todebug

November 27, 2025

[61]:
```
import sys
import numpy as np
import matplotlib.pyplot as plt
import pickle
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchvision import transforms
from torch.utils.data import Dataset
from torch.utils.data import DataLoader, random_split
import os

np.random.seed(0)
```

[62]:
```
if torch.backends.mps.is_available():
    device = torch.device("mps")
    use_mps = True
else:
    device = torch.device("cpu")
    use_mps = False

print(device)
```

```
mps
```

[63]:
```
class PKLDataset(Dataset):
    def __init__(self, path, transform=None):
        with open(path, "rb") as f:
            data = pickle.load(f)

        self.images = data["images"]        # shape: (N, 28, 28, 3)
        self.labels = data["labels"].reshape(-1)   # shape: (N,) instead of␣
↪(N,1)
        self.transform = transform

    def __len__(self):
        return len(self.images)
```

```
    def __getitem__(self, idx):
        img = self.images[idx]              # numpy array (28,28,3)
        label = int(self.labels[idx])       # convert to Python int

        # Convert to tensor and permute to (C, H, W)
        img = torch.tensor(img, dtype=torch.float32).permute(2, 0, 1) / 255.0

        if self.transform:
            img = self.transform(img)

        return img, label
```

[64]:
```
import pickle

with open("ift-3395-6390-kaggle-2-competition-fall-2025/train_data.pkl", "rb")␣
 ↪as f:
    data = pickle.load(f)

print(type(data))
print(len(data) if hasattr(data, "__len__") else "no len")
print(data)
```

```
<class 'dict'>
2
{'images': array([[[[ 6,  4,  0],
         [ 9,  5,  0],
         [ 8,  4,  0],
         …,
         [ 9,  6,  0],
         [ 9,  6,  0],
         [ 7,  4,  0]],

        [[11,  6,  0],
         [ 4,  4,  0],
         [ 3,  3,  0],
         …,
         [ 9,  6,  0],
         [ 6,  4,  0],
         [ 4,  2,  0]],

        [[11,  6,  0],
         [ 4,  4,  0],
         [ 3,  3,  0],
         …,
         [ 6,  4,  0],
         [ 6,  4,  0],
         [ 4,  2,  0]],
```

```
 …,

 [[ 1,  1,  0],
  [ 0,  0,  0],
  [ 0,  0,  1],
  …,
  [ 5,  4,  0],
  [ 6,  5,  0],
  [ 6,  5,  0]],

 [[ 3,  1,  1],
  [ 0,  0,  0],
  [ 0,  0,  1],
  …,
  [ 6,  5,  0],
  [ 6,  5,  0],
  [ 7,  6,  0]],

 [[10,  2,  2],
  [ 0,  0,  1],
  [ 0,  0,  1],
  …,
  [ 6,  5,  0],
  [ 7,  6,  0],
  [ 7,  6,  0]]],


[[[11,  9,  0],
  [ 9,  7,  0],
  [ 9,  7,  0],
  …,
  [ 0,  0,  1],
  [ 0,  0,  1],
  [ 0,  0,  1]],

 [[12,  9,  0],
  [11,  7,  0],
  [ 9,  6,  0],
  …,
  [ 0,  0,  2],
  [ 0,  0,  1],
  [ 0,  0,  1]],

 [[ 9,  7,  0],
  [12,  8,  0],
  [10,  6,  0],
  …,
```

```
  [ 0,  0,  1],
  [ 0,  0,  1],
  [ 0,  0,  0]],

 …,

 [[ 0,  0,  3],
  [ 0,  0,  3],
  [ 0,  0,  4],
  …,
  [ 0,  0,  2],
  [ 0,  0,  1],
  [ 0,  0,  0]],

 [[ 0,  0,  2],
  [ 0,  0,  2],
  [ 0,  0,  5],
  …,
  [ 0,  0,  2],
  [ 0,  0,  1],
  [ 1,  0,  0]],

 [[ 0,  0,  1],
  [ 0,  0,  2],
  [ 0,  0,  4],
  …,
  [ 0,  0,  1],
  [ 1,  0,  0],
  [ 1,  0,  0]]],


[[[18, 12,  0],
  [12,  9,  0],
  [17, 11,  0],
  …,
  [ 0,  0,  3],
  [ 5,  0,  2],
  [ 5,  0,  2]],

 [[15, 11,  0],
  [10,  9,  0],
  [10,  8,  0],
  …,
  [ 2,  0,  2],
  [ 7,  0,  1],
  [ 8,  0,  1]],

 [[17, 10,  0],
```

```
  [ 7,  6,  0],
  [ 4,  4,  0],
  …,
  [ 0,  0,  2],
  [ 5,  0,  1],
  [ 8,  0,  1]],

 …,

 [[ 2,  0,  0],
  [ 1,  0,  0],
  [ 0,  0,  0],
  …,
  [ 0,  0,  1],
  [ 0,  0,  0],
  [ 0,  0,  1]],

 [[ 2,  0,  0],
  [ 3,  1,  0],
  [ 0,  0,  0],
  …,
  [ 0,  0,  0],
  [ 0,  0,  1],
  [ 1,  0,  0]],

 [[ 3,  0,  0],
  [ 2,  0,  0],
  [ 2,  1,  0],
  …,
  [ 0,  0,  1],
  [ 1,  0,  0],
  [ 1,  0,  0]]],


…,


[[[56,  8, 20],
  [19,  0, 11],
  [ 0,  0,  1],
  …,
  [ 6,  3,  0],
  [11,  7,  0],
  [16,  8,  0]],

 [[19,  0, 11],
  [ 5,  0,  7],
  [ 0,  0,  3],
```

```
  …,
   [ 5,  3,  0],
   [ 5,  3,  0],
   [ 8,  4,  0]],

 [[ 0,  0,  0],
   [ 0,  0,  2],
   [ 1,  0,  6],
  …,
   [ 7,  4,  0],
   [ 5,  3,  0],
   [ 4,  2,  0]],

 …,

 [[ 4,  3,  0],
   [ 4,  2,  0],
   [ 4,  2,  0],
  …,
   [ 0,  0,  1],
   [ 1,  0,  0],
   [ 0,  0,  0]],

 [[ 1,  1,  0],
   [ 1,  1,  0],
   [ 4,  2,  0],
  …,
   [ 0,  0,  0],
   [ 3,  1,  0],
   [ 1,  1,  0]],

 [[ 1,  1,  0],
   [ 1,  1,  0],
   [ 0,  0,  0],
  …,
   [ 1,  0,  0],
   [ 3,  2,  0],
   [ 3,  3,  0]]],


[[[ 9,  6,  0],
   [ 8,  6,  0],
   [ 6,  4,  0],
  …,
   [ 3,  2,  0],
   [ 3,  2,  0],
   [ 0,  0,  0]],
```

```
 [[ 6,  6,  0],
  [ 4,  4,  0],
  [ 6,  4,  0],
  …,
  [ 3,  2,  0],
  [ 1,  0,  0],
  [ 2,  0,  0]],

 [[ 4,  4,  0],
  [ 6,  4,  0],
  [ 6,  4,  0],
  …,
  [ 1,  0,  0],
  [ 2,  0,  0],
  [ 3,  0,  0]],

 …,

 [[ 3,  3,  0],
  [ 3,  3,  0],
  [ 3,  1,  0],
  …,
  [ 0,  0,  1],
  [ 0,  0,  0],
  [ 0,  0,  0]],

 [[ 3,  3,  0],
  [ 3,  3,  0],
  [ 4,  2,  0],
  …,
  [ 2,  1,  0],
  [ 2,  1,  0],
  [ 1,  1,  0]],

 [[ 3,  3,  0],
  [ 3,  3,  0],
  [ 4,  2,  0],
  …,
  [ 2,  1,  0],
  [ 2,  1,  0],
  [ 1,  1,  0]]],


[[[ 6,  3,  0],
  [ 3,  2,  0],
  [ 2,  0,  2],
  …,
  [ 7,  6,  0],
```

```
         [ 7,  6,  0],
         [ 6,  6,  0]],

        [[ 4,  2,  0],
         [ 1,  0,  1],
         [ 2,  0,  2],
         …,
         [ 8,  5,  0],
         [ 7,  5,  0],
         [ 6,  4,  0]],

        [[ 2,  0,  0],
         [ 0,  0,  1],
         [ 0,  0,  3],
         …,
         [ 5,  3,  0],
         [ 7,  4,  0],
         [ 7,  4,  0]],

        …,

        [[ 4,  1,  0],
         [ 2,  0,  0],
         [ 1,  0,  0],
         …,
         [ 4,  2,  0],
         [ 6,  4,  0],
         [ 6,  4,  0]],

        [[ 7,  3,  0],
         [ 7,  4,  0],
         [ 6,  2,  0],
         …,
         [ 6,  4,  0],
         [ 7,  4,  0],
         [ 7,  4,  0]],

        [[11,  6,  0],
         [10,  5,  0],
         [12,  5,  0],
         …,
         [ 7,  4,  0],
         [ 7,  4,  0],
         [ 7,  5,  0]]]], shape=(1080, 28, 28, 3), dtype=uint8), 'labels':
array([[0],
       [0],
       [0],
       …,
```

```
       [2],
       [2],
       [3]], shape=(1080, 1), dtype=uint8)}
```

