InstruNet AI

Milestone – 2

Train a CNN model to Classify Instruments:

Libraries used:

- Tensorflow
- keras

Logic used:

- The dataset used is “Musical Instrument Sounds for classification” which is available on Kaggle.
- So, the data we have converted to mel-spectrogram images are used in this milestone to train the CNN model.
- The CNN model has two layers of convolutional layers and max pooling layers and the activation function used for the neural network is ‘relu’ and at last dense layer is ‘softmax’.
- The loss function used is ‘sparse_categorical_crossentropy’ and then we compile the model and use model.fit().

Code:

```
import tensorflow as tf

from keras import layers, models

from google.colab import files


uploaded = files.upload()

import numpy as np


data = np.load("instrument_melspec_dataset.npz", allow_pickle=True)


X_train = data["X_train"]

Y_train = data["Y_train"]

X_test  = data["X_test"]
```

```
Y_test  = data["Y_test"]
label_map = data["label_map"].item()

print(X_train.shape, Y_train.shape)
print(label_map)

X_train = np.expand_dims(X_train, axis=-1)
X_test = np.expand_dims(X_test, axis=-1)

print(X_train.shape)
print(X_test.shape)
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(128,128,1)),
    layers.BatchNormalization(),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(64, (3,3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(128, (3,3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(2,2),

    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.3),

    layers.Dense(28, activation='softmax')
])
model.compile(
```

```
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
model.fit(X_train,Y_train, epochs=10,validation_data=(X_test,Y_test))
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true_classes = np.argmax(Y_test, axis=1)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_true_classes, y_pred_classes)
print(cm)
test_loss, test_accuracy = model.evaluate(X_test, Y_test, verbose=0)
print(f"Test Loss: {test_loss:.4f}")
print(f"Test Accuracy: {test_accuracy:.4f}")
model.save("instrument_cnn_model.keras")
```

Outcomes:

Accuracy of the model: 97.27%

Loss function of the model: 0.1211

Confusion matrix: