

```

#zoom random 5e-4 500 epochs then 1100

#Have commented out a lot of print statements to try and save time
#03/07/25 Code updated to remove z score norm and check channel dims
#LR scheduler added
#Stratified split not random
#27/06/25 PVA calcs added and to print at end of all training done

import os
import re
import torch
import numpy as np
import torch.nn as nn
import torch.optim as optim
from collections import Counter
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms
import torchvision.models
from sklearn.model_selection import train_test_split
import timm
import matplotlib.pyplot as plt
from google.colab import files
uploaded = files.upload()
import sys
from math import sqrt
#From [name of imported file] import [name of class within that file]
from MBConvBlock import MBConvBlock
#From [name of imported file] import [name of class within that file]
from ScaledDotProductAttention import ScaledDotProductAttention
sys.path.append('.')
from torch.utils.data import random_split
from sklearn.metrics import confusion_matrix
import seaborn as sns
from sklearn.metrics import classification_report
from torch.utils.data import Subset
from torch.optim import lr_scheduler
from torch.nn.functional import pad
from torch.optim import lr_scheduler
import random

# Set device to GPU if available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
def set_seed(seed=42):
    random.seed(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed(seed)
    torch.backends.cudnn.deterministic = True
    torch.backends.cudnn.benchmark = False

set_seed(42)
#The pre-processing pipeline already performed z-score normalisation and channel changes, therefore
#the tensors being loaded are already shape [1, 224, 224]

def extract_number(filename):
    """Extracts numbers for sorting files like 'file_23.pt'."""
    match = re.search(r'(\d+)', filename)
    return int(match.group(1)) if match else 0

def generate_labels_from_filenames(mel_spectrogram_files, files_per_class=500):
    """
    Generates integer class labels based on file order.
    Example: 0 for first 25 files, 1 for next 25, etc.
    """
    mel_spectrogram_files.sort(key=extract_number)
    labels = [idx // files_per_class for idx in range(len(mel_spectrogram_files))]

    for idx, file in enumerate(mel_spectrogram_files):
        print(f"File: {file}, Label: {labels[idx]}")
    return labels

def check_labels(mel_spectrogram_files, labels):
    print("Checking file-label mapping:")
    for file, label in zip(mel_spectrogram_files, labels):
        print(f"File: {file} -> Label: {label}")

def collate_pad(batch):
    tensors, labels = zip(*batch)

    # Find max time dimension
    max_len = max(tensor.shape[-1] for tensor in tensors)

```

```

# Pad all tensors to max_len
padded_tensors = []
for tensor in tensors:
    pad_len = max_len - tensor.shape[-1]
    padded_tensor = pad(tensor, (0, pad_len)) # pad last dimension
    padded_tensors.append(padded_tensor)

return torch.stack(padded_tensors), torch.tensor(labels)
class MelSpectrogramDataset(Dataset):
    def __init__(self, mel_spectrogram_dir, mel_spectrogram_files, labels, transform=None):
        self.mel_spectrogram_files = mel_spectrogram_files
        self.labels = labels
        self.transform = transform
        self.mel_spectrogram_dir = mel_spectrogram_dir

        if len(self.mel_spectrogram_files) != len(self.labels):
            raise ValueError("Mismatch between number of files and labels.")

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

    def __getitem__(self, idx):
        file_name = self.mel_spectrogram_files[idx]
        path = os.path.join(self.mel_spectrogram_dir, file_name)

        mel = torch.load(path)
        label = int(self.labels[idx]) # Ensure label is integer
        #shouldn't need this next line anymore as my tensors should be
        #shape 1, 224, 224
        #if len(mel.shape) == 2:
        #    mel = mel.unsqueeze(0) # [1, H, W] - #✅ # Add channel dimensions i.e. change from [224, 224] to [1, 224, 224]

        # if self.transform:
        #    mel = nn.functional.interpolate(mel.unsqueeze(0), size=(224, 224), mode='bilinear', align_corners=False).squeeze(0)
        #    mel = self.transform(mel) # Resize + Normalize

        if mel.shape[0] == 1:
            mel = mel.repeat(3, 1, 1) #Duplicate channels to match CoAtNet input ([3, H, W]) i.e. [3, 224, 224]
        #print(f"Tensor shape in Dataset __getitem__: {mel.shape}") #Check that tensor is [3, 224, 224]
        return mel, label

class CoAtNet(nn.Module):
    def __init__(self, in_ch, image_size, num_classes=36, out_chs=[64,96,192,384,768]):
        super(CoAtNet, self).__init__()
        self.out_chs = out_chs
        self.maxpool2d = nn.MaxPool2d(kernel_size=2, stride=2)
        self.maxpool1d = nn.MaxPool1d(kernel_size=2, stride=2)

        self.s0 = nn.Sequential(
            nn.Conv2d(in_ch, in_ch, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.Conv2d(in_ch, in_ch, kernel_size=3, padding=1)
        )
        self.mlp0 = nn.Sequential(
            nn.Conv2d(in_ch, out_chs[0], kernel_size=1),
            nn.ReLU(),
            nn.Conv2d(out_chs[0], out_chs[0], kernel_size=1)
        )
        self.s1 = MBConvBlock(ksize=3, input_filters=out_chs[0], output_filters=out_chs[0], image_size=image_size//2)
        self.mlp1 = nn.Sequential(
            nn.Conv2d(out_chs[0], out_chs[1], kernel_size=1),
            nn.ReLU(),
            nn.Conv2d(out_chs[1], out_chs[1], kernel_size=1)
        )
        self.s2 = MBConvBlock(ksize=3, input_filters=out_chs[1], output_filters=out_chs[1], image_size=image_size//4)
        self.mlp2 = nn.Sequential(
            nn.Conv2d(out_chs[1], out_chs[2], kernel_size=1),
            nn.ReLU(),
            nn.Conv2d(out_chs[2], out_chs[2], kernel_size=1)
        )
        self.s3 = ScaledDotProductAttention(out_chs[2], out_chs[2]//8, out_chs[2]//8, 8)
        self.mlp3 = nn.Sequential(
            nn.Linear(out_chs[2], out_chs[3]),
            nn.ReLU(),
            nn.Linear(out_chs[3], out_chs[3])
        )
        self.s4 = ScaledDotProductAttention(out_chs[3], out_chs[3]//8, out_chs[3]//8, 8)
        self.mlp4 = nn.Sequential(
            nn.Linear(out_chs[3], out_chs[4]),
            nn.ReLU(),
            nn.Linear(out_chs[4], out_chs[4])

