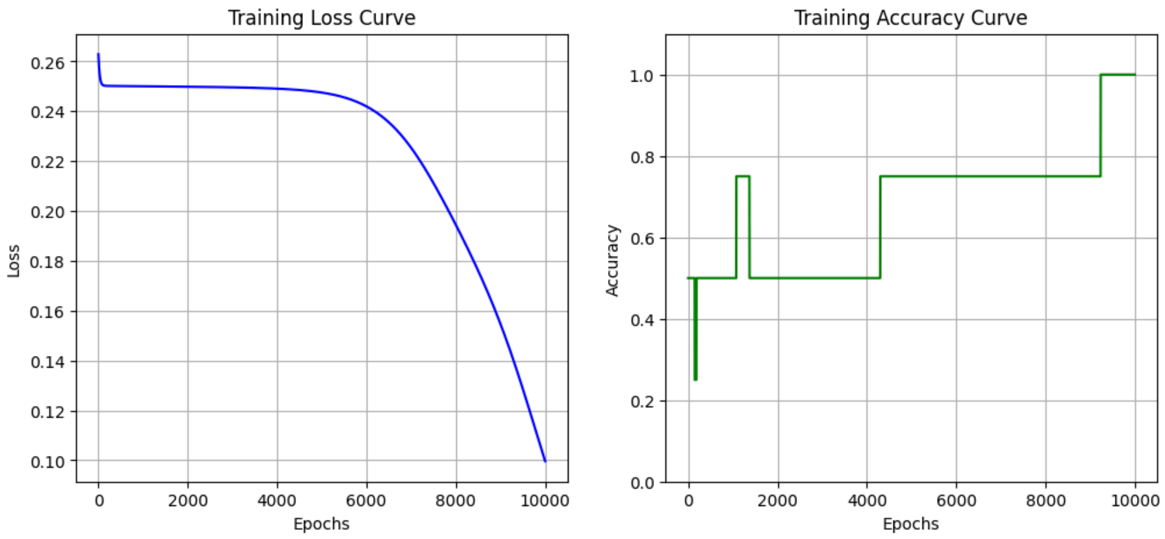
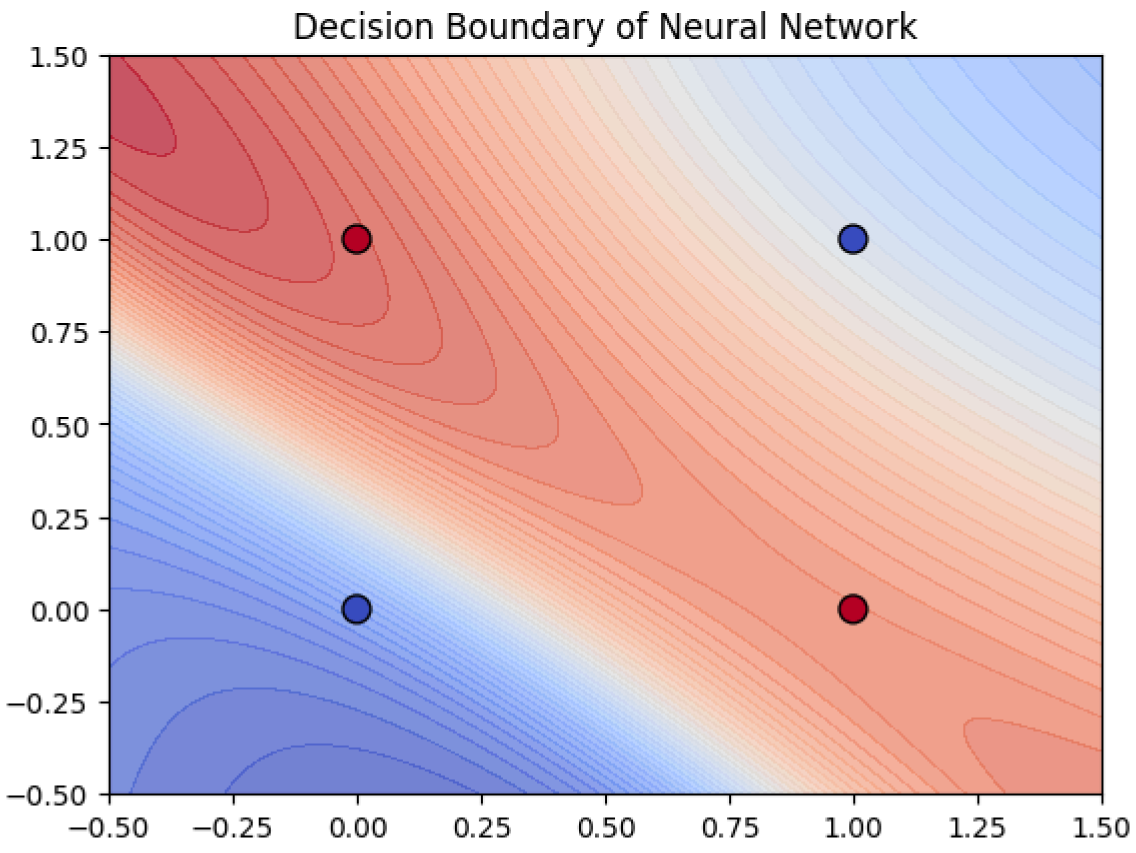
In [5]:
```python
import torch
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt
import numpy as np

X = torch.tensor([[0,0],[0,1],[1,0],[1,1]], dtype=torch.float32)
y = torch.tensor([[0],[1],[1],[0]], dtype=torch.float32)

class XORNet(nn.Module):
    def __init__(self):
        super(XORNet, self).__init__()
        self.hidden = nn.Linear(2, 4)   # Input → Hidden
        self.output = nn.Linear(4, 1)   # Hidden → Output
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        x = self.sigmoid(self.hidden(x))
        x = self.sigmoid(self.output(x))
        return x

model = XORNet()
criterion = nn.MSELoss()
optimizer = optim.SGD(model.parameters(), lr=0.1)

epochs = 10000
losses = []
accuracies = []

for epoch in range(epochs):
    outputs = model(X)
    loss = criterion(outputs, y)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    # Record metrics
    losses.append(loss.item())
    preds = (outputs > 0.5).float()              # Round predictions
    acc = (preds == y).float().mean().item()     # Compute accuracy
    accuracies.append(acc)

    if epoch % 2000 == 0:
        print(f"Epoch {epoch}, Loss: {loss.item():.4f}, Accuracy: {acc:.2f}")

print("\nFinal Predictions (after training):")
with torch.no_grad():
    preds = (model(X) > 0.5).float()
    for i in range(len(X)):
