effnet_rcnn_artist

March 25, 2025

[1]:
```
import torch
import pandas as pd
import numpy as np
from tqdm import tqdm
import wandb
```

[2]:
```
wandb.
 ↪init(entity="ameyar3103-iiit-hyderabad",project="recurrent_conv_art_effnet",␣
 ↪config={
    "epochs": 5,
    "batch_size": 4,
    "learning_rate": 0.001,
    "model": "EFFNET_RCNN"
})
```

```
wandb: Using wandb-core as the SDK backend.  Please refer to
https://wandb.me/wandb-core for more information.
wandb: Currently logged in as: ameyar3103
(ameyar3103-iiit-hyderabad) to https://api.wandb.ai. Use
`wandb login --relogin` to force relogin
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

[2]:
```
<wandb.sdk.wandb_run.Run at 0x784a39c45660>
```

0.1 Data loading

[3]:
```
df_train = pd.read_csv('wikiart_csv/artist_train.csv',header=None,␣
 ↪names=["image_path", "artist_id"])
df_val = pd.read_csv('wikiart_csv/artist_val.csv',header=None,␣
 ↪names=["image_path", "artist_id"])
```

[4]:
```
# get the number of classes
num_classes = 23 # from artist_class.txt
```

[5]:
```
# Gather input data
train_images = df_train['image_path'].values
train_labels = df_train['artist_id'].values

val_images = df_val['image_path'].values
val_labels = df_val['artist_id'].values
```

[6]:
```
from torchvision import transforms
import cv2
```

0.2 Preprocess data and create test and train dataset

[7]:
```
# create test and train dataset for dataloader

def get_image(image_path,image_size=224):
    try:
        img = cv2.imread('./wikiart/' + image_path)
        if img is None:
            raise ValueError(f"Image not loaded: ./wikiart/{image_path}")
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        h, w, _ = img.shape
        scale = 256 / min(h, w)
        new_w = int(w * scale)
        new_h = int(h * scale)
        img_resized = cv2.resize(img, (new_w, new_h))
        start_x = (new_w - image_size) // 2
        start_y = (new_h - image_size) // 2
        img_cropped = img_resized[start_y:start_y+image_size, start_x:
 ↪start_x+image_size]
        img_cropped = img_cropped.astype(np.float32) / 255.0
        img_tensor = torch.from_numpy(img_cropped).permute(2, 0, 1)
        mean = torch.tensor([0.485, 0.456, 0.406]).view(3, 1, 1)
        std  = torch.tensor([0.229, 0.224, 0.225]).view(3, 1, 1)
        img_tensor = (img_tensor - mean) / std
        return img_tensor
    except Exception as e:
        print(f"Error processing {image_path}: {e}")
        return torch.zeros(3, image_size, image_size)

class WikiArtDataset(torch.utils.data.Dataset):
    def __init__(self, images, labels):
        self.images = images
        self.labels = labels
```

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

    def __getitem__(self, idx):
        # image_vectors = []
        # for image in self.images:
        #     image_emb = get_image(image)
        #     image_vectors.append(image_emb)
        # image = torch.stack(image_vectors)
        image = self.images[idx]
        # label should be a one-hot encoded vector
        label = torch.zeros(num_classes)
        label[self.labels[idx]] = 1

        return image, label

train_dataset = WikiArtDataset(train_images, train_labels)
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=32,␣
 ↪shuffle=True)
val_dataset = WikiArtDataset(val_images, val_labels)
val_loader = torch.utils.data.DataLoader(val_dataset, batch_size=32,␣
 ↪shuffle=False)

for i, (images, labels) in enumerate(train_loader):
    print(images)
    print(labels)
    break
```

```