```

```

    )

    self.avgpool = nn.AdaptiveAvgPool1d(1) # Avg pool over the sequence length (N)
    self.fc = nn.Linear(out_chs[4], num_classes)

    # Define softmax for output probabilities
    self.softmax = nn.Softmax(dim=1)

def forward(self, x):
    B, C, H, W = x.shape
    #print(f"Input shape: {x.shape}") # Expect [B, 3, 224, 224] #Debugging to check input shape as expected

    # Stage 0: Conv + MLP + MaxPool
    y = self.mlp0(self.s0(x))
    #print(f"After s0 and mlp0: {y.shape}") # Should keep spatial dims same as s0 output
    # show_feature_map(y, "Stage 0")
    y = self.maxpool2d(y)
    #print(f"After maxpool2d 0: {y.shape}") # spatial dims should halve here

    # Stage 1: MBConv + MLP + MaxPool
    y = self.mlp1(self.s1(y))
    #print(f"After s1 and mlp1: {y.shape}")
    #show_feature_map(y, "Stage 1")
    y = self.maxpool2d(y)
    #print(f"After maxpool2d 1: {y.shape}")

    # Stage 2: MBConv + MLP + MaxPool
    y = self.mlp2(self.s2(y))
    #print(f"After s2 and mlp2: {y.shape}")
    #show_feature_map(y, "Stage 2")
    y = self.maxpool2d(y)
    #print(f"After maxpool2d 2: {y.shape}")

    B, C, H, W = y.shape
    # Stage 3: Self Attention + MLP + MaxPool1d
    y = y.reshape(B, self.out_chs[2], -1).permute(0, 2, 1) # (B, N, C)
    #print(f"After reshape and permute for attention (stage 3): {y.shape}")
    y = self.mlp3(self.s3(y, y))
    #print(f"After s3 and mlp3: {y.shape}")
    y = self.maxpool1d(y.permute(0, 2, 1)).permute(0, 2, 1) # MaxPool over N
    #print(f"After maxpool1d 3: {y.shape}")

    # Stage 4: Self Attention + MLP + Global Average Pool + FC + Softmax
    y = self.mlp4(self.s4(y, y)) # y: (B, N, C)
    #print(f"After s4 and mlp4: {y.shape}")

    #print("Shape before permute:", y.shape) # (B, N, C)
    y = y.permute(0, 2, 1) # (B, C, N)
    #print("Shape after permute:", y.shape)

    y = self.avgpool(y) # (B, C, 1)
    #print("Shape after avgpool:", y.shape)

    y = y.squeeze(-1) # (B, C)
    #print("Shape after squeeze:", y.shape)

    y = self.fc(y) # (B, C)
    #print("Shape after fc:", y.shape)

    # Plot class probabilities for the first example in batch
    class_names = [f"Class {i}" for i in range(y.shape[1])]
    probs = y[0] # since batch size = 1

    #plt.figure(figsize=(10, 4))
    #plt.bar(class_names, probs.detach().cpu().numpy())
    #plt.title("Class Probabilities")
    #plt.xlabel("Classes")
    #plt.ylabel("Probability")
    #plt.xticks(rotation=45)
    #plt.show()

    return y

def main():
    tensor_folder = "/content/drive/MyDrive/ColabNotebooks/ZoomRecordings/ZoomTensorsOnly" #These are the Direct Phone recordings - all
    mel_files = [f for f in os.listdir(tensor_folder) if f.endswith(".pt")]

    # Create labels for 5 mel specs per keystroke i.e. 125 tensors per class
    labels = generate_labels_from_filenames(mel_files, files_per_class=500)
    num_classes = len(set(labels))
    print(f"Number of classes: {num_classes}")

