train

March 8, 2025

[12]:
```python
import torch
import numpy as np
from torch.utils.data import DataLoader, random_split, TensorDataset
import torch.nn as nn
import torch.optim as optim
from transformers import ViTForImageClassification, ViTConfig
import time
```

[13]:
```python
# Use CPU
device = "cpu"
print(f"Using device: {device}")
```

```
Using device: cpu
```

[14]:
```python
# Load and modify the ViT configuration
print("Initializing ViT model...")
config = ViTConfig.from_pretrained("google/vit-base-patch16-224")
config.num_channels = 1  # Change from RGB (3 channels) to Grayscale (1 channel)
config.image_size = 224  # Ensure model expects 224x224 input
```

```
Initializing ViT model…
```

[15]:
```python
# Initialize model with modified config
model = ViTForImageClassification(config)

# Adjust patch embeddings to accept single-channel input
model.vit.embeddings.patch_embeddings.projection = nn.Conv2d(
    in_channels=1, out_channels=config.hidden_size, kernel_size=16, stride=16
)

# Move model to CPU
model.to(device)
```

[15]:
```
ViTForImageClassification(
  (vit): ViTModel(
    (embeddings): ViTEmbeddings(
      (patch_embeddings): ViTPatchEmbeddings(
        (projection): Conv2d(1, 768, kernel_size=(16, 16), stride=(16, 16))
      )
```

```
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (encoder): ViTEncoder(
        (layer): ModuleList(
          (0-11): 12 x ViTLayer(
            (attention): ViTSdpaAttention(
              (attention): ViTSdpaSelfAttention(
                (query): Linear(in_features=768, out_features=768, bias=True)
                (key): Linear(in_features=768, out_features=768, bias=True)
                (value): Linear(in_features=768, out_features=768, bias=True)
                (dropout): Dropout(p=0.0, inplace=False)
              )
              (output): ViTSelfOutput(
                (dense): Linear(in_features=768, out_features=768, bias=True)
                (dropout): Dropout(p=0.0, inplace=False)
              )
            )
            (intermediate): ViTIntermediate(
              (dense): Linear(in_features=768, out_features=3072, bias=True)
              (intermediate_act_fn): GELUActivation()
            )
            (output): ViTOutput(
              (dense): Linear(in_features=3072, out_features=768, bias=True)
              (dropout): Dropout(p=0.0, inplace=False)
            )
            (layernorm_before): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
            (layernorm_after): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
          )
        )
      )
      (layernorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    )
    (classifier): Linear(in_features=768, out_features=1000, bias=True)
)
```

[16]:

```
# Function to pad images from (64, 72) to (224, 224)
def pad_images(images):
    padded = torch.zeros((images.shape[0], 1, 224, 224), dtype=torch.float32)
    padded[:, :, :64, :72] = images  # Place original images in top-left corner
    return padded

# Load datasets
print("Loading datasets...")
data1 = np.load("Run355456_Dataset_jqkne.npy")  # Shape: (N1, 64, 72)
data2 = np.load("Run357479_Dataset_iodic.npy")  # Shape: (N2, 64, 72)
```

```
# Normalize data (important for convergence)
print("Normalizing datasets...")
data1 = data1 / data1.max()
data2 = data2 / data2.max()

# Assign labels
labels1 = np.zeros(len(data1), dtype=np.int64)  # Class 0
labels2 = np.ones(len(data2), dtype=np.int64)   # Class 1

# Merge datasets
print("Merging datasets...")
data = np.concatenate((data1, data2), axis=0)  # (Total_N, 64, 72)
labels = np.concatenate((labels1, labels2), axis=0)  # (Total_N,)

# Convert to PyTorch tensors
print("Converting datasets to PyTorch tensors...")
data = torch.tensor(data, dtype=torch.float32).unsqueeze(1)  # (N, 1, 64, 72)
data = pad_images(data)  # Convert (1, 64, 72) → (1, 224, 224)
labels = torch.tensor(labels, dtype=torch.long)
```

```
Loading datasets…
Normalizing datasets…
Merging datasets…
Converting datasets to PyTorch tensors…
```

[17]:

```
# Train/Val/Test split (70-20-10)
train_size = int(0.7 * len(data))
val_size = int(0.2 * len(data))
test_size = len(data) - train_size - val_size

print(f"Dataset split: {train_size} train, {val_size} val, {test_size} test.")
train_data, val_data, test_data = random_split(TensorDataset(data, labels),␣
 ↪[train_size, val_size, test_size])

# Initialize DataLoaders with smaller batch size for CPU
batch_size = 16 # Reduced batch size for CPU
print(f"Initializing DataLoaders with batch size {batch_size}...")
train_loader = DataLoader(train_data, batch_size=batch_size, shuffle=True,␣
 ↪num_workers=0)
val_loader = DataLoader(val_data, batch_size=batch_size, num_workers=0)
test_loader = DataLoader(test_data, batch_size=batch_size, num_workers=0)
```

```
Dataset split: 14000 train, 4000 val, 2000 test.
Initializing DataLoaders with batch size 16…
```

[18]:

```
# Loss and Optimizer
print("Setting up optimizer and loss function...")
```

```python
criterion = nn.CrossEntropyLoss()
optimizer = optim.AdamW(model.parameters(), lr=2e-4, weight_decay=1e-4)

# Learning rate scheduler
scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(
    optimizer, mode='max', factor=0.5, patience=2, verbose=True
)
```

Setting up optimizer and loss function…

[19]:
```python
# Training parameters
epochs = 5
best_val_acc = 0.0
best_train_acc = 0.0
```

[20]:
```python
# Helper function for validation
def validate(model, val_loader):
    model.eval()
    val_loss = 0.0
    correct = 0
    total = 0

    with torch.no_grad():
        for images, labels in val_loader:
            images, labels = images.to(device), labels.to(device)
            outputs = model(images).logits
            loss = criterion(outputs, labels)

            val_loss += loss.item()
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    return val_loss / len(val_loader), correct / total
```

[24]:
```python
print("\n--- Training Starts ---")
for epoch in range(epochs):
    start_time = time.time()
    print(f"\n  Epoch {epoch+1}/{epochs} starts...")

    # Training phase
    model.train()
    total_loss = 0.0
    correct = 0
    total = 0

    for batch_idx, (images, labels) in enumerate(train_loader):
        images, labels = images.to(device), labels.to(device)
```

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

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

        # Statistics
        total_loss += loss.item()
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

        # Print progress
        if (batch_idx + 1) % 10 == 0 or batch_idx + 1 == len(train_loader):
            print(f"  Batch {batch_idx + 1}/{len(train_loader)} | "
                  f"Loss: {total_loss/(batch_idx+1):.4f} | "
                  f"Acc: {correct/total:.4f}")

    train_loss = total_loss / len(train_loader)
    train_acc = correct / total

    # Validation phase
    val_loss, val_acc = validate(model, val_loader)

    # Update scheduler based on validation accuracy
    scheduler.step(val_acc)

    # Save best model
    if val_acc > best_val_acc or train_acc > best_train_acc:
        best_val_acc = val_acc
        best_train_acc = train_acc
        torch.save(model.state_dict(), "best_vit_model_jupyternotebook.pth")
        print(f"    New best model saved with validation accuracy: {val_acc:.
↪4f}")

    # Print epoch summary
    epoch_time = time.time() - start_time
    print(f" Epoch {epoch+1} completed in {epoch_time:.1f}s | "
          f"Train Loss: {train_loss:.4f} | Train Acc: {train_acc:.4f} | "
          f"Val Loss: {val_loss:.4f} | Val Acc: {val_acc:.4f}")
```

```
--- Training Starts ---
```

```
Epoch 1/5 starts…
 Batch 10/875 | Loss: 0.7614 | Acc: 0.4375
 Batch 20/875 | Loss: 0.7382 | Acc: 0.4813
 Batch 30/875 | Loss: 0.7245 | Acc: 0.5021
 Batch 40/875 | Loss: 0.6743 | Acc: 0.5750
 Batch 50/875 | Loss: 0.5607 | Acc: 0.6600
 Batch 60/875 | Loss: 0.4675 | Acc: 0.7167
 Batch 70/875 | Loss: 0.4065 | Acc: 0.7562
 Batch 80/875 | Loss: 0.3717 | Acc: 0.7805
 Batch 90/875 | Loss: 0.3440 | Acc: 0.8028
 Batch 100/875 | Loss: 0.3125 | Acc: 0.8213
 Batch 110/875 | Loss: 0.2857 | Acc: 0.8375
 Batch 120/875 | Loss: 0.2623 | Acc: 0.8510
 Batch 130/875 | Loss: 0.2428 | Acc: 0.8620
 Batch 140/875 | Loss: 0.2289 | Acc: 0.8710
 Batch 150/875 | Loss: 0.2152 | Acc: 0.8792
 Batch 160/875 | Loss: 0.2023 | Acc: 0.8867
 Batch 170/875 | Loss: 0.1904 | Acc: 0.8934
 Batch 180/875 | Loss: 0.1799 | Acc: 0.8993
 Batch 190/875 | Loss: 0.1704 | Acc: 0.9046
 Batch 200/875 | Loss: 0.1619 | Acc: 0.9094
 Batch 210/875 | Loss: 0.1567 | Acc: 0.9134
 Batch 220/875 | Loss: 0.1498 | Acc: 0.9173
 Batch 230/875 | Loss: 0.1434 | Acc: 0.9209
 Batch 240/875 | Loss: 0.1375 | Acc: 0.9242
 Batch 250/875 | Loss: 0.1320 | Acc: 0.9273
 Batch 260/875 | Loss: 0.1269 | Acc: 0.9300
 Batch 270/875 | Loss: 0.1223 | Acc: 0.9326
 Batch 280/875 | Loss: 0.1179 | Acc: 0.9350
 Batch 290/875 | Loss: 0.1139 | Acc: 0.9373
 Batch 300/875 | Loss: 0.1101 | Acc: 0.9394
 Batch 310/875 | Loss: 0.1066 | Acc: 0.9413
 Batch 320/875 | Loss: 0.1032 | Acc: 0.9432
 Batch 330/875 | Loss: 0.1001 | Acc: 0.9449
 Batch 340/875 | Loss: 0.0972 | Acc: 0.9465
 Batch 350/875 | Loss: 0.0944 | Acc: 0.9480
 Batch 360/875 | Loss: 0.0918 | Acc: 0.9495
 Batch 370/875 | Loss: 0.0893 | Acc: 0.9508
 Batch 380/875 | Loss: 0.0870 | Acc: 0.9521
 Batch 390/875 | Loss: 0.0854 | Acc: 0.9532
 Batch 400/875 | Loss: 0.0833 | Acc: 0.9544
 Batch 410/875 | Loss: 0.0813 | Acc: 0.9555
 Batch 420/875 | Loss: 0.0794 | Acc: 0.9565
 Batch 430/875 | Loss: 0.0775 | Acc: 0.9576
 Batch 440/875 | Loss: 0.0758 | Acc: 0.9585
 Batch 450/875 | Loss: 0.0741 | Acc: 0.9594
 Batch 460/875 | Loss: 0.0725 | Acc: 0.9603
 Batch 470/875 | Loss: 0.0710 | Acc: 0.9612
```