[65]:
```python
from sklearn.model_selection import train_test_split
from torchvision import transforms
from torch.utils.data import Subset



# Charger le dataset SANS transformation d'abord
dataset = PKLDataset(
                "ift-3395-6390-kaggle-2-competition-fall-2025/train_data.pkl",
)

loader = DataLoader(dataset, batch_size=64, shuffle=False)

d_mean = torch.zeros(3)
d_std = torch.zeros(3)
nb_samples = 0.0

for images, _ in loader:
    batch_samples = images.size(0)

    d_mean += images.mean(dim=[0,2,3]) * batch_samples
    d_std += images.std(dim=[0,2,3]) * batch_samples
    nb_samples += batch_samples

d_mean /= nb_samples
d_std /= nb_samples

d_mean = d_mean.tolist()
d_std = d_std.tolist()

print("Mean:", d_mean)
print("Std:", d_std)
```

```
Mean: [0.21014535427093506, 0.005530359563231468, 0.2285669893026352]
Std: [0.18871904909610748, 0.01642582379281521, 0.16962255537509918]
```

[66]:
```python
# Définir les transformations
transform_train = transforms.Compose([
    transforms.Normalize(mean=d_mean, std=[d_std]),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
    transforms.ColorJitter(brightness=0.2, contrast=0.2),
])
```

```
transform_val = transforms.Compose([
])
```

[67]:
```
labels = dataset.labels

# Split stratifié
indices = np.arange(len(dataset))
train_idx, valid_idx = train_test_split(
    indices,
    test_size=0.2,
    random_state=42,
    stratify=labels
)

# Créer des subsets
train_data = Subset(dataset, train_idx)
valid_data = Subset(dataset, valid_idx)

class TransformSubset(Dataset):
    def __init__(self, subset, transform=None):
        self.subset = subset
        self.transform = transform

    def __getitem__(self, idx):
        image, label = self.subset[idx]
        if self.transform:
            image = self.transform(image)
        return image, label

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

# Appliquer les transformations
train_data = TransformSubset(train_data, transform=transform_train)
valid_data = TransformSubset(valid_data, transform=transform_val)

# DataLoaders
train_loader = DataLoader(train_data, batch_size=8, shuffle=True)
valid_loader = DataLoader(valid_data, batch_size=16, shuffle=False)
```

[68]:
```
import numpy as np

labels = dataset.labels
classes, counts = np.unique(labels, return_counts=True)

from sklearn.utils.class_weight import compute_class_weight
```

```
weights = compute_class_weight(
    class_weight="balanced",
    classes=np.unique(labels),
    y=labels
)
class_weights = torch.tensor(weights, dtype=torch.float32).to(device)

loss_fn = nn.CrossEntropyLoss(weight=class_weights)

test_loss_fn = nn.CrossEntropyLoss(reduction='sum')

# spot to save your learning curves, and potentially checkpoint your models
savedir = 'results'
if not os.path.exists(savedir):
    os.makedirs(savedir)
```

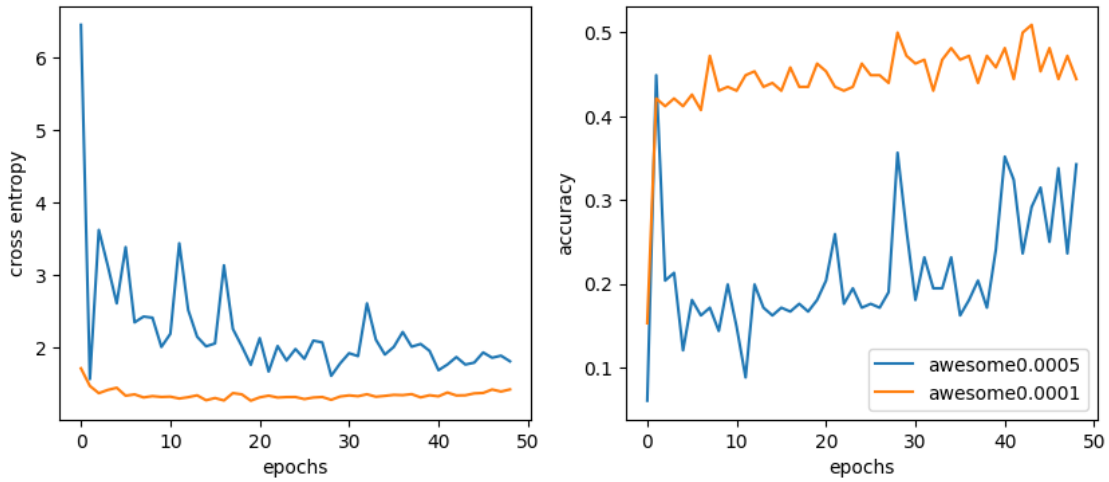
[69]:
```
def train(model, train_loader, optimizer, epoch):
    """Perform one epoch of training."""
    model.train()

    for batch_idx, (inputs, target) in enumerate(train_loader):
        inputs, target = inputs.to(device), target.to(device)

        # 1) Reset gradients
        optimizer.zero_grad()

        # 2) Forward pass
        output = model(inputs)

        # 3) Compute loss
        loss = loss_fn(output, target)

        # 4) Backpropagation
        loss.backward()

        # 5) Update weights
        optimizer.step()

        # Logging
        if batch_idx % 10 == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                epoch,
                batch_idx * len(inputs),
                len(train_loader.dataset),
                100. * batch_idx / len(train_loader),
                loss.item()
            ))
```

[70]:
```
def test(model, test_loader):
    """Evaluate the model by doing one pass over a dataset"""
    model.eval()

    test_loss = 0   # total loss over test set
    correct = 0     # total number of correct test predictions
    test_size = 0   # number of test samples used

    with torch.no_grad():  # no backprop, faster evaluation
        for inputs, target in test_loader:
            inputs, target = inputs.to(device), target.to(device)

            # Forward pass
            output = model(inputs)

            # Accumulate loss (sum, not mean)
            loss = test_loss_fn(output, target)  # already reduction='sum'
            test_loss += loss.item()

            # Predictions
            pred = output.argmax(dim=1)  # index of highest logit
            correct += (pred == target).sum().item()

            # Keep track of sample count
            test_size += target.size(0)

    # Final metrics
    test_loss /= test_size
    accuracy = correct / test_size

    print('Test set: Average loss: {:.4f}, Accuracy: {}/{} ({:.0f}%)\n'.format(
        test_loss, correct, test_size, 100. * accuracy))

    return test_loss, accuracy
```

[71]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 4))

for filename in os.listdir(savedir):
    if filename.endswith('.pkl'):
        with open(os.path.join(savedir, filename),'rb') as fin:
            results = pickle.load(fin)
            ax1.plot(results['loss'])
            ax1.set_ylabel('cross entropy')
            ax1.set_xlabel('epochs')

            ax2.plot(results['accuracy'], label = filename[:-4])
            ax2.set_ylabel('accuracy')
```

```
            ax2.set_xlabel('epochs')

plt.legend()
```

[71]: <matplotlib.legend.Legend at 0x133c1dbd0>



[72]:
```
class CNNNet(nn.Module):

    def __init__(self):
        super().__init__()

        self.conv1 = nn.Conv2d(3,32,3,padding = 1)
        self.bn1 = nn.BatchNorm2d(32)
        self.conv2 = nn.Conv2d(32,32,3,padding = 1)
        self.bn2 = nn.BatchNorm2d(32)

        self.conv3 = nn.Conv2d(32,64,3,padding = 1)
        self.bn3 = nn.BatchNorm2d(64)
        self.conv4 = nn.Conv2d(64,64,3,padding = 1)
        self.bn4 = nn.BatchNorm2d(64)

        self.fc1 = nn.Linear(64*7*7,256)
        self.bn5 = nn.BatchNorm1d(256)
        self.dropout = nn.Dropout(0.5)
        self.fc2 = nn.Linear(256,5)

    def forward(self, x):
      x = F.relu(self.bn1(self.conv1(x)))
      x = F.relu(self.bn2(self.conv2(x)))
      x = F.max_pool2d(x,2)
```