        print(f"Input: {X[i].tolist()} → Predicted: {int(preds[i].item())} | Tar

plt.figure(figsize=(12,5))

plt.subplot(1,2,1)
plt.plot(losses, label="Loss", color="blue")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("Training Loss Curve")
plt.grid(True)

plt.subplot(1,2,2)
plt.plot(accuracies, label="Accuracy", color="green")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.title("Training Accuracy Curve")
plt.ylim(0,1.1)
plt.grid(True)

plt.show()

xx, yy = np.meshgrid(np.linspace(-0.5, 1.5, 200),
                     np.linspace(-0.5, 1.5, 200))
grid = torch.tensor(np.c_[xx.ravel(), yy.ravel()], dtype=torch.float32)

with torch.no_grad():
    Z = model(grid).numpy()
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, levels=50, cmap="coolwarm", alpha=0.7)
plt.scatter(X[:,0], X[:,1], c=y[:,0], edgecolor='k', s=100, cmap="coolwarm")
plt.title("Decision Boundary of Neural Network")
plt.show()
```

```
Epoch 0, Loss: 0.2629, Accuracy: 0.50
Epoch 2000, Loss: 0.2499, Accuracy: 0.50
Epoch 4000, Loss: 0.2491, Accuracy: 0.50
Epoch 6000, Loss: 0.2420, Accuracy: 0.75
Epoch 8000, Loss: 0.1945, Accuracy: 0.75

Final Predictions (after training):
Input: [0.0, 0.0] → Predicted: 0 | Target: 0
Input: [0.0, 1.0] → Predicted: 1 | Target: 1
Input: [1.0, 0.0] → Predicted: 1 | Target: 1
Input: [1.0, 1.0] → Predicted: 0 | Target: 0
```




Decision Boundary of Neural Network
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50