```

```

#Check labels
mel_files = [f for f in os.listdir(tensor_folder) if f.endswith(".pt")]
labels = generate_labels_from_filenames(mel_files, files_per_class=500)
check_labels(mel_files, labels)

# Create Dataset & DataLoader
#dataset = MelSpectrogramDataset(tensor_folder, mel_files, labels, transform=transform)
# Full dataset
full_dataset = MelSpectrogramDataset(tensor_folder, mel_files, labels, transform=None)

# Convert labels to numpy for sklearn
labels_np = np.array(labels)
indices = np.arange(len(labels))
#Stratified split of data
# First split: Train (80%) vs Temp (20%)
train_indices, temp_indices, y_train, y_temp = train_test_split(
    indices,
    labels_np,
    test_size=0.2,
    stratify=labels_np,
    random_state=42
)

# # Second split: Temp → Validation (10%) and Test (10%)
val_indices, test_indices, y_val, y_test = train_test_split(
    temp_indices,
    y_temp,
    test_size=0.5,
    stratify=y_temp,
    random_state=42
)

# train_indices, temp_indices = train_test_split(
#     indices,
#     test_size=0.2,
#     random_state=42,
#     shuffle=True
# )

# Second split: Temp → Validation (10%) and Test (10%)
#val_indices, test_indices = train_test_split(
#     temp_indices,
#     test_size=0.5,
#     random_state=42,
#     shuffle=True
# )

# Create Subsets
train_dataset = Subset(full_dataset, train_indices)
validation_dataset = Subset(full_dataset, val_indices)
test_dataset = Subset(full_dataset, test_indices)

#Print the length of the dataset
print("Total number of samples in the dataset:", len(full_dataset))
train_ratio=0.6
validation_ratio=0.2
test_ratio=0.2
dataset_size = len(full_dataset)
train_size = int(train_ratio * dataset_size)
test_size=int(test_ratio * dataset_size)
validation_size = int(validation_ratio * dataset_size)

print("Train labels distribution:", np.bincount([label for _, label in train_dataset]))
print("Validation labels distribution:", np.bincount([label for _, label in validation_dataset]))
print("Test labels distribution:", np.bincount([label for _, label in test_dataset]))

train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True, collate_fn=collate_pad)
validation_loader=DataLoader(validation_dataset, batch_size=32, shuffle=False)
test_loader=DataLoader(test_dataset, batch_size=32, shuffle=False)
print(f'Total dataset size: {dataset_size}')
print(f'Training dataset size: {len(train_dataset)}')
print(f'Validation dataset size: {len(validation_dataset)}')
print(f'Test dataset size: {len(test_dataset)}')

def count_labels(subset, name):
    subset_labels = [full_dataset[i][1] for i in subset.indices]
    label_count = Counter(subset_labels)
    #print(f"{name} labels distribution:")

```

```

    #print(sorted(label_count.items()))
    #print()

count_labels(train_dataset, "Train")
count_labels(validation_dataset, "Validation")
count_labels(test_dataset, "Test")

#The CoAtNet model is defined in it's own CoAtNet custom class above
#3 input channels, image dimensions 224x224, no. output classes for classification
input_height= 224#no. mel freq bins
model = CoAtNet(in_ch=3, image_size=input_height, num_classes=num_classes) #Calls to my Custom CoAtNet model/matches format of it

##Moves model to GPU or CPU for training
model.to(device)

#Loss function is set to Cross entropy loss critereon
criterion = nn.CrossEntropyLoss()
#Sets optimiser to Adam and learning rate is specified
optimizer = optim.Adam(model.parameters(), lr=0.001)
train_losses = []
val_losses = []
train_accuracies = []
val_accuracies = []
#TRAINING LOOP#
scheduler = lr_scheduler.StepLR(optimizer, step_size=20, gamma=0.5) #LR scheduler
num_epochs = 500
best_val_accuracy=0.0 #Track peak validation accuracy (PVA)

#Saving data to checkpoint as model keeps timing out
checkpoint_path = "/content/drive/MyDrive/CoAtNet45495465484454262_checkpoint.pth"
start_epoch = 0

# Load checkpoint if it exists
if os.path.exists(checkpoint_path):
    print("Loading checkpoint...")
    checkpoint = torch.load(checkpoint_path, map_location=device)
    file_to_label=checkpoint.get('file_to_label', None)
    model.load_state_dict(checkpoint['model_state_dict'])
    optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
    scheduler.load_state_dict(checkpoint['scheduler_state_dict'])
    start_epoch = checkpoint['epoch'] + 1 # resume from next epoch
    best_val_accuracy = checkpoint.get('best_val_accuracy', 0.0)
    print(f"Resumed from epoch {start_epoch}, best validation accuracy so far: {best_val_accuracy:.2f}%")
total_epochs = 500
for epoch in range(start_epoch, num_epochs):
    model.train() #Sets the model to training mode enabling related features
    running_loss = 0.0 #Cumulative loss for the epoch
    correct = 0 #Correct prediction count
    total = 0 #Total sample count
#Iterates over batches of training data from train_loader
#Each batch contains images (input data) and labels (ground truth)

    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)
        #print("Images shape:", images.shape) # e.g., torch.Size([64, 3, 224, 224])
        #print("Labels:", labels[:10]) # should be integers in [0, 35]
#Passes the input images through the model to get predictions
        outputs = model(images)
        #Computes the loss (how far the model predictions (outputs) are from the actual labels using a loss function called crite
        loss = criterion(outputs, labels)
#Clears any gradients from the previous step to avoid the accumulation of gradients
        optimizer.zero_grad()
        #Performs back propagation
        loss.backward()
        #Updates weights
        optimizer.step()
#Track the loss and accuracy
        running_loss += loss.item()#Adds current loss to total running loss
        _, predicted = outputs.max(1)#Checks how many predictions are correct
        total += labels.size(0)#No. samples processed
        correct += predicted.eq(labels).sum().item()#Number of correct predictions
#Prints the summary of each epoch
        accuracy = 100 * correct / total
        train_losses.append(running_loss / len(train_loader))
        train_accuracies.append(accuracy)
        scheduler.step() #Steps the learning rate scheduler after each epoch (not after each batch)

    torch.save({
        'epoch': epoch,
        'model_state_dict': model.state_dict(),
        'optimizer_state_dict': optimizer.state_dict(),
        'scheduler_state_dict': scheduler.state_dict(),
        'best_val_accuracy': best_val_accuracy,

