Assignment 3

Akshaya

June 13, 2025

Overview

Working on this assignment showed me how computers can quickly understand and compare images by extracting important features using pre-trained models. I learned that turning images into feature vectors—lists of numbers—makes searching for similar images much faster and easier. Using tools like Annoy for similarity search helps handle large datasets efficiently. Overall, I realized how useful these techniques are for organizing and retrieving images, and how they can be applied in real-world applications.

Question 1

Task 1

```
# 2. Train-test split (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# 3. Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# 4. One-hot encode target labels using to_categorical()
y_train_encoded = to_categorical(y_train, num_classes=3)
y_test_encoded = to_categorical(y_test, num_classes=3)

# 5. Convert to PyTorch tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train_encoded, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test_encoded, dtype=torch.float32)

# Check tensor shapes
print("X_train:", X_train_tensor.shape)
print("y_train:", y_train_tensor.shape)
```

```
X_train: torch.Size([120, 4])
y_train: torch.Size([120, 3])
```

Figure 1:

Task 2

```
        # Input layer to hidden layer (4 -> 8)
        self.fc1 = nn.Linear(4, 8)

        # Hidden layer to output layer (8 -> 3)
        self.fc2 = nn.Linear(8, 3)

    def forward(self, x):
        # Hidden layer with ReLU activation
        x = F.relu(self.fc1(x))

        # Output layer with Softmax activation
        x = F.softmax(self.fc2(x), dim=1)  # softmax along the class dimension
        return x

# Instantiate the model
model = IrisNet()

# Print model architecture
print(model)
```

```
IrisNet(
  (fc1): Linear(in_features=4, out_features=8, bias=True)
  (fc2): Linear(in_features=8, out_features=3, bias=True)
)
```

Figure 2:

Task 3

```
        outputs = model(batch_x)

        # Compute loss
        loss = loss_function(outputs, batch_y)

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

        total_loss += loss.item()

    # Print loss every 10 epochs
    if (epoch+1) % 10 == 0:
        print(f"Epoch [{epoch+1}/{epochs}], Loss: {total_loss:.4f}")
```

```
Epoch [10/100], Loss: 1.5419
Epoch [20/100], Loss: 1.1415
Epoch [30/100], Loss: 0.8097
Epoch [40/100], Loss: 0.7421
Epoch [50/100], Loss: 0.7157
Epoch [60/100], Loss: 0.6253
Epoch [70/100], Loss: 0.5547
Epoch [80/100], Loss: 0.5081
Epoch [90/100], Loss: 0.5449
Epoch [100/100], Loss: 0.5474
```

Figure 3:

Task 4

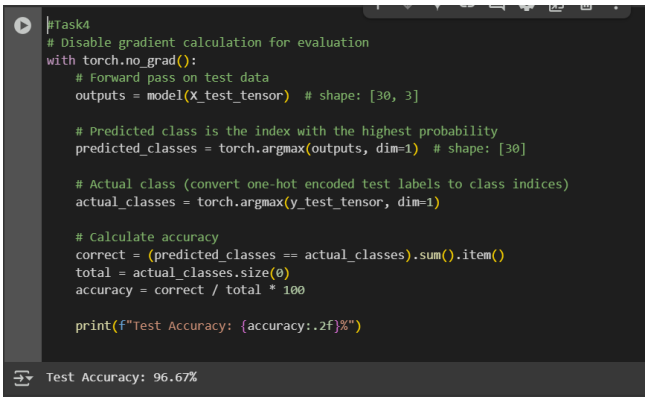

Figure 4:

Question 2

Working on this assignment showed me how computers can quickly understand and compare images by extracting important features using pre-trained models. I learned that turning images into feature vectors—lists of numbers—makes searching for similar images much faster and easier. Using tools like Annoy for similarity search helps handle large datasets efficiently. Overall, I realized how useful these techniques are for organizing and retrieving images, and how they can be applied in real-world applications.

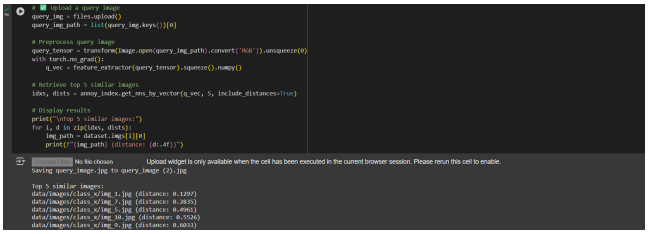

Figure 5:

Links for Questions

- `https://github.com/Guthikonda-Akshaya/GanForge/blob/assignment-3/a3q1.ipynb`
- `https://github.com/Guthikonda-Akshaya/GanForge/blob/assignment-3/a3q2.ipynb`