```
  Batch 480/875 | Loss: 0.0695 | Acc: 0.9620
  Batch 490/875 | Loss: 0.0681 | Acc: 0.9628
  Batch 500/875 | Loss: 0.0667 | Acc: 0.9635
  Batch 510/875 | Loss: 0.0654 | Acc: 0.9642
  Batch 520/875 | Loss: 0.0642 | Acc: 0.9649
  Batch 530/875 | Loss: 0.0630 | Acc: 0.9656
  Batch 540/875 | Loss: 0.0618 | Acc: 0.9662
  Batch 550/875 | Loss: 0.0607 | Acc: 0.9668
  Batch 560/875 | Loss: 0.0596 | Acc: 0.9674
  Batch 570/875 | Loss: 0.0586 | Acc: 0.9680
  Batch 580/875 | Loss: 0.0575 | Acc: 0.9685
  Batch 590/875 | Loss: 0.0566 | Acc: 0.9691
  Batch 600/875 | Loss: 0.0556 | Acc: 0.9696
  Batch 610/875 | Loss: 0.0547 | Acc: 0.9701
  Batch 620/875 | Loss: 0.0538 | Acc: 0.9706
  Batch 630/875 | Loss: 0.0530 | Acc: 0.9710
  Batch 640/875 | Loss: 0.0522 | Acc: 0.9715
  Batch 650/875 | Loss: 0.0514 | Acc: 0.9719
  Batch 660/875 | Loss: 0.0506 | Acc: 0.9723
  Batch 670/875 | Loss: 0.0498 | Acc: 0.9728
  Batch 680/875 | Loss: 0.0491 | Acc: 0.9732
  Batch 690/875 | Loss: 0.0484 | Acc: 0.9736
  Batch 700/875 | Loss: 0.0477 | Acc: 0.9739
  Batch 710/875 | Loss: 0.0470 | Acc: 0.9743
  Batch 720/875 | Loss: 0.0464 | Acc: 0.9747
  Batch 730/875 | Loss: 0.0458 | Acc: 0.9750
  Batch 740/875 | Loss: 0.0451 | Acc: 0.9753
  Batch 750/875 | Loss: 0.0445 | Acc: 0.9757
  Batch 760/875 | Loss: 0.0440 | Acc: 0.9760
  Batch 770/875 | Loss: 0.0434 | Acc: 0.9763
  Batch 780/875 | Loss: 0.0428 | Acc: 0.9766
  Batch 790/875 | Loss: 0.0423 | Acc: 0.9769
  Batch 800/875 | Loss: 0.0418 | Acc: 0.9772
  Batch 810/875 | Loss: 0.0412 | Acc: 0.9775
  Batch 820/875 | Loss: 0.0407 | Acc: 0.9777
  Batch 830/875 | Loss: 0.0403 | Acc: 0.9780
  Batch 840/875 | Loss: 0.0398 | Acc: 0.9783
  Batch 850/875 | Loss: 0.0393 | Acc: 0.9785
  Batch 860/875 | Loss: 0.0389 | Acc: 0.9788
  Batch 870/875 | Loss: 0.0384 | Acc: 0.9790
  Batch 875/875 | Loss: 0.0382 | Acc: 0.9791
    New best model saved with validation accuracy: 1.0000
  Epoch 1 completed in 2399.9s | Train Loss: 0.0382 | Train Acc: 0.9791 | Val
Loss: 0.0001 | Val Acc: 1.0000

