# MY MODEL

## 1: Data Loading and Preprocessing

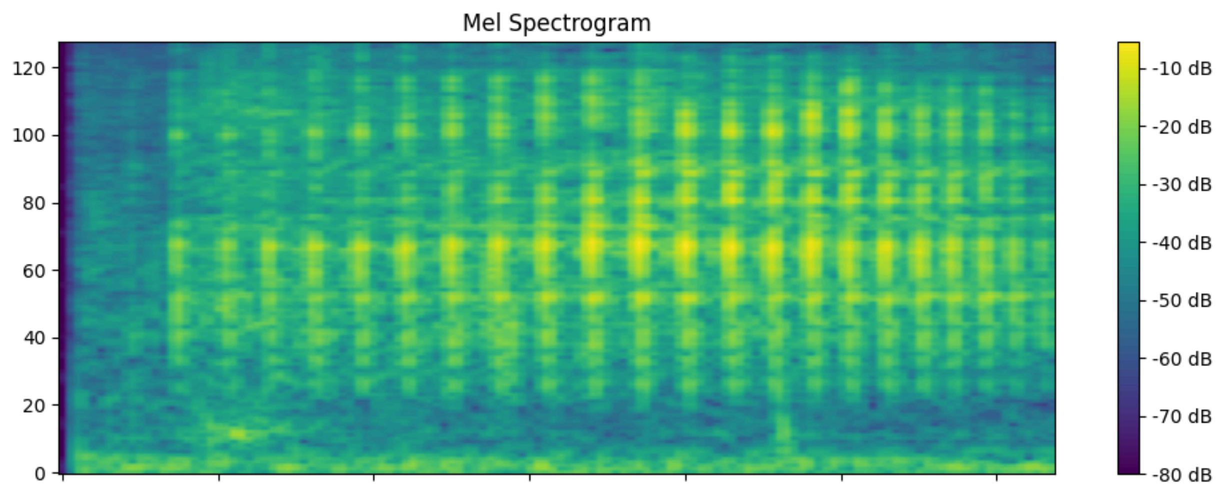
```
import numpy as np
import pandas as pd
import librosa
import os
import pickle
from sklearn.preprocessing import LabelEncoder
from sklearn.utils import resample

# Load metadata
metadata_path = '/content/drive/MyDrive/dataset for bird/264 birds/birdsong_metadata.csv'
df = pd.read_csv(metadata_path)

# Path to audio files
audio_files_path = '/content/drive/MyDrive/dataset for bird/264 birds/songs'

# Function to extract Mel spectrogram from FLAC audio file
def extract_features(file_path, max_pad_len=128):
    try:
        audio, sr = librosa.load(file_path, sr=None)
        mels = librosa.feature.melspectrogram(y=audio, sr=sr, n_mels=128, fmax=8000)
        mels = librosa.power_to_db(mels, ref=np.max)

        # Pad or truncate Mel spectrogram
        if mels.shape[1] < max_pad_len:
            pad_width = max_pad_len - mels.shape[1]
            mels = np.pad(mels, pad_width=((0, 0), (0, pad_width)), mode='constant')
        else:
            mels = mels[:, :max_pad_len]

        return mels
    except Exception as e:
        print(f"Error processing {file_path}: {str(e)}")
        return None

# Analyze class distribution
class_counts = df['species'].value_counts()
print("Original class distribution:")
print(class_counts)

# Determine maximum number of samples in any class (majority class)
max_samples = class_counts.max()

# Oversample to balance the dataset
balanced_features = []
balanced_labels = []

for species in class_counts.index:
    species_files = df[df['species'] == species]['file_id']

    # Extract features for each file in the species
    species_features = []
    for file_id in species_files:
        file_path = os.path.join(audio_files_path, f"xc{file_id}.flac")
        if os.path.exists(file_path):
            feature = extract_features(file_path)
            if feature is not None:
                species_features.append(feature)
        else:
            print(f"File not found: {file_path}")

    # If species has fewer samples than the max_samples, oversample by duplicating
    if len(species_features) < max_samples:
        species_features = resample(species_features, replace=True, n_samples=max_samples, random_state=42)

    balanced_features.extend(species_features)
    balanced_labels.extend([species] * max_samples)
```

```
# Convert balanced features and labels to numpy arrays
balanced_features = np.array(balanced_features, dtype=np.float32)
balanced_labels = np.array(balanced_labels)

# Label Encoding for balanced labels
label_encoder = LabelEncoder()
encoded_labels = label_encoder.fit_transform(balanced_labels)

# Save the label encoder for future use
with open('label_encoder.pkl', 'wb') as f:
    pickle.dump(label_encoder, f)

# Print final balanced data shapes
print(f"Balanced features shape: {balanced_features.shape}")
print(f"Balanced labels shape: {encoded_labels.shape}")
```

```
Original class distribution:
species
major         6
palustris     6
montanus      6
trochilus     3
sibilatrix    3
             ..
caeruleus     3
canorus       3
frugilegus    3
corone        3
coelebs       3
Name: count, Length: 85, dtype: int64
Balanced features shape: (510, 128, 128)
Balanced labels shape: (510,)
```

## Visualization:

```
import matplotlib.pyplot as plt

# Visualize a random Mel spectrogram
plt.figure(figsize=(10, 4))
plt.imshow(balanced_features[0], aspect='auto', origin='lower')
plt.title('Mel Spectrogram')
plt.colorbar(format='%+2.0f dB')
plt.tight_layout()
plt.show()
```




## 2: Model Building

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout


# Define the CNN model
model = Sequential([
Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 1)), # Assuming (128, 128) Mel spectrogram
MaxPooling2D(2, 2),
Conv2D(64, (3, 3), activation='relu'),
MaxPooling2D(2, 2),
Flatten(),
Dense(128, activation='relu'),
Dropout(0.5),
Dense(len(label_encoder.classes_), activation='softmax') # Number of output neurons = number of species
])
# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
# Model summary
model.summary()
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`inpu
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential_8"
```

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_8 (Conv2D) | (None, 126, 126, 32) | 320 |
| max_pooling2d_8 (MaxPooling2D) | (None, 63, 63, 32) | 0 |
| conv2d_9 (Conv2D) | (None, 61, 61, 64) | 18,496 |
| max_pooling2d_9 (MaxPooling2D) | (None, 30, 30, 64) | 0 |
| flatten_4 (Flatten) | (None, 57600) | 0 |
| dense_16 (Dense) | (None, 128) | 7,372,928 |
| dropout_4 (Dropout) | (None, 128) | 0 |
| dense_17 (Dense) | (None, 85) | 10,965 |

```
Total params: 7,402,709 (28.24 MB)
Trainable params: 7,402,709 (28.24 MB)
```

## 3: Model Training and Validation

```
# Train the model
history = model.fit(balanced_features, encoded_labels, epochs=50, batch_size=32, validation_split=0.2)
# Save the model
model.save('bird_species_model.keras') # Save model for later use
```

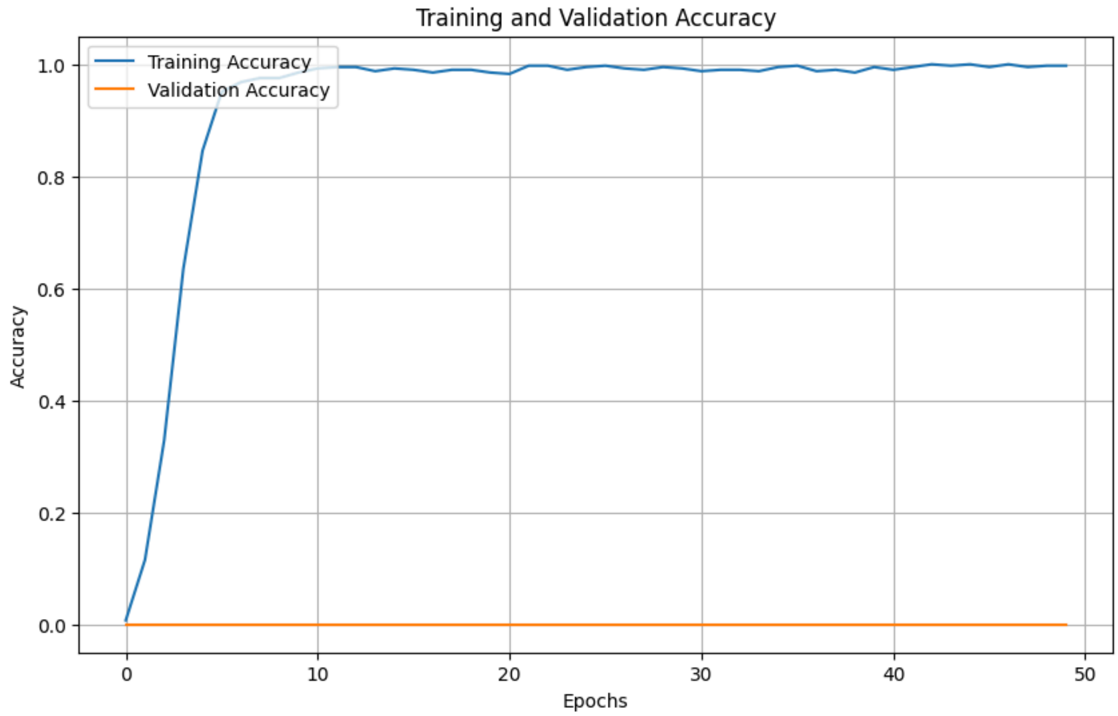
```
Epoch 1/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 8s 444ms/step - accuracy: 0.0041 - loss: 56.1417 - val_accuracy: 0.0000e+00 - val_loss: 4.4429
Epoch 2/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 4s 26ms/step - accuracy: 0.0830 - loss: 4.2989 - val_accuracy: 0.0000e+00 - val_loss: 4.6454
Epoch 3/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 19ms/step - accuracy: 0.3040 - loss: 3.2685 - val_accuracy: 0.0000e+00 - val_loss: 4.9327
Epoch 4/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.5888 - loss: 2.0589 - val_accuracy: 0.0000e+00 - val_loss: 8.4073
Epoch 5/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - accuracy: 0.8282 - loss: 0.9021 - val_accuracy: 0.0000e+00 - val_loss: 10.1100
Epoch 6/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.9549 - loss: 0.2960 - val_accuracy: 0.0000e+00 - val_loss: 12.1584
Epoch 7/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - accuracy: 0.9731 - loss: 0.1765 - val_accuracy: 0.0000e+00 - val_loss: 11.8808
Epoch 8/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.9821 - loss: 0.0994 - val_accuracy: 0.0000e+00 - val_loss: 10.5967
Epoch 9/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9868 - loss: 0.1235 - val_accuracy: 0.0000e+00 - val_loss: 10.4527
Epoch 10/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9805 - loss: 0.0943 - val_accuracy: 0.0000e+00 - val_loss: 11.0569
Epoch 11/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9950 - loss: 0.0514 - val_accuracy: 0.0000e+00 - val_loss: 10.8663
```

```
Epoch 12/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.9910 - loss: 0.0494 - val_accuracy: 0.0000e+00 - val_loss: 13.4339
Epoch 13/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.9910 - loss: 0.0691 - val_accuracy: 0.0000e+00 - val_loss: 18.4669
Epoch 14/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9840 - loss: 0.0741 - val_accuracy: 0.0000e+00 - val_loss: 13.4831
Epoch 15/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.9934 - loss: 0.0524 - val_accuracy: 0.0000e+00 - val_loss: 12.9841
Epoch 16/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9898 - loss: 0.0676 - val_accuracy: 0.0000e+00 - val_loss: 12.3813
Epoch 17/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9889 - loss: 0.0571 - val_accuracy: 0.0000e+00 - val_loss: 13.0866
Epoch 18/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9882 - loss: 0.0417 - val_accuracy: 0.0000e+00 - val_loss: 12.4625
Epoch 19/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.9879 - loss: 0.0910 - val_accuracy: 0.0000e+00 - val_loss: 13.2601
Epoch 20/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9904 - loss: 0.0783 - val_accuracy: 0.0000e+00 - val_loss: 11.1171
Epoch 21/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9820 - loss: 0.0973 - val_accuracy: 0.0000e+00 - val_loss: 11.4379
Epoch 22/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9978 - loss: 0.0356 - val_accuracy: 0.0000e+00 - val_loss: 13.1843
Epoch 23/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.9974 - loss: 0.0417 - val_accuracy: 0.0000e+00 - val_loss: 12.5246
Epoch 24/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.9877 - loss: 0.0316 - val_accuracy: 0.0000e+00 - val_loss: 10.7373
Epoch 25/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.9975 - loss: 0.0261 - val_accuracy: 0.0000e+00 - val_loss: 12.6505
Epoch 26/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9995 - loss: 0.0121 - val_accuracy: 0.0000e+00 - val_loss: 13.2682
Epoch 27/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9955 - loss: 0.0199 - val_accuracy: 0.0000e+00 - val_loss: 11.6348
Epoch 28/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9881 - loss: 0.0358 - val_accuracy: 0.0000e+00 - val_loss: 12.5774
Epoch 29/50
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.9934 - loss: 0.0328 - val_accuracy: 0.0000e+00 - val_loss: 13.5234
```

## Visualization:

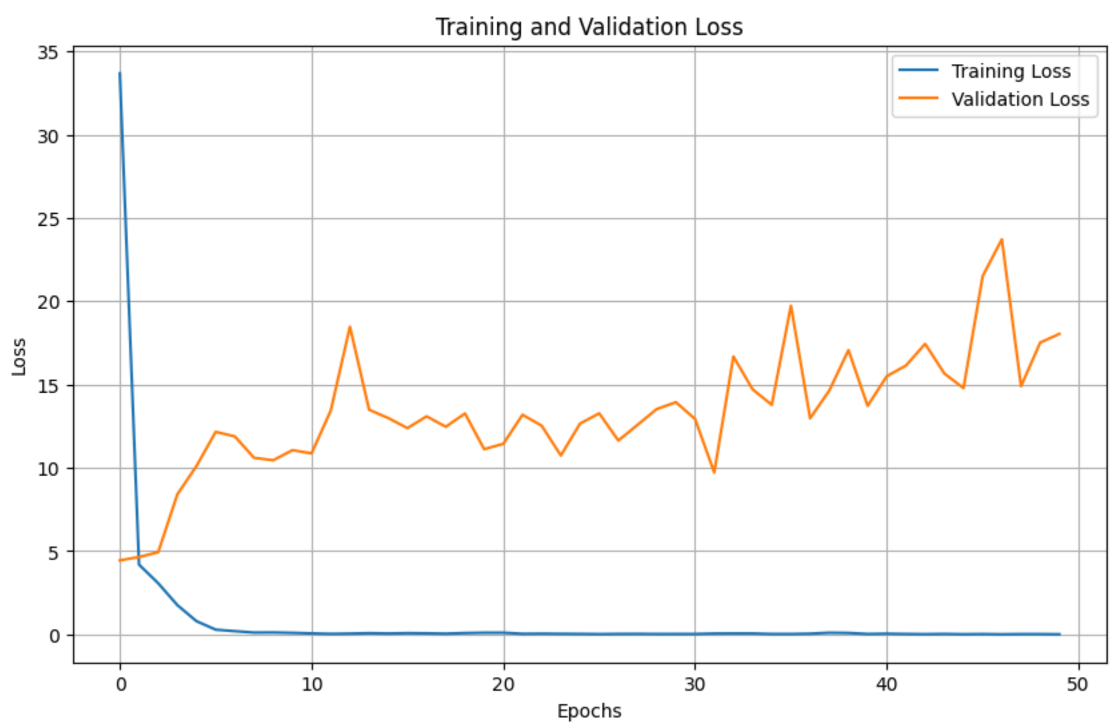
### Training and Validation Accuracy Plot

```
import matplotlib.pyplot as plt
# Plot training & validation accuracy over epochs
plt.figure(figsize=(10, 6))
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(loc='upper left')
plt.grid(True)
plt.show()
```




## Training and Validation Loss Plot

```
# Plot training & validation loss over epochs
plt.figure(figsize=(10, 6))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(loc='upper right')
plt.grid(True)
plt.show()
```



## 4: Model Evaluation

```
# Evaluate the model on training data
train_loss, train_acc = model.evaluate(balanced_features , encoded_labels)
print(f"Training accuracy: {train_acc * 100:.2f}%")
```