```
        x = F.relu(self.bn3(self.conv3(x)))
        x = F.relu(self.bn4(self.conv4(x)))
        x = F.max_pool2d(x,2)

        x = x.view(x.size(0),-1)
        x = F.relu(self.bn5(self.fc1(x)))
        x = self.dropout(x)
        x = self.fc2(x)
        return x
```

[73]:
```
subset = Subset(train_data, list(range(50)))
subset_loader = DataLoader(subset, batch_size=10, shuffle=True)
model = CNNNet().to(device)
lr = 0.001
optimizer = optim.Adam(model.parameters(), lr=lr)

for epoch in range(50):
    train(model, subset_loader, optimizer, epoch)
    loss, acc = test(model, subset_loader)
    print(loss, acc)
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
Cell In[73], line 8
      5 optimizer = optim.Adam(model.parameters(), lr=lr)
      7 for epoch in range(50):
----> 8     train(model, subset_loader, optimizer, epoch)
      9     loss, acc = test(model, subset_loader)
     10     print(loss, acc)

Cell In[69], line 5, in train(model, train_loader, optimizer, epoch)
      2 """Perform one epoch of training."""
      3 model.train()
----> 5 for batch_idx, (inputs, target) in enumerate(train_loader):
      6     inputs, target = inputs.to(device), target.to(device)
      8     # 1) Reset gradients

File ~/Documents/UDEM/A25/IFT3395/kaggle2/kaggle2/lib/python3.13/site-packages/
 ↪torch/utils/data/dataloader.py:732, in _BaseDataLoaderIter.__next__(self)
    729 if self._sampler_iter is None:
    730     # TODO(https://github.com/pytorch/pytorch/issues/76750)
    731     self._reset()  # type: ignore[call-arg]
--> 732 data = self._next_data()
    733 self._num_yielded += 1
    734 if (
    735     self._dataset_kind == _DatasetKind.Iterable
    736     and self._IterableDataset_len_called is not None
```

```
    737     and self._num_yielded > self._IterableDataset_len_called
    738 ):

File ~/Documents/UDEM/A25/IFT3395/kaggle2/kaggle2/lib/python3.13/site-packages/
 ↪torch/utils/data/dataloader.py:788, in _SingleProcessDataLoaderIter.
 ↪_next_data(self)
    786 def _next_data(self):
    787     index = self._next_index()  # may raise StopIteration
--> 788     data = self._dataset_fetcher.fetch(index)  # may raise StopIteratio
    789     if self._pin_memory:
    790         data = _utils.pin_memory.pin_memory(data, self.
 ↪_pin_memory_device)

File ~/Documents/UDEM/A25/IFT3395/kaggle2/kaggle2/lib/python3.13/site-packages/
 ↪torch/utils/data/_utils/fetch.py:50, in _MapDatasetFetcher.fetch(self,␣
 ↪possibly_batched_index)
     48 if self.auto_collation:
     49     if hasattr(self.dataset, "__getitems__") and self.dataset.
 ↪__getitems__:
---> 50         data = self.dataset.__getitems__(possibly_batched_index)
     51     else:
     52         data = [self.dataset[idx] for idx in possibly_batched_index]

File ~/Documents/UDEM/A25/IFT3395/kaggle2/kaggle2/lib/python3.13/site-packages/
 ↪torch/utils/data/dataset.py:416, in Subset.__getitems__(self, indices)
    414     return self.dataset.__getitems__([self.indices[idx] for idx in␣
 ↪indices])  # type: ignore[attr-defined]
    415 else:
--> 416     return [self.dataset[self.indices[idx]] for idx in indices]

Cell In[67], line 24, in TransformSubset.__getitem__(self, idx)
     22 image, label = self.subset[idx]
     23 if self.transform:
---> 24     image = self.transform(image)
     25 return image, label

File ~/Documents/UDEM/A25/IFT3395/kaggle2/kaggle2/lib/python3.13/site-packages/
 ↪torchvision/transforms/transforms.py:95, in Compose.__call__(self, img)
     93 def __call__(self, img):
     94     for t in self.transforms:
---> 95         img = t(img)
     96     return img

File ~/Documents/UDEM/A25/IFT3395/kaggle2/kaggle2/lib/python3.13/site-packages/
 ↪torch/nn/modules/module.py:1775, in Module._wrapped_call_impl(self, *args,␣
 ↪**kwargs)
   1773     return self._compiled_call_impl(*args, **kwargs)  # type:␣
 ↪ignore[misc]
```

```
   1774 else:
-> 1775     return self._call_impl(*args, **kwargs)

File ~/Documents/UDEM/A25/IFT3395/kaggle2/kaggle2/lib/python3.13/site-packages/
 ↪torch/nn/modules/module.py:1786, in Module._call_impl(self, *args, **kwargs)
   1781 # If we don't have any hooks, we want to skip the rest of the logic in
   1782 # this function, and just call forward.
   1783 if not (self._backward_hooks or self._backward_pre_hooks or self.
 ↪_forward_hooks or self._forward_pre_hooks
   1784         or _global_backward_pre_hooks or _global_backward_hooks
   1785         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1786     return forward_call(*args, **kwargs)
   1788 result = None
   1789 called_always_called_hooks = set()

File ~/Documents/UDEM/A25/IFT3395/kaggle2/kaggle2/lib/python3.13/site-packages/
 ↪torchvision/transforms/transforms.py:277, in Normalize.forward(self, tensor)
    269 def forward(self, tensor: Tensor) -> Tensor:
    270     """
    271     Args:
    272         tensor (Tensor): Tensor image to be normalized.
   (…)    275         Tensor: Normalized Tensor image.
    276     """
--> 277     return F.normalize(tensor, self.mean, self.std, self.inplace)

File ~/Documents/UDEM/A25/IFT3395/kaggle2/kaggle2/lib/python3.13/site-packages/
 ↪torchvision/transforms/functional.py:350, in normalize(tensor, mean, std,␣
 ↪inplace)
    347 if not isinstance(tensor, torch.Tensor):
    348     raise TypeError(f"img should be Tensor Image. Got {type(tensor)}")
--> 350 return F_t.normalize(tensor, mean=mean, std=std, inplace=inplace)

File ~/Documents/UDEM/A25/IFT3395/kaggle2/kaggle2/lib/python3.13/site-packages/
 ↪torchvision/transforms/_functional_tensor.py:928, in normalize(tensor, mean,␣
 ↪std, inplace)
    926 if std.ndim == 1:
    927     std = std.view(-1, 1, 1)
--> 928 return tensor.sub_(mean).div_(std)

RuntimeError: The size of tensor a (28) must match the size of tensor b (3) at␣
 ↪non-singleton dimension 2
```

[ ]:
```
# TRAINING
model = CNNNet().to(device)

lr = 0.0005
optimizer = optim.Adam(model.parameters(), lr=lr)
```

```
results = {'name':'awesome', 'lr': lr, 'loss': [], 'accuracy':[]}
savefile = os.path.join(savedir, results['name']+str(results['lr'])+'.pkl' )

for epoch in range(1, 50):
    train(model, train_loader, optimizer, epoch)
    loss, acc = test(model, valid_loader)

    # save results
    results['loss'].append(loss)
    results['accuracy'].append(acc)
    with open(savefile, 'wb') as fout:
        pickle.dump(results, fout)
```

```
Train Epoch: 1 [0/864 (0%)]     Loss: 1.991542
Train Epoch: 1 [80/864 (9%)]    Loss: 2.254312
Train Epoch: 1 [160/864 (19%)]  Loss: 1.832831
Train Epoch: 1 [240/864 (28%)]  Loss: 1.832648
Train Epoch: 1 [320/864 (37%)]  Loss: 1.568615
Train Epoch: 1 [400/864 (46%)]  Loss: 1.418531
Train Epoch: 1 [480/864 (56%)]  Loss: 1.877491
Train Epoch: 1 [560/864 (65%)]  Loss: 1.838115
Train Epoch: 1 [640/864 (74%)]  Loss: 1.295010
Train Epoch: 1 [720/864 (83%)]  Loss: 2.292656
Train Epoch: 1 [800/864 (93%)]  Loss: 1.859918
Test set: Average loss: 6.4588, Accuracy: 13/216 (6%)