  Epoch 2/5 starts…
  Batch 10/875 | Loss: 0.0001 | Acc: 1.0000
  Batch 20/875 | Loss: 0.0001 | Acc: 1.0000
```

```
Batch 30/875 | Loss: 0.0001 | Acc: 1.0000
Batch 40/875 | Loss: 0.0001 | Acc: 1.0000
Batch 50/875 | Loss: 0.0001 | Acc: 1.0000
Batch 60/875 | Loss: 0.0001 | Acc: 1.0000
Batch 70/875 | Loss: 0.0001 | Acc: 1.0000
Batch 80/875 | Loss: 0.0001 | Acc: 1.0000
Batch 90/875 | Loss: 0.0001 | Acc: 1.0000
Batch 100/875 | Loss: 0.0001 | Acc: 1.0000
Batch 110/875 | Loss: 0.0001 | Acc: 1.0000
Batch 120/875 | Loss: 0.0001 | Acc: 1.0000
Batch 130/875 | Loss: 0.0001 | Acc: 1.0000
Batch 140/875 | Loss: 0.0001 | Acc: 1.0000
Batch 150/875 | Loss: 0.0001 | Acc: 1.0000
Batch 160/875 | Loss: 0.0001 | Acc: 1.0000
Batch 170/875 | Loss: 0.0001 | Acc: 1.0000
Batch 180/875 | Loss: 0.0001 | Acc: 1.0000
Batch 190/875 | Loss: 0.0001 | Acc: 1.0000
Batch 200/875 | Loss: 0.0001 | Acc: 1.0000
Batch 210/875 | Loss: 0.0001 | Acc: 1.0000
Batch 220/875 | Loss: 0.0001 | Acc: 1.0000
Batch 230/875 | Loss: 0.0001 | Acc: 1.0000
Batch 240/875 | Loss: 0.0001 | Acc: 1.0000
Batch 250/875 | Loss: 0.0001 | Acc: 1.0000
Batch 260/875 | Loss: 0.0001 | Acc: 1.0000
Batch 270/875 | Loss: 0.0001 | Acc: 1.0000
Batch 280/875 | Loss: 0.0001 | Acc: 1.0000
Batch 290/875 | Loss: 0.0001 | Acc: 1.0000
Batch 300/875 | Loss: 0.0001 | Acc: 1.0000
Batch 310/875 | Loss: 0.0001 | Acc: 1.0000
Batch 320/875 | Loss: 0.0001 | Acc: 1.0000
Batch 330/875 | Loss: 0.0001 | Acc: 1.0000
Batch 340/875 | Loss: 0.0001 | Acc: 1.0000
Batch 350/875 | Loss: 0.0001 | Acc: 1.0000
Batch 360/875 | Loss: 0.0001 | Acc: 1.0000
Batch 370/875 | Loss: 0.0001 | Acc: 1.0000
Batch 380/875 | Loss: 0.0001 | Acc: 1.0000
Batch 390/875 | Loss: 0.0001 | Acc: 1.0000
Batch 400/875 | Loss: 0.0001 | Acc: 1.0000
Batch 410/875 | Loss: 0.0001 | Acc: 1.0000
Batch 420/875 | Loss: 0.0001 | Acc: 1.0000
Batch 430/875 | Loss: 0.0001 | Acc: 1.0000
Batch 440/875 | Loss: 0.0001 | Acc: 1.0000
Batch 450/875 | Loss: 0.0001 | Acc: 1.0000
Batch 460/875 | Loss: 0.0001 | Acc: 1.0000
Batch 470/875 | Loss: 0.0001 | Acc: 1.0000
Batch 480/875 | Loss: 0.0001 | Acc: 1.0000
Batch 490/875 | Loss: 0.0001 | Acc: 1.0000
Batch 500/875 | Loss: 0.0001 | Acc: 1.0000
```

```
 Batch 510/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 520/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 530/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 540/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 550/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 560/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 570/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 580/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 590/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 600/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 610/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 620/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 630/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 640/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 650/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 660/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 670/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 680/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 690/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 700/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 710/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 720/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 730/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 740/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 750/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 760/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 770/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 780/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 790/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 800/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 810/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 820/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 830/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 840/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 850/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 860/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 870/875 | Loss: 0.0001 | Acc: 1.0000
 Batch 875/875 | Loss: 0.0001 | Acc: 1.0000
   New best model saved with validation accuracy: 1.0000
 Epoch 2 completed in 2600.9s | Train Loss: 0.0001 | Train Acc: 1.0000 | Val
Loss: 0.0000 | Val Acc: 1.0000