```

```

'file_to_label': {f: l for f, l in zip(mel_files, labels)}
}, checkpoint_path)
print(f"Checkpoint saved at epoch {epoch + 1}")

#EVALUATION LOOP#This is called immediately after the training loop within the same epoch "for" loop
model.eval() #set the model to evaluation mode (same as Validation)
val_loss = 0.0
val_correct = 0
val_total = 0
with torch.no_grad():
    for images, labels in validation_loader:
        images, labels = images.to(device), labels.to(device)
        outputs = model(images)
        loss = criterion(outputs, labels)
        val_loss +=loss.item()

        _, predicted = outputs.max(1)
        val_total += labels.size(0)
        val_correct += predicted.eq(labels).sum().item()

    avg_val_loss = val_loss / len(validation_loader)
    val_accuracy = 100 * val_correct / val_total
    val_losses.append(avg_val_loss)
    val_accuracies.append(val_accuracy)

    # Update best validation accuracy if current is better
    if val_accuracy > best_val_accuracy:
        best_val_accuracy = val_accuracy

    print(f"Epoch [{epoch+1}/{num_epochs}] - Loss: {running_loss:.4f}, Accuracy: {accuracy:.2f}%")
    print(f"Validation - Loss: {avg_val_loss:.4f}, Accuracy: {val_accuracy:.2f}%")

#Print PVA
print(f"\nPeak Validation Accuracy: {best_val_accuracy:.2f}%")

#Plot line plots of training & validation loss & accuracy per epoch
plt.plot(train_losses, label='Train Loss')
plt.plot(val_losses, label='Validation Loss')
plt.legend()
plt.title('Loss over Epochs')
plt.show()

plt.plot(train_accuracies, label='Train Acc')
plt.plot(val_accuracies, label='Val Acc')
plt.legend()
plt.title('Accuracy over Epochs')
plt.show()

# Evaluation on test set
model.eval()
all_preds = []
all_labels = []

with torch.no_grad():
    for images, labels in test_loader:
        images, labels = images.to(device), labels.to(device)
        outputs = model(images)
        _, predicted = outputs.max(1)
        all_preds.extend(predicted.cpu().numpy())
        all_labels.extend(labels.cpu().numpy())

# Confusion matrix
cm = confusion_matrix(all_labels, all_preds)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()

# Classification report
print(classification_report(all_labels, all_preds))
#Produce a confusion matrix to analyse the results after the test loop
#Produce a classification report to analyse the results after the test loop

if __name__ == "__main__":
    main()

```



Choose files 2 files

- ScaledDotProductAttention.py(text/x-python) - 3527 bytes, last modified: 05/07/2025 - 100% done
- MBConvBlock.py(text/x-python) - 7149 bytes, last modified: 06/06/2025 - 100% done

Streaming output truncated to the last 5000 lines.