Train Epoch: 2 [0/864 (0%)]     Loss: 1.569145
Train Epoch: 2 [80/864 (9%)]    Loss: 1.079422
Train Epoch: 2 [160/864 (19%)]  Loss: 1.386957
Train Epoch: 2 [240/864 (28%)]  Loss: 1.936242
Train Epoch: 2 [320/864 (37%)]  Loss: 1.415684
Train Epoch: 2 [400/864 (46%)]  Loss: 1.663964
Train Epoch: 2 [480/864 (56%)]  Loss: 1.463775
Train Epoch: 2 [560/864 (65%)]  Loss: 1.893300
Train Epoch: 2 [640/864 (74%)]  Loss: 1.636451
Train Epoch: 2 [720/864 (83%)]  Loss: 1.704341
Train Epoch: 2 [800/864 (93%)]  Loss: 1.493666
Test set: Average loss: 1.5655, Accuracy: 97/216 (45%)

Train Epoch: 3 [0/864 (0%)]     Loss: 1.323246
Train Epoch: 3 [80/864 (9%)]    Loss: 1.619669
Train Epoch: 3 [160/864 (19%)]  Loss: 1.320122
Train Epoch: 3 [240/864 (28%)]  Loss: 1.550691
Train Epoch: 3 [320/864 (37%)]  Loss: 1.608774
Train Epoch: 3 [400/864 (46%)]  Loss: 1.657238
Train Epoch: 3 [480/864 (56%)]  Loss: 1.560713
Train Epoch: 3 [560/864 (65%)]  Loss: 1.100687
```

```
Train Epoch: 3 [640/864 (74%)]  Loss: 1.709621
Train Epoch: 3 [720/864 (83%)]  Loss: 1.876141
Train Epoch: 3 [800/864 (93%)]  Loss: 1.307635
Test set: Average loss: 3.6260, Accuracy: 44/216 (20%)

Train Epoch: 4 [0/864 (0%)]     Loss: 1.596151
Train Epoch: 4 [80/864 (9%)]    Loss: 1.741009
Train Epoch: 4 [160/864 (19%)]  Loss: 1.127683
Train Epoch: 4 [240/864 (28%)]  Loss: 1.268621
Train Epoch: 4 [320/864 (37%)]  Loss: 1.473926
Train Epoch: 4 [400/864 (46%)]  Loss: 1.142202
Train Epoch: 4 [480/864 (56%)]  Loss: 1.351836
Train Epoch: 4 [560/864 (65%)]  Loss: 1.713647
Train Epoch: 4 [640/864 (74%)]  Loss: 1.347172
Train Epoch: 4 [720/864 (83%)]  Loss: 1.281329
Train Epoch: 4 [800/864 (93%)]  Loss: 1.058985
Test set: Average loss: 3.1323, Accuracy: 46/216 (21%)

Train Epoch: 5 [0/864 (0%)]     Loss: 1.480044
Train Epoch: 5 [80/864 (9%)]    Loss: 1.352029
Train Epoch: 5 [160/864 (19%)]  Loss: 1.966451
Train Epoch: 5 [240/864 (28%)]  Loss: 1.116824
Train Epoch: 5 [320/864 (37%)]  Loss: 1.807473
Train Epoch: 5 [400/864 (46%)]  Loss: 1.237751
Train Epoch: 5 [480/864 (56%)]  Loss: 1.442364
Train Epoch: 5 [560/864 (65%)]  Loss: 1.210736
Train Epoch: 5 [640/864 (74%)]  Loss: 1.480225
Train Epoch: 5 [720/864 (83%)]  Loss: 1.377017
Train Epoch: 5 [800/864 (93%)]  Loss: 1.619799
Test set: Average loss: 2.6111, Accuracy: 26/216 (12%)

Train Epoch: 6 [0/864 (0%)]     Loss: 1.720623
Train Epoch: 6 [80/864 (9%)]    Loss: 1.565661
Train Epoch: 6 [160/864 (19%)]  Loss: 1.426810
Train Epoch: 6 [240/864 (28%)]  Loss: 1.570710
Train Epoch: 6 [320/864 (37%)]  Loss: 1.718529
Train Epoch: 6 [400/864 (46%)]  Loss: 1.399586
Train Epoch: 6 [480/864 (56%)]  Loss: 1.032720
Train Epoch: 6 [560/864 (65%)]  Loss: 1.575224
Train Epoch: 6 [640/864 (74%)]  Loss: 1.905921
Train Epoch: 6 [720/864 (83%)]  Loss: 1.016033
Train Epoch: 6 [800/864 (93%)]  Loss: 1.721959
Test set: Average loss: 3.3885, Accuracy: 39/216 (18%)

Train Epoch: 7 [0/864 (0%)]     Loss: 1.640982
Train Epoch: 7 [80/864 (9%)]    Loss: 1.141848
Train Epoch: 7 [160/864 (19%)]  Loss: 1.236024
Train Epoch: 7 [240/864 (28%)]  Loss: 1.380845
```

```
Train Epoch: 7 [320/864 (37%)]  Loss: 1.496209
Train Epoch: 7 [400/864 (46%)]  Loss: 1.945813
Train Epoch: 7 [480/864 (56%)]  Loss: 1.298412
Train Epoch: 7 [560/864 (65%)]  Loss: 1.230159
Train Epoch: 7 [640/864 (74%)]  Loss: 1.615985
Train Epoch: 7 [720/864 (83%)]  Loss: 1.750019
Train Epoch: 7 [800/864 (93%)]  Loss: 1.317471
Test set: Average loss: 2.3490, Accuracy: 35/216 (16%)

Train Epoch: 8 [0/864 (0%)]     Loss: 1.175210
Train Epoch: 8 [80/864 (9%)]    Loss: 1.793330
Train Epoch: 8 [160/864 (19%)]  Loss: 1.572250
Train Epoch: 8 [240/864 (28%)]  Loss: 2.153195
Train Epoch: 8 [320/864 (37%)]  Loss: 1.290256
Train Epoch: 8 [400/864 (46%)]  Loss: 1.735583
Train Epoch: 8 [480/864 (56%)]  Loss: 1.286321
Train Epoch: 8 [560/864 (65%)]  Loss: 1.411595
Train Epoch: 8 [640/864 (74%)]  Loss: 1.891258
Train Epoch: 8 [720/864 (83%)]  Loss: 2.380080
Train Epoch: 8 [800/864 (93%)]  Loss: 2.065113
Test set: Average loss: 2.4289, Accuracy: 37/216 (17%)

Train Epoch: 9 [0/864 (0%)]     Loss: 1.829203
Train Epoch: 9 [80/864 (9%)]    Loss: 1.522376
Train Epoch: 9 [160/864 (19%)]  Loss: 0.789232
Train Epoch: 9 [240/864 (28%)]  Loss: 1.272131
Train Epoch: 9 [320/864 (37%)]  Loss: 1.675686
Train Epoch: 9 [400/864 (46%)]  Loss: 1.406569
Train Epoch: 9 [480/864 (56%)]  Loss: 1.262880
Train Epoch: 9 [560/864 (65%)]  Loss: 1.836727
Train Epoch: 9 [640/864 (74%)]  Loss: 1.166757
Train Epoch: 9 [720/864 (83%)]  Loss: 1.529144
Train Epoch: 9 [800/864 (93%)]  Loss: 1.076416
Test set: Average loss: 2.4138, Accuracy: 31/216 (14%)

Train Epoch: 10 [0/864 (0%)]    Loss: 0.959074
Train Epoch: 10 [80/864 (9%)]   Loss: 1.479104
Train Epoch: 10 [160/864 (19%)] Loss: 1.372345
Train Epoch: 10 [240/864 (28%)] Loss: 1.290429
Train Epoch: 10 [320/864 (37%)] Loss: 1.544037
Train Epoch: 10 [400/864 (46%)] Loss: 1.628568
Train Epoch: 10 [480/864 (56%)] Loss: 1.713122
Train Epoch: 10 [560/864 (65%)] Loss: 1.433753
Train Epoch: 10 [640/864 (74%)] Loss: 1.713912
Train Epoch: 10 [720/864 (83%)] Loss: 1.886128
Train Epoch: 10 [800/864 (93%)] Loss: 1.379290
Test set: Average loss: 2.0097, Accuracy: 43/216 (20%)
```

```
Train Epoch: 11 [0/864 (0%)]    Loss: 1.251420
Train Epoch: 11 [80/864 (9%)]   Loss: 1.300847
Train Epoch: 11 [160/864 (19%)] Loss: 1.709773
Train Epoch: 11 [240/864 (28%)] Loss: 0.939897
Train Epoch: 11 [320/864 (37%)] Loss: 1.423993
Train Epoch: 11 [400/864 (46%)] Loss: 1.159987
Train Epoch: 11 [480/864 (56%)] Loss: 1.318067
Train Epoch: 11 [560/864 (65%)] Loss: 1.737222
Train Epoch: 11 [640/864 (74%)] Loss: 1.084751
Train Epoch: 11 [720/864 (83%)] Loss: 1.326241
Train Epoch: 11 [800/864 (93%)] Loss: 1.339702
Test set: Average loss: 2.1908, Accuracy: 32/216 (15%)