```
16/16 ━━━━━━━━━━━━━━━━━━━━ 1s 47ms/step - accuracy: 0.9613 - loss: 0.6895
Training accuracy: 80.00%
```

## 5: Bird Species Prediction

```
# Path to the new audio file
new_audio_path = '/content/drive/MyDrive/dataset for bird/264 birds/songs/xc101935.flac'

# Extract features from the new audio file
new_feature = extract_features(new_audio_path)
new_feature = np.expand_dims(new_feature, axis=0)  # Add batch dimension
new_feature = np.expand_dims(new_feature, axis=-1)  # Add channel dimension (for grayscale)

# Predict using the trained model
prediction = model.predict(new_feature)

# Get the predicted label (the index of the highest probability)
predicted_label = np.argmax(prediction, axis=1)

# Decode the predicted label back to the species name
predicted_species = label_encoder.inverse_transform(predicted_label)
print(f"Predicted Bird Species: {predicted_species[0]}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 236ms/step
Predicted Bird Species: sibilatrix
```

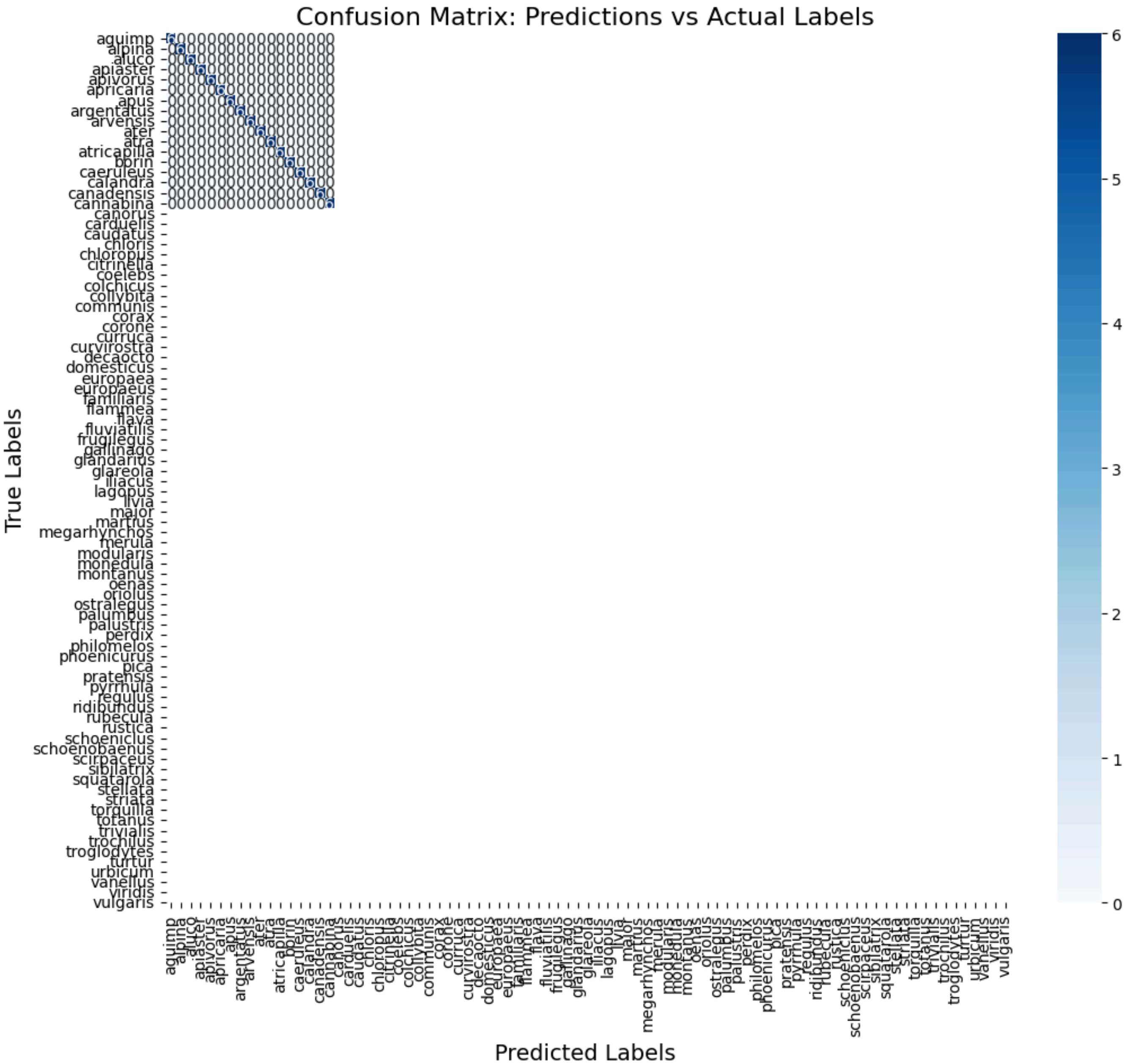
## Visualization:

### Plotting the Confusion Matrix with Axes Subplot

```
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

# Ensure validation features have the right shape: (batch_size, 128, 128, 1)
validation_features = balanced_features[:int(0.2 * len(balanced_features))]  # Assuming 20% validation split
validation_labels = encoded_labels[:int(0.2 * len(encoded_labels))]

# Expand dimensions to match the model's input shape (batch_size, 128, 128, 1)
validation_features = np.expand_dims(validation_features, axis=-1)  # Add the channel dimension

# Generate predictions
predictions = model.predict(validation_features)
predicted_labels = np.argmax(predictions, axis=1)

# Create confusion matrix
cm = confusion_matrix(validation_labels, predicted_labels)

# Set up the plot grid and axes subplot
fig, ax = plt.subplots(figsize=(12, 10))

# Plot the confusion matrix using seaborn heatmap
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', ax=ax,
            xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)

# Set titles and labels for better understanding
ax.set_title('Confusion Matrix: Predictions vs Actual Labels', fontsize=16)
ax.set_xlabel('Predicted Labels', fontsize=14)
```

```
ax.set_ylabel('True Labels', fontsize=14)

# Display the plot
plt.show()
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
```

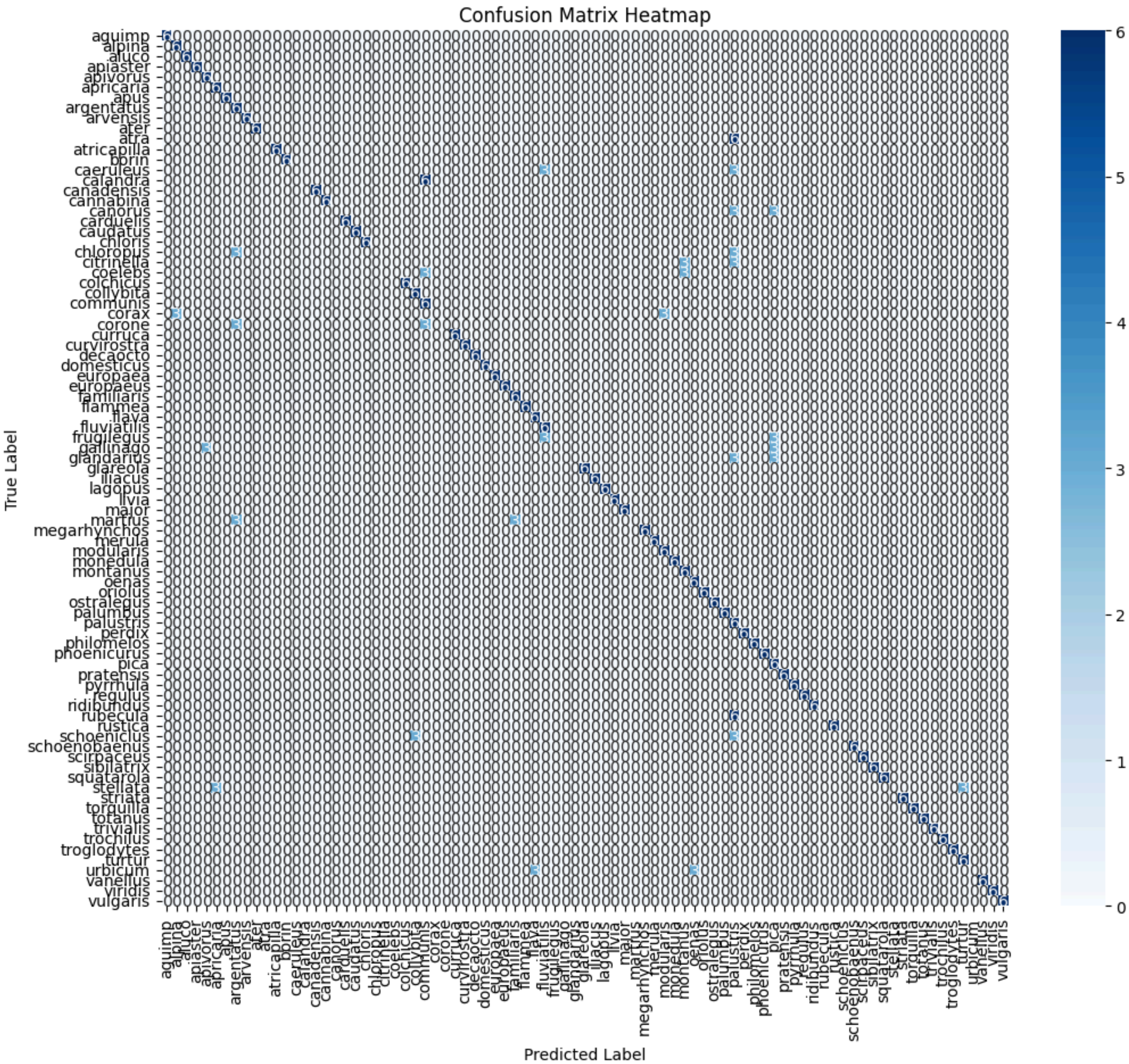



## Confusion Matrix with Heatmap

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import numpy as np

# Ensure the input features have the correct shape (batch_size, 128, 128, 1)
features_with_channel = np.expand_dims(balanced_features, axis=-1)  # Add channel dimension for grayscale

# Generate predictions
y_pred = model.predict(features_with_channel)  # Make predictions on the features with the correct shape
y_pred_classes = np.argmax(y_pred, axis=1)  # Get the predicted class labels

# Compute confusion matrix
conf_matrix = confusion_matrix(encoded_labels, y_pred_classes)
```

```
# Plot heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.title('Confusion Matrix Heatmap')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')

# Show the plot
plt.show()
```

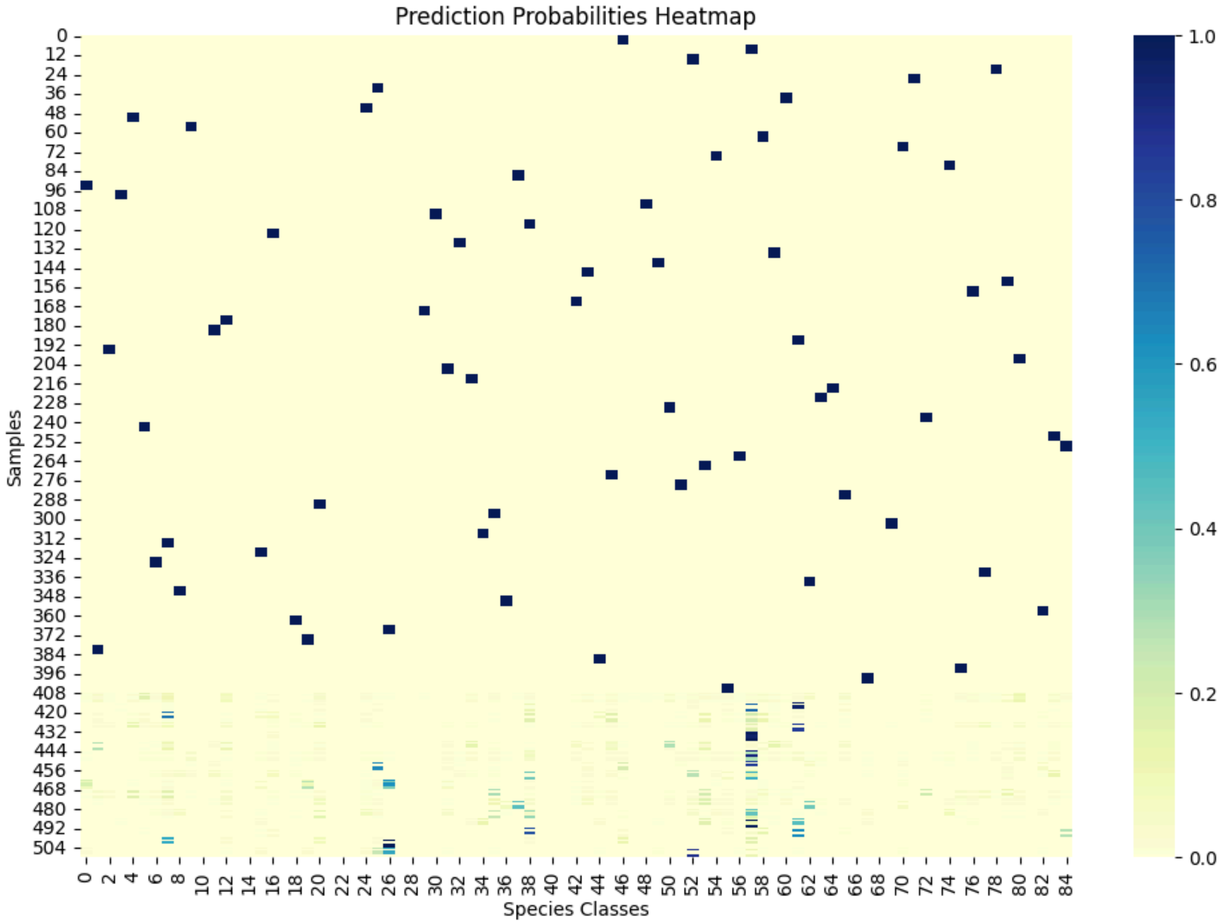
```
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step
```




## Heatmap of the Model's Prediction Probabilities

```
# Assuming y_pred is the prediction probabilities (from model.predict())
plt.figure(figsize=(12, 8))
sns.heatmap(y_pred, cmap="YlGnBu", annot=False)
plt.title('Prediction Probabilities Heatmap')
plt.xlabel('Species Classes')
plt.ylabel('Samples')

# Show the plot
```