File: r\_mel\_spec\_683\_v11.pt -> Label: 27  
File: r\_mel\_spec\_683\_v12.pt -> Label: 27  
File: r\_mel\_spec\_683\_v13.pt -> Label: 27  
File: r\_mel\_spec\_683\_v14.pt -> Label: 27  
File: r\_mel\_spec\_683\_v15.pt -> Label: 27  
File: r\_mel\_spec\_683\_v16.pt -> Label: 27  
File: r\_mel\_spec\_683\_v17.pt -> Label: 27  
File: r\_mel\_spec\_683\_v18.pt -> Label: 27  
File: r\_mel\_spec\_683\_v19.pt -> Label: 27  
File: r\_mel\_spec\_684\_v0.pt -> Label: 27  
File: r\_mel\_spec\_684\_v1.pt -> Label: 27  
File: r\_mel\_spec\_684\_v2.pt -> Label: 27  
File: r\_mel\_spec\_684\_v3.pt -> Label: 27  
File: r\_mel\_spec\_684\_v4.pt -> Label: 27  
File: r\_mel\_spec\_684\_v5.pt -> Label: 27  
File: r\_mel\_spec\_684\_v6.pt -> Label: 27  
File: r\_mel\_spec\_684\_v7.pt -> Label: 27  
File: r\_mel\_spec\_684\_v8.pt -> Label: 27  
File: r\_mel\_spec\_684\_v9.pt -> Label: 27  
File: r\_mel\_spec\_684\_v10.pt -> Label: 27  
File: r\_mel\_spec\_684\_v11.pt -> Label: 27  
File: r\_mel\_spec\_684\_v12.pt -> Label: 27  
File: r\_mel\_spec\_684\_v13.pt -> Label: 27  
File: r\_mel\_spec\_684\_v14.pt -> Label: 27  
File: r\_mel\_spec\_684\_v15.pt -> Label: 27  
File: r\_mel\_spec\_684\_v16.pt -> Label: 27  
File: r\_mel\_spec\_684\_v17.pt -> Label: 27  
File: r\_mel\_spec\_684\_v18.pt -> Label: 27  
File: r\_mel\_spec\_684\_v19.pt -> Label: 27  
File: r\_mel\_spec\_685\_v0.pt -> Label: 27  
File: r\_mel\_spec\_685\_v1.pt -> Label: 27  
File: r\_mel\_spec\_685\_v2.pt -> Label: 27  
File: r\_mel\_spec\_685\_v3.pt -> Label: 27  
File: r\_mel\_spec\_685\_v4.pt -> Label: 27  
File: r\_mel\_spec\_685\_v5.pt -> Label: 27  
File: r\_mel\_spec\_685\_v6.pt -> Label: 27  
File: r\_mel\_spec\_685\_v7.pt -> Label: 27  
File: r\_mel\_spec\_685\_v8.pt -> Label: 27  
File: r\_mel\_spec\_685\_v9.pt -> Label: 27  
File: r\_mel\_spec\_685\_v10.pt -> Label: 27  
File: r\_mel\_spec\_685\_v11.pt -> Label: 27  
File: r\_mel\_spec\_685\_v12.pt -> Label: 27  
File: r\_mel\_spec\_685\_v13.pt -> Label: 27  
File: r\_mel\_spec\_685\_v14.pt -> Label: 27  
File: r\_mel\_spec\_685\_v15.pt -> Label: 27  
File: r\_mel\_spec\_685\_v16.pt -> Label: 27  
File: r\_mel\_spec\_685\_v17.pt -> Label: 27  
File: r\_mel\_spec\_685\_v18.pt -> Label: 27  
File: r\_mel\_spec\_685\_v19.pt -> Label: 27  
File: r\_mel\_spec\_686\_v0.pt -> Label: 27  
File: r\_mel\_spec\_686\_v1.pt -> Label: 27  
File: r\_mel\_spec\_686\_v2.pt -> Label: 27  
File: r\_mel\_spec\_686\_v3.pt -> Label: 27  
File: r\_mel\_spec\_686\_v4.pt -> Label: 27  
File: r\_mel\_spec\_686\_v5.pt -> Label: 27  
File: r\_mel\_spec\_686\_v6.pt -> Label: 27  
File: r\_mel\_spec\_686\_v7.pt -> Label: 27  
File: r\_mel\_spec\_686\_v8.pt -> Label: 27  
File: r\_mel\_spec\_686\_v9.pt -> Label: 27  
File: r\_mel\_spec\_686\_v10.pt -> Label: 27  
File: r\_mel\_spec\_686\_v11.pt -> Label: 27  
File: r\_mel\_spec\_686\_v12.pt -> Label: 27  
File: r\_mel\_spec\_686\_v13.pt -> Label: 27  
File: r\_mel\_spec\_686\_v14.pt -> Label: 27  
File: r\_mel\_spec\_686\_v15.pt -> Label: 27  
File: r\_mel\_spec\_686\_v16.pt -> Label: 27  
File: r\_mel\_spec\_686\_v17.pt -> Label: 27  
File: r\_mel\_spec\_686\_v18.pt -> Label: 27  
File: r\_mel\_spec\_686\_v19.pt -> Label: 27  
File: r\_mel\_spec\_687\_v0.pt -> Label: 27  
File: r\_mel\_spec\_687\_v1.pt -> Label: 27  
File: r\_mel\_spec\_687\_v2.pt -> Label: 27  
File: r\_mel\_spec\_687\_v3.pt -> Label: 27  
File: r\_mel\_spec\_687\_v4.pt -> Label: 27  
File: r\_mel\_spec\_687\_v5.pt -> Label: 27  
File: r\_mel\_spec\_687\_v6.pt -> Label: 27  
File: r\_mel\_spec\_687\_v7.pt -> Label: 27  
File: r\_mel\_spec\_687\_v8.pt -> Label: 27  
File: r\_mel\_spec\_687\_v9.pt -> Label: 27  
File: r\_mel\_spec\_687\_v10.pt -> Label: 27  
File: r\_mel\_spec\_687\_v11.pt -> Label: 27  
File: r\_mel\_spec\_687\_v12.pt -> Label: 27  
File: r\_mel\_spec\_687\_v13.pt -> Label: 27  
File: r\_mel\_spec\_687\_v14.pt -> Label: 27  
File: r\_mel\_spec\_687\_v15.pt -> Label: 27  
File: r\_mel\_spec\_687\_v16.pt -> Label: 27