Train Epoch: 12 [0/864 (0%)]    Loss: 0.985143
Train Epoch: 12 [80/864 (9%)]   Loss: 1.061999
Train Epoch: 12 [160/864 (19%)] Loss: 1.226785
Train Epoch: 12 [240/864 (28%)] Loss: 1.517105
Train Epoch: 12 [320/864 (37%)] Loss: 1.369205
Train Epoch: 12 [400/864 (46%)] Loss: 1.357145
Train Epoch: 12 [480/864 (56%)] Loss: 1.077093
Train Epoch: 12 [560/864 (65%)] Loss: 1.279563
Train Epoch: 12 [640/864 (74%)] Loss: 1.331511
Train Epoch: 12 [720/864 (83%)] Loss: 1.722929
Train Epoch: 12 [800/864 (93%)] Loss: 1.339792
Test set: Average loss: 3.4414, Accuracy: 19/216 (9%)

Train Epoch: 13 [0/864 (0%)]    Loss: 0.889527
Train Epoch: 13 [80/864 (9%)]   Loss: 1.190298
Train Epoch: 13 [160/864 (19%)] Loss: 0.918201
Train Epoch: 13 [240/864 (28%)] Loss: 1.446275
Train Epoch: 13 [320/864 (37%)] Loss: 1.153134
Train Epoch: 13 [400/864 (46%)] Loss: 1.700194
Train Epoch: 13 [480/864 (56%)] Loss: 1.338820
Train Epoch: 13 [560/864 (65%)] Loss: 0.972073
Train Epoch: 13 [640/864 (74%)] Loss: 0.976929
Train Epoch: 13 [720/864 (83%)] Loss: 1.288064
Train Epoch: 13 [800/864 (93%)] Loss: 1.059741
Test set: Average loss: 2.5197, Accuracy: 43/216 (20%)

Train Epoch: 14 [0/864 (0%)]    Loss: 1.839000
Train Epoch: 14 [80/864 (9%)]   Loss: 1.445063
Train Epoch: 14 [160/864 (19%)] Loss: 1.245382
Train Epoch: 14 [240/864 (28%)] Loss: 1.311540
Train Epoch: 14 [320/864 (37%)] Loss: 1.201957
Train Epoch: 14 [400/864 (46%)] Loss: 1.227921
Train Epoch: 14 [480/864 (56%)] Loss: 1.100287
Train Epoch: 14 [560/864 (65%)] Loss: 1.501113
Train Epoch: 14 [640/864 (74%)] Loss: 0.998743
```

```
Train Epoch: 14 [720/864 (83%)] Loss: 1.722950
Train Epoch: 14 [800/864 (93%)] Loss: 1.324681
Test set: Average loss: 2.1503, Accuracy: 37/216 (17%)

Train Epoch: 15 [0/864 (0%)]    Loss: 1.476520
Train Epoch: 15 [80/864 (9%)]   Loss: 1.277718
Train Epoch: 15 [160/864 (19%)] Loss: 1.304910
Train Epoch: 15 [240/864 (28%)] Loss: 0.985599
Train Epoch: 15 [320/864 (37%)] Loss: 1.348862
Train Epoch: 15 [400/864 (46%)] Loss: 1.392395
Train Epoch: 15 [480/864 (56%)] Loss: 1.077910
Train Epoch: 15 [560/864 (65%)] Loss: 0.953464
Train Epoch: 15 [640/864 (74%)] Loss: 1.242846
Train Epoch: 15 [720/864 (83%)] Loss: 1.386460
Train Epoch: 15 [800/864 (93%)] Loss: 0.885433
Test set: Average loss: 2.0181, Accuracy: 35/216 (16%)

Train Epoch: 16 [0/864 (0%)]    Loss: 1.396358
Train Epoch: 16 [80/864 (9%)]   Loss: 1.195690
Train Epoch: 16 [160/864 (19%)] Loss: 1.092486
Train Epoch: 16 [240/864 (28%)] Loss: 0.883033
Train Epoch: 16 [320/864 (37%)] Loss: 0.878203
Train Epoch: 16 [400/864 (46%)] Loss: 1.155583
Train Epoch: 16 [480/864 (56%)] Loss: 1.350154
Train Epoch: 16 [560/864 (65%)] Loss: 1.482650
Train Epoch: 16 [640/864 (74%)] Loss: 1.975385
Train Epoch: 16 [720/864 (83%)] Loss: 1.227506
Train Epoch: 16 [800/864 (93%)] Loss: 1.215125
Test set: Average loss: 2.0584, Accuracy: 37/216 (17%)

Train Epoch: 17 [0/864 (0%)]    Loss: 1.191734
Train Epoch: 17 [80/864 (9%)]   Loss: 1.414641
Train Epoch: 17 [160/864 (19%)] Loss: 1.125356
Train Epoch: 17 [240/864 (28%)] Loss: 1.681561
Train Epoch: 17 [320/864 (37%)] Loss: 1.246128
Train Epoch: 17 [400/864 (46%)] Loss: 1.276996
Train Epoch: 17 [480/864 (56%)] Loss: 1.182513
Train Epoch: 17 [560/864 (65%)] Loss: 1.481760
Train Epoch: 17 [640/864 (74%)] Loss: 1.028136
Train Epoch: 17 [720/864 (83%)] Loss: 1.273778
Train Epoch: 17 [800/864 (93%)] Loss: 1.940253
Test set: Average loss: 3.1372, Accuracy: 36/216 (17%)

Train Epoch: 18 [0/864 (0%)]    Loss: 0.941799
Train Epoch: 18 [80/864 (9%)]   Loss: 1.034058
Train Epoch: 18 [160/864 (19%)] Loss: 1.408517
Train Epoch: 18 [240/864 (28%)] Loss: 1.224389
Train Epoch: 18 [320/864 (37%)] Loss: 0.978239
```

```
Train Epoch: 18 [400/864 (46%)] Loss: 1.297216
Train Epoch: 18 [480/864 (56%)] Loss: 1.167396
Train Epoch: 18 [560/864 (65%)] Loss: 1.753606
Train Epoch: 18 [640/864 (74%)] Loss: 0.817784
Train Epoch: 18 [720/864 (83%)] Loss: 0.685467
Train Epoch: 18 [800/864 (93%)] Loss: 2.112607
Test set: Average loss: 2.2580, Accuracy: 38/216 (18%)

Train Epoch: 19 [0/864 (0%)]    Loss: 1.135335
Train Epoch: 19 [80/864 (9%)]   Loss: 0.996446
Train Epoch: 19 [160/864 (19%)] Loss: 0.829147
Train Epoch: 19 [240/864 (28%)] Loss: 1.357130
Train Epoch: 19 [320/864 (37%)] Loss: 0.902131
Train Epoch: 19 [400/864 (46%)] Loss: 1.519347
Train Epoch: 19 [480/864 (56%)] Loss: 1.461728
Train Epoch: 19 [560/864 (65%)] Loss: 1.335772
Train Epoch: 19 [640/864 (74%)] Loss: 1.641492
Train Epoch: 19 [720/864 (83%)] Loss: 0.782825
Train Epoch: 19 [800/864 (93%)] Loss: 1.526076
Test set: Average loss: 2.0195, Accuracy: 36/216 (17%)

Train Epoch: 20 [0/864 (0%)]    Loss: 1.349173
Train Epoch: 20 [80/864 (9%)]   Loss: 1.530367
Train Epoch: 20 [160/864 (19%)] Loss: 1.285702
Train Epoch: 20 [240/864 (28%)] Loss: 1.244889
Train Epoch: 20 [320/864 (37%)] Loss: 1.220057
Train Epoch: 20 [400/864 (46%)] Loss: 1.813722
Train Epoch: 20 [480/864 (56%)] Loss: 0.953515
Train Epoch: 20 [560/864 (65%)] Loss: 1.821404
Train Epoch: 20 [640/864 (74%)] Loss: 1.125508
Train Epoch: 20 [720/864 (83%)] Loss: 1.066594
Train Epoch: 20 [800/864 (93%)] Loss: 0.660583
Test set: Average loss: 1.7628, Accuracy: 39/216 (18%)

