CNN model 1

|  |
| --- |
| train\_path = 'data/training/train' test\_path = 'data/testing/test' pred\_path = 'data/prediction'  IMAGE\_SIZE = (224, 224) BATCH\_SIZE = 32 EPOCHS = 30  train\_datagen = ImageDataGenerator(  rotation\_range=20,  width\_shift\_range=0.1,  height\_shift\_range=0.1,  shear\_range=0.1,  zoom\_range=0.1,  horizontal\_flip=True,  fill\_mode='nearest', )  test\_datagen = ImageDataGenerator()  train\_ds = train\_datagen.flow\_from\_directory(  train\_path,  target\_size=IMAGE\_SIZE,  batch\_size=BATCH\_SIZE,  color\_mode='grayscale',  class\_mode='categorical' )  # Define the model architecture model = Sequential()  # Normalization model.add(Rescaling(1. / 255, input\_shape=(224, 224, 1)))  # Convolutional layers  model.add(Conv2D(32, kernel\_size=(5, 5), activation='relu')) model.add(BatchNormalization()) model.add(MaxPooling2D(2, 2))  model.add(Conv2D(64, kernel\_size=(5, 5), activation='relu')) model.add(BatchNormalization()) model.add(MaxPooling2D(2, 2))  model.add(Conv2D(128, kernel\_size=(3, 3), activation='relu')) model.add(BatchNormalization()) model.add(MaxPooling2D(2, 2))  model.add(Conv2D(256, kernel\_size=(3, 3), activation='relu')) model.add(BatchNormalization()) model.add(MaxPooling2D(2, 2))  # Flatten and dense layers with Dropout model.add(Flatten()) model.add(Dense(units=512, activation='relu')) model.add(Dropout(0.4)) model.add(Dense(units=128, activation='relu')) model.add(Dropout(0.4))  # Output layer model.add(Dense(units=len(class\_names), activation='softmax'))  # Compile model model.compile(optimizer=tf.keras.optimizers.Adam(learning\_rate=0.001),   loss='categorical\_crossentropy',   metrics=['accuracy'])  # Learning Rate Reduction reduce\_lr = ReduceLROnPlateau(  monitor='val\_loss',  factor=0.5,  patience=2,  min\_lr=1e-5 )  # Early stopping early\_stopping = EarlyStopping(  monitor='val\_loss',  patience=5,  restore\_best\_weights=True,  min\_delta=0.001,  mode='min' )  # Train model history = model.fit(  train\_ds,  validation\_data=test\_ds,  epochs=EPOCHS,  callbacks=[early\_stopping, reduce\_lr] )  # Save model model.save('output/model/cnn\_model\_v1.h5') |

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