('Post_Impressionism/vincent-van-gogh_pollard-willows-and-setting-sun-1888.jpg',
'Art_Nouveau_Modern/boris-kustodiev_village-holiday-1910.jpg',
'Romanticism/gustave-dore_he-sprang-unpon-the-old-woman-and-ate-her-up.jpg',
'Northern_Renaissance/albrecht-durer_three-studies-from-nature-for-adam-s-
arms-1504.jpg', 'Symbolism/nicholas-roerich_untitled-1915.jpg',
'Symbolism/nicholas-roerich_himalayas-22.jpg', 'Impressionism/eugene-
boudin_cows-near-the-toques.jpg', 'Realism/john-singer-sargent_sheepfold-in-the-
tirol-1915.jpg', 'Expressionism/pablo-picasso_the-absinthe-drinker-portrait-of-
angel-fernandez-de-soto-1903.jpg', 'Symbolism/nicholas-
roerich_kangchenjunga-2.jpg', 'Realism/vincent-van-gogh_beach-and-
sea-1882(1).jpg', 'Symbolism/nicholas-roerich_ladakh-golden-clouds-over-blue-
mountains-1943.jpg', 'Romanticism/gustave-dore_abraham-god-and-two-angels-
png-1852.jpg', 'Impressionism/camille-pissarro_windmill-at-knokke-
belgium-1894.jpg', 'Realism/vincent-van-gogh_the-spire-of-the-church-of-our-
lady-1885.jpg', 'Impressionism/camille-pissarro_portrait-of-madame-felicie-
vellay-estruc.jpg', 'Realism/ivan-shishkin_forest-path-1863.jpg', 'Realism/ivan-
shishkin_landscape-1896.jpg', 'Post_Impressionism/vincent-van-gogh_two-
trees.jpg', 'Post_Impressionism/vincent-van-gogh_skull.jpg',
'Symbolism/martiros-saryan_irises-1903.jpg', 'Impressionism/camille-
```

```
pissarro_the-louvre-morning-sun-1901.jpg', 'Post_Impressionism/vincent-van-
gogh_orchard-with-blossoming-apricot-trees-1888.jpg', 'Expressionism/martiros-
saryan_yerevan-1923.jpg', 'Realism/vincent-van-gogh_fisherman-in-jacket-with-
upturned-collar-1883(1).jpg', 'Impressionism/camille-pissarro_the-thaw-
eragny-1893.jpg', 'Art_Nouveau_Modern/nicholas-roerich_prologue-
forest-1908.jpg', 'Impressionism/claude-monet_relaxing-in-the-garden-
argenteuil.jpg', 'Realism/martiros-saryan_portrait-of-a-i-alikhanov.jpg',
'Impressionism/pierre-auguste-renoir_laundry-boat-by-the-banks-of-the-seine-
near-paris-1873.jpg', 'Impressionism/pierre-auguste-renoir_studies-of-the-
children-of-paul-berard-1881.jpg', 'Impressionism/pierre-auguste-renoir_reading-
couple-edmond-renoir-and-marguerite-legrand-1877.jpg')
tensor([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 1.],
        [0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0.],
        [1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
         0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
         0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
         0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
         0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 1.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
         0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0.],
        [0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 1.],
        [0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
```

```
        0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 1.],
       [0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
        0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
        0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
        0., 0., 0., 0., 0.]])
```

[ ]:
```python
# CNN model

import torch.nn as nn
import torch.nn.functional as F
import torchvision.models as models
import torch

class EffNetLSTM(nn.Module):
    def __init__(self, num_classes):
        super().__init__()

        # EfficientNet-B0 backbone (outputs 1280 channels)
        effnet = models.efficientnet_b0(weights=models.EfficientNet_B0_Weights.
