import numpy as np

# Ground truth labels (one-hot encoded)

y\_true = np.array([

[1, 0, 0], # Sample 1: Class 1

[0, 1, 0], # Sample 2: Class 2

[0, 0, 1], # Sample 3: Class 3

[2, 3, 5], # Sample 4: Class 4

[3, 4, 9] # Sample 5: Class 5

])

# Model predictions (logits)

logits = np.array([

[2.0, 1.0, 0.1], # Sample 1

[0.5, 2.5, 0.2], # Sample 2

[0.1, 0.2, 3.0], # Sample 3

[2.5, 1.8, 9.1], # Sample 4

[4, 5, 1.9] # Sample 1

])

# Softmax function

def softmax(x):

exp\_x = np.exp(x - np.max(x, axis=-1, keepdims=True)) # Numerical stability

return exp\_x / np.sum(exp\_x, axis=-1, keepdims=True)

# Compute softmax probabilities

probs = softmax(logits)

print("Softmax Probabilities:\n", probs)

# Compute log(probs)

log\_probs = np.log(probs)

print("\nLog of Softmax Probabilities:\n", log\_probs)

# Compute y\_true \* log(probs)

y\_true\_log\_probs = y\_true \* log\_probs

print("\ny\_true \* log(probs):\n", y\_true\_log\_probs)

# Compute CE loss for each sample

ce\_loss = -np.sum(y\_true\_log\_probs, axis=-1)

print("\nCE Loss per Sample:", ce\_loss)

# Average CE loss for the dataset

avg\_ce\_loss = np.mean(ce\_loss)

print("\nAverage CE Loss for the Dataset:", avg\_ce\_loss)