DEEP LEARNING LAB EX-1

TASK 1

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons
```

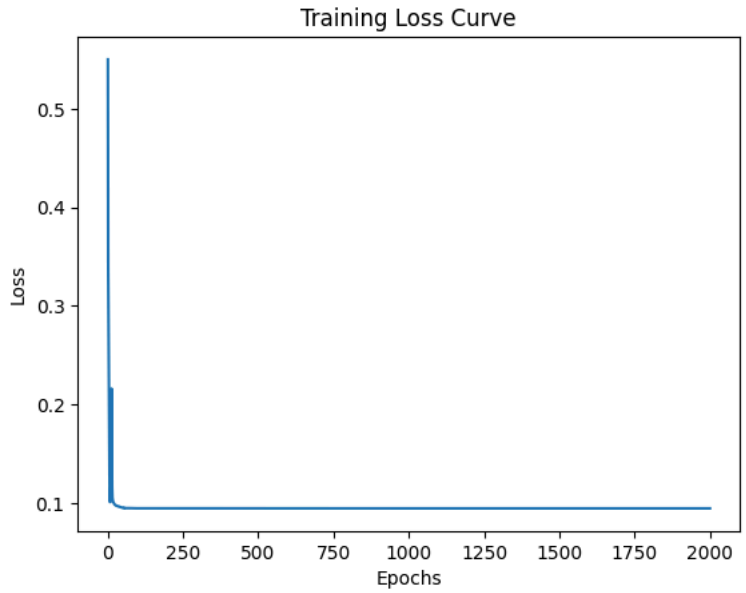
```
# Dataset
X, y = make_moons(n_samples=500, noise=0.2)
y = y.reshape(-1, 1)
```

```
# Initialization
np.random.seed(42)

W1 = np.random.randn(2, 8)
b1 = np.zeros((1, 8))
W2 = np.random.randn(8, 1)
b2 = np.zeros((1, 1))

lr = 0.01
epochs = 2000
losses = []
```

```
# Activation functions
def relu(z):
  return np.maximum(0, z)

def relu_derivative(z):
  return (z > 0).astype(float)

def sigmoid(z):
  return 1 / (1 + np.exp(-z))
```

```
# Training loop
for _ in range(epochs):
  z1 = X @ W1 + b1
  a1 = relu(z1)
  z2 = a1 @ W2 + b2
  y_hat = sigmoid(z2)

  loss = np.mean((y_hat - y) ** 2)

  losses.append(loss)

  dz2 = y_hat - y
  dW2 = a1.T @ dz2
  db2 = np.sum(dz2, axis=0, keepdims=True)

  da1 = dz2 @ W2.T
  dz1 = da1 * relu_derivative(z1)
  dW1 = X.T @ dz1
  db1 = np.sum(dz1, axis=0, keepdims=True)

  W2 -= lr * dW2
  b2 -= lr * db2
  W1 -= lr * dW1
  b1 -= lr * db1
```

```
# Loss plot
plt.plot(losses)
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("Training Loss Curve")
plt.show()
```



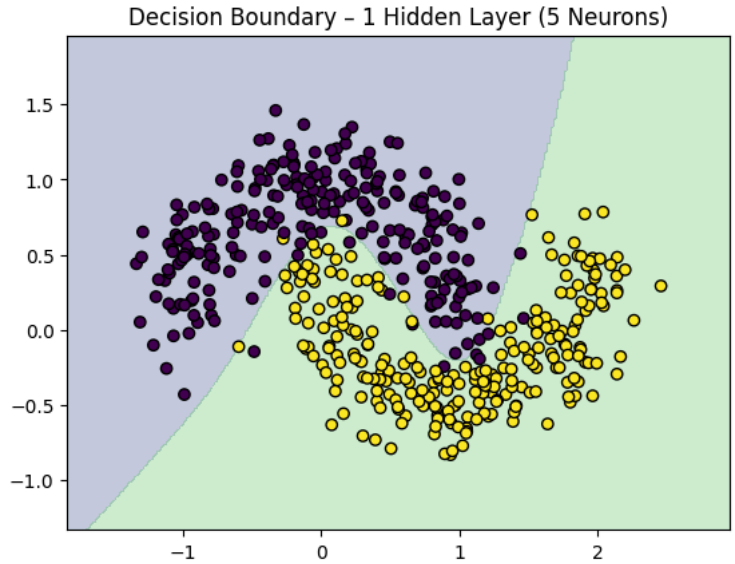
TASK 2

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons

# Dataset
X, y = make_moons(n_samples=500, noise=0.2)
y = y.reshape(-1, 1)

# Activation functions
def tanh(z):
    return np.tanh(z)

def tanh_derivative(z):
    return 1 - np.tanh(z) ** 2

def sigmoid(z):
    return 1 / (1 + np.exp(-z))
```

```
def plot_decision_boundary(model, X, y, title):
    x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
    y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5
    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 300),
                         np.linspace(y_min, y_max, 300))

    grid = np.c_[xx.ravel(), yy.ravel()]
    preds = model(grid).reshape(xx.shape)

    plt.contourf(xx, yy, preds > 0.5, alpha=0.3)
    plt.scatter(X[:, 0], X[:, 1], c=y.flatten(), edgecolors="k")
    plt.title(title)
    plt.show()
```