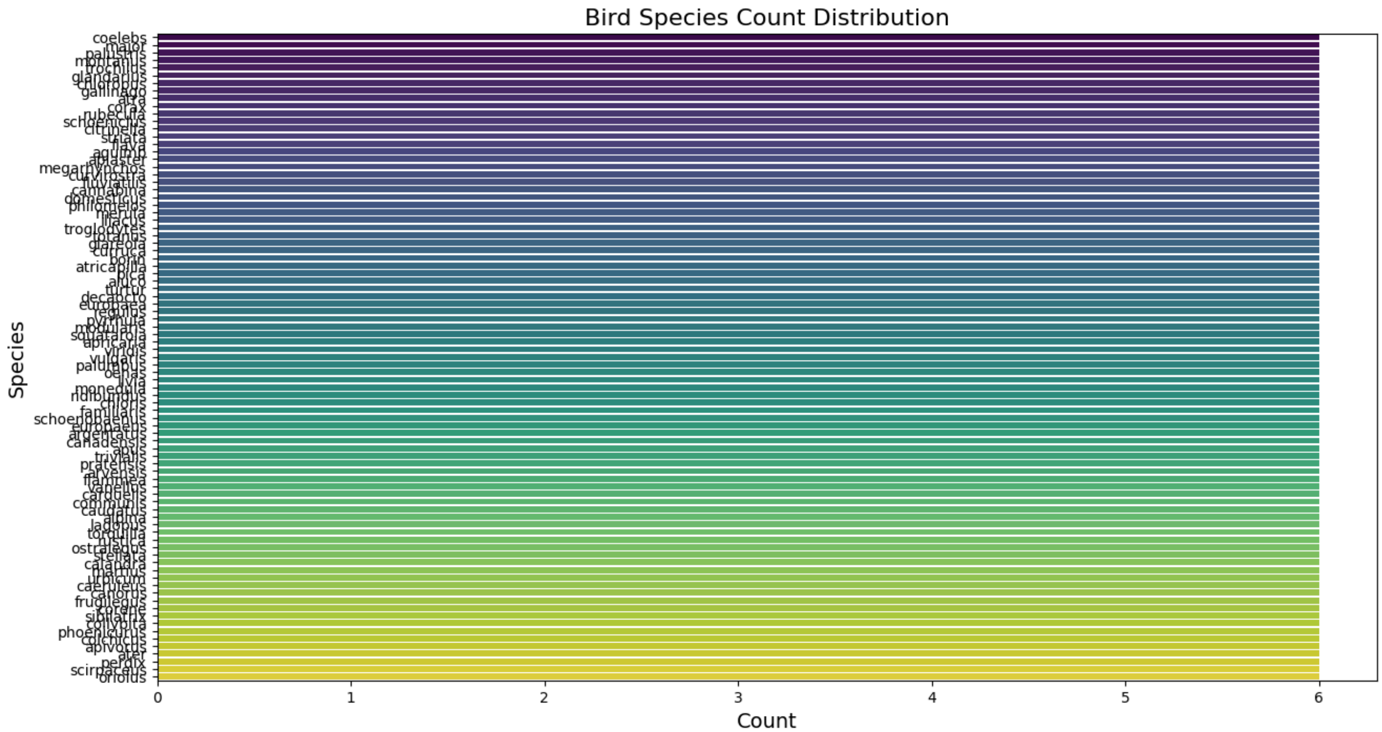
```
plt.show()
```




## Countplot for Bird Species Distribution

```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming you have the labels and they are already encoded
species_labels = label_encoder.inverse_transform(encoded_labels)  # Decode the labels back to species names

# Create a countplot
plt.figure(figsize=(15, 8))
sns.countplot(y=species_labels, order=pd.Series(species_labels).value_counts().index, palette="viridis")

# Set titles and labels for the plot
plt.title('Bird Species Count Distribution', fontsize=16)
plt.xlabel('Count', fontsize=14)
plt.ylabel('Species', fontsize=14)

# Show the plot
plt.show()
```

```
<ipython-input-67-000f6f6048e9>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend

  sns.countplot(y=species_labels, order=pd.Series(species_labels).value_counts().index, palette="viridis")
```

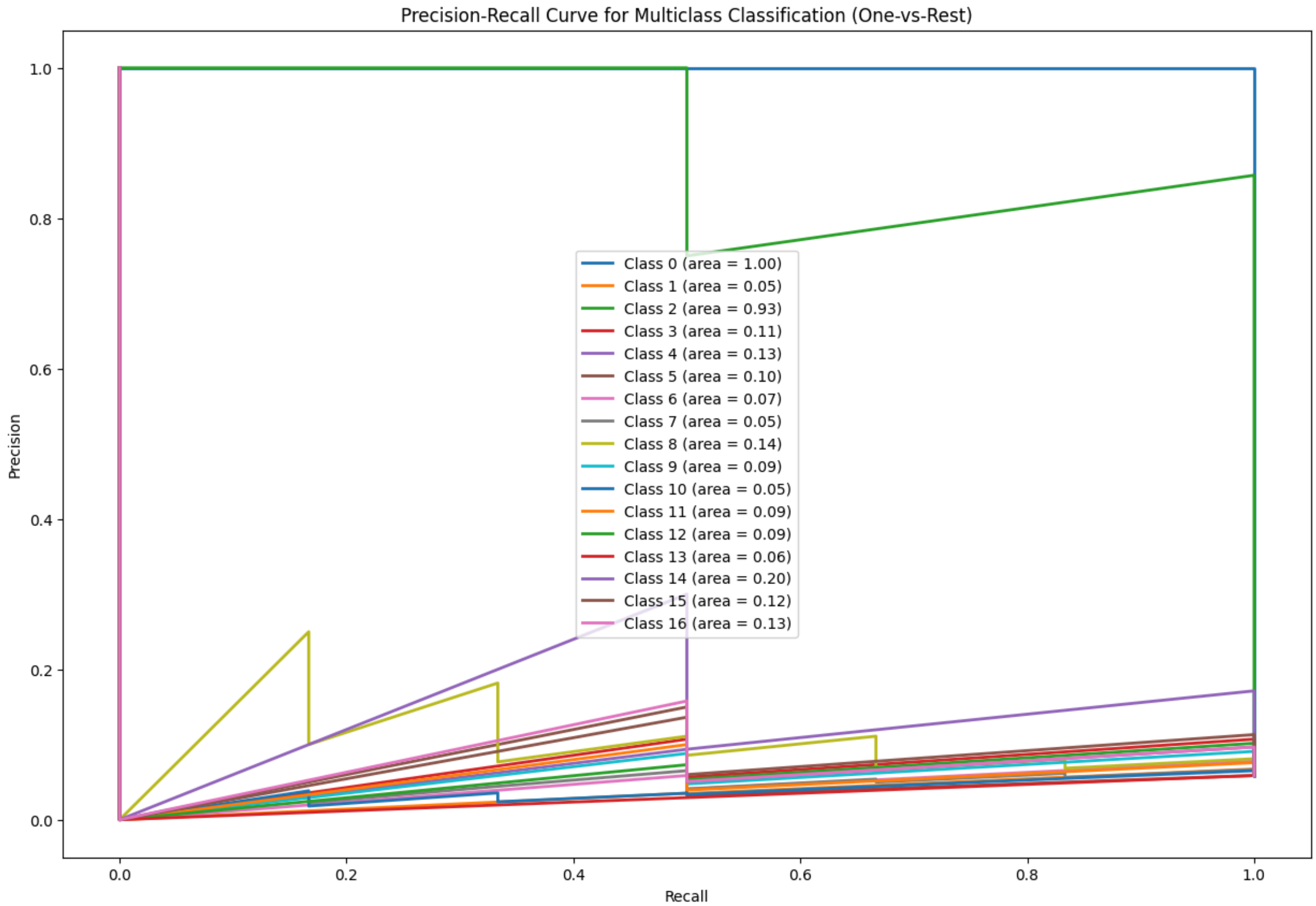



## Precision-Recall Curve

```
from sklearn.metrics import precision_recall_curve, average_precision_score
from sklearn.preprocessing import label_binarize
import matplotlib.pyplot as plt
import numpy as np

# Ensure labels are one-hot encoded if they are not
if len(validation_labels.shape) == 1:
    val_labels_binarized = label_binarize(validation_labels, classes=np.unique(validation_labels))
else:
    val_labels_binarized = validation_labels  # Already one-hot encoded

# Get predicted probabilities for the validation set
y_scores = model.predict(validation_features)

# Define the number of classes based on training data or unique labels in encoded
n_classes = val_labels_binarized.shape[1]  # This should work now

# Calculate Precision-Recall for each class
precision = dict()
recall = dict()
average_precision = dict()

for i in range(n_classes):
    if np.sum(val_labels_binarized[:, i]) == 0:
        print(f"Warning: No positive samples in class {i}, skipping Precision-Recall computation.")
        continue
    precision[i], recall[i], _ = precision_recall_curve(val_labels_binarized[:, i], y_scores[:, i])
    average_precision[i] = average_precision_score(val_labels_binarized[:, i], y_scores[:, i])

# Plot Precision-Recall curve for each class
```

```
plt.figure(figsize=(15, 10))
for i in range(n_classes):
    if i in precision:
        plt.plot(recall[i], precision[i], lw=2, label=f'Class {i} (area = {average_precision[i]:.2f})')

plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve for Multiclass Classification (One-vs-Rest)")
plt.legend(loc="best")
plt.show()
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
```

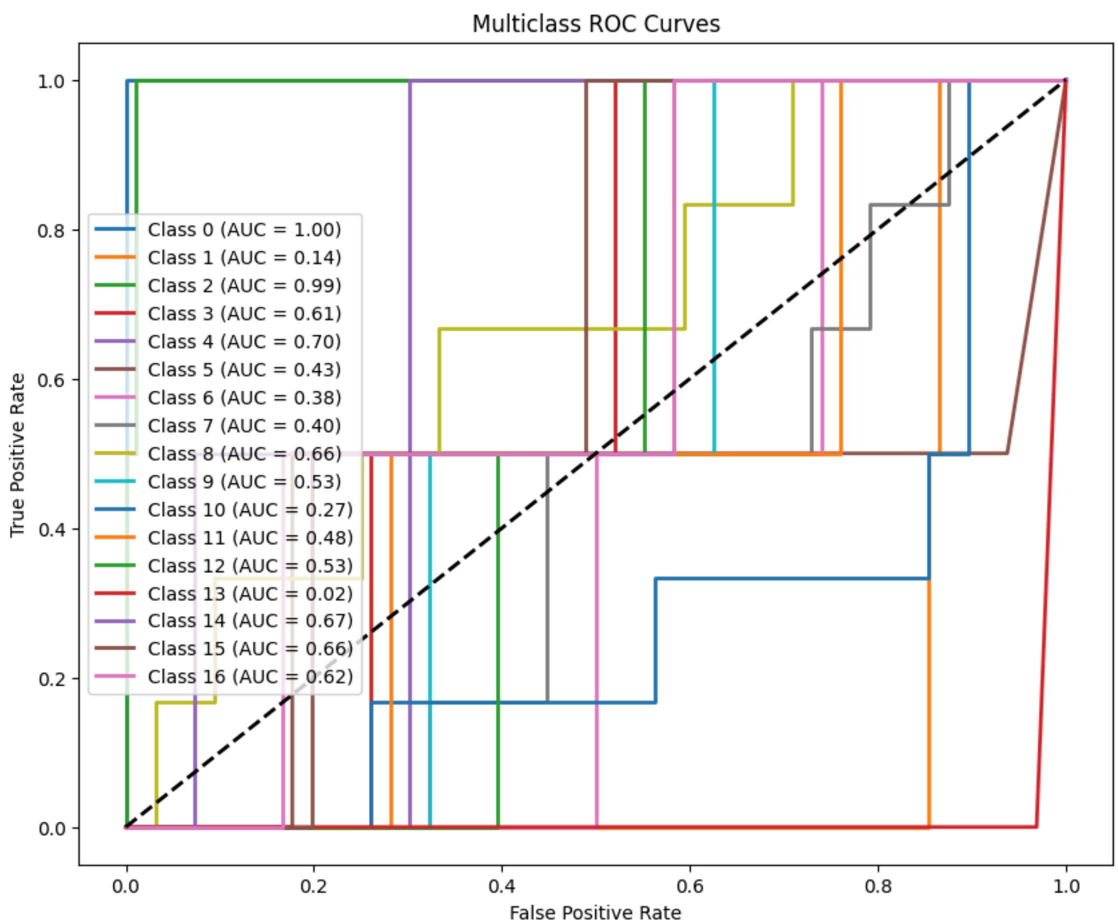



## Receiver Operating Characteristic (ROC) Curve

```
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize
import matplotlib.pyplot as plt
import numpy as np

# Ensure labels are one-hot encoded if they are not
if len(validation_labels.shape) == 1:
    val_labels_binarized = label_binarize(validation_labels, classes=np.unique(validation_labels))
else:
    val_labels_binarized = validation_labels  # Already one-hot encoded

# Get predicted probabilities for the validation set
y_scores = model.predict(validation_features)
```

```
# Define the number of classes based on the validation labels or training data
n_classes = val_labels_binarized.shape[1]  # This should work now

# Calculate ROC curve and AUC for each class
fpr = dict()
tpr = dict()
roc_auc = dict()

for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(val_labels_binarized[:, i], y_scores[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Plot ROC curve for each class
plt.figure(figsize=(10, 8))
for i in range(n_classes):
    plt.plot(fpr[i], tpr[i], lw=2, label=f'Class {i} (AUC = {roc_auc[i]:.2f})')

# Plot the diagonal line (random guessing)
plt.plot([0, 1], [0, 1], 'k--', lw=2)

plt.title('Multiclass ROC Curves')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc="best")
plt.show()
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
```

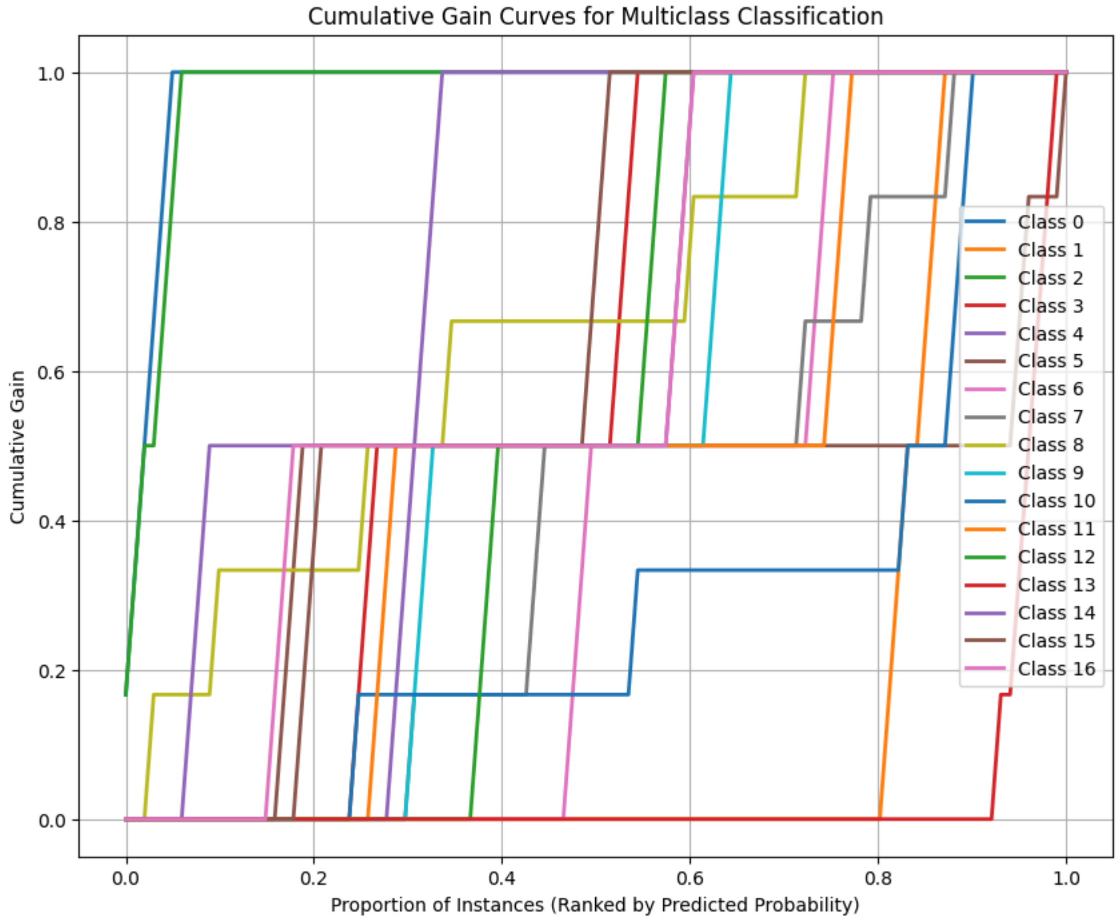



## Cumulative Gain Curve (Multiclass Classification)

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import label_binarize
from sklearn.metrics import precision_recall_curve

# Ensure the labels are one-hot encoded
if len(validation_labels.shape) == 1:
```

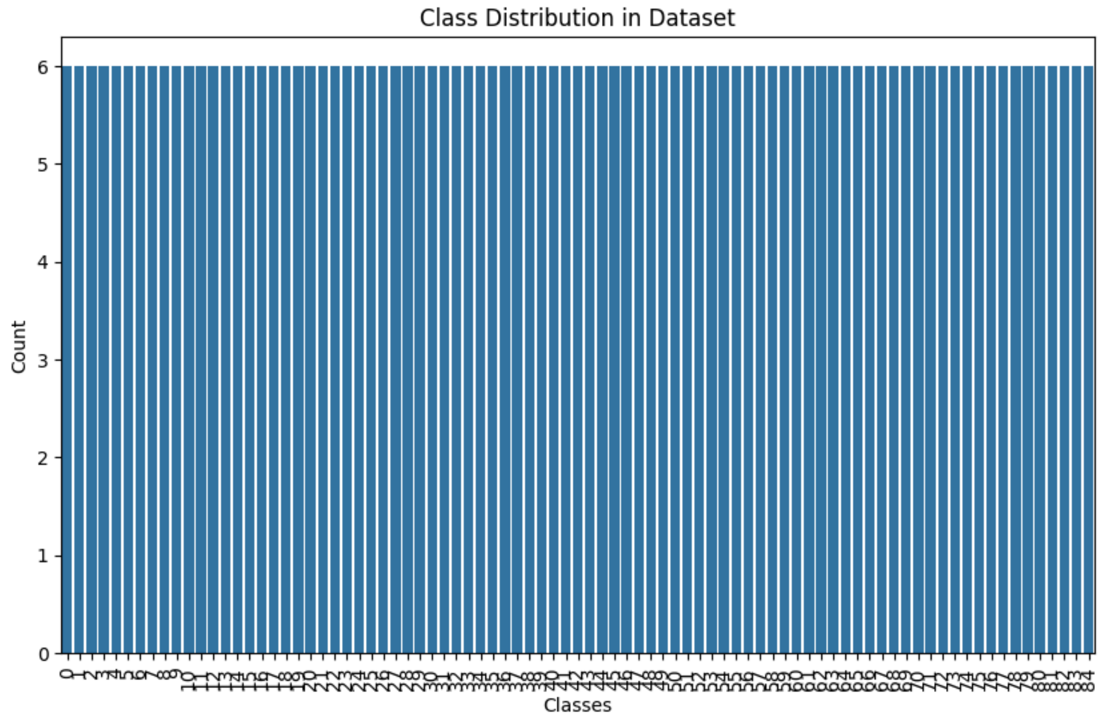
```
        val_labels_binarized = label_binarize(validation_labels, classes=np.unique(validation_labels))
    else:
        val_labels_binarized = validation_labels  # Already one-hot encoded

    # Get predicted probabilities for the validation set
    y_scores = model.predict(validation_features)

    # Define the number of classes based on the validation labels
    n_classes = val_labels_binarized.shape[1]

    # Calculate the Cumulative Gain curve for each class
    cumulative_gain = dict()

    for i in range(n_classes):
        # Get predicted probabilities and true labels for the class
        probs = y_scores[:, i]
        true_labels = val_labels_binarized[:, i]

        # Sort by predicted probabilities in descending order
        sorted_indices = np.argsort(probs)[::-1]
        sorted_true_labels = true_labels[sorted_indices]

        # Calculate cumulative gain (i.e., cumulative sum of true positives)
        cumulative_gain[i] = np.cumsum(sorted_true_labels).astype(float)  # Ensure it's a float array

        # Normalize cumulative gain (to get the percentage of true positives)
        cumulative_gain[i] /= cumulative_gain[i][-1]

    # Plot Cumulative Gain Curves for each class
    plt.figure(figsize=(10, 8))

    for i in range(n_classes):
        plt.plot(np.linspace(0, 1, len(cumulative_gain[i])), cumulative_gain[i], lw=2, label=f'Class {i}')

    plt.xlabel('Proportion of Instances (Ranked by Predicted Probability)')
    plt.ylabel('Cumulative Gain')
    plt.title('Cumulative Gain Curves for Multiclass Classification')
    plt.legend(loc='best')
    plt.grid(True)
    plt.show()
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
```




## Class Distribution Plot (Count Plot)

```
import seaborn as sns

# Plot the distribution of labels
plt.figure(figsize=(10, 6))
sns.countplot(x=encoded_labels)
plt.title('Class Distribution in Dataset')
plt.xlabel('Classes')
plt.ylabel('Count')
plt.xticks(rotation=90)  # Rotate labels if necessary
plt.show()
```




## Learning Rate Finder Plot

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import (
    confusion_matrix,
    precision_recall_curve,
    average_precision_score,
    roc_curve,
    auc
)
from keras.callbacks import Callback
import keras
from sklearn.preprocessing import MultiLabelBinarizer

# Custom Callback to log learning rates
class LearningRateLogger(Callback):
    def __init__(self):
        super(LearningRateLogger, self).__init__()
        self.learning_rates = []

    def on_epoch_end(self, epoch, logs=None):
        # Access the learning rate directly from the optimizer
        lr = self.model.optimizer.learning_rate.numpy()  # Use .numpy() to get the value
        self.learning_rates.append(lr)

# Example: Generate synthetic data for demonstration (replace this with your actual data)
num_samples = 1000  # Total number of samples
num_classes = 18    # Number of classes

# Generate random features and labels (for demonstration)
np.random.seed(42)
features = np.random.rand(num_samples, 64)  # Assuming 64 features
encoded_labels = np.random.randint(0, num_classes, size=(num_samples,))  # Random class labels

# Convert to one-hot encoding (binarization)
mlb = MultiLabelBinarizer()
encoded_labels = mlb.fit_transform(encoded_labels.reshape(-1, 1))

# Split the data into training and validation sets
train_size = int(0.8 * num_samples)
training_features = features[:train_size]
training_labels = encoded_labels[:train_size]
```

```
validation_features = features[train_size:]
validation_labels = encoded_labels[train_size:]

# Create an instance of the learning rate logger
lr_logger = LearningRateLogger()

# Assuming 'model' is your trained Keras model
# Here is a sample model definition (replace with your actual model)
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(128, activation='relu', input_shape=(training_features.shape[1],)))
model.add(Dense(num_classes, activation='softmax'))  # Output layer for multi-class classification
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model with the learning rate logger
history = model.fit(
    training_features,
    training_labels,
    epochs=300,  # Adjust as needed
    validation_data=(validation_features, validation_labels),
    callbacks=[lr_logger]
)

# Generate predictions for confusion matrix and other metrics
predictions = model.predict(validation_features)
predicted_labels = np.argmax(predictions, axis=1)

# Create confusion matrix
cm = confusion_matrix(np.argmax(validation_labels, axis=1), predicted_labels)

# Plot confusion matrix
plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=mlb.classes_, yticklabels=mlb.classes_)
plt.title('Confusion Matrix: Predictions vs Actual Labels')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()

# Ensure predictions are properly formatted for precision-recall
y_scores = predictions  # No need to use [:, 1] as it's multiclass

# Compute precision-recall curve
precision, recall, _ = precision_recall_curve(validation_labels.ravel(), y_scores.ravel())
average_precision = average_precision_score(validation_labels, y_scores, average="micro")

# Plot Precision-Recall Curve
plt.figure(figsize=(8, 6))
plt.plot(recall, precision, label=f'Average Precision = {average_precision:.2f}')
plt.title('Precision-Recall Curve')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend()
plt.show()

# Compute ROC curve and AUC for each class
for i in range(validation_labels.shape[1]):
    fpr, tpr, _ = roc_curve(validation_labels[:, i], y_scores[:, i])
    roc_auc = auc(fpr, tpr)

    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, label=f'AUC for class {mlb.classes_[i]} = {roc_auc:.2f}')
    plt.plot([0, 1], [0, 1], 'k--')  # Dashed diagonal line
    plt.title(f'ROC Curve for class {mlb.classes_[i]}')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.legend()
    plt.show()

# Plot Learning Rate vs Loss
lrs = lr_logger.learning_rates
losses = history.history['loss']

plt.figure(figsize=(8, 6))
plt.plot(lrs, losses)
```

```
plt.xscale('log')
plt.xlabel('Learning Rate')
plt.ylabel('Loss')
plt.title('Learning Rate vs Loss')
plt.show()


# Proceed with metrics and plots
# Confusion matrix plotting
plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=mlb.classes_, yticklabels=mlb.classes_)
plt.title('Confusion Matrix: Predictions vs Actual Labels')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```

```
Epoch 1/300
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argumen
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
25/25 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - accuracy: 0.0627 - loss: 2.9519 - val_accuracy: 0.0450 - val_loss: 2.8942
Epoch 2/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.0598 - loss: 2.8739 - val_accuracy: 0.0800 - val_loss: 2.8988
Epoch 3/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0905 - loss: 2.8517 - val_accuracy: 0.0700 - val_loss: 2.8998
Epoch 4/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0950 - loss: 2.8326 - val_accuracy: 0.0700 - val_loss: 2.9019
Epoch 5/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.1125 - loss: 2.8019 - val_accuracy: 0.0650 - val_loss: 2.9030
Epoch 6/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.1065 - loss: 2.7979 - val_accuracy: 0.0550 - val_loss: 2.8988
Epoch 7/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.1244 - loss: 2.7700 - val_accuracy: 0.0650 - val_loss: 2.9140
Epoch 8/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.1389 - loss: 2.7630 - val_accuracy: 0.0850 - val_loss: 2.9161
Epoch 9/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.1500 - loss: 2.7244 - val_accuracy: 0.0550 - val_loss: 2.9159
Epoch 10/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.1870 - loss: 2.6955 - val_accuracy: 0.0600 - val_loss: 2.9214
Epoch 11/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2094 - loss: 2.6721 - val_accuracy: 0.0700 - val_loss: 2.9271
Epoch 12/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.2266 - loss: 2.6551 - val_accuracy: 0.0550 - val_loss: 2.9326
Epoch 13/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2167 - loss: 2.6494 - val_accuracy: 0.0500 - val_loss: 2.9308
Epoch 14/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.1999 - loss: 2.6118 - val_accuracy: 0.0550 - val_loss: 2.9335
Epoch 15/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2433 - loss: 2.5903 - val_accuracy: 0.0550 - val_loss: 2.9487
Epoch 16/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2169 - loss: 2.5803 - val_accuracy: 0.0400 - val_loss: 2.9798
Epoch 17/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2428 - loss: 2.5435 - val_accuracy: 0.0600 - val_loss: 2.9672
Epoch 18/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2742 - loss: 2.5152 - val_accuracy: 0.0500 - val_loss: 2.9600
Epoch 19/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2886 - loss: 2.4906 - val_accuracy: 0.0450 - val_loss: 2.9965
Epoch 20/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2624 - loss: 2.4765 - val_accuracy: 0.0650 - val_loss: 2.9839
Epoch 21/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2769 - loss: 2.4760 - val_accuracy: 0.0350 - val_loss: 2.9986
Epoch 22/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2764 - loss: 2.4535 - val_accuracy: 0.0500 - val_loss: 3.0089
Epoch 23/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.3399 - loss: 2.3876 - val_accuracy: 0.0300 - val_loss: 3.0365
Epoch 24/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2814 - loss: 2.4093 - val_accuracy: 0.0550 - val_loss: 3.0203
Epoch 25/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2912 - loss: 2.3778 - val_accuracy: 0.0400 - val_loss: 3.0487
Epoch 26/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3570 - loss: 2.3234 - val_accuracy: 0.0600 - val_loss: 3.0492
Epoch 27/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3110 - loss: 2.3259 - val_accuracy: 0.0400 - val_loss: 3.0691
Epoch 28/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3438 - loss: 2.3189 - val_accuracy: 0.0450 - val_loss: 3.0672
Epoch 29/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3612 - loss: 2.2501 - val_accuracy: 0.0500 - val_loss: 3.0770
Epoch 30/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3645 - loss: 2.2325 - val_accuracy: 0.0350 - val_loss: 3.0898
Epoch 31/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3719 - loss: 2.2255 - val_accuracy: 0.0450 - val_loss: 3.0975
Epoch 32/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3582 - loss: 2.2402 - val_accuracy: 0.0400 - val_loss: 3.1260
Epoch 33/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3483 - loss: 2.2131 - val_accuracy: 0.0400 - val_loss: 3.1388
Epoch 34/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.3599 - loss: 2.1893 - val_accuracy: 0.0350 - val_loss: 3.1450
Epoch 35/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3484 - loss: 2.1720 - val_accuracy: 0.0450 - val_loss: 3.1687
Epoch 36/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4020 - loss: 2.1391 - val_accuracy: 0.0400 - val_loss: 3.1711
Epoch 37/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3798 - loss: 2.1503 - val_accuracy: 0.0300 - val_loss: 3.1943
Epoch 38/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4005 - loss: 2.1081 - val_accuracy: 0.0300 - val_loss: 3.2051
Epoch 39/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.4176 - loss: 2.0783 - val_accuracy: 0.0250 - val_loss: 3.2273
Epoch 40/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3755 - loss: 2.0966 - val_accuracy: 0.0450 - val_loss: 3.2259
Epoch 41/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4146 - loss: 2.0568 - val_accuracy: 0.0300 - val_loss: 3.2308
```

```
Epoch 42/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4121 - loss: 2.0086 - val_accuracy: 0.0350 - val_loss: 3.2423
Epoch 43/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.4235 - loss: 2.0211 - val_accuracy: 0.0250 - val_loss: 3.2626
Epoch 44/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3979 - loss: 2.0005 - val_accuracy: 0.0500 - val_loss: 3.2895
Epoch 45/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.3936 - loss: 2.0065 - val_accuracy: 0.0250 - val_loss: 3.2908
Epoch 46/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4428 - loss: 1.9623 - val_accuracy: 0.0200 - val_loss: 3.2925
Epoch 47/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4373 - loss: 1.9521 - val_accuracy: 0.0350 - val_loss: 3.3172
Epoch 48/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.4400 - loss: 1.9659 - val_accuracy: 0.0350 - val_loss: 3.3076
Epoch 49/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4419 - loss: 1.9455 - val_accuracy: 0.0300 - val_loss: 3.3518
Epoch 50/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.4313 - loss: 1.9313 - val_accuracy: 0.0300 - val_loss: 3.3426
Epoch 51/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4368 - loss: 1.9185 - val_accuracy: 0.0350 - val_loss: 3.3508
Epoch 52/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4765 - loss: 1.8787 - val_accuracy: 0.0300 - val_loss: 3.3879
Epoch 53/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4728 - loss: 1.8496 - val_accuracy: 0.0300 - val_loss: 3.3812
Epoch 54/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4920 - loss: 1.8431 - val_accuracy: 0.0250 - val_loss: 3.3851
Epoch 55/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4729 - loss: 1.8444 - val_accuracy: 0.0450 - val_loss: 3.4372
Epoch 56/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.4907 - loss: 1.8597 - val_accuracy: 0.0350 - val_loss: 3.4365
Epoch 57/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5083 - loss: 1.7769 - val_accuracy: 0.0450 - val_loss: 3.4307
Epoch 58/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5272 - loss: 1.7865 - val_accuracy: 0.0300 - val_loss: 3.4391
Epoch 59/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5166 - loss: 1.7773 - val_accuracy: 0.0300 - val_loss: 3.4680
Epoch 60/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4919 - loss: 1.7856 - val_accuracy: 0.0350 - val_loss: 3.4785
Epoch 61/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4934 - loss: 1.7512 - val_accuracy: 0.0250 - val_loss: 3.4895
Epoch 62/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5128 - loss: 1.7509 - val_accuracy: 0.0250 - val_loss: 3.5080
Epoch 63/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5532 - loss: 1.6870 - val_accuracy: 0.0450 - val_loss: 3.5077
Epoch 64/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5311 - loss: 1.6749 - val_accuracy: 0.0350 - val_loss: 3.5609
Epoch 65/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5442 - loss: 1.6836 - val_accuracy: 0.0400 - val_loss: 3.5444
Epoch 66/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5669 - loss: 1.6484 - val_accuracy: 0.0400 - val_loss: 3.5274
Epoch 67/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5199 - loss: 1.6484 - val_accuracy: 0.0400 - val_loss: 3.5925
Epoch 68/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.5504 - loss: 1.6208 - val_accuracy: 0.0300 - val_loss: 3.5913
Epoch 69/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5360 - loss: 1.6398 - val_accuracy: 0.0300 - val_loss: 3.5846
Epoch 70/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5318 - loss: 1.6224 - val_accuracy: 0.0250 - val_loss: 3.5925
Epoch 71/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5513 - loss: 1.5755 - val_accuracy: 0.0350 - val_loss: 3.6274
Epoch 72/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5820 - loss: 1.5943 - val_accuracy: 0.0250 - val_loss: 3.6338
Epoch 73/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5313 - loss: 1.5944 - val_accuracy: 0.0300 - val_loss: 3.6454
Epoch 74/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5556 - loss: 1.5482 - val_accuracy: 0.0200 - val_loss: 3.6659
Epoch 75/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5900 - loss: 1.5269 - val_accuracy: 0.0300 - val_loss: 3.6675
Epoch 76/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5423 - loss: 1.5614 - val_accuracy: 0.0250 - val_loss: 3.6930
Epoch 77/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5782 - loss: 1.5161 - val_accuracy: 0.0300 - val_loss: 3.6763
Epoch 78/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6192 - loss: 1.4498 - val_accuracy: 0.0250 - val_loss: 3.7087
Epoch 79/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6179 - loss: 1.4713 - val_accuracy: 0.0250 - val_loss: 3.7160
Epoch 80/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5974 - loss: 1.4764 - val_accuracy: 0.0250 - val_loss: 3.7520
Epoch 81/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6380 - loss: 1.4319 - val_accuracy: 0.0350 - val_loss: 3.7172
Epoch 82/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6223 - loss: 1.4218 - val_accuracy: 0.0250 - val_loss: 3.7787
Epoch 83/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6214 - loss: 1.4382 - val_accuracy: 0.0250 - val_loss: 3.7735
Epoch 84/300
```

```
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6365 - loss: 1.3585 - val_accuracy: 0.0250 - val_loss: 3.7575
Epoch 85/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6343 - loss: 1.3837 - val_accuracy: 0.0250 - val_loss: 3.7711
Epoch 86/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6120 - loss: 1.3991 - val_accuracy: 0.0350 - val_loss: 3.8060
Epoch 87/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6770 - loss: 1.3374 - val_accuracy: 0.0250 - val_loss: 3.8216
Epoch 88/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6538 - loss: 1.3608 - val_accuracy: 0.0250 - val_loss: 3.8393
Epoch 89/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6630 - loss: 1.3286 - val_accuracy: 0.0300 - val_loss: 3.8358
Epoch 90/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6622 - loss: 1.3122 - val_accuracy: 0.0300 - val_loss: 3.8349
Epoch 91/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6730 - loss: 1.2984 - val_accuracy: 0.0200 - val_loss: 3.8541
Epoch 92/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6937 - loss: 1.2836 - val_accuracy: 0.0250 - val_loss: 3.8743
Epoch 93/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6863 - loss: 1.2554 - val_accuracy: 0.0250 - val_loss: 3.9104
Epoch 94/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6944 - loss: 1.2402 - val_accuracy: 0.0200 - val_loss: 3.8837
Epoch 95/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6869 - loss: 1.2360 - val_accuracy: 0.0300 - val_loss: 3.9280
Epoch 96/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7122 - loss: 1.2106 - val_accuracy: 0.0150 - val_loss: 3.9273
Epoch 97/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7262 - loss: 1.1916 - val_accuracy: 0.0250 - val_loss: 3.9405
Epoch 98/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7176 - loss: 1.1804 - val_accuracy: 0.0300 - val_loss: 3.9718
Epoch 99/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7370 - loss: 1.1807 - val_accuracy: 0.0150 - val_loss: 3.9693
Epoch 100/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7386 - loss: 1.1556 - val_accuracy: 0.0350 - val_loss: 3.9575
Epoch 101/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7490 - loss: 1.1664 - val_accuracy: 0.0150 - val_loss: 3.9536
Epoch 102/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7670 - loss: 1.1200 - val_accuracy: 0.0100 - val_loss: 4.0083
Epoch 103/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7390 - loss: 1.1253 - val_accuracy: 0.0200 - val_loss: 3.9941
Epoch 104/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7577 - loss: 1.0978 - val_accuracy: 0.0200 - val_loss: 4.0153
Epoch 105/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7653 - loss: 1.0854 - val_accuracy: 0.0200 - val_loss: 4.0253
Epoch 106/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7398 - loss: 1.1345 - val_accuracy: 0.0150 - val_loss: 4.0561
Epoch 107/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7525 - loss: 1.0960 - val_accuracy: 0.0250 - val_loss: 4.0529
Epoch 108/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7608 - loss: 1.0690 - val_accuracy: 0.0200 - val_loss: 4.1216
Epoch 109/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7590 - loss: 1.0680 - val_accuracy: 0.0100 - val_loss: 4.0769
Epoch 110/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7436 - loss: 1.0681 - val_accuracy: 0.0150 - val_loss: 4.1202
Epoch 111/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7786 - loss: 1.0335 - val_accuracy: 0.0250 - val_loss: 4.1346
Epoch 112/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7863 - loss: 0.9923 - val_accuracy: 0.0100 - val_loss: 4.1387
Epoch 113/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7618 - loss: 1.0304 - val_accuracy: 0.0100 - val_loss: 4.1303
Epoch 114/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7804 - loss: 1.0003 - val_accuracy: 0.0300 - val_loss: 4.1800
Epoch 115/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8132 - loss: 0.9689 - val_accuracy: 0.0150 - val_loss: 4.1470
Epoch 116/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8166 - loss: 0.9343 - val_accuracy: 0.0150 - val_loss: 4.1744
Epoch 117/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7984 - loss: 0.9687 - val_accuracy: 0.0200 - val_loss: 4.2051
Epoch 118/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8001 - loss: 0.9445 - val_accuracy: 0.0150 - val_loss: 4.1976
Epoch 119/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8088 - loss: 0.9626 - val_accuracy: 0.0150 - val_loss: 4.2344
Epoch 120/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8310 - loss: 0.8855 - val_accuracy: 0.0150 - val_loss: 4.2233
Epoch 121/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8333 - loss: 0.9256 - val_accuracy: 0.0200 - val_loss: 4.2835
Epoch 122/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8508 - loss: 0.8596 - val_accuracy: 0.0150 - val_loss: 4.2787
Epoch 123/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8536 - loss: 0.8768 - val_accuracy: 0.0200 - val_loss: 4.2773
Epoch 124/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8502 - loss: 0.8619 - val_accuracy: 0.0250 - val_loss: 4.3110
Epoch 125/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8501 - loss: 0.8776 - val_accuracy: 0.0100 - val_loss: 4.2893
Epoch 126/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8642 - loss: 0.8444 - val_accuracy: 0.0100 - val_loss: 4.3027
```

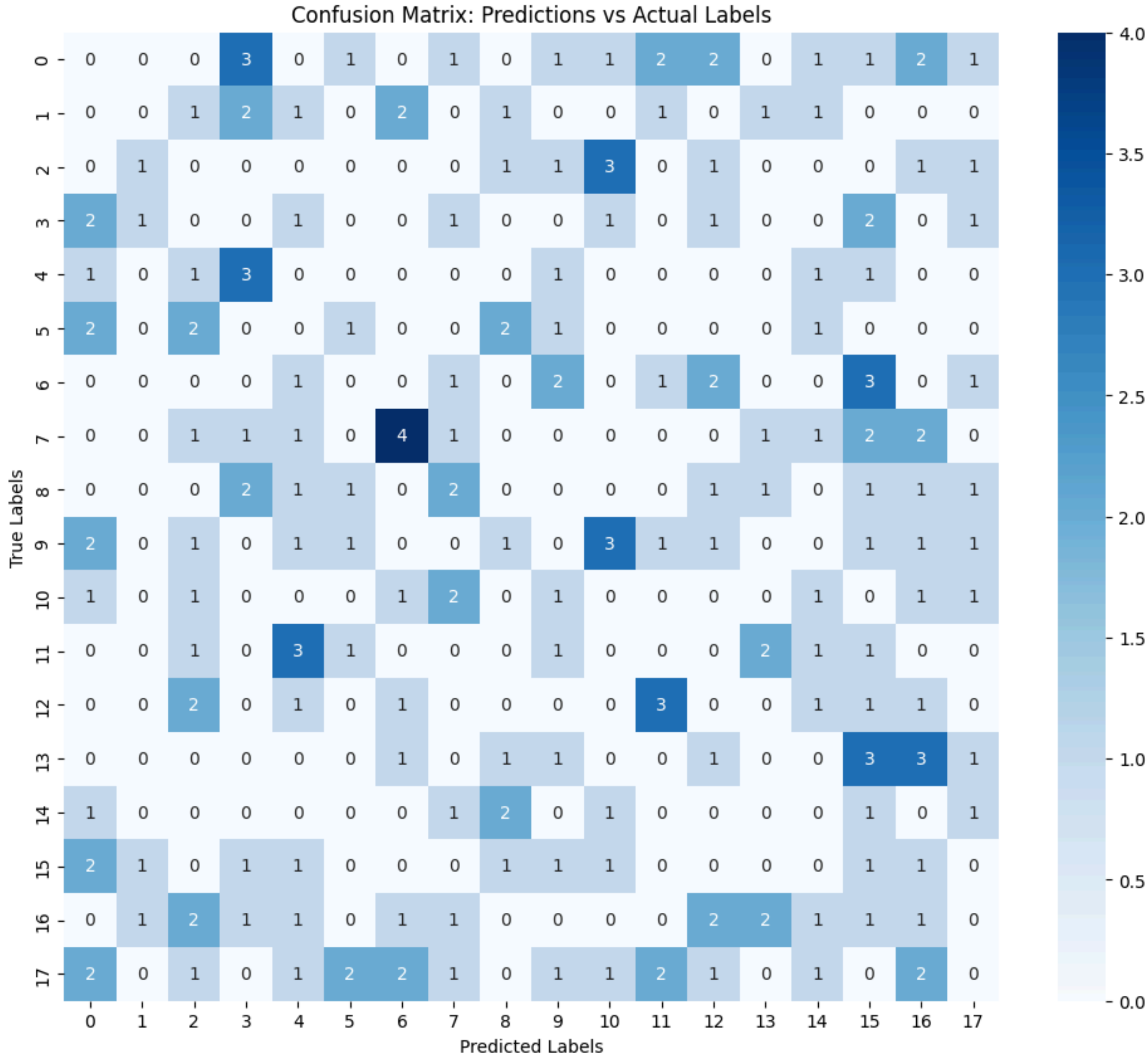
```
Epoch 127/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8625 - loss: 0.8327 - val_accuracy: 0.0150 - val_loss: 4.3209
Epoch 128/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8809 - loss: 0.8375 - val_accuracy: 0.0150 - val_loss: 4.3594
Epoch 129/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8859 - loss: 0.8041 - val_accuracy: 0.0250 - val_loss: 4.3685
Epoch 130/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8660 - loss: 0.8180 - val_accuracy: 0.0250 - val_loss: 4.4124
Epoch 131/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8761 - loss: 0.7987 - val_accuracy: 0.0250 - val_loss: 4.3759
Epoch 132/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8858 - loss: 0.7931 - val_accuracy: 0.0200 - val_loss: 4.4092
Epoch 133/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8849 - loss: 0.7452 - val_accuracy: 0.0200 - val_loss: 4.3964
Epoch 134/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8897 - loss: 0.7448 - val_accuracy: 0.0300 - val_loss: 4.4435
Epoch 135/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8965 - loss: 0.7522 - val_accuracy: 0.0100 - val_loss: 4.4488
Epoch 136/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9077 - loss: 0.7188 - val_accuracy: 0.0250 - val_loss: 4.4525
Epoch 137/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9001 - loss: 0.7209 - val_accuracy: 0.0300 - val_loss: 4.4856
Epoch 138/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8969 - loss: 0.7057 - val_accuracy: 0.0150 - val_loss: 4.4910
Epoch 139/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9126 - loss: 0.7031 - val_accuracy: 0.0150 - val_loss: 4.4899
Epoch 140/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9162 - loss: 0.7044 - val_accuracy: 0.0150 - val_loss: 4.5193
Epoch 141/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9264 - loss: 0.6688 - val_accuracy: 0.0250 - val_loss: 4.5624
Epoch 142/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9051 - loss: 0.6940 - val_accuracy: 0.0250 - val_loss: 4.5255
Epoch 143/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9481 - loss: 0.6527 - val_accuracy: 0.0250 - val_loss: 4.5582
Epoch 144/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9341 - loss: 0.6578 - val_accuracy: 0.0150 - val_loss: 4.5782
Epoch 145/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9367 - loss: 0.6521 - val_accuracy: 0.0300 - val_loss: 4.6095
Epoch 146/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9382 - loss: 0.6512 - val_accuracy: 0.0300 - val_loss: 4.6115
Epoch 147/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9325 - loss: 0.6472 - val_accuracy: 0.0150 - val_loss: 4.5912
Epoch 148/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9473 - loss: 0.6169 - val_accuracy: 0.0300 - val_loss: 4.6354
Epoch 149/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9519 - loss: 0.6126 - val_accuracy: 0.0200 - val_loss: 4.6494
Epoch 150/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9490 - loss: 0.6014 - val_accuracy: 0.0250 - val_loss: 4.6401
Epoch 151/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9594 - loss: 0.5798 - val_accuracy: 0.0250 - val_loss: 4.6611
Epoch 152/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9446 - loss: 0.5832 - val_accuracy: 0.0250 - val_loss: 4.7063
Epoch 153/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9687 - loss: 0.5538 - val_accuracy: 0.0250 - val_loss: 4.6828
Epoch 154/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9693 - loss: 0.5662 - val_accuracy: 0.0150 - val_loss: 4.7226
Epoch 155/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9663 - loss: 0.5548 - val_accuracy: 0.0250 - val_loss: 4.7363
Epoch 156/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9592 - loss: 0.5694 - val_accuracy: 0.0200 - val_loss: 4.7434
Epoch 157/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9774 - loss: 0.5205 - val_accuracy: 0.0250 - val_loss: 4.7642
Epoch 158/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9671 - loss: 0.5327 - val_accuracy: 0.0250 - val_loss: 4.7914
Epoch 159/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9730 - loss: 0.5166 - val_accuracy: 0.0200 - val_loss: 4.8172
Epoch 160/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9744 - loss: 0.5057 - val_accuracy: 0.0200 - val_loss: 4.8064
Epoch 161/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9699 - loss: 0.5005 - val_accuracy: 0.0200 - val_loss: 4.8295
Epoch 162/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9840 - loss: 0.4780 - val_accuracy: 0.0200 - val_loss: 4.8475
Epoch 163/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9747 - loss: 0.4946 - val_accuracy: 0.0150 - val_loss: 4.8802
Epoch 164/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9801 - loss: 0.5003 - val_accuracy: 0.0300 - val_loss: 4.8755
Epoch 165/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9822 - loss: 0.4629 - val_accuracy: 0.0200 - val_loss: 4.8486
Epoch 166/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9832 - loss: 0.4577 - val_accuracy: 0.0250 - val_loss: 4.8913
Epoch 167/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9750 - loss: 0.4634 - val_accuracy: 0.0200 - val_loss: 4.8992
Epoch 168/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9821 - loss: 0.4458 - val_accuracy: 0.0250 - val_loss: 4.9154
Epoch 169/300
```

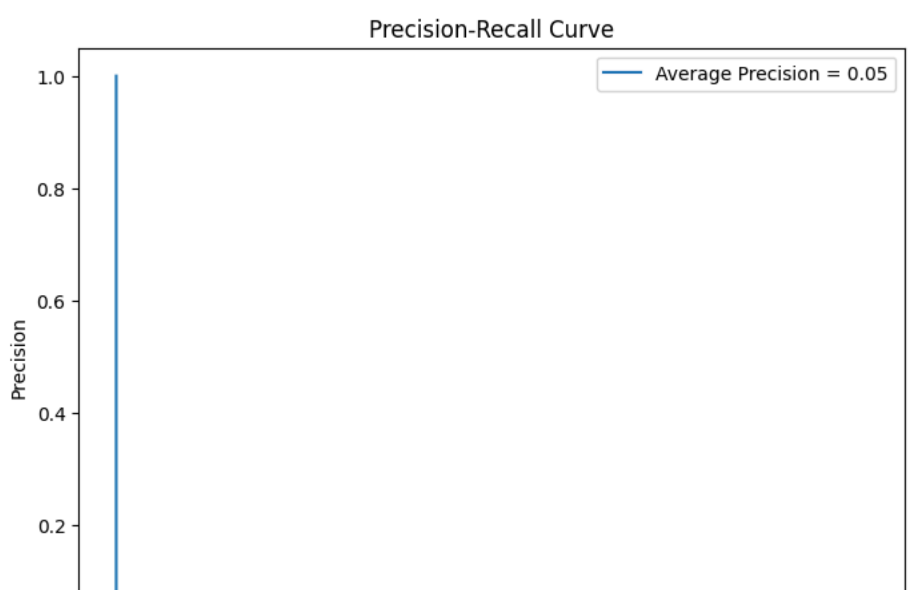
```
Epoch 169/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9858 - loss: 0.4525 - val_accuracy: 0.0200 - val_loss: 4.9535
Epoch 170/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9839 - loss: 0.4335 - val_accuracy: 0.0300 - val_loss: 4.9589
Epoch 171/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9919 - loss: 0.4103 - val_accuracy: 0.0200 - val_loss: 4.9959
Epoch 172/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9882 - loss: 0.4000 - val_accuracy: 0.0200 - val_loss: 4.9884
Epoch 173/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9889 - loss: 0.4157 - val_accuracy: 0.0300 - val_loss: 4.9861
Epoch 174/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9941 - loss: 0.4043 - val_accuracy: 0.0350 - val_loss: 5.0185
Epoch 175/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9960 - loss: 0.3883 - val_accuracy: 0.0200 - val_loss: 5.0528
Epoch 176/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9887 - loss: 0.3951 - val_accuracy: 0.0300 - val_loss: 5.0539
Epoch 177/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9947 - loss: 0.3815 - val_accuracy: 0.0200 - val_loss: 5.1041
Epoch 178/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9926 - loss: 0.3806 - val_accuracy: 0.0300 - val_loss: 5.0838
Epoch 179/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9943 - loss: 0.3807 - val_accuracy: 0.0300 - val_loss: 5.1443
Epoch 180/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9930 - loss: 0.3625 - val_accuracy: 0.0200 - val_loss: 5.1358
Epoch 181/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9944 - loss: 0.3591 - val_accuracy: 0.0300 - val_loss: 5.1656
Epoch 182/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9948 - loss: 0.3416 - val_accuracy: 0.0250 - val_loss: 5.1617
Epoch 183/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9996 - loss: 0.3365 - val_accuracy: 0.0250 - val_loss: 5.1633
Epoch 184/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9934 - loss: 0.3374 - val_accuracy: 0.0250 - val_loss: 5.1843
Epoch 185/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9985 - loss: 0.3350 - val_accuracy: 0.0300 - val_loss: 5.2149
Epoch 186/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9988 - loss: 0.3364 - val_accuracy: 0.0300 - val_loss: 5.2646
Epoch 187/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9982 - loss: 0.3228 - val_accuracy: 0.0250 - val_loss: 5.1986
Epoch 188/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9954 - loss: 0.3115 - val_accuracy: 0.0250 - val_loss: 5.2430
Epoch 189/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9970 - loss: 0.2942 - val_accuracy: 0.0250 - val_loss: 5.3041
Epoch 190/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9895 - loss: 0.3060 - val_accuracy: 0.0300 - val_loss: 5.2698
Epoch 191/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9966 - loss: 0.3027 - val_accuracy: 0.0250 - val_loss: 5.2967
Epoch 192/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9979 - loss: 0.3032 - val_accuracy: 0.0250 - val_loss: 5.2912
Epoch 193/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9973 - loss: 0.2892 - val_accuracy: 0.0250 - val_loss: 5.3759
Epoch 194/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9984 - loss: 0.2858 - val_accuracy: 0.0300 - val_loss: 5.3487
Epoch 195/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9987 - loss: 0.2827 - val_accuracy: 0.0300 - val_loss: 5.3489
Epoch 196/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.2726 - val_accuracy: 0.0250 - val_loss: 5.3879
Epoch 197/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.2682 - val_accuracy: 0.0250 - val_loss: 5.4063
Epoch 198/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.2745 - val_accuracy: 0.0250 - val_loss: 5.4017
Epoch 199/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.2606 - val_accuracy: 0.0300 - val_loss: 5.4544
Epoch 200/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.2554 - val_accuracy: 0.0250 - val_loss: 5.4223
Epoch 201/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.2522 - val_accuracy: 0.0300 - val_loss: 5.4406
Epoch 202/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.2514 - val_accuracy: 0.0250 - val_loss: 5.4573
Epoch 203/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9997 - loss: 0.2524 - val_accuracy: 0.0250 - val_loss: 5.4838
Epoch 204/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.2414 - val_accuracy: 0.0300 - val_loss: 5.4841
Epoch 205/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 1.0000 - loss: 0.2383 - val_accuracy: 0.0300 - val_loss: 5.5339
Epoch 206/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.2300 - val_accuracy: 0.0250 - val_loss: 5.5169
Epoch 207/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.2286 - val_accuracy: 0.0300 - val_loss: 5.5620
Epoch 208/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.2291 - val_accuracy: 0.0200 - val_loss: 5.5687
Epoch 209/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.2193 - val_accuracy: 0.0250 - val_loss: 5.5768
Epoch 210/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.2117 - val_accuracy: 0.0250 - val_loss: 5.5751
Epoch 211/300
```

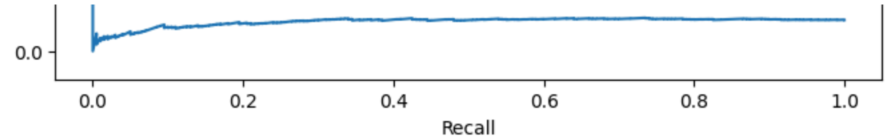
```
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.2083 - val_accuracy: 0.0250 - val_loss: 5.6216
Epoch 212/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.2046 - val_accuracy: 0.0250 - val_loss: 5.6303
Epoch 213/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.2062 - val_accuracy: 0.0250 - val_loss: 5.6572
Epoch 214/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1975 - val_accuracy: 0.0250 - val_loss: 5.6723
Epoch 215/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1992 - val_accuracy: 0.0300 - val_loss: 5.6680
Epoch 216/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1940 - val_accuracy: 0.0250 - val_loss: 5.6590
Epoch 217/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1949 - val_accuracy: 0.0300 - val_loss: 5.6947
Epoch 218/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1900 - val_accuracy: 0.0300 - val_loss: 5.7323
Epoch 219/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.1883 - val_accuracy: 0.0250 - val_loss: 5.7241
Epoch 220/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1762 - val_accuracy: 0.0300 - val_loss: 5.7552
Epoch 221/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1785 - val_accuracy: 0.0350 - val_loss: 5.7853
Epoch 222/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1759 - val_accuracy: 0.0300 - val_loss: 5.7781
Epoch 223/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1728 - val_accuracy: 0.0300 - val_loss: 5.8116
Epoch 224/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1707 - val_accuracy: 0.0300 - val_loss: 5.8554
Epoch 225/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1667 - val_accuracy: 0.0300 - val_loss: 5.8419
Epoch 226/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1653 - val_accuracy: 0.0250 - val_loss: 5.8558
Epoch 227/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1579 - val_accuracy: 0.0300 - val_loss: 5.8705
Epoch 228/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1508 - val_accuracy: 0.0300 - val_loss: 5.8732
Epoch 229/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1549 - val_accuracy: 0.0350 - val_loss: 5.9001
Epoch 230/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1536 - val_accuracy: 0.0300 - val_loss: 5.9113
Epoch 231/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1502 - val_accuracy: 0.0250 - val_loss: 5.9118
Epoch 232/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1452 - val_accuracy: 0.0250 - val_loss: 5.9593
Epoch 233/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1502 - val_accuracy: 0.0250 - val_loss: 5.9476
Epoch 234/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1447 - val_accuracy: 0.0250 - val_loss: 5.9517
Epoch 235/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1394 - val_accuracy: 0.0200 - val_loss: 6.0125
Epoch 236/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1367 - val_accuracy: 0.0250 - val_loss: 6.0165
Epoch 237/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1333 - val_accuracy: 0.0250 - val_loss: 6.0030
Epoch 238/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1333 - val_accuracy: 0.0300 - val_loss: 6.0379
Epoch 239/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1330 - val_accuracy: 0.0250 - val_loss: 6.0640
Epoch 240/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1309 - val_accuracy: 0.0300 - val_loss: 6.0501
Epoch 241/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1267 - val_accuracy: 0.0250 - val_loss: 6.0647
Epoch 242/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1228 - val_accuracy: 0.0250 - val_loss: 6.1092
Epoch 243/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1245 - val_accuracy: 0.0300 - val_loss: 6.1102
Epoch 244/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1247 - val_accuracy: 0.0300 - val_loss: 6.1147
Epoch 245/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1174 - val_accuracy: 0.0250 - val_loss: 6.0896
Epoch 246/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1134 - val_accuracy: 0.0300 - val_loss: 6.1703
Epoch 247/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1167 - val_accuracy: 0.0250 - val_loss: 6.1888
Epoch 248/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1183 - val_accuracy: 0.0200 - val_loss: 6.1739
Epoch 249/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1090 - val_accuracy: 0.0250 - val_loss: 6.1971
Epoch 250/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 0.1096 - val_accuracy: 0.0300 - val_loss: 6.2050
Epoch 251/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1052 - val_accuracy: 0.0250 - val_loss: 6.2423
Epoch 252/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1055 - val_accuracy: 0.0250 - val_loss: 6.2509
Epoch 253/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1081 - val_accuracy: 0.0300 - val_loss: 6.2712
```

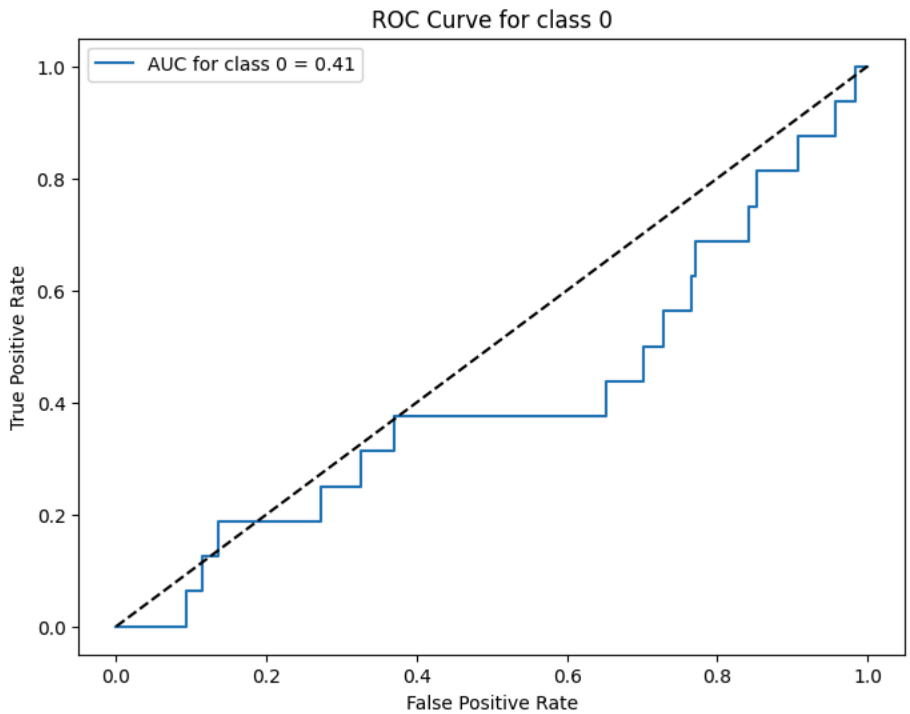
```
Epoch 254/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.1009 - val_accuracy: 0.0300 - val_loss: 6.2711
Epoch 255/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.1035 - val_accuracy: 0.0300 - val_loss: 6.2782
Epoch 256/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0977 - val_accuracy: 0.0250 - val_loss: 6.3300
Epoch 257/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0948 - val_accuracy: 0.0250 - val_loss: 6.3267
Epoch 258/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0928 - val_accuracy: 0.0250 - val_loss: 6.3308
Epoch 259/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0914 - val_accuracy: 0.0250 - val_loss: 6.3367
Epoch 260/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0937 - val_accuracy: 0.0250 - val_loss: 6.3623
Epoch 261/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0886 - val_accuracy: 0.0250 - val_loss: 6.3815
Epoch 262/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0882 - val_accuracy: 0.0350 - val_loss: 6.3977
Epoch 263/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0868 - val_accuracy: 0.0250 - val_loss: 6.3962
Epoch 264/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0869 - val_accuracy: 0.0250 - val_loss: 6.4072
Epoch 265/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0859 - val_accuracy: 0.0250 - val_loss: 6.4431
Epoch 266/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0833 - val_accuracy: 0.0300 - val_loss: 6.4579
Epoch 267/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0836 - val_accuracy: 0.0250 - val_loss: 6.4620
Epoch 268/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0810 - val_accuracy: 0.0250 - val_loss: 6.4896
Epoch 269/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0811 - val_accuracy: 0.0250 - val_loss: 6.5004
Epoch 270/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0785 - val_accuracy: 0.0250 - val_loss: 6.5130
Epoch 271/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0793 - val_accuracy: 0.0300 - val_loss: 6.5484
Epoch 272/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0774 - val_accuracy: 0.0250 - val_loss: 6.5285
Epoch 273/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0742 - val_accuracy: 0.0250 - val_loss: 6.5371
Epoch 274/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0781 - val_accuracy: 0.0300 - val_loss: 6.5595
Epoch 275/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0736 - val_accuracy: 0.0250 - val_loss: 6.5834
Epoch 276/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0717 - val_accuracy: 0.0250 - val_loss: 6.6108
Epoch 277/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0699 - val_accuracy: 0.0250 - val_loss: 6.6417
Epoch 278/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0683 - val_accuracy: 0.0250 - val_loss: 6.6227
Epoch 279/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0659 - val_accuracy: 0.0300 - val_loss: 6.6326
Epoch 280/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0666 - val_accuracy: 0.0200 - val_loss: 6.6652
Epoch 281/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0664 - val_accuracy: 0.0300 - val_loss: 6.6665
Epoch 282/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0638 - val_accuracy: 0.0250 - val_loss: 6.6946
Epoch 283/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0623 - val_accuracy: 0.0300 - val_loss: 6.7004
Epoch 284/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0625 - val_accuracy: 0.0250 - val_loss: 6.7022
Epoch 285/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0618 - val_accuracy: 0.0250 - val_loss: 6.7420
Epoch 286/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0587 - val_accuracy: 0.0300 - val_loss: 6.7471
Epoch 287/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0608 - val_accuracy: 0.0250 - val_loss: 6.7515
Epoch 288/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0577 - val_accuracy: 0.0250 - val_loss: 6.7696
Epoch 289/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0591 - val_accuracy: 0.0250 - val_loss: 6.7606
Epoch 290/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0583 - val_accuracy: 0.0250 - val_loss: 6.7988
Epoch 291/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0564 - val_accuracy: 0.0300 - val_loss: 6.8139
Epoch 292/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0546 - val_accuracy: 0.0200 - val_loss: 6.8266
Epoch 293/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0552 - val_accuracy: 0.0300 - val_loss: 6.8395
Epoch 294/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0539 - val_accuracy: 0.0300 - val_loss: 6.8562
Epoch 295/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0532 - val_accuracy: 0.0250 - val_loss: 6.8548
Epoch 296/300
```

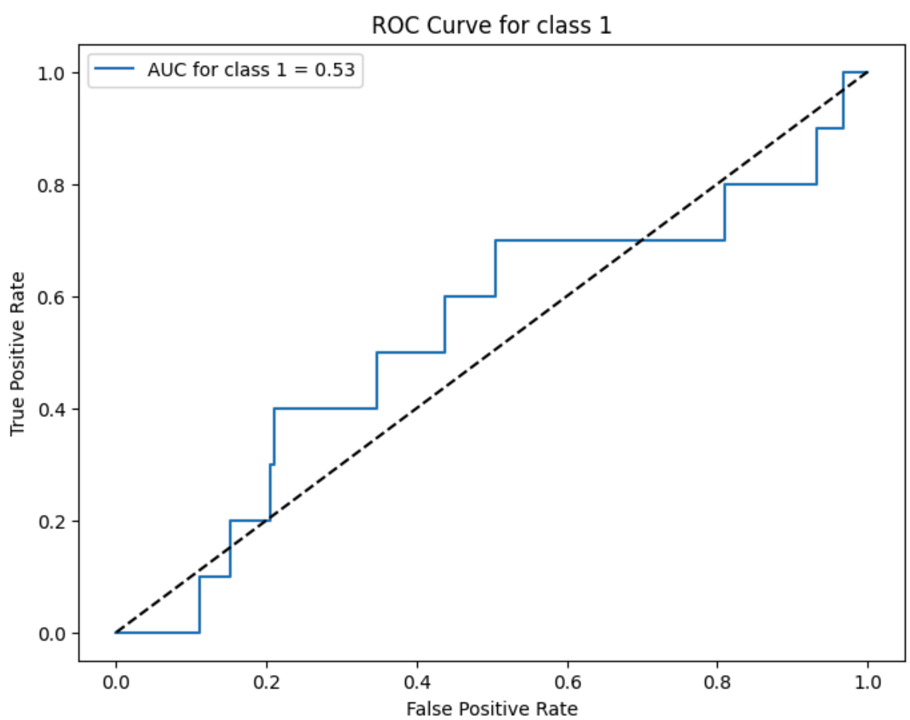
```
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0533 - val_accuracy: 0.0300 - val_loss: 6.8833
Epoch 297/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0505 - val_accuracy: 0.0250 - val_loss: 6.8826
Epoch 298/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0509 - val_accuracy: 0.0250 - val_loss: 6.9151
Epoch 299/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0494 - val_accuracy: 0.0250 - val_loss: 6.9388
Epoch 300/300
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0491 - val_accuracy: 0.0200 - val_loss: 6.9188
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
```

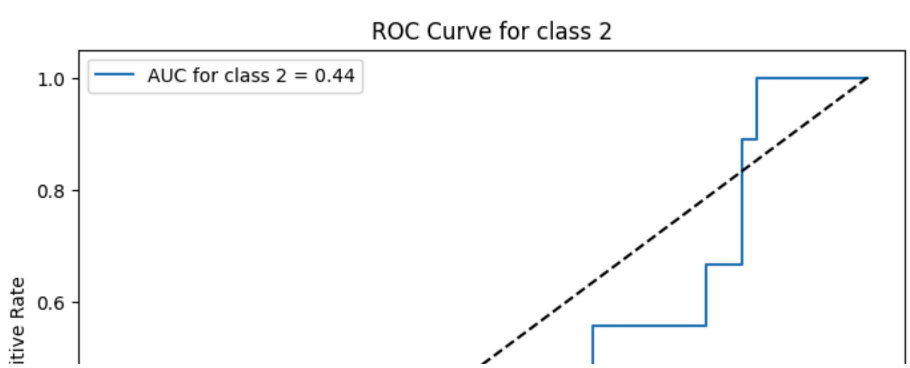


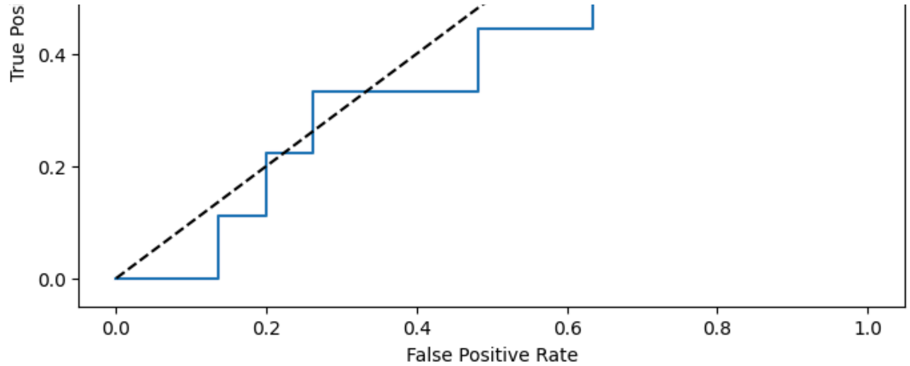



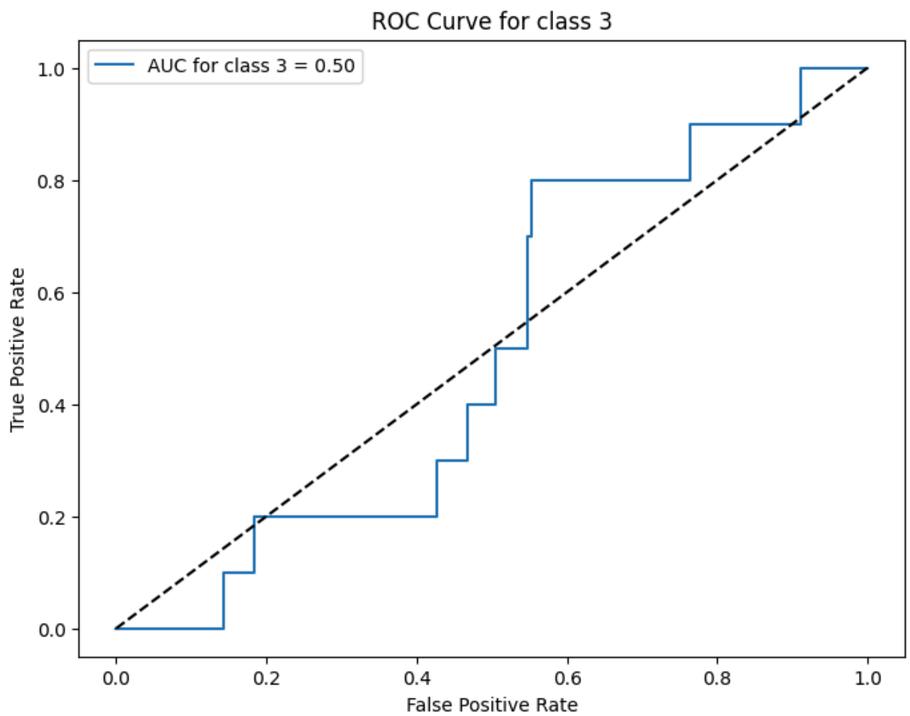

0.0
0.0 0.2 0.4 0.6 0.8 1.0
Recall

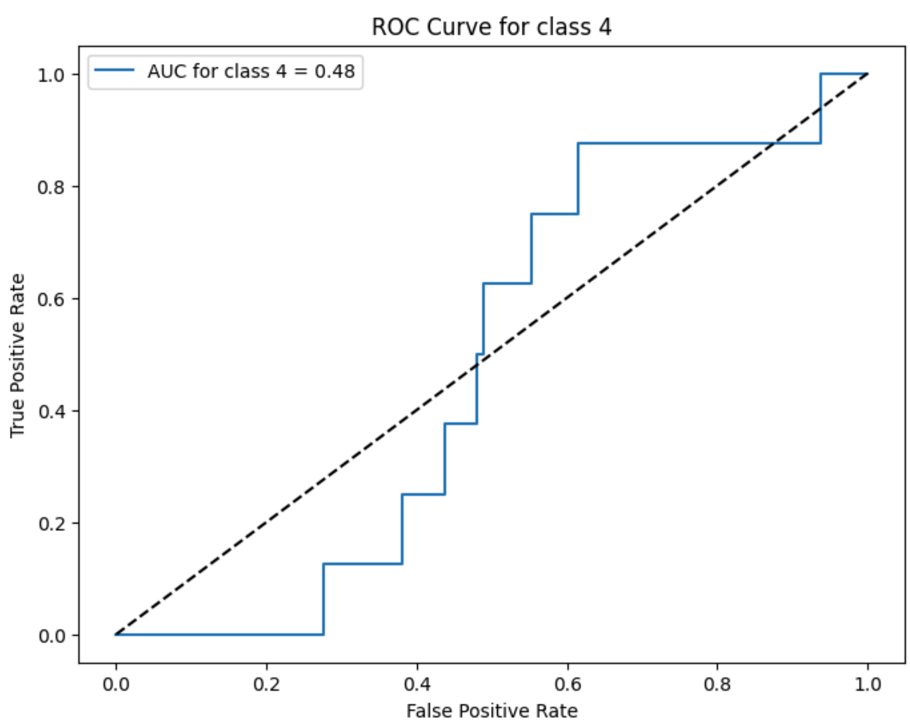

ROC Curve for class 0
AUC for class 0 = 0.41
True Positive Rate
False Positive Rate
0.0 0.2 0.4 0.6 0.8 1.0

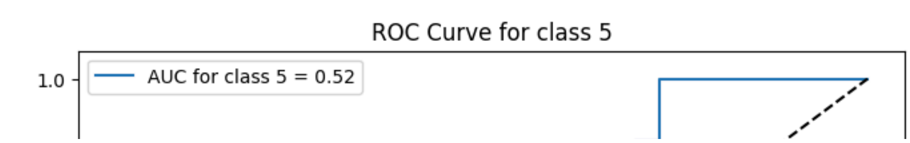

ROC Curve for class 1
AUC for class 1 = 0.53
True Positive Rate
False Positive Rate
0.0 0.2 0.4 0.6 0.8 1.0

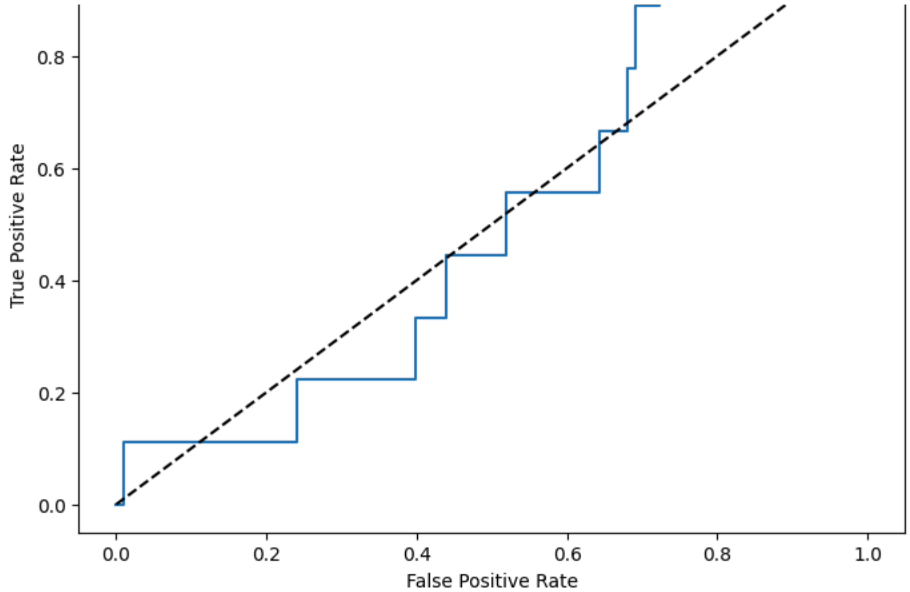

ROC Curve for class 2
AUC for class 2 = 0.44
1.0
0.8
0.6

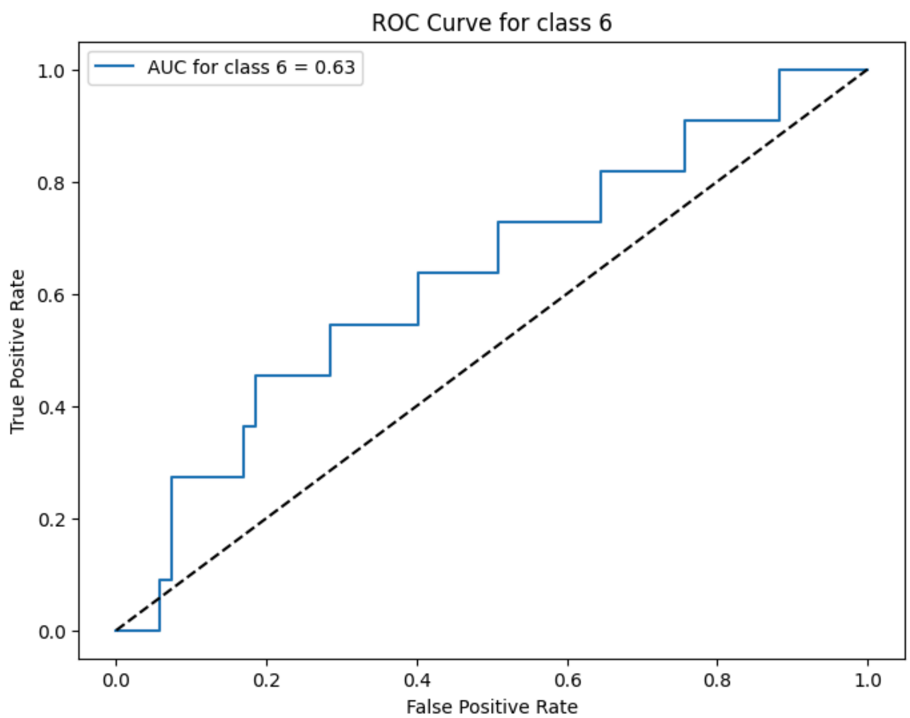

True Pos
False Positive Rate

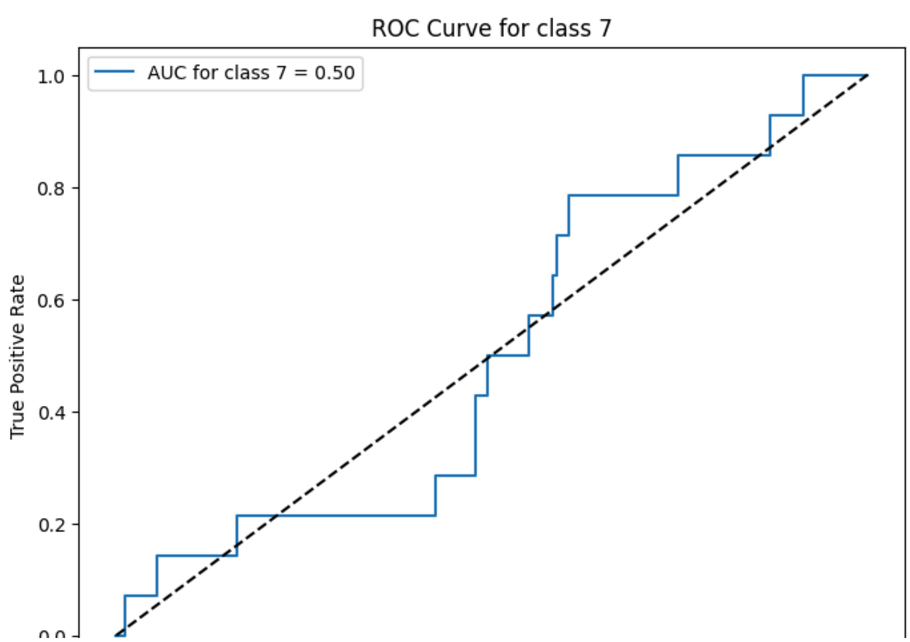

ROC Curve for class 3
AUC for class 3 = 0.50
True Positive Rate
False Positive Rate

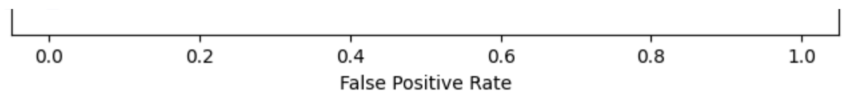

ROC Curve for class 4
AUC for class 4 = 0.48
True Positive Rate
False Positive Rate

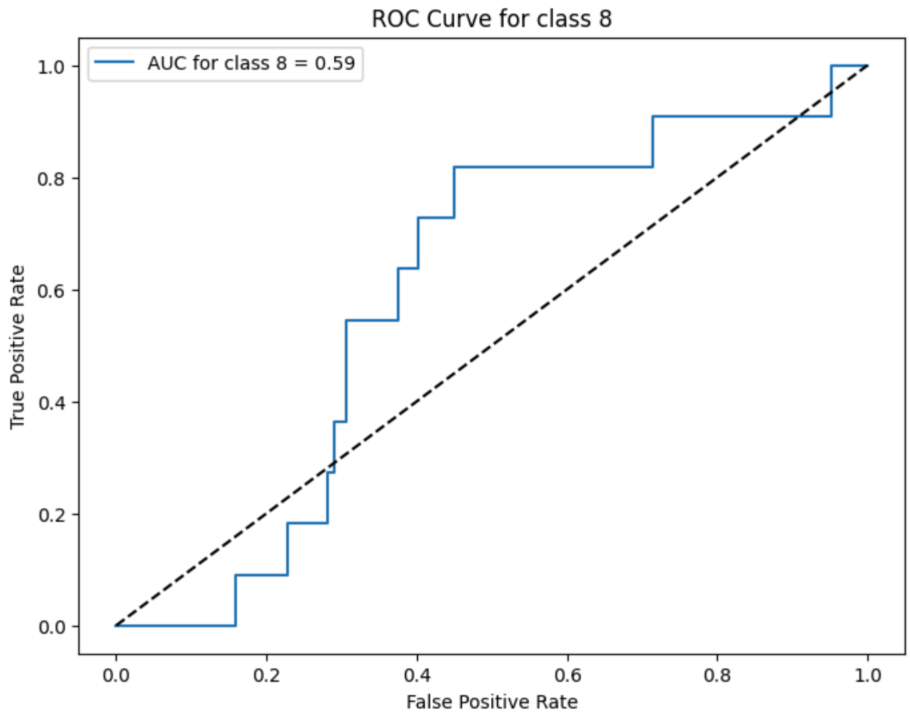

ROC Curve for class 5
AUC for class 5 = 0.52

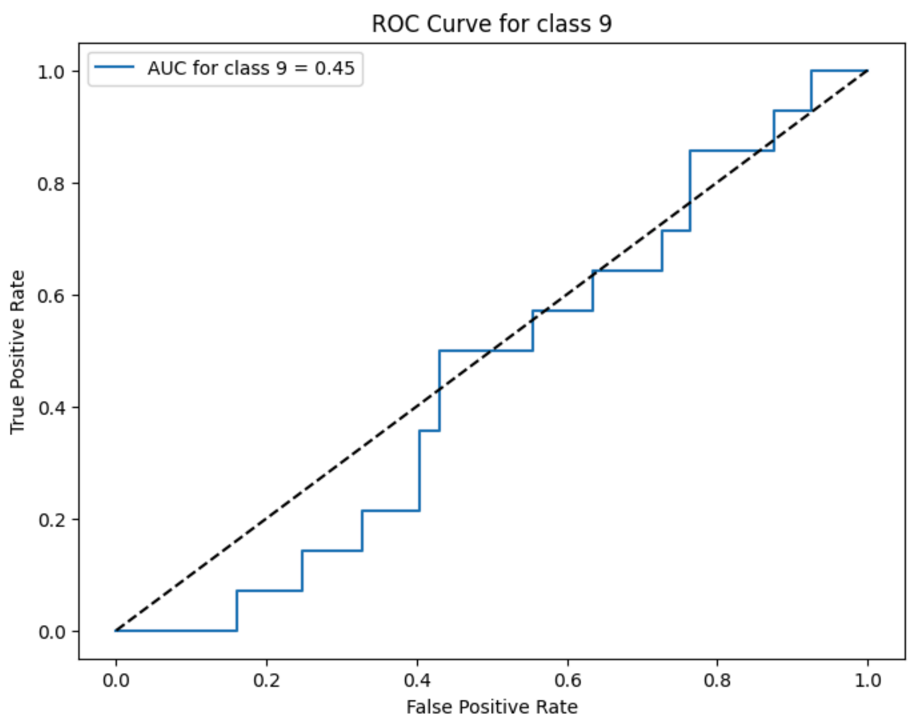

True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

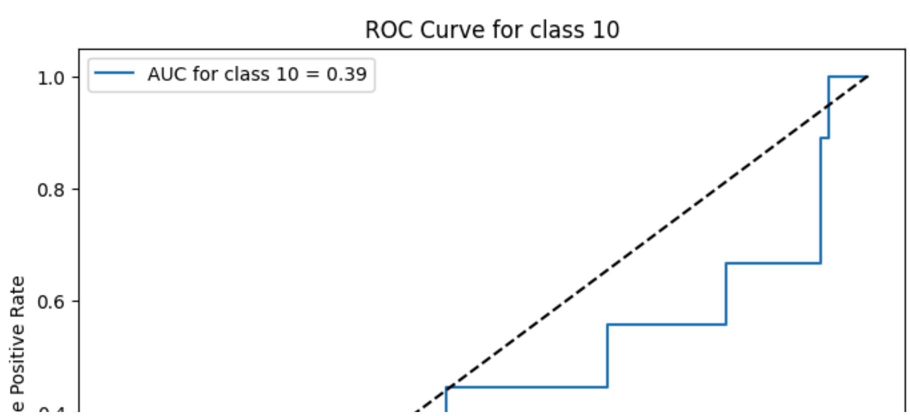

ROC Curve for class 6
AUC for class 6 = 0.63
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

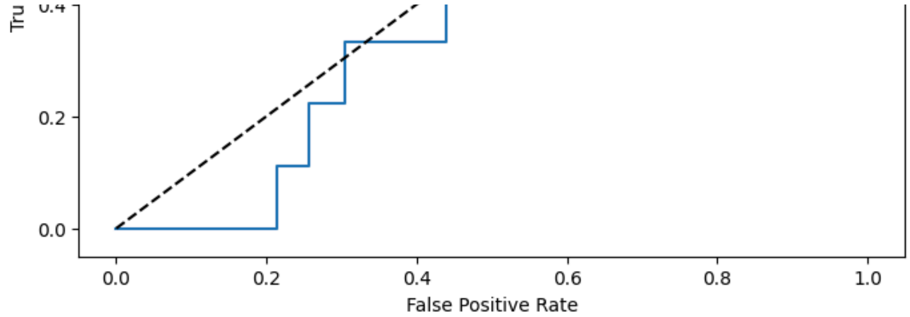

ROC Curve for class 7
AUC for class 7 = 0.50
True Positive Rate
0.2
0.4
0.6
0.8
1.0

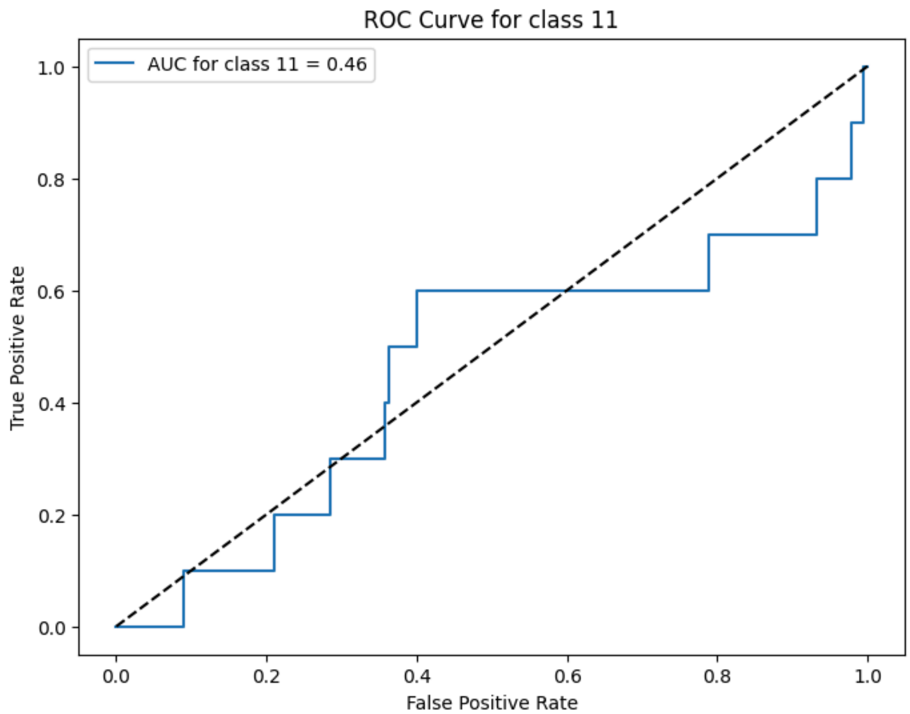

0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate

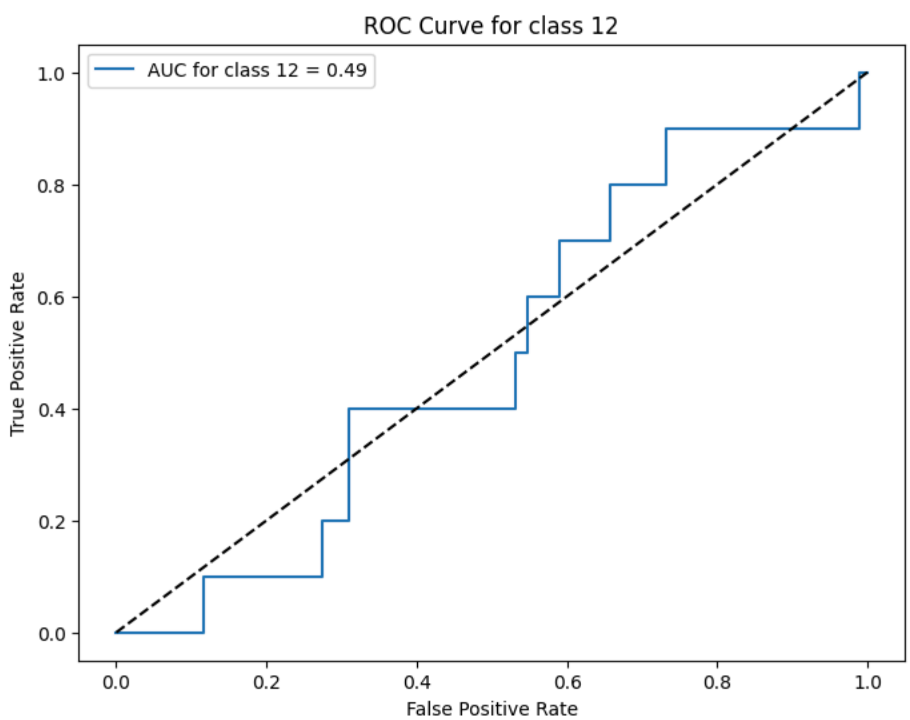

ROC Curve for class 8
AUC for class 8 = 0.59
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

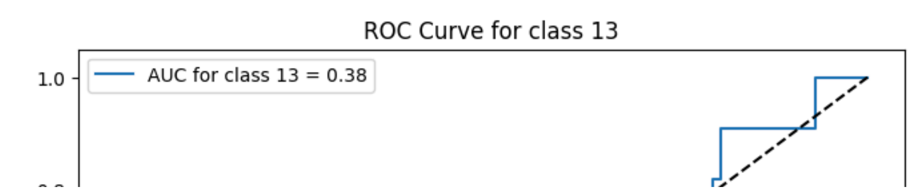

ROC Curve for class 9
AUC for class 9 = 0.45
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

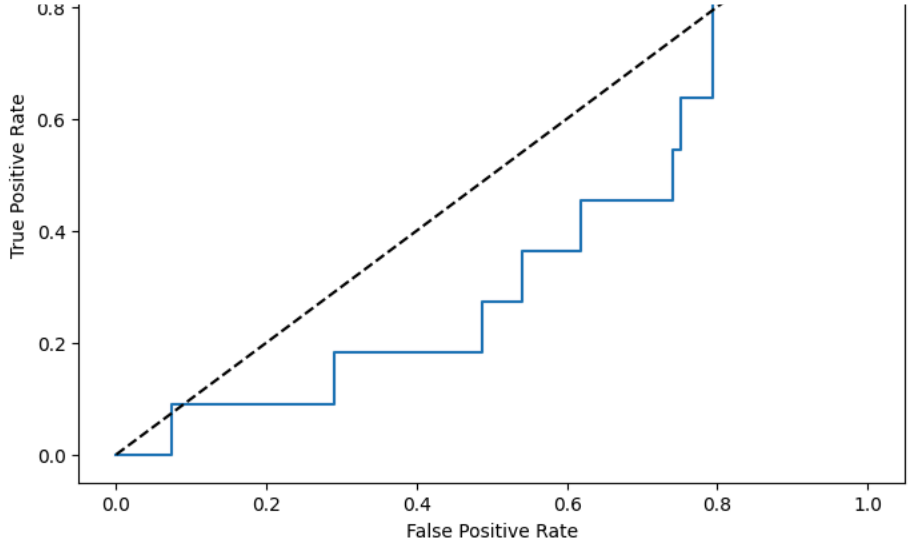

ROC Curve for class 10
AUC for class 10 = 0.39
1.0
0.8
0.6

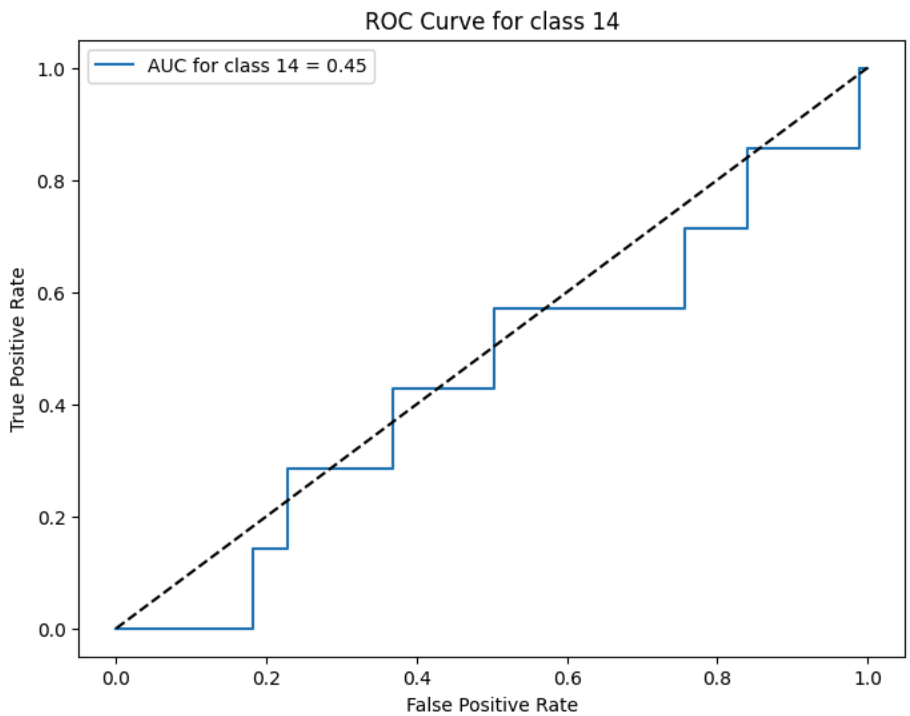

0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate

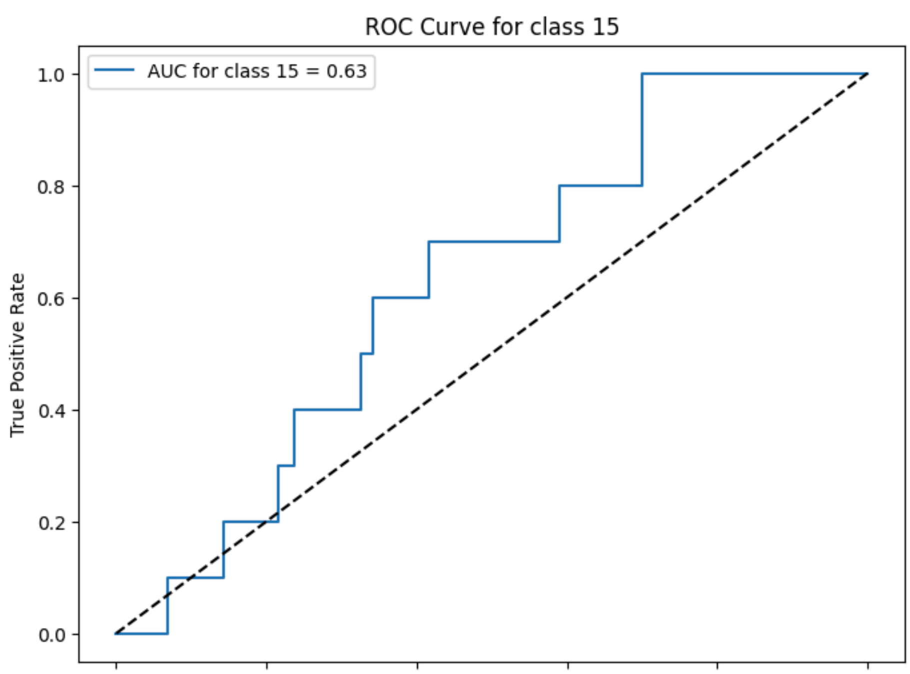

ROC Curve for class 11
AUC for class 11 = 0.46
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve for class 12
AUC for class 12 = 0.49
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve for class 13
AUC for class 13 = 0.38
1.0


True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve for class 14
AUC for class 14 = 0.45
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve for class 15
AUC for class 15 = 0.63
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0