

```

*****
# Simple CNN model
*****
def build_model_simple(input_shape, num_classes):
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
        MaxPooling2D((2, 2)),
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Conv2D(128, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(num_classes, activation='softmax')
    ])

    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    return model

*****
# ResNet50 model
*****
def build_model_resnet50(input_shape, num_classes):
    base_model = ResNet50(weights='imagenet', include_top=False, input_shape=input_shape)
    base_model.trainable = False #freeze the base model

    model = Sequential([
        base_model,
        GlobalAveragePooling2D(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(num_classes, activation='softmax')