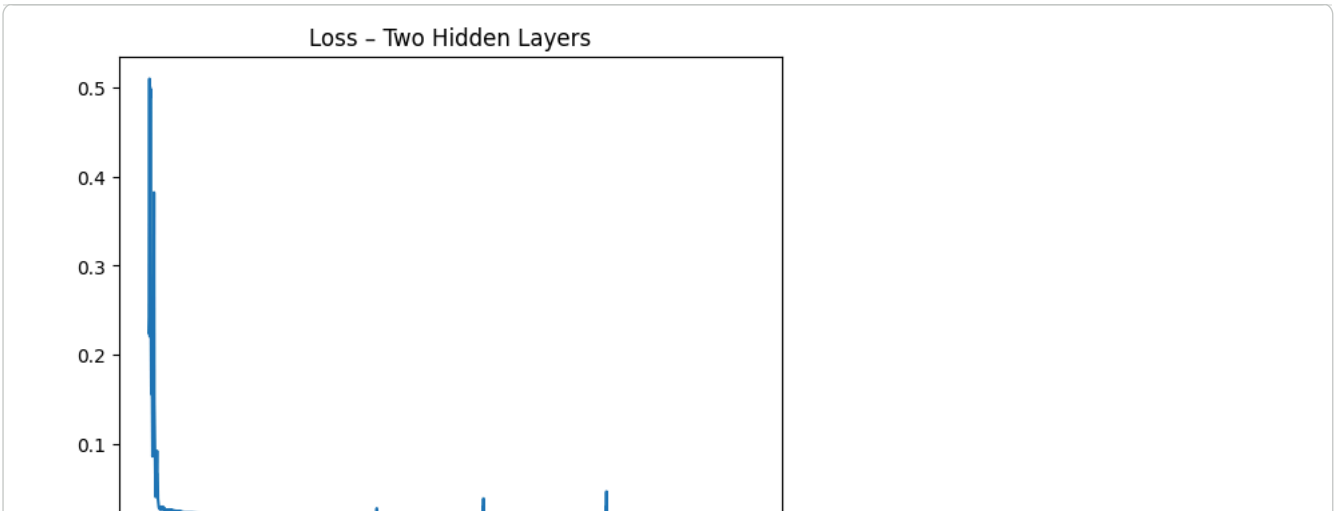
```
np.random.seed(42)

W1 = np.random.randn(2, 5)
b1 = np.zeros((1, 5))
W2 = np.random.randn(5, 1)
b2 = np.zeros((1, 1))

lr = 0.01
epochs = 2000
losses = []

for _ in range(epochs):
    z1 = X @ W1 + b1
    a1 = tanh(z1)
    z2 = a1 @ W2 + b2
    y_hat = sigmoid(z2)

    loss = np.mean((y_hat - y) ** 2)
```

```
    losses.append(loss)

    dz2 = y_hat - y
    dW2 = a1.T @ dz2
    db2 = np.sum(dz2, axis=0, keepdims=True)

    dz1 = (dz2 @ W2.T) * tanh_derivative(z1)
    dW1 = X.T @ dz1
    db1 = np.sum(dz1, axis=0, keepdims=True)

    W2 -= lr * dW2
    b2 -= lr * db2
    W1 -= lr * dW1
    b1 -= lr * db1

plt.plot(losses)
plt.title("Loss – Single Hidden Layer")
plt.show()

plot_decision_boundary(
    lambda x: sigmoid(tanh(x @ W1 + b1) @ W2 + b2),
    X, y,
    "Decision Boundary – 1 Hidden Layer (5 Neurons)"
)
```




```
W1 = np.random.randn(2, 16)
b1 = np.zeros((1, 16))
W2 = np.random.randn(16, 16)
b2 = np.zeros((1, 16))
W3 = np.random.randn(16, 1)
b3 = np.zeros((1, 1))

losses = []
```

```
for _ in range(epochs):
    z1 = X @ W1 + b1
    a1 = tanh(z1)
    z2 = a1 @ W2 + b2
    a2 = tanh(z2)
    z3 = a2 @ W3 + b3
    y_hat = sigmoid(z3)

    loss = np.mean((y_hat - y) ** 2)
    losses.append(loss)

    dz3 = y_hat - y
    dW3 = a2.T @ dz3
    db3 = np.sum(dz3, axis=0, keepdims=True)

    dz2 = (dz3 @ W3.T) * tanh_derivative(z2)
    dW2 = a1.T @ dz2
    db2 = np.sum(dz2, axis=0, keepdims=True)

    dz1 = (dz2 @ W2.T) * tanh_derivative(z1)
    dW1 = X.T @ dz1
    db1 = np.sum(dz1, axis=0, keepdims=True)

    W3 -= lr * dW3
    b3 -= lr * db3
    W2 -= lr * dW2
    b2 -= lr * db2
    W1 -= lr * dW1
    b1 -= lr * db1

plt.plot(losses)
plt.title("Loss – Two Hidden Layers")
plt.show()

plot_decision_boundary(
    lambda x: sigmoid(
        tanh(tanh(x @ W1 + b1) @ W2 + b2) @ W3 + b3
    ),
    X, y,
    "Decision Boundary – 2 Hidden Layers (16,16)"
)
```


Loss – Two Hidden Layers
0.5
0.4
0.3
0.2
0.1