Train Epoch: 21 [0/864 (0%)]    Loss: 1.977732
Train Epoch: 21 [80/864 (9%)]   Loss: 1.152168
Train Epoch: 21 [160/864 (19%)] Loss: 1.495065
Train Epoch: 21 [240/864 (28%)] Loss: 1.630612
Train Epoch: 21 [320/864 (37%)] Loss: 0.959674
Train Epoch: 21 [400/864 (46%)] Loss: 1.109067
Train Epoch: 21 [480/864 (56%)] Loss: 1.041780
Train Epoch: 21 [560/864 (65%)] Loss: 1.230733
Train Epoch: 21 [640/864 (74%)] Loss: 1.555616
Train Epoch: 21 [720/864 (83%)] Loss: 1.165013
Train Epoch: 21 [800/864 (93%)] Loss: 1.360699
Test set: Average loss: 2.1308, Accuracy: 44/216 (20%)

Train Epoch: 22 [0/864 (0%)]    Loss: 0.987675
```

```
Train Epoch: 22 [80/864 (9%)]   Loss: 1.609592
Train Epoch: 22 [160/864 (19%)] Loss: 1.340687
Train Epoch: 22 [240/864 (28%)] Loss: 1.103451
Train Epoch: 22 [320/864 (37%)] Loss: 1.474086
Train Epoch: 22 [400/864 (46%)] Loss: 1.094319
Train Epoch: 22 [480/864 (56%)] Loss: 1.378117
Train Epoch: 22 [560/864 (65%)] Loss: 0.932062
Train Epoch: 22 [640/864 (74%)] Loss: 1.224720
Train Epoch: 22 [720/864 (83%)] Loss: 1.290799
Train Epoch: 22 [800/864 (93%)] Loss: 1.338561
Test set: Average loss: 1.6695, Accuracy: 56/216 (26%)

Train Epoch: 23 [0/864 (0%)]    Loss: 1.242312
Train Epoch: 23 [80/864 (9%)]   Loss: 0.891826
Train Epoch: 23 [160/864 (19%)] Loss: 1.221789
Train Epoch: 23 [240/864 (28%)] Loss: 1.402700
Train Epoch: 23 [320/864 (37%)] Loss: 1.220321
Train Epoch: 23 [400/864 (46%)] Loss: 0.894767
Train Epoch: 23 [480/864 (56%)] Loss: 1.255121
Train Epoch: 23 [560/864 (65%)] Loss: 0.710624
Train Epoch: 23 [640/864 (74%)] Loss: 1.153483
Train Epoch: 23 [720/864 (83%)] Loss: 1.073201
Train Epoch: 23 [800/864 (93%)] Loss: 1.313385
Test set: Average loss: 2.0220, Accuracy: 38/216 (18%)

Train Epoch: 24 [0/864 (0%)]    Loss: 1.925090
Train Epoch: 24 [80/864 (9%)]   Loss: 0.915415
Train Epoch: 24 [160/864 (19%)] Loss: 0.789014
Train Epoch: 24 [240/864 (28%)] Loss: 0.828106
Train Epoch: 24 [320/864 (37%)] Loss: 1.413995
Train Epoch: 24 [400/864 (46%)] Loss: 1.175473
Train Epoch: 24 [480/864 (56%)] Loss: 1.207781
Train Epoch: 24 [560/864 (65%)] Loss: 1.202285
Train Epoch: 24 [640/864 (74%)] Loss: 1.424632
Train Epoch: 24 [720/864 (83%)] Loss: 0.948456
Train Epoch: 24 [800/864 (93%)] Loss: 0.786046
Test set: Average loss: 1.8228, Accuracy: 42/216 (19%)

Train Epoch: 25 [0/864 (0%)]    Loss: 0.683155
Train Epoch: 25 [80/864 (9%)]   Loss: 0.553415
Train Epoch: 25 [160/864 (19%)] Loss: 0.770175
Train Epoch: 25 [240/864 (28%)] Loss: 0.890284
Train Epoch: 25 [320/864 (37%)] Loss: 0.446839
Train Epoch: 25 [400/864 (46%)] Loss: 0.731889
Train Epoch: 25 [480/864 (56%)] Loss: 1.132219
Train Epoch: 25 [560/864 (65%)] Loss: 1.940603
Train Epoch: 25 [640/864 (74%)] Loss: 1.007398
Train Epoch: 25 [720/864 (83%)] Loss: 0.734575
```

```
Train Epoch: 25 [800/864 (93%)] Loss: 1.018053
Test set: Average loss: 1.9807, Accuracy: 37/216 (17%)

Train Epoch: 26 [0/864 (0%)]    Loss: 0.689595
Train Epoch: 26 [80/864 (9%)]   Loss: 0.837676
Train Epoch: 26 [160/864 (19%)] Loss: 1.202846
Train Epoch: 26 [240/864 (28%)] Loss: 1.102704
Train Epoch: 26 [320/864 (37%)] Loss: 1.096965
Train Epoch: 26 [400/864 (46%)] Loss: 0.943445
Train Epoch: 26 [480/864 (56%)] Loss: 0.776607
Train Epoch: 26 [560/864 (65%)] Loss: 1.252443
Train Epoch: 26 [640/864 (74%)] Loss: 1.387441
Train Epoch: 26 [720/864 (83%)] Loss: 1.498628
Train Epoch: 26 [800/864 (93%)] Loss: 0.724472
Test set: Average loss: 1.8450, Accuracy: 38/216 (18%)

Train Epoch: 27 [0/864 (0%)]    Loss: 1.233772
Train Epoch: 27 [80/864 (9%)]   Loss: 0.937928
Train Epoch: 27 [160/864 (19%)] Loss: 0.777443
Train Epoch: 27 [240/864 (28%)] Loss: 0.859444
Train Epoch: 27 [320/864 (37%)] Loss: 1.038062
Train Epoch: 27 [400/864 (46%)] Loss: 0.878869
Train Epoch: 27 [480/864 (56%)] Loss: 1.155051
Train Epoch: 27 [560/864 (65%)] Loss: 0.891075
Train Epoch: 27 [640/864 (74%)] Loss: 0.776698
Train Epoch: 27 [720/864 (83%)] Loss: 1.046387
Train Epoch: 27 [800/864 (93%)] Loss: 1.181279
Test set: Average loss: 2.0974, Accuracy: 37/216 (17%)

Train Epoch: 28 [0/864 (0%)]    Loss: 1.030900
Train Epoch: 28 [80/864 (9%)]   Loss: 1.160361
Train Epoch: 28 [160/864 (19%)] Loss: 1.585746
Train Epoch: 28 [240/864 (28%)] Loss: 1.653333
Train Epoch: 28 [320/864 (37%)] Loss: 0.774403
Train Epoch: 28 [400/864 (46%)] Loss: 1.087800
Train Epoch: 28 [480/864 (56%)] Loss: 1.567277
Train Epoch: 28 [560/864 (65%)] Loss: 0.860462
Train Epoch: 28 [640/864 (74%)] Loss: 0.958073
Train Epoch: 28 [720/864 (83%)] Loss: 0.763303
Train Epoch: 28 [800/864 (93%)] Loss: 1.058998
Test set: Average loss: 2.0721, Accuracy: 41/216 (19%)

Train Epoch: 29 [0/864 (0%)]    Loss: 1.086542
Train Epoch: 29 [80/864 (9%)]   Loss: 0.886506
Train Epoch: 29 [160/864 (19%)] Loss: 0.974974
Train Epoch: 29 [240/864 (28%)] Loss: 1.574549
Train Epoch: 29 [320/864 (37%)] Loss: 0.815419
Train Epoch: 29 [400/864 (46%)] Loss: 1.201386
```

```
Train Epoch: 29 [480/864 (56%)] Loss: 1.362534
Train Epoch: 29 [560/864 (65%)] Loss: 0.894004
Train Epoch: 29 [640/864 (74%)] Loss: 1.263924
Train Epoch: 29 [720/864 (83%)] Loss: 1.303571
Train Epoch: 29 [800/864 (93%)] Loss: 0.865217
Test set: Average loss: 1.6105, Accuracy: 77/216 (36%)

Train Epoch: 30 [0/864 (0%)]    Loss: 1.122183
Train Epoch: 30 [80/864 (9%)]   Loss: 1.315028
Train Epoch: 30 [160/864 (19%)] Loss: 0.665185
Train Epoch: 30 [240/864 (28%)] Loss: 1.644213
Train Epoch: 30 [320/864 (37%)] Loss: 1.065828
Train Epoch: 30 [400/864 (46%)] Loss: 0.810349
Train Epoch: 30 [480/864 (56%)] Loss: 0.978588
Train Epoch: 30 [560/864 (65%)] Loss: 1.000837
Train Epoch: 30 [640/864 (74%)] Loss: 1.055631
Train Epoch: 30 [720/864 (83%)] Loss: 0.744072
Train Epoch: 30 [800/864 (93%)] Loss: 0.571915
Test set: Average loss: 1.7824, Accuracy: 57/216 (26%)