 Epoch 3/5 starts…
 Batch 10/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 20/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 30/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 40/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 50/875 | Loss: 0.0000 | Acc: 1.0000
```

```
Batch 60/875 | Loss: 0.0000 | Acc: 1.0000
Batch 70/875 | Loss: 0.0000 | Acc: 1.0000
Batch 80/875 | Loss: 0.0000 | Acc: 1.0000
Batch 90/875 | Loss: 0.0000 | Acc: 1.0000
Batch 100/875 | Loss: 0.0000 | Acc: 1.0000
Batch 110/875 | Loss: 0.0000 | Acc: 1.0000
Batch 120/875 | Loss: 0.0000 | Acc: 1.0000
Batch 130/875 | Loss: 0.0000 | Acc: 1.0000
Batch 140/875 | Loss: 0.0000 | Acc: 1.0000
Batch 150/875 | Loss: 0.0000 | Acc: 1.0000
Batch 160/875 | Loss: 0.0000 | Acc: 1.0000
Batch 170/875 | Loss: 0.0000 | Acc: 1.0000
Batch 180/875 | Loss: 0.0000 | Acc: 1.0000
Batch 190/875 | Loss: 0.0000 | Acc: 1.0000
Batch 200/875 | Loss: 0.0000 | Acc: 1.0000
Batch 210/875 | Loss: 0.0000 | Acc: 1.0000
Batch 220/875 | Loss: 0.0000 | Acc: 1.0000
Batch 230/875 | Loss: 0.0000 | Acc: 1.0000
Batch 240/875 | Loss: 0.0000 | Acc: 1.0000
Batch 250/875 | Loss: 0.0000 | Acc: 1.0000
Batch 260/875 | Loss: 0.0000 | Acc: 1.0000
Batch 270/875 | Loss: 0.0000 | Acc: 1.0000
Batch 280/875 | Loss: 0.0000 | Acc: 1.0000
Batch 290/875 | Loss: 0.0000 | Acc: 1.0000
Batch 300/875 | Loss: 0.0000 | Acc: 1.0000
Batch 310/875 | Loss: 0.0000 | Acc: 1.0000
Batch 320/875 | Loss: 0.0000 | Acc: 1.0000
Batch 330/875 | Loss: 0.0000 | Acc: 1.0000
Batch 340/875 | Loss: 0.0000 | Acc: 1.0000
Batch 350/875 | Loss: 0.0000 | Acc: 1.0000
Batch 360/875 | Loss: 0.0000 | Acc: 1.0000
Batch 370/875 | Loss: 0.0000 | Acc: 1.0000
Batch 380/875 | Loss: 0.0000 | Acc: 1.0000
Batch 390/875 | Loss: 0.0000 | Acc: 1.0000
Batch 400/875 | Loss: 0.0000 | Acc: 1.0000
Batch 410/875 | Loss: 0.0000 | Acc: 1.0000
Batch 420/875 | Loss: 0.0000 | Acc: 1.0000
Batch 430/875 | Loss: 0.0000 | Acc: 1.0000
Batch 440/875 | Loss: 0.0000 | Acc: 1.0000
Batch 450/875 | Loss: 0.0000 | Acc: 1.0000
Batch 460/875 | Loss: 0.0000 | Acc: 1.0000
Batch 470/875 | Loss: 0.0000 | Acc: 1.0000
Batch 480/875 | Loss: 0.0000 | Acc: 1.0000
Batch 490/875 | Loss: 0.0000 | Acc: 1.0000
Batch 500/875 | Loss: 0.0000 | Acc: 1.0000
Batch 510/875 | Loss: 0.0000 | Acc: 1.0000
Batch 520/875 | Loss: 0.0000 | Acc: 1.0000
Batch 530/875 | Loss: 0.0000 | Acc: 1.0000
```

```
 Batch 540/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 550/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 560/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 570/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 580/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 590/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 600/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 610/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 620/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 630/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 640/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 650/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 660/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 670/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 680/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 690/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 700/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 710/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 720/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 730/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 740/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 750/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 760/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 770/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 780/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 790/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 800/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 810/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 820/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 830/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 840/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 850/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 860/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 870/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 875/875 | Loss: 0.0000 | Acc: 1.0000
Epoch 3 completed in 2807.7s | Train Loss: 0.0000 | Train Acc: 1.0000 | Val
Loss: 0.0000 | Val Acc: 1.0000

Epoch 4/5 starts…
 Batch 10/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 20/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 30/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 40/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 50/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 60/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 70/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 80/875 | Loss: 0.0000 | Acc: 1.0000
 Batch 90/875 | Loss: 0.0000 | Acc: 1.0000
```