 ↪IMAGENET1K_V1)
        self.cnn = effnet.features

        self.channel_reducer = nn.Sequential(
            nn.Conv2d(1280, 512, kernel_size=1),
            nn.BatchNorm2d(512),
```

```
            nn.ReLU()
        )

        self.lstm = nn.LSTM(
            input_size=512,
            hidden_size=256,
            num_layers=2,
            bidirectional=True,
            batch_first=True
        )

        self.classifier = nn.Sequential(
            nn.Linear(512, 256),
            nn.ReLU(),
            nn.Dropout(0.5),
            nn.Linear(256, num_classes)
        )

    def forward(self, x):
        features = self.cnn(x)

        x = self.channel_reducer(features)
        bs, c, h, w = x.size()
        x = x.permute(0, 2, 3, 1).reshape(bs, h*w, c)

        lstm_out, (h_n, c_n) = self.lstm(x)
        last_hidden = torch.cat((h_n[-2], h_n[-1]), dim=1)

        return self.classifier(last_hidden)

model = EffNetLSTM(num_classes)
model.to('cuda')

# Loss and optimizer
import torch.optim as optim

wandb.watch(model, log="all")
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam([
    {'params': model.cnn.parameters(), 'lr': 1e-5},
    {'params': model.channel_reducer.parameters(), 'lr': 1e-4},
    {'params': model.lstm.parameters(), 'lr': 1e-4},
    {'params': model.classifier.parameters(), 'lr': 1e-4}
])
```

0.3 Training the model

[9]:
```
# Train the model
num_epochs = 20

for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0
    train_bar = tqdm(train_loader, desc=f"Epoch {epoch+1}/{num_epochs}")
    for image_paths, labels in train_bar:
        image_tensors = torch.stack([get_image(image_path) for image_path in␣
 ↪image_paths])
        images = image_tensors.to('cuda')
        labels = labels.to('cuda')

        # Forward pass
        outputs = model(images)
        loss = criterion(outputs, labels)

        # Backward and optimize
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
        train_bar.set_postfix(loss=loss.item())

    avg_train_loss = running_loss / len(train_loader)
    wandb.log({"epoch": epoch+1, "train_loss": avg_train_loss})

    # Validation Loop
    model.eval()
    val_loss = 0.0
    correct = 0
    total = 0
    with torch.no_grad():
        val_bar = tqdm(val_loader, desc="Validation")
        for image_paths, labels in val_bar:
            image_tensors = torch.stack([get_image(image_path) for image_path␣
 ↪in image_paths])
            image_tensors = image_tensors.to('cuda')
            labels = labels.to('cuda')
            outputs = model(image_tensors)
            loss = criterion(outputs, labels)
            val_loss += loss.item()
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
```

```
            correct += (predicted == labels.argmax(dim=1)).sum().item()
            val_bar.set_postfix(loss=loss.item())

    avg_val_loss = val_loss / len(val_loader)
    val_accuracy = 100 * correct / total
    wandb.log({"val_loss": avg_val_loss, "val_accuracy": val_accuracy})
    print(f"Epoch {epoch+1}/{num_epochs} - Train Loss: {avg_train_loss:.4f},␣