Train Epoch: 31 [0/864 (0%)]    Loss: 0.663633
Train Epoch: 31 [80/864 (9%)]   Loss: 0.650128
Train Epoch: 31 [160/864 (19%)] Loss: 1.223528
Train Epoch: 31 [240/864 (28%)] Loss: 0.844951
Train Epoch: 31 [320/864 (37%)] Loss: 0.733583
Train Epoch: 31 [400/864 (46%)] Loss: 0.934548
Train Epoch: 31 [480/864 (56%)] Loss: 0.779864
Train Epoch: 31 [560/864 (65%)] Loss: 0.739311
Train Epoch: 31 [640/864 (74%)] Loss: 1.488568
Train Epoch: 31 [720/864 (83%)] Loss: 1.602744
Train Epoch: 31 [800/864 (93%)] Loss: 1.329901
Test set: Average loss: 1.9235, Accuracy: 39/216 (18%)

Train Epoch: 32 [0/864 (0%)]    Loss: 0.880136
Train Epoch: 32 [80/864 (9%)]   Loss: 1.220631
Train Epoch: 32 [160/864 (19%)] Loss: 0.702603
Train Epoch: 32 [240/864 (28%)] Loss: 0.633186
Train Epoch: 32 [320/864 (37%)] Loss: 0.878188
Train Epoch: 32 [400/864 (46%)] Loss: 0.644070
Train Epoch: 32 [480/864 (56%)] Loss: 0.896346
Train Epoch: 32 [560/864 (65%)] Loss: 0.941773
Train Epoch: 32 [640/864 (74%)] Loss: 0.818888
Train Epoch: 32 [720/864 (83%)] Loss: 1.004514
Train Epoch: 32 [800/864 (93%)] Loss: 1.521662
Test set: Average loss: 1.8826, Accuracy: 50/216 (23%)

Train Epoch: 33 [0/864 (0%)]    Loss: 1.277029
Train Epoch: 33 [80/864 (9%)]   Loss: 0.897807
```

```
Train Epoch: 33 [160/864 (19%)] Loss: 1.511446
Train Epoch: 33 [240/864 (28%)] Loss: 1.252344
Train Epoch: 33 [320/864 (37%)] Loss: 0.989129
Train Epoch: 33 [400/864 (46%)] Loss: 0.680739
Train Epoch: 33 [480/864 (56%)] Loss: 0.721159
Train Epoch: 33 [560/864 (65%)] Loss: 1.017320
Train Epoch: 33 [640/864 (74%)] Loss: 0.946005
Train Epoch: 33 [720/864 (83%)] Loss: 1.125751
Train Epoch: 33 [800/864 (93%)] Loss: 0.817886
Test set: Average loss: 2.6117, Accuracy: 42/216 (19%)

Train Epoch: 34 [0/864 (0%)]    Loss: 0.866242
Train Epoch: 34 [80/864 (9%)]   Loss: 0.776665
Train Epoch: 34 [160/864 (19%)] Loss: 1.136506
Train Epoch: 34 [240/864 (28%)] Loss: 1.466198
Train Epoch: 34 [320/864 (37%)] Loss: 1.719156
Train Epoch: 34 [400/864 (46%)] Loss: 1.466884
Train Epoch: 34 [480/864 (56%)] Loss: 1.302451
Train Epoch: 34 [560/864 (65%)] Loss: 0.657323
Train Epoch: 34 [640/864 (74%)] Loss: 0.585181
Train Epoch: 34 [720/864 (83%)] Loss: 0.869353
Train Epoch: 34 [800/864 (93%)] Loss: 1.461225
Test set: Average loss: 2.1089, Accuracy: 42/216 (19%)

Train Epoch: 35 [0/864 (0%)]    Loss: 0.970444
Train Epoch: 35 [80/864 (9%)]   Loss: 1.181359
Train Epoch: 35 [160/864 (19%)] Loss: 0.643281
Train Epoch: 35 [240/864 (28%)] Loss: 0.968138
Train Epoch: 35 [320/864 (37%)] Loss: 0.741853
Train Epoch: 35 [400/864 (46%)] Loss: 1.520805
Train Epoch: 35 [480/864 (56%)] Loss: 1.507978
Train Epoch: 35 [560/864 (65%)] Loss: 0.800475
Train Epoch: 35 [640/864 (74%)] Loss: 1.267937
Train Epoch: 35 [720/864 (83%)] Loss: 0.916158
Train Epoch: 35 [800/864 (93%)] Loss: 1.382514
Test set: Average loss: 1.9048, Accuracy: 50/216 (23%)

Train Epoch: 36 [0/864 (0%)]    Loss: 1.256270
Train Epoch: 36 [80/864 (9%)]   Loss: 1.297433
Train Epoch: 36 [160/864 (19%)] Loss: 1.390424
Train Epoch: 36 [240/864 (28%)] Loss: 0.855650
Train Epoch: 36 [320/864 (37%)] Loss: 0.655614
Train Epoch: 36 [400/864 (46%)] Loss: 0.444343
Train Epoch: 36 [480/864 (56%)] Loss: 1.038353
Train Epoch: 36 [560/864 (65%)] Loss: 1.140210
Train Epoch: 36 [640/864 (74%)] Loss: 1.236717
Train Epoch: 36 [720/864 (83%)] Loss: 0.890200
Train Epoch: 36 [800/864 (93%)] Loss: 1.234260
```

```
Test set: Average loss: 2.0099, Accuracy: 35/216 (16%)

Train Epoch: 37 [0/864 (0%)]    Loss: 0.410066
Train Epoch: 37 [80/864 (9%)]   Loss: 0.443311
Train Epoch: 37 [160/864 (19%)] Loss: 0.452770
Train Epoch: 37 [240/864 (28%)] Loss: 1.039239
Train Epoch: 37 [320/864 (37%)] Loss: 0.711895
Train Epoch: 37 [400/864 (46%)] Loss: 0.981887
Train Epoch: 37 [480/864 (56%)] Loss: 0.685884
Train Epoch: 37 [560/864 (65%)] Loss: 0.758848
Train Epoch: 37 [640/864 (74%)] Loss: 0.624572
Train Epoch: 37 [720/864 (83%)] Loss: 1.391013
Train Epoch: 37 [800/864 (93%)] Loss: 1.324786
Test set: Average loss: 2.2162, Accuracy: 39/216 (18%)

Train Epoch: 38 [0/864 (0%)]    Loss: 0.847880
Train Epoch: 38 [80/864 (9%)]   Loss: 0.654297
Train Epoch: 38 [160/864 (19%)] Loss: 0.633060
Train Epoch: 38 [240/864 (28%)] Loss: 1.709511
Train Epoch: 38 [320/864 (37%)] Loss: 0.824055
Train Epoch: 38 [400/864 (46%)] Loss: 0.861953
Train Epoch: 38 [480/864 (56%)] Loss: 0.562377
Train Epoch: 38 [560/864 (65%)] Loss: 0.684887
Train Epoch: 38 [640/864 (74%)] Loss: 1.156634
Train Epoch: 38 [720/864 (83%)] Loss: 0.337143
Train Epoch: 38 [800/864 (93%)] Loss: 0.666703
Test set: Average loss: 2.0127, Accuracy: 44/216 (20%)

Train Epoch: 39 [0/864 (0%)]    Loss: 1.280517
Train Epoch: 39 [80/864 (9%)]   Loss: 1.002397
Train Epoch: 39 [160/864 (19%)] Loss: 0.812761
Train Epoch: 39 [240/864 (28%)] Loss: 0.646624
Train Epoch: 39 [320/864 (37%)] Loss: 1.396016
Train Epoch: 39 [400/864 (46%)] Loss: 0.605108
Train Epoch: 39 [480/864 (56%)] Loss: 0.705610
Train Epoch: 39 [560/864 (65%)] Loss: 0.682916
Train Epoch: 39 [640/864 (74%)] Loss: 0.367920
Train Epoch: 39 [720/864 (83%)] Loss: 0.318818
Train Epoch: 39 [800/864 (93%)] Loss: 1.658422
Test set: Average loss: 2.0510, Accuracy: 37/216 (17%)

Train Epoch: 40 [0/864 (0%)]    Loss: 0.501520
Train Epoch: 40 [80/864 (9%)]   Loss: 0.644471
Train Epoch: 40 [160/864 (19%)] Loss: 0.940771
Train Epoch: 40 [240/864 (28%)] Loss: 0.479545
Train Epoch: 40 [320/864 (37%)] Loss: 1.238307
Train Epoch: 40 [400/864 (46%)] Loss: 1.185899
Train Epoch: 40 [480/864 (56%)] Loss: 0.964339
```

```
Train Epoch: 40 [560/864 (65%)] Loss: 0.837590
Train Epoch: 40 [640/864 (74%)] Loss: 0.633787
Train Epoch: 40 [720/864 (83%)] Loss: 0.671699
Train Epoch: 40 [800/864 (93%)] Loss: 0.878178
Test set: Average loss: 1.9557, Accuracy: 52/216 (24%)

Train Epoch: 41 [0/864 (0%)]    Loss: 0.708588
Train Epoch: 41 [80/864 (9%)]   Loss: 1.809117
Train Epoch: 41 [160/864 (19%)] Loss: 0.961572
Train Epoch: 41 [240/864 (28%)] Loss: 1.133524
Train Epoch: 41 [320/864 (37%)] Loss: 1.370834
Train Epoch: 41 [400/864 (46%)] Loss: 1.308713
Train Epoch: 41 [480/864 (56%)] Loss: 1.050987
Train Epoch: 41 [560/864 (65%)] Loss: 0.678046
Train Epoch: 41 [640/864 (74%)] Loss: 1.245096
Train Epoch: 41 [720/864 (83%)] Loss: 0.631279
Train Epoch: 41 [800/864 (93%)] Loss: 0.966099
Test set: Average loss: 1.6881, Accuracy: 76/216 (35%)

Train Epoch: 42 [0/864 (0%)]    Loss: 0.846886
Train Epoch: 42 [80/864 (9%)]   Loss: 0.913458
Train Epoch: 42 [160/864 (19%)] Loss: 0.526626
Train Epoch: 42 [240/864 (28%)] Loss: 0.577329
Train Epoch: 42 [320/864 (37%)] Loss: 0.609855
Train Epoch: 42 [400/864 (46%)] Loss: 1.322748
Train Epoch: 42 [480/864 (56%)] Loss: 0.751441
Train Epoch: 42 [560/864 (65%)] Loss: 1.080683
Train Epoch: 42 [640/864 (74%)] Loss: 0.747974
Train Epoch: 42 [720/864 (83%)] Loss: 1.324106
Train Epoch: 42 [800/864 (93%)] Loss: 0.244648
Test set: Average loss: 1.7669, Accuracy: 70/216 (32%)

Train Epoch: 43 [0/864 (0%)]    Loss: 1.379042
Train Epoch: 43 [80/864 (9%)]   Loss: 0.634120
Train Epoch: 43 [160/864 (19%)] Loss: 1.157544
Train Epoch: 43 [240/864 (28%)] Loss: 0.762299
Train Epoch: 43 [320/864 (37%)] Loss: 0.708578
Train Epoch: 43 [400/864 (46%)] Loss: 1.052669
Train Epoch: 43 [480/864 (56%)] Loss: 1.162360
Train Epoch: 43 [560/864 (65%)] Loss: 0.589235
Train Epoch: 43 [640/864 (74%)] Loss: 0.683069
Train Epoch: 43 [720/864 (83%)] Loss: 0.260671
Train Epoch: 43 [800/864 (93%)] Loss: 0.823041
Test set: Average loss: 1.8701, Accuracy: 51/216 (24%)

Train Epoch: 44 [0/864 (0%)]    Loss: 0.769664
Train Epoch: 44 [80/864 (9%)]   Loss: 1.338789
Train Epoch: 44 [160/864 (19%)] Loss: 0.558688
```

```
Train Epoch: 44 [240/864 (28%)] Loss: 1.464984
Train Epoch: 44 [320/864 (37%)] Loss: 0.916632
Train Epoch: 44 [400/864 (46%)] Loss: 0.672945
Train Epoch: 44 [480/864 (56%)] Loss: 1.226530
Train Epoch: 44 [560/864 (65%)] Loss: 0.440256
Train Epoch: 44 [640/864 (74%)] Loss: 0.627471
Train Epoch: 44 [720/864 (83%)] Loss: 0.973583
Train Epoch: 44 [800/864 (93%)] Loss: 0.469617
Test set: Average loss: 1.7663, Accuracy: 63/216 (29%)

Train Epoch: 45 [0/864 (0%)]    Loss: 0.549963
Train Epoch: 45 [80/864 (9%)]   Loss: 0.548218
Train Epoch: 45 [160/864 (19%)] Loss: 0.949370
Train Epoch: 45 [240/864 (28%)] Loss: 0.486026
Train Epoch: 45 [320/864 (37%)] Loss: 0.686796
Train Epoch: 45 [400/864 (46%)] Loss: 0.591055
Train Epoch: 45 [480/864 (56%)] Loss: 1.013602
Train Epoch: 45 [560/864 (65%)] Loss: 0.591888
Train Epoch: 45 [640/864 (74%)] Loss: 0.265185
Train Epoch: 45 [720/864 (83%)] Loss: 0.774450
Train Epoch: 45 [800/864 (93%)] Loss: 1.328625
Test set: Average loss: 1.7929, Accuracy: 68/216 (31%)

Train Epoch: 46 [0/864 (0%)]    Loss: 0.888694
Train Epoch: 46 [80/864 (9%)]   Loss: 0.279275
Train Epoch: 46 [160/864 (19%)] Loss: 0.814365
Train Epoch: 46 [240/864 (28%)] Loss: 0.656564
Train Epoch: 46 [320/864 (37%)] Loss: 1.091509
Train Epoch: 46 [400/864 (46%)] Loss: 0.372904
Train Epoch: 46 [480/864 (56%)] Loss: 0.488587
Train Epoch: 46 [560/864 (65%)] Loss: 0.540185
Train Epoch: 46 [640/864 (74%)] Loss: 0.881176
Train Epoch: 46 [720/864 (83%)] Loss: 0.882236
Train Epoch: 46 [800/864 (93%)] Loss: 1.096201
Test set: Average loss: 1.9332, Accuracy: 54/216 (25%)

Train Epoch: 47 [0/864 (0%)]    Loss: 1.251382
Train Epoch: 47 [80/864 (9%)]   Loss: 0.486575
Train Epoch: 47 [160/864 (19%)] Loss: 0.790431
Train Epoch: 47 [240/864 (28%)] Loss: 0.326583
Train Epoch: 47 [320/864 (37%)] Loss: 0.718742
Train Epoch: 47 [400/864 (46%)] Loss: 0.371737
Train Epoch: 47 [480/864 (56%)] Loss: 1.318235
Train Epoch: 47 [560/864 (65%)] Loss: 0.510003
Train Epoch: 47 [640/864 (74%)] Loss: 0.717171
Train Epoch: 47 [720/864 (83%)] Loss: 0.484478
Train Epoch: 47 [800/864 (93%)] Loss: 0.847138
Test set: Average loss: 1.8605, Accuracy: 73/216 (34%)
```

```
Train Epoch: 48 [0/864 (0%)]    Loss: 0.427431
Train Epoch: 48 [80/864 (9%)]   Loss: 0.487201
Train Epoch: 48 [160/864 (19%)] Loss: 0.564830
Train Epoch: 48 [240/864 (28%)] Loss: 0.686130
Train Epoch: 48 [320/864 (37%)] Loss: 0.712798
Train Epoch: 48 [400/864 (46%)] Loss: 0.843749
Train Epoch: 48 [480/864 (56%)] Loss: 1.008767
Train Epoch: 48 [560/864 (65%)] Loss: 1.701514
Train Epoch: 48 [640/864 (74%)] Loss: 0.713939
Train Epoch: 48 [720/864 (83%)] Loss: 0.366782
Train Epoch: 48 [800/864 (93%)] Loss: 0.894736
Test set: Average loss: 1.8887, Accuracy: 51/216 (24%)

Train Epoch: 49 [0/864 (0%)]    Loss: 1.494120
Train Epoch: 49 [80/864 (9%)]   Loss: 0.976958
Train Epoch: 49 [160/864 (19%)] Loss: 0.866933
Train Epoch: 49 [240/864 (28%)] Loss: 0.418380
Train Epoch: 49 [320/864 (37%)] Loss: 0.420312
Train Epoch: 49 [400/864 (46%)] Loss: 0.821285
Train Epoch: 49 [480/864 (56%)] Loss: 0.551852
Train Epoch: 49 [560/864 (65%)] Loss: 0.534683
Train Epoch: 49 [640/864 (74%)] Loss: 0.946050
Train Epoch: 49 [720/864 (83%)] Loss: 0.585346
Train Epoch: 49 [800/864 (93%)] Loss: 0.956052
Test set: Average loss: 1.8111, Accuracy: 74/216 (34%)
```