```
Batch 100/875 | Loss: 0.0000 | Acc: 1.0000
Batch 110/875 | Loss: 0.0000 | Acc: 1.0000
Batch 120/875 | Loss: 0.0000 | Acc: 1.0000
Batch 130/875 | Loss: 0.0000 | Acc: 1.0000
Batch 140/875 | Loss: 0.0000 | Acc: 1.0000
Batch 150/875 | Loss: 0.0000 | Acc: 1.0000
Batch 160/875 | Loss: 0.0000 | Acc: 1.0000
Batch 170/875 | Loss: 0.0000 | Acc: 1.0000
Batch 180/875 | Loss: 0.0000 | Acc: 1.0000
Batch 190/875 | Loss: 0.0000 | Acc: 1.0000
Batch 200/875 | Loss: 0.0000 | Acc: 1.0000
Batch 210/875 | Loss: 0.0000 | Acc: 1.0000
Batch 220/875 | Loss: 0.0000 | Acc: 1.0000
Batch 230/875 | Loss: 0.0000 | Acc: 1.0000
Batch 240/875 | Loss: 0.0000 | Acc: 1.0000
Batch 250/875 | Loss: 0.0000 | Acc: 1.0000
Batch 260/875 | Loss: 0.0000 | Acc: 1.0000
Batch 270/875 | Loss: 0.0000 | Acc: 1.0000
Batch 280/875 | Loss: 0.0000 | Acc: 1.0000
Batch 290/875 | Loss: 0.0000 | Acc: 1.0000
Batch 300/875 | Loss: 0.0000 | Acc: 1.0000
Batch 310/875 | Loss: 0.0000 | Acc: 1.0000
Batch 320/875 | Loss: 0.0000 | Acc: 1.0000
Batch 330/875 | Loss: 0.0000 | Acc: 1.0000
Batch 340/875 | Loss: 0.0000 | Acc: 1.0000
Batch 350/875 | Loss: 0.0000 | Acc: 1.0000
Batch 360/875 | Loss: 0.0000 | Acc: 1.0000
Batch 370/875 | Loss: 0.0000 | Acc: 1.0000
Batch 380/875 | Loss: 0.0000 | Acc: 1.0000
Batch 390/875 | Loss: 0.0000 | Acc: 1.0000
Batch 400/875 | Loss: 0.0000 | Acc: 1.0000
Batch 410/875 | Loss: 0.0000 | Acc: 1.0000
Batch 420/875 | Loss: 0.0000 | Acc: 1.0000
Batch 430/875 | Loss: 0.0000 | Acc: 1.0000
Batch 440/875 | Loss: 0.0000 | Acc: 1.0000
Batch 450/875 | Loss: 0.0000 | Acc: 1.0000
Batch 460/875 | Loss: 0.0000 | Acc: 1.0000
Batch 470/875 | Loss: 0.0000 | Acc: 1.0000
Batch 480/875 | Loss: 0.0000 | Acc: 1.0000
Batch 490/875 | Loss: 0.0000 | Acc: 1.0000
Batch 500/875 | Loss: 0.0000 | Acc: 1.0000
Batch 510/875 | Loss: 0.0000 | Acc: 1.0000
Batch 520/875 | Loss: 0.0000 | Acc: 1.0000
Batch 530/875 | Loss: 0.0000 | Acc: 1.0000
Batch 540/875 | Loss: 0.0000 | Acc: 1.0000
Batch 550/875 | Loss: 0.0000 | Acc: 1.0000
Batch 560/875 | Loss: 0.0000 | Acc: 1.0000
Batch 570/875 | Loss: 0.0000 | Acc: 1.0000
```

```
  Batch 580/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 590/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 600/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 610/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 620/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 630/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 640/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 650/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 660/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 670/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 680/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 690/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 700/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 710/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 720/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 730/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 740/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 750/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 760/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 770/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 780/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 790/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 800/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 810/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 820/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 830/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 840/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 850/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 860/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 870/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 875/875 | Loss: 0.0000 | Acc: 1.0000
 Epoch 4 completed in 2210.7s | Train Loss: 0.0000 | Train Acc: 1.0000 | Val
Loss: 0.0000 | Val Acc: 1.0000

 Epoch 5/5 starts…
  Batch 10/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 20/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 30/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 40/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 50/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 60/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 70/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 80/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 90/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 100/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 110/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 120/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 130/875 | Loss: 0.0000 | Acc: 1.0000
```