**THE UNIVERSITY OF CHICAGO**



[illegible]

111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1063 1064 1065 1066 1067 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083 1084 1085 1086 1087 1088 1089 1090 1091 1092 1093 1094 1095 1096 1097 1098 1099 1100 1101 1102 1103 1104 1105 1106 110

<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPikXwanzc9qBIY>

111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1063 1064 1065 1066 1067 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083 1084 1085 1086 1087 1088 1089 1090 1091 1092 1093 1094 1095 1096 1097 1098 1099 1100 1101 1102 1103 1104 1105 1106 110

[illegible]



lab.research.google.com/drive/1A49QqufsdA86oiQMDMPIkXwanzc9qBIY

— 117 —



<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPIkXwanzc9qBIY>

— 117 —

<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPIkXwanzc9qBIY>

— 199 —

[illegible]



<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPIkXwanzc9qBIY>





<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPIkXwanzc9qBIY>

<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPikXwanzc9qBIY>

<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPikXwanzc9qBIY>

— 116 —

<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPikXwanzc9qBIY>

<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPikXwanzc9qBIY>

<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPikXwanzc9qBIY>

<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPIkXwanzc9qBIY>



<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPIkXwanzc9qBIY>

<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPIkXwanzc9qBIY>

[illegible]

<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPIkXwanzc9qBIY>

[illegible]

<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPIkXwanzc9qBIY>

[illegible]

22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050 1051



[illegible]

<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPIkXwanzc9qBIY>

<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPIkXwanzc9qBIY>

<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPIkXwanzc9qBIY>

[illegible]

<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPIkXwanzc9qBIY>

[illegible]

<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPIkXwanzc9qBIY>



<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPIkXwanzc9qBIY>



<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPIkXwanzc9qBIY>

— 11 —

[illegible]

<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPikXwanzc9qBIY>

```
File: z_mel_spec_900_v10.pt -> Label: 35
File: z_mel_spec_900_v11.pt -> Label: 35
File: z_mel_spec_900_v12.pt -> Label: 35
File: z_mel_spec_900_v13.pt -> Label: 35
File: z_mel_spec_900_v14.pt -> Label: 35
File: z_mel_spec_900_v15.pt -> Label: 35
File: z_mel_spec_900_v16.pt -> Label: 35
File: z_mel_spec_900_v17.pt -> Label: 35
File: z_mel_spec_900_v18.pt -> Label: 35
File: z_mel_spec_900_v19.pt -> Label: 35
Total number of samples in the dataset: 18000
Train labels distribution: [400 400 400 400 400 400 400 400 400 400 400 400 400 400 400 400 400 400 400 400]
Validation labels distribution: [50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50]
Test labels distribution: [50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50]
Total dataset size: 18000
Training dataset size: 14400
Validation dataset size: 1800
Test dataset size: 1800
Loading checkpoint...
Resumed from epoch 288, best validation accuracy so far: 2.78%
Checkpoint saved at epoch 289
Epoch [289/500] - Loss: 1612.5836, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 290
Epoch [290/500] - Loss: 1612.5836, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 291
Epoch [291/500] - Loss: 1612.5836, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 292
Epoch [292/500] - Loss: 1612.5836, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 293
Epoch [293/500] - Loss: 1612.5836, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 294
Epoch [294/500] - Loss: 1612.5835, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 295
Epoch [295/500] - Loss: 1612.5836, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 296
Epoch [296/500] - Loss: 1612.5836, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 297
Epoch [297/500] - Loss: 1612.5836, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 298
Epoch [298/500] - Loss: 1612.5835, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 299
Epoch [299/500] - Loss: 1612.5836, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 300
Epoch [300/500] - Loss: 1612.5836, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 301
Epoch [301/500] - Loss: 1612.5836, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 302
Epoch [302/500] - Loss: 1612.5836, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 303
Epoch [303/500] - Loss: 1612.5836, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 304
Epoch [304/500] - Loss: 1612.5836, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 305
Epoch [305/500] - Loss: 1612.5836, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 306
Epoch [306/500] - Loss: 1612.5836, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 307
Epoch [307/500] - Loss: 1612.5836, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 308
Epoch [308/500] - Loss: 1612.5836, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 309
Epoch [309/500] - Loss: 1612.5835, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 310
Epoch [310/500] - Loss: 1612.5836, Accuracy: 2.78%
Validation - Loss: 3.5835, Accuracy: 2.78%
Checkpoint saved at epoch 311
Epoch [311/500] - Loss: 1612.5836, Accuracy: 2.78%
```

