Accuracy: 100.00% | Test loss: 0.00210, Test acc: 10
0.00%
Epoch: 730 | Loss: 0.00047, Accuracy: 100.00% | Test loss: 0.00209, Test acc: 10
0.00%
Epoch: 740 | Loss: 0.00046, Accuracy: 100.00% | Test loss: 0.00210, Test acc: 10
0.00%
Epoch: 750 | Loss: 0.00045, Accuracy: 100.00% | Test loss: 0.00212, Test acc: 10
0.00%
Epoch: 760 | Loss: 0.00044, Accuracy: 100.00% | Test loss: 0.00213, Test acc: 10
0.00%
Epoch: 770 | Loss: 0.00043, Accuracy: 100.00% | Test loss: 0.00213, Test acc: 10
0.00%
Epoch: 780 | Loss: 0.00042, Accuracy: 100.00% | Test loss: 0.00214, Test acc: 10
0.00%
Epoch: 790 | Loss: 0.00041, Accuracy: 100.00% | Test loss: 0.00215, Test acc: 10
0.00%
Epoch: 800 | Loss: 0.00041, Accuracy: 100.00% | Test loss: 0.00210, Test acc: 10
0.00%
Epoch: 810 | Loss: 0.00040, Accuracy: 100.00% | Test loss: 0.00214, Test acc: 10
0.00%
Epoch: 820 | Loss: 0.00039, Accuracy: 100.00% | Test loss: 0.00214, Test acc: 10
0.00%
Epoch: 830 | Loss: 0.00038, Accuracy: 100.00% | Test loss: 0.00215, Test acc: 10
0.00%
Epoch: 840 | Loss: 0.00038, Accuracy: 100.00% | Test loss: 0.00217, Test acc: 10
0.00%
Epoch: 850 | Loss: 0.00037, Accuracy: 100.00% | Test loss: 0.00220, Test acc: 10
0.00%
Epoch: 860 | Loss: 0.00036, Accuracy: 100.00% | Test loss: 0.00224, Test acc: 10
0.00%
Epoch: 870 | Loss: 0.00036, Accuracy: 100.00% | Test loss: 0.00218, Test acc: 10
0.00%
Epoch: 880 | Loss: 0.00035, Accuracy: 100.00% | Test loss: 0.00222, Test acc: 10
0.00%
Epoch: 890 | Loss: 0.00034, Accuracy: 100.00% | Test loss: 0.00224, Test acc: 10
0.00%
Epoch: 900 | Loss: 0.00034, Accuracy: 100.00% | Test loss: 0.00225, Test acc: 10
```

```
0.00%
Epoch: 910 | Loss: 0.00033, Accuracy: 100.00% | Test loss: 0.00221, Test acc: 10
0.00%
Epoch: 920 | Loss: 0.00032, Accuracy: 100.00% | Test loss: 0.00222, Test acc: 10
0.00%
Epoch: 930 | Loss: 0.00032, Accuracy: 100.00% | Test loss: 0.00224, Test acc: 10
0.00%
Epoch: 940 | Loss: 0.00031, Accuracy: 100.00% | Test loss: 0.00223, Test acc: 10
0.00%
Epoch: 950 | Loss: 0.00031, Accuracy: 100.00% | Test loss: 0.00224, Test acc: 10
0.00%
Epoch: 960 | Loss: 0.00030, Accuracy: 100.00% | Test loss: 0.00215, Test acc: 10
0.00%
Epoch: 970 | Loss: 0.00030, Accuracy: 100.00% | Test loss: 0.00224, Test acc: 10
0.00%
Epoch: 980 | Loss: 0.00029, Accuracy: 100.00% | Test loss: 0.00219, Test acc: 10
0.00%
Epoch: 990 | Loss: 0.00029, Accuracy: 100.00% | Test loss: 0.00221, Test acc: 10
0.00%
Adam final test acc: 100.00%
```

In [35]:
```python
# comparison plots using stored SGD/Adam results
plt.figure(figsize=(10,5))
# loss curves
epochs = range(len(tr_losses_sgd))
plt.plot(epochs, tr_losses_sgd, label="SGD train loss")
plt.plot(epochs, te_losses_sgd, label="SGD test loss")
plt.plot(epochs, tr_losses_adam, label="Adam train loss")
plt.plot(epochs, te_losses_adam, label="Adam test loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.title("Loss Curve Comparison: SGD vs Adam")
plt.legend()
plt.grid(True)
plt.show()

plt.figure(figsize=(10,5))
# accuracy curves
plt.plot(epochs, tr_acc_sgd, label="SGD train acc")
plt.plot(epochs, te_acc_sgd, label="SGD test acc")
plt.plot(epochs, tr_acc_adam, label="Adam train acc")
plt.plot(epochs, te_acc_adam, label="Adam test acc")
plt.xlabel("Epoch")
plt.ylabel("Accuracy (%)")
plt.title("Accuracy Curve Comparison: SGD vs Adam")
plt.legend()
plt.grid(True)
plt.show()

# decision boundaries
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.title("SGD")
plot_decision_boundary(m_sgd, X_test, y_test)
plt.subplot(1,2,2)
plt.title("Adam")
plot_decision_boundary(m_adam, X_test, y_test)
plt.show()
```







Discussion and Conclusion

In this lab we built several neural network architectures to tackle a binary classification problem on a synthetic circles dataset. Initially **Model 0** contained two linear layers with no activation functions. It was severely underfitting: the decision boundaries were almost straight lines and both training and test accuracy remained low (~50–60%). Adding depth and width in **Model 1** improved capacity and accuracy, but without non-linearities the model still struggled to learn the circular decision boundary.

Introducing ReLU activations in **Model 2** made a dramatic difference. Even with a modest number of parameters the network learned non-linear patterns, producing curved decision regions that matched the data much more closely. Training for longer epochs (1000–1500) stabilized the loss and pushed test accuracy well above 90 %. The loss curves confirmed faster convergence and reduced gap between train and test losses, indicating that the model was no longer underfitting.

Optimizer Comparison: SGD vs Adam

We trained `ModelV2` using both **SGD** and **Adam** with the same learning rate. Adam consistently reached higher accuracy in fewer epochs due to its adaptive moment estimates, whereas SGD required more epochs to approach similar performance and sometimes exhibited noisier loss curves. The decision boundaries from the Adam-trained model appeared slightly smoother, although both optimizers ultimately produced acceptable classifiers. For simple problems like this, SGD can be adequate—but Adam accelerates training and is generally more forgiving when tuning hyperparameters.

Thus, the model complexity (depth, width, and activations) and appropriate optimization choices are key to transforming an underfitting linear model into a high-accuracy classifier. Overall, this lab demonstrated how incremental improvements lead to progressively better decision boundaries and how optimizer selection affects training dynamics.