```
Batch 140/875 | Loss: 0.0000 | Acc: 1.0000
Batch 150/875 | Loss: 0.0000 | Acc: 1.0000
Batch 160/875 | Loss: 0.0000 | Acc: 1.0000
Batch 170/875 | Loss: 0.0000 | Acc: 1.0000
Batch 180/875 | Loss: 0.0000 | Acc: 1.0000
Batch 190/875 | Loss: 0.0000 | Acc: 1.0000
Batch 200/875 | Loss: 0.0000 | Acc: 1.0000
Batch 210/875 | Loss: 0.0000 | Acc: 1.0000
Batch 220/875 | Loss: 0.0000 | Acc: 1.0000
Batch 230/875 | Loss: 0.0000 | Acc: 1.0000
Batch 240/875 | Loss: 0.0000 | Acc: 1.0000
Batch 250/875 | Loss: 0.0000 | Acc: 1.0000
Batch 260/875 | Loss: 0.0000 | Acc: 1.0000
Batch 270/875 | Loss: 0.0000 | Acc: 1.0000
Batch 280/875 | Loss: 0.0000 | Acc: 1.0000
Batch 290/875 | Loss: 0.0000 | Acc: 1.0000
Batch 300/875 | Loss: 0.0000 | Acc: 1.0000
Batch 310/875 | Loss: 0.0000 | Acc: 1.0000
Batch 320/875 | Loss: 0.0000 | Acc: 1.0000
Batch 330/875 | Loss: 0.0000 | Acc: 1.0000
Batch 340/875 | Loss: 0.0000 | Acc: 1.0000
Batch 350/875 | Loss: 0.0000 | Acc: 1.0000
Batch 360/875 | Loss: 0.0000 | Acc: 1.0000
Batch 370/875 | Loss: 0.0000 | Acc: 1.0000
Batch 380/875 | Loss: 0.0000 | Acc: 1.0000
Batch 390/875 | Loss: 0.0000 | Acc: 1.0000
Batch 400/875 | Loss: 0.0000 | Acc: 1.0000
Batch 410/875 | Loss: 0.0000 | Acc: 1.0000
Batch 420/875 | Loss: 0.0000 | Acc: 1.0000
Batch 430/875 | Loss: 0.0000 | Acc: 1.0000
Batch 440/875 | Loss: 0.0000 | Acc: 1.0000
Batch 450/875 | Loss: 0.0000 | Acc: 1.0000
Batch 460/875 | Loss: 0.0000 | Acc: 1.0000
Batch 470/875 | Loss: 0.0000 | Acc: 1.0000
Batch 480/875 | Loss: 0.0000 | Acc: 1.0000
Batch 490/875 | Loss: 0.0000 | Acc: 1.0000
Batch 500/875 | Loss: 0.0000 | Acc: 1.0000
Batch 510/875 | Loss: 0.0000 | Acc: 1.0000
Batch 520/875 | Loss: 0.0000 | Acc: 1.0000
Batch 530/875 | Loss: 0.0000 | Acc: 1.0000
Batch 540/875 | Loss: 0.0000 | Acc: 1.0000
Batch 550/875 | Loss: 0.0000 | Acc: 1.0000
Batch 560/875 | Loss: 0.0000 | Acc: 1.0000
Batch 570/875 | Loss: 0.0000 | Acc: 1.0000
Batch 580/875 | Loss: 0.0000 | Acc: 1.0000
Batch 590/875 | Loss: 0.0000 | Acc: 1.0000
Batch 600/875 | Loss: 0.0000 | Acc: 1.0000
Batch 610/875 | Loss: 0.0000 | Acc: 1.0000
```

```
  Batch 620/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 630/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 640/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 650/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 660/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 670/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 680/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 690/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 700/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 710/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 720/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 730/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 740/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 750/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 760/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 770/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 780/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 790/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 800/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 810/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 820/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 830/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 840/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 850/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 860/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 870/875 | Loss: 0.0000 | Acc: 1.0000
  Batch 875/875 | Loss: 0.0000 | Acc: 1.0000
 Epoch 5 completed in 2348.5s | Train Loss: 0.0000 | Train Acc: 1.0000 | Val
Loss: 0.0000 | Val Acc: 1.0000
```

[28]:
```python
# Load best model for final evaluation
print("\n--- Final Evaluation ---")
model.load_state_dict(torch.load("best_vit_model_jupyternotebook.pth"))
model.eval()
```