11/11/11 11:11:11



validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 342  
Epoch [342/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 343  
Epoch [343/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 344  
Epoch [344/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 345  
Epoch [345/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 346  
Epoch [346/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 347  
Epoch [347/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 348  
Epoch [348/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 349  
Epoch [349/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 350  
Epoch [350/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 351  
Epoch [351/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 352  
Epoch [352/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 353  
Epoch [353/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 354  
Epoch [354/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 355  
Epoch [355/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 356  
Epoch [356/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 357  
Epoch [357/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 358  
Epoch [358/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 359  
Epoch [359/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 360  
Epoch [360/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 361  
Epoch [361/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 362  
Epoch [362/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 363  
Epoch [363/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 364  
Epoch [364/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 365  
Epoch [365/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 366  
Epoch [366/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 367  
Epoch [367/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 368  
Epoch [368/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 369  
Epoch [369/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 370  
Epoch [370/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 371  
Epoch [371/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%

Checkpoint saved at epoch 372  
Epoch [372/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 373  
Epoch [373/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 374  
Epoch [374/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 375  
Epoch [375/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 376  
Epoch [376/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 377  
Epoch [377/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 378  
Epoch [378/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 379  
Epoch [379/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 380  
Epoch [380/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 381  
Epoch [381/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 382  
Epoch [382/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 383  
Epoch [383/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 384  
Epoch [384/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 385  
Epoch [385/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 386  
Epoch [386/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 387  
Epoch [387/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 388  
Epoch [388/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 389  
Epoch [389/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 390  
Epoch [390/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 391  
Epoch [391/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 392  
Epoch [392/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 393  
Epoch [393/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 394  
Epoch [394/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 395  
Epoch [395/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 396  
Epoch [396/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 397  
Epoch [397/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 398  
Epoch [398/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 399  
Epoch [399/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 400  
Epoch [400/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 401  
Epoch [401/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%

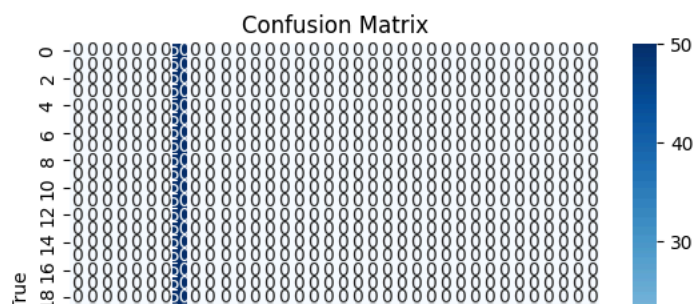
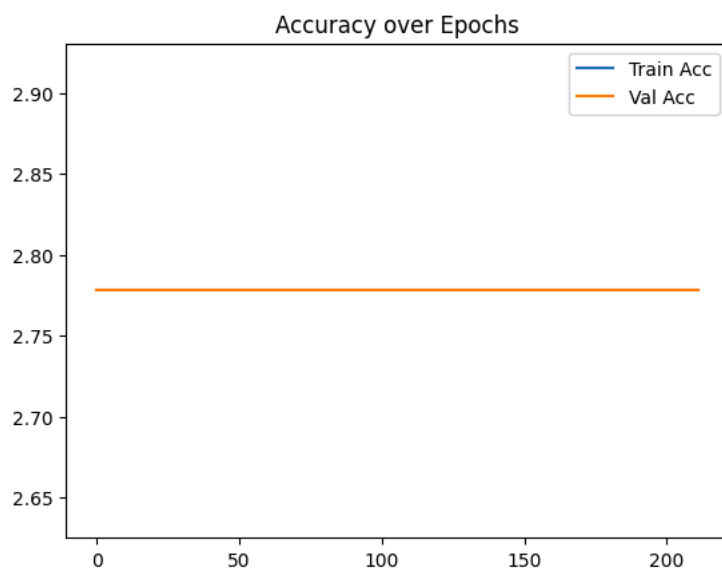
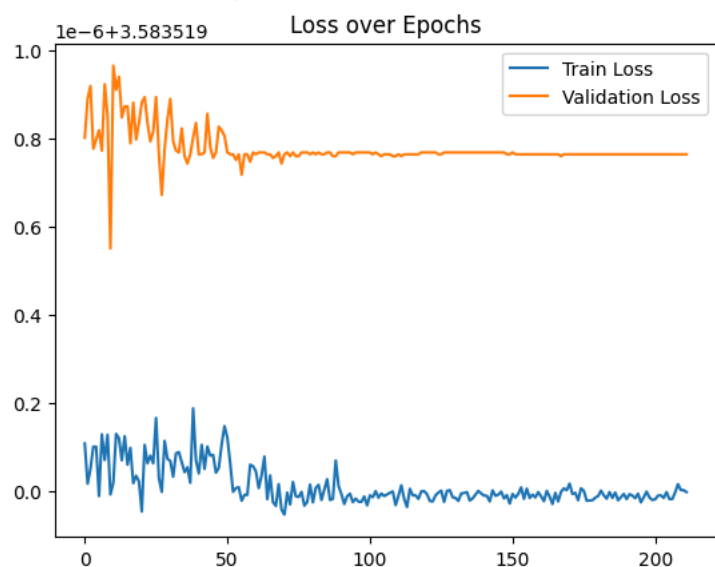
Checkpoint saved at epoch 402  
Epoch [402/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 403  
Epoch [403/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 404  
Epoch [404/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 405  
Epoch [405/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 406  
Epoch [406/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 407  
Epoch [407/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 408  
Epoch [408/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 409  
Epoch [409/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 410  
Epoch [410/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 411  
Epoch [411/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 412  
Epoch [412/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 413  
Epoch [413/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 414  
Epoch [414/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 415  
Epoch [415/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 416  
Epoch [416/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 417  
Epoch [417/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 418  
Epoch [418/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 419  
Epoch [419/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 420  
Epoch [420/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 421  
Epoch [421/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 422  
Epoch [422/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 423  
Epoch [423/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 424  
Epoch [424/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 425  
Epoch [425/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 426  
Epoch [426/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 427  
Epoch [427/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 428  
Epoch [428/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 429  
Epoch [429/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 430  
Epoch [430/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 431  
Epoch [431/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 432