↪Val Loss: {avg_val_loss:.4f}, Val Accuracy: {val_accuracy:.2f}%")
    if(epoch%10==0):
        torch.save(model.state_dict(), f"effnet_rcnn_epoch_{epoch+1}.pth")
        torch.save(optimizer.state_dict(),␣
↪f"effnet_rcnn_optimizer_epoch_{epoch+1}.pth")
```

```
Epoch 1/20:  29%|         | 121/418 [01:17<02:47,  1.78it/s, loss=2.85]Corrupt
JPEG data: bad Huffman code
Epoch 1/20:  59%|        | 247/418 [02:36<01:40,  1.70it/s, loss=2.15]Corrupt
JPEG data: premature end of data segment
Epoch 1/20: 100%|      | 418/418 [04:23<00:00,  1.59it/s, loss=2.46]
Validation: 100%|      | 179/179 [01:27<00:00,  2.05it/s, loss=2.17]

Epoch 1/20 - Train Loss: 2.4257, Val Loss: 1.7476, Val Accuracy: 49.07%

Epoch 2/20:  31%|         | 130/418 [01:18<02:46,  1.73it/s, loss=1.48]Corrupt
JPEG data: premature end of data segment
Epoch 2/20:  46%|        | 193/418 [01:56<02:19,  1.61it/s, loss=1.69]Corrupt
JPEG data: bad Huffman code
Epoch 2/20: 100%|      | 418/418 [04:12<00:00,  1.66it/s, loss=3.52]
Validation: 100%|      | 179/179 [01:21<00:00,  2.20it/s, loss=2.07]

Epoch 2/20 - Train Loss: 1.6529, Val Loss: 1.4455, Val Accuracy: 58.52%

Epoch 3/20:  38%|        | 158/418 [01:38<02:46,  1.57it/s, loss=1.3]  Corrupt
JPEG data: bad Huffman code
Epoch 3/20:  84%|      | 352/418 [03:42<00:37,  1.76it/s, loss=0.982]Corrupt
JPEG data: premature end of data segment
Epoch 3/20: 100%|      | 418/418 [04:21<00:00,  1.60it/s, loss=1.7]
Validation: 100%|      | 179/179 [01:23<00:00,  2.14it/s, loss=1.85]

Epoch 3/20 - Train Loss: 1.3759, Val Loss: 1.2750, Val Accuracy: 63.27%

Epoch 4/20:   8%|          | 34/418 [00:21<03:51,  1.66it/s, loss=1.3]  Corrupt
JPEG data: premature end of data segment
Epoch 4/20:  68%|       | 283/418 [02:58<01:26,  1.57it/s, loss=1.31] Corrupt
JPEG data: bad Huffman code
Epoch 4/20: 100%|      | 418/418 [04:22<00:00,  1.59it/s, loss=4.01]
Validation: 100%|      | 179/179 [01:23<00:00,  2.14it/s, loss=1.76]

Epoch 4/20 - Train Loss: 1.1793, Val Loss: 1.1903, Val Accuracy: 65.70%

Epoch 5/20:  26%|         | 109/418 [01:06<03:29,  1.47it/s, loss=0.775]Corrupt
JPEG data: bad Huffman code
```

```
Epoch 5/20:  38%|        | 158/418 [01:36<02:52,  1.51it/s, loss=1.04] Corrupt
JPEG data: premature end of data segment
Epoch 5/20: 100%|      | 418/418 [04:18<00:00,  1.62it/s, loss=2.71]
Validation: 100%|      | 179/179 [01:24<00:00,  2.12it/s, loss=1.79]

Epoch 5/20 - Train Loss: 1.0197, Val Loss: 1.1001, Val Accuracy: 68.21%

Epoch 6/20:  64%|       | 269/418 [02:46<01:28,  1.69it/s, loss=0.958]Corrupt
JPEG data: bad Huffman code
Epoch 6/20:  79%|       | 332/418 [03:25<00:49,  1.74it/s, loss=0.853]Corrupt
JPEG data: premature end of data segment
Epoch 6/20: 100%|      | 418/418 [04:17<00:00,  1.62it/s, loss=2.03]
Validation: 100%|      | 179/179 [01:24<00:00,  2.13it/s, loss=1.55]

Epoch 6/20 - Train Loss: 0.9011, Val Loss: 1.0193, Val Accuracy: 71.29%

Epoch 7/20:  63%|       | 263/418 [02:41<01:46,  1.46it/s, loss=0.368]Corrupt
JPEG data: premature end of data segment