```
--- Final Evaluation ---
```

[28]:
```
ViTForImageClassification(
  (vit): ViTModel(
    (embeddings): ViTEmbeddings(
      (patch_embeddings): ViTPatchEmbeddings(
        (projection): Conv2d(1, 768, kernel_size=(16, 16), stride=(16, 16))
      )
      (dropout): Dropout(p=0.0, inplace=False)
    )
    (encoder): ViTEncoder(
      (layer): ModuleList(
```

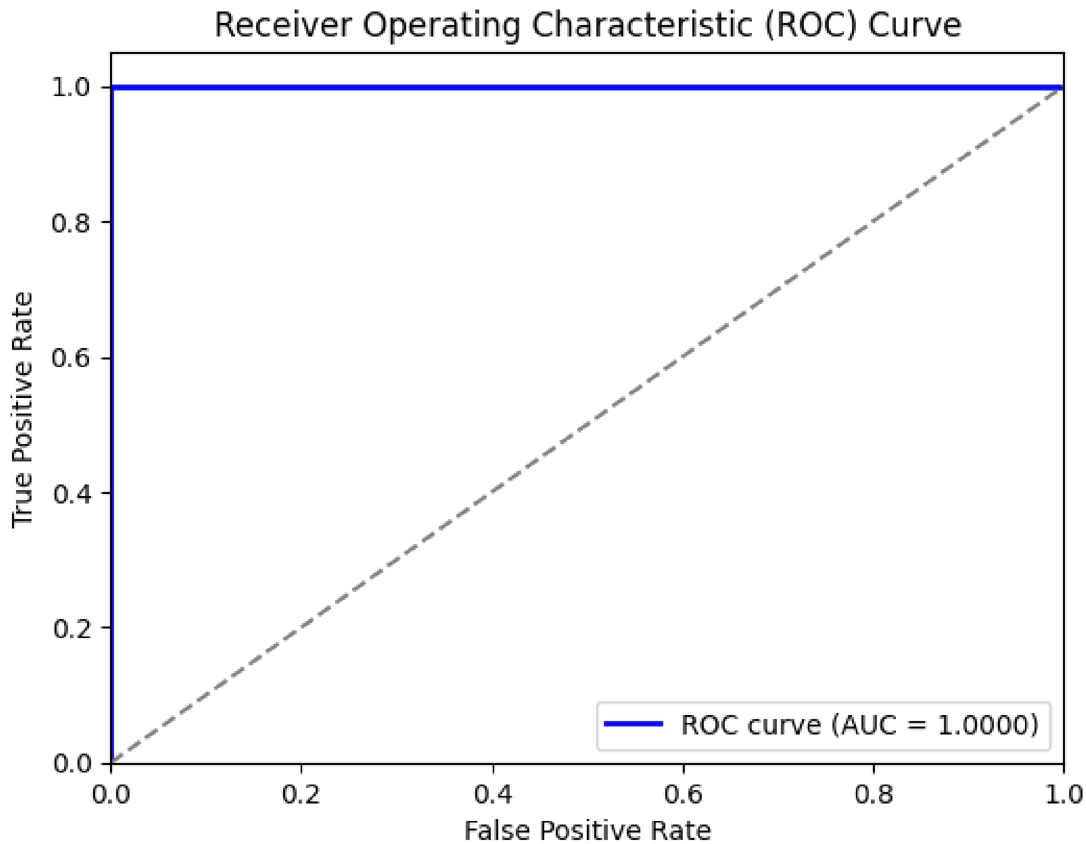
```
        (0-11): 12 x ViTLayer(
          (attention): ViTSdpaAttention(
            (attention): ViTSdpaSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.0, inplace=False)
            )
            (output): ViTSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.0, inplace=False)
            )
          )
          (intermediate): ViTIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): ViTOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (dropout): Dropout(p=0.0, inplace=False)
          )
          (layernorm_before): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
          (layernorm_after): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
        )
      )
    )
    (layernorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
  )
  (classifier): Linear(in_features=768, out_features=1000, bias=True)
)
```

[29]:

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Initialize test loss and tracking variables
test_loss = 0.0
correct = 0
total = 0
all_labels = []
all_probs = []

with torch.no_grad():
    for images, labels in test_loader:
        images, labels = images.to(device), labels.to(device)
```

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

        # Accumulate test loss
        test_loss += loss.item()

        # Get predictions
        probs = torch.softmax(outputs, dim=1)[:, 1]  # Probability of class 1
        _, predicted = torch.max(outputs.data, 1)

        # Store values for evaluation
        all_labels.extend(labels.cpu().numpy())
        all_probs.extend(probs.cpu().numpy())

        # Compute accuracy
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

# Compute final loss and accuracy
test_loss = test_loss / len(test_loader)
test_acc = correct / total

# Compute ROC curve and AUC
fpr, tpr, _ = roc_curve(all_labels, all_probs)
roc_auc = auc(fpr, tpr)

# Print final test results
print(f"\n Final Test Results | Loss: {test_loss:.4f} | Accuracy: {test_acc:.
 ↪4f} | AUC: {roc_auc:.4f}")

# Plot ROC Curve
plt.figure()
plt.plot(fpr, tpr, color="blue", lw=2, label=f"ROC curve (AUC = {roc_auc:.4f})")
plt.plot([0, 1], [0, 1], color="gray", linestyle="--")  # Diagonal reference␣
 ↪line
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver Operating Characteristic (ROC) Curve")
plt.legend(loc="lower right")
plt.show()
```

```
Final Test Results | Loss: 0.0000 | Accuracy: 1.0000 | AUC: 1.0000
```


Receiver Operating Characteristic (ROC) Curve
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
ROC curve (AUC = 1.0000)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate