```
#Jack Baxter
#Final Project: AAI 511
#11 August 2025
```

```
#import necessary packages
import pretty_midi
import pandas as pd
import numpy as np
from google.colab import drive
import os
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.layers import LSTM, TimeDistributed
from sklearn.metrics import accuracy_score, precision_score, recall_score
```

```
#initialize connection to google drive data
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=Tr
```

```
#create main dataframe for piano rolls and composer id
main_music = pd.DataFrame(columns = ['song', 'composer_id'])
```

```
#initialize standard midi & piano roll parameters
hz = 100
timesteps = 4000
pitchrange = 128
```

```
#define function to get piano roll from midi file
#function created by LLM (xAI, 2025)
def get_piano_roll(midi_path, fs=100, fixed_length=4000):
    try:
        midi_data = pretty_midi.PrettyMIDI(midi_path)
        piano_roll = midi_data.get_piano_roll(fs=fs)
        piano_roll = piano_roll / 127.0
        current_length = piano_roll.shape[1]
        if current_length < fixed_length:
            padded = np.zeros((pitchrange, fixed_length))
            padded[:, :current_length] = piano_roll
            return padded
        else:
            return piano_roll[:, :fixed_length]
    except Exception as e:
        print(f"Error processing {midi_path}: {e}")
        return None
```

```
#previous dataset append function - not used in final
piano_roll = get_piano_roll(midi_path, hz, pitchrange)
if piano_roll is not None:
    new_row = pd.DataFrame({
        'song': [piano_roll],
        'composer_id': [composer_id]
    })
    main_music = pd.concat([main_music, new_row], ignore_index=True)
print(main_music)
print("\nDataFrame Info:")
print(main_music.dtypes)
print("\nShape of first piano roll:", main_music['song'][0].shape if not main_music.empty else "No data")
print(main_music.head())
```

```
                                               song composer_id
0  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...           1

DataFrame Info:
song           object
composer_id    object
dtype: object

Shape of first piano roll: (128, 128)
                                               song composer_id
```

```
     0  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...           1
```

```
#define function to evaluate final model
#function created by LLM (xAI, 2025)
def evaluate_model(model, X_test, y_test, model_name):
    y_pred = model.predict(X_test)
    y_pred_classes = np.argmax(y_pred, axis=1)
    y_test_classes = np.argmax(y_test, axis=1)
    accuracy = accuracy_score(y_test_classes, y_pred_classes)
    precision = precision_score(y_test_classes, y_pred_classes, average='weighted')
    recall = recall_score(y_test_classes, y_pred_classes, average='weighted')
    print(f"{model_name} Performance:")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print()
    return accuracy, precision, recall
```

```
#establish data file path, specifiy piano roll hyper param, initialize list for
#data, iterate through directory and convert midi files to piano rolls
#bach data
data_dir = '/content/drive/My Drive/Bach'
data = []
fs = 100
fixed_length = 4000
compid = 1
for file_name in os.listdir(data_dir):
    midi_path = os.path.join(data_dir, file_name)
    piano_roll = get_piano_roll(midi_path, fs, fixed_length)
    if piano_roll is not None:
        composer_id = compid
        data.append({'song': piano_roll, 'composer_id': composer_id})
```

```
#establish data file path, specifiy piano roll hyper param, initialize list for
#data, iterate through directory and convert midi files to piano rolls
#beethoven data
data_dir = '/content/drive/My Drive/Beethoven'
fs = 100
fixed_length = 4000
compid = 2
max_entries = 20
for file_name in os.listdir(data_dir):
    midi_path = os.path.join(data_dir, file_name)
    piano_roll = get_piano_roll(midi_path, fs, fixed_length)
    if piano_roll is not None:
        composer_id = compid
        data.append({'song': piano_roll, 'composer_id': composer_id})
    if len(data) >= max_entries:  # Stop after collecting 10 valid entries
        break
```

```
#establish data file path, specifiy piano roll hyper param, initialize list for
#data, iterate through directory and convert midi files to piano rolls
#chopin data
data_dir = '/content/drive/My Drive/Chopin'
fs = 100
fixed_length = 4000
compid = 3
max_entries = 30
for file_name in os.listdir(data_dir):
    midi_path = os.path.join(data_dir, file_name)
    piano_roll = get_piano_roll(midi_path, fs, fixed_length)
    if piano_roll is not None:
        composer_id = compid
        data.append({'song': piano_roll, 'composer_id': composer_id})
    if len(data) >= max_entries:  # Stop after collecting 10 valid entries
        break
```

```
/usr/local/lib/python3.11/dist-packages/pretty_midi/pretty_midi.py:100: RuntimeWarning: Tempo, Key or Time signature change
  warnings.warn(
```

```
#establish data file path, specifiy piano roll hyper param, initialize list for
#data, iterate through directory and convert midi files to piano rolls
#mozart data
```

```
data_dir = '/content/drive/My Drive/Mozart'
fs = 100
fixed_length = 4000
compid = 4
max_entries = 40
for file_name in os.listdir(data_dir):
    midi_path = os.path.join(data_dir, file_name)
    piano_roll = get_piano_roll(midi_path, fs, fixed_length)
    if piano_roll is not None:
        composer_id = compid
        data.append({'song': piano_roll, 'composer_id': composer_id})
    if len(data) >= max_entries:  # Stop after collecting 10 valid entries
        break


#inspect dataframe
main_music = pd.DataFrame(data)


#inspect/evaluate dataframe
print(main_music)
print(main_music.dtypes)
```

```
                                                 song  composer_id
0   [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            1
1   [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            1
2   [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            1
3   [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            1
4   [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            1
5   [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            1
6   [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            1
7   [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            1
8   [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            1
9   [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            1
10  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            2
11  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            2
12  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            2
13  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            2
14  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            2
15  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            2
16  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            2
17  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            2
18  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            2
19  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            2
20  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            3
21  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            3
22  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            3
23  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            3
24  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            3
25  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            3
26  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            3
27  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            3
28  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            3
29  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            3
30  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            4
31  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            4
32  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            4
33  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            4
34  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            4
35  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            4
36  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            4
37  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            4
38  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            4
39  [[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...            4
song           object
composer_id     int64
dtype: object
```

```
#define X and Y vars for training
#one-hot encode y vars for training clarity
x = np.array(main_music['song'].tolist())
y = main_music['composer_id'].values - 1
y = to_categorical(y, num_classes=4)
#investigate format
print(y)
print(x.shape)
```

```
[[1. 0. 0. 0.]
 [1. 0. 0. 0.]
 [1. 0. 0. 0.]
 [1. 0. 0. 0.]
```

```
      [1. 0. 0. 0.]
      [1. 0. 0. 0.]
      [1. 0. 0. 0.]
      [1. 0. 0. 0.]
      [1. 0. 0. 0.]
      [1. 0. 0. 0.]
      [0. 1. 0. 0.]
      [0. 1. 0. 0.]
      [0. 1. 0. 0.]
      [0. 1. 0. 0.]
      [0. 1. 0. 0.]
      [0. 1. 0. 0.]
      [0. 1. 0. 0.]
      [0. 1. 0. 0.]
      [0. 1. 0. 0.]
      [0. 1. 0. 0.]
      [0. 0. 1. 0.]
      [0. 0. 1. 0.]
      [0. 0. 1. 0.]
      [0. 0. 1. 0.]
      [0. 0. 1. 0.]
      [0. 0. 1. 0.]
      [0. 0. 1. 0.]
      [0. 0. 1. 0.]
      [0. 0. 1. 0.]
      [0. 0. 1. 0.]
      [0. 0. 0. 1.]
      [0. 0. 0. 1.]
      [0. 0. 0. 1.]
      [0. 0. 0. 1.]
      [0. 0. 0. 1.]
      [0. 0. 0. 1.]
      [0. 0. 0. 1.]
      [0. 0. 0. 1.]
      [0. 0. 0. 1.]
      [0. 0. 0. 1.]]
     (40, 128, 4000)
```

```
#split train/test data
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=42)
```

```
#format data for CNN expansion (additional channel for CNN format)
#both training and test data
xtraincnn = np.expand_dims(xtrain, axis=-1)
xtestcnn = np.expand_dims(xtest, axis=-1)
```

```
#initialize CNN model, sequential with multiple layers and activation functions
cnn_model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(xtrain.shape[1], xtrain.shape[2], 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(4, activation='softmax')
])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do not pass an `input_
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
#initialize LSTM model, sequential with multiple layers and activation functions
lstm_model = Sequential([
    LSTM(128, input_shape=(xtrain.shape[1], xtrain.shape[2]), return_sequences=True),
    LSTM(64),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(4, activation='softmax')
])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWarning: Do not pass an `input_shape`/`input_di
  super().__init__(**kwargs)
```

```
#compile models with adam optimizer and CCentropy
#include epoch accuracy metric for real time data monitoring
cnn_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
lstm_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
cnn_model.fit(xtraincnn, ytrain, epochs=10, batch_size=32, validation_split=0.2)
lstm_model.fit(xtrain, ytrain, epochs=10, batch_size=32, validation_split=0.2)
```

```
Epoch 1/10
```

```
#evaluation metrics CNN
cnn_accuracy, cnn_precision, cnn_recall = evaluate_model(cnn_model, xtestcnn, ytest, "CNN Model")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 529ms/step
CNN Model Performance:
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
```

```
#evaluation metrics LSTM
lstm_accuracy, lstm_precision, lstm_recall = evaluate_model(lstm_model, xtest, ytest, "LSTM Model")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 800ms/step
LSTM Model Performance:
Accuracy: 0.0000
Precision: 0.0000
Recall: 0.0000

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-de
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall is ill-defin
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
#hyperparam adjustments and retraining for CNN:
from tensorflow.keras.layers import BatchNormalization, Activation, AveragePooling2D
cnn_model2 = Sequential([
    Conv2D(32, (3, 3), input_shape=(xtrain.shape[1], xtrain.shape[2], 1)),
    BatchNormalization(),
    Activation('relu'),
    AveragePooling2D((2, 2)),
    Conv2D(64, (3, 3)),
    BatchNormalization(),
    Activation('relu'),
    AveragePooling2D((2, 2)),
    Conv2D(128, (3, 3)),
    BatchNormalization(),
    Activation('relu'),
    AveragePooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(4, activation='softmax')
])
```

```
#adjusted epochs 10->20
cnn_model2.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
cnn_model2.fit(xtraincnn, ytrain, epochs=20, batch_size=32, validation_split=0.2)
cnn_accuracy, cnn_precision, cnn_recall = evaluate_model(cnn_model2, xtestcnn, ytest, "CNN Model 2")
```

```
#hyperparam adjustments and retraining for LSTM:
lstm_model2 = Sequential([
    LSTM(128, input_shape=(xtrain.shape[1], xtrain.shape[2]),
         return_sequences=True, dropout=0.3),
    LSTM(128, dropout=0.3),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(4, activation='softmax')
])
```

```
#adjusted epochs 10->20
lstm_model2.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
lstm_model2.fit(xtrain, ytrain, epochs=20, batch_size=32, validation_split=0.2)
lstm_accuracy, lstm_precision, lstm_recall = evaluate_model(lstm_model2, xtest, ytest, "LSTM Model 2")
```