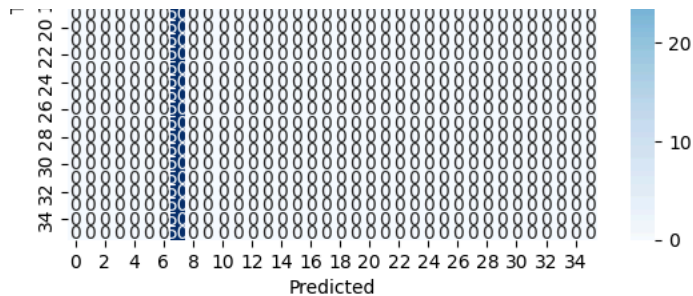
Epoch [432/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 433  
Epoch [433/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 434  
Epoch [434/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 435  
Epoch [435/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 436  
Epoch [436/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 437  
Epoch [437/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 438  
Epoch [438/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 439  
Epoch [439/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 440  
Epoch [440/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 441  
Epoch [441/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 442  
Epoch [442/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 443  
Epoch [443/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 444  
Epoch [444/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 445  
Epoch [445/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 446  
Epoch [446/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 447  
Epoch [447/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 448  
Epoch [448/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 449  
Epoch [449/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 450  
Epoch [450/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 451  
Epoch [451/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 452  
Epoch [452/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 453  
Epoch [453/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 454  
Epoch [454/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 455  
Epoch [455/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 456  
Epoch [456/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 457  
Epoch [457/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 458  
Epoch [458/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 459  
Epoch [459/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 460  
Epoch [460/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 461  
Epoch [461/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 462

<https://colab.research.google.com/drive/1A49QqufsdA86oiQMDMPIkXwanzc9qBIY>

Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 493  
Epoch [493/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 494  
Epoch [494/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 495  
Epoch [495/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 496  
Epoch [496/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 497  
Epoch [497/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 498  
Epoch [498/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 499  
Epoch [499/500] - Loss: 1612.5836, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%  
Checkpoint saved at epoch 500  
Epoch [500/500] - Loss: 1612.5835, Accuracy: 2.78%  
Validation - Loss: 3.5835, Accuracy: 2.78%

Peak Validation Accuracy: 2.78%





	precision	recall	f1-score	support
0	0.00	0.00	0.00	50
1	0.00	0.00	0.00	50
2	0.00	0.00	0.00	50
3	0.00	0.00	0.00	50
4	0.00	0.00	0.00	50
5	0.00	0.00	0.00	50
6	0.00	0.00	0.00	50
7	0.03	1.00	0.05	50
8	0.00	0.00	0.00	50
9	0.00	0.00	0.00	50
10	0.00	0.00	0.00	50
11	0.00	0.00	0.00	50
12	0.00	0.00	0.00	50
13	0.00	0.00	0.00	50
14	0.00	0.00	0.00	50
15	0.00	0.00	0.00	50
16	0.00	0.00	0.00	50
17	0.00	0.00	0.00	50
18	0.00	0.00	0.00	50
19	0.00	0.00	0.00	50
20	0.00	0.00	0.00	50
21	0.00	0.00	0.00	50
22	0.00	0.00	0.00	50
23	0.00	0.00	0.00	50
24	0.00	0.00	0.00	50
25	0.00	0.00	0.00	50
26	0.00	0.00	0.00	50
27	0.00	0.00	0.00	50
28	0.00	0.00	0.00	50
29	0.00	0.00	0.00	50
30	0.00	0.00	0.00	50
31	0.00	0.00	0.00	50
32	0.00	0.00	0.00	50
33	0.00	0.00	0.00	50
34	0.00	0.00	0.00	50
35	0.00	0.00	0.00	50
accuracy			0.03	1800
macro avg	0.00	0.03	0.00	1800
weighted avg	0.00	0.03	0.00	1800

```
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined ar
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined ar
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined ar
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```