Epoch 7/20:  82%|       | 341/418 [03:30<00:47,  1.62it/s, loss=0.648]Corrupt
JPEG data: bad Huffman code
Epoch 7/20: 100%|      | 418/418 [04:17<00:00,  1.62it/s, loss=2.85]
Validation: 100%|      | 179/179 [01:25<00:00,  2.10it/s, loss=1.1]

Epoch 7/20 - Train Loss: 0.7918, Val Loss: 1.0017, Val Accuracy: 71.94%

Epoch 8/20:  51%|       | 215/418 [02:12<02:01,  1.67it/s, loss=0.845]Corrupt
JPEG data: bad Huffman code
Epoch 8/20:  91%|       | 380/418 [03:55<00:21,  1.75it/s, loss=0.504]Corrupt
JPEG data: premature end of data segment
Epoch 8/20: 100%|      | 418/418 [04:19<00:00,  1.61it/s, loss=4.05]
Validation: 100%|      | 179/179 [01:23<00:00,  2.15it/s, loss=1.21]

Epoch 8/20 - Train Loss: 0.7042, Val Loss: 0.9478, Val Accuracy: 73.27%

Epoch 9/20:  24%|        | 100/418 [00:59<02:57,  1.79it/s, loss=0.236]Corrupt
JPEG data: bad Huffman code
Epoch 9/20:  45%|        | 187/418 [01:51<02:08,  1.79it/s, loss=0.509]Corrupt
JPEG data: premature end of data segment
Epoch 9/20: 100%|      | 418/418 [04:05<00:00,  1.70it/s, loss=5.12]
Validation: 100%|      | 179/179 [01:18<00:00,  2.28it/s, loss=1.3]

Epoch 9/20 - Train Loss: 0.6483, Val Loss: 0.9496, Val Accuracy: 73.20%

Epoch 10/20:   4%|         | 16/418 [00:08<03:54,  1.71it/s, loss=0.684]Corrupt
JPEG data: premature end of data segment
Epoch 10/20:  12%|        | 50/418 [00:28<03:33,  1.72it/s, loss=0.374]Corrupt
JPEG data: bad Huffman code
Epoch 10/20: 100%|      | 418/418 [04:08<00:00,  1.68it/s, loss=1.88]
Validation: 100%|      | 179/179 [01:18<00:00,  2.28it/s, loss=2.12]

Epoch 10/20 - Train Loss: 0.5608, Val Loss: 0.9432, Val Accuracy: 74.04%

Epoch 11/20:  42%|       | 174/418 [01:40<02:17,  1.78it/s,
loss=0.461]Corrupt JPEG data: bad Huffman code
```

```
Epoch 11/20:  53%|       | 220/418 [02:08<01:58,  1.68it/s,
loss=0.468]Corrupt JPEG data: premature end of data segment
Epoch 11/20: 100%|      | 418/418 [04:02<00:00,  1.72it/s, loss=3.39]
Validation: 100%|      | 179/179 [01:17<00:00,  2.30it/s, loss=0.906]

Epoch 11/20 - Train Loss: 0.5230, Val Loss: 0.9450, Val Accuracy: 74.62%

Epoch 12/20:  49%|       | 204/418 [01:58<01:58,  1.81it/s,
loss=0.493]Corrupt JPEG data: bad Huffman code
Epoch 12/20:  64%|       | 267/418 [02:35<01:25,  1.77it/s, loss=0.5]
Corrupt JPEG data: premature end of data segment
Epoch 12/20: 100%|      | 418/418 [04:03<00:00,  1.72it/s, loss=2.96]
Validation: 100%|      | 179/179 [01:18<00:00,  2.28it/s, loss=1.19]

Epoch 12/20 - Train Loss: 0.4724, Val Loss: 0.9030, Val Accuracy: 75.94%

Epoch 13/20:  64%|       | 269/418 [02:34<01:27,  1.69it/s, loss=0.31]
Corrupt JPEG data: bad Huffman code
Epoch 13/20:  74%|       | 309/418 [02:58<01:05,  1.66it/s,
loss=0.0834]Corrupt JPEG data: premature end of data segment
Epoch 13/20: 100%|      | 418/418 [04:02<00:00,  1.72it/s, loss=2.82]
Validation: 100%|      | 179/179 [01:17<00:00,  2.31it/s, loss=1.12]

Epoch 13/20 - Train Loss: 0.4297, Val Loss: 0.9521, Val Accuracy: 75.31%

Epoch 14/20:  53%|       | 222/418 [02:08<01:52,  1.74it/s, loss=0.378]
Corrupt JPEG data: bad Huffman code
Epoch 14/20:  77%|       | 322/418 [03:06<00:58,  1.63it/s, loss=0.523]
Corrupt JPEG data: premature end of data segment
Epoch 14/20: 100%|      | 418/418 [04:01<00:00,  1.73it/s, loss=2.25]
Validation: 100%|      | 179/179 [01:18<00:00,  2.27it/s, loss=0.955]

Epoch 14/20 - Train Loss: 0.3955, Val Loss: 0.9157, Val Accuracy: 75.50%

Epoch 15/20:  13%|         | 55/418 [00:31<03:31,  1.72it/s, loss=0.259]Corrupt
JPEG data: premature end of data segment
Epoch 15/20:  28%|         | 117/418 [01:08<02:52,  1.75it/s, loss=0.12]
Corrupt JPEG data: bad Huffman code
Epoch 15/20: 100%|      | 418/418 [04:01<00:00,  1.73it/s, loss=3.1]
Validation: 100%|      | 179/179 [01:17<00:00,  2.30it/s, loss=1.15]

Epoch 15/20 - Train Loss: 0.3445, Val Loss: 0.9240, Val Accuracy: 76.17%

Epoch 16/20:  32%|        | 135/418 [01:18<02:35,  1.82it/s,
loss=0.501]Corrupt JPEG data: premature end of data segment
Epoch 16/20:  89%|      | 372/418 [03:35<00:26,  1.71it/s, loss=0.199]
Corrupt JPEG data: bad Huffman code
Epoch 16/20: 100%|      | 418/418 [04:01<00:00,  1.73it/s, loss=2.94]
Validation: 100%|      | 179/179 [01:17<00:00,  2.31it/s, loss=1.04]

Epoch 16/20 - Train Loss: 0.3159, Val Loss: 0.9512, Val Accuracy: 76.04%

Epoch 17/20:  17%|         | 73/418 [00:41<03:10,  1.81it/s, loss=0.419]Corrupt
JPEG data: premature end of data segment
```

```
Epoch 17/20:  41%|        | 170/418 [01:38<02:18,  1.80it/s, loss=0.242]
Corrupt JPEG data: bad Huffman code
Epoch 17/20: 100%|      | 418/418 [04:01<00:00,  1.73it/s, loss=0.0301]
Validation: 100%|      | 179/179 [01:18<00:00,  2.29it/s, loss=1.4]

Epoch 17/20 - Train Loss: 0.3042, Val Loss: 0.9682, Val Accuracy: 77.01%

Epoch 18/20:  21%|         | 87/418 [00:50<03:18,  1.67it/s, loss=0.547]
Corrupt JPEG data: bad Huffman code
Epoch 18/20:  67%|       | 280/418 [02:43<01:19,  1.74it/s, loss=0.197]
Corrupt JPEG data: premature end of data segment
Epoch 18/20: 100%|      | 418/418 [04:01<00:00,  1.73it/s, loss=0.507]
Validation: 100%|      | 179/179 [01:18<00:00,  2.29it/s, loss=0.746]

Epoch 18/20 - Train Loss: 0.2655, Val Loss: 0.9756, Val Accuracy: 76.31%

Epoch 19/20:  70%|       | 292/418 [02:51<01:09,  1.82it/s, loss=0.69]
Corrupt JPEG data: premature end of data segment
Epoch 19/20:  90%|      | 377/418 [03:39<00:24,  1.69it/s, loss=0.445]
Corrupt JPEG data: bad Huffman code
Epoch 19/20: 100%|      | 418/418 [04:01<00:00,  1.73it/s, loss=3.1]
Validation: 100%|      | 179/179 [01:17<00:00,  2.32it/s, loss=1.28]

Epoch 19/20 - Train Loss: 0.2548, Val Loss: 0.9450, Val Accuracy: 76.64%

Epoch 20/20:   8%|         | 32/418 [00:19<03:55,  1.64it/s, loss=0.512]Corrupt
JPEG data: premature end of data segment
Epoch 20/20:  64%|       | 267/418 [02:34<01:26,  1.74it/s,
loss=0.0613]Corrupt JPEG data: bad Huffman code
Epoch 20/20: 100%|      | 418/418 [04:01<00:00,  1.73it/s, loss=2]
Validation: 100%|      | 179/179 [01:17<00:00,  2.31it/s, loss=0.977]

Epoch 20/20 - Train Loss: 0.2577, Val Loss: 0.9774, Val Accuracy: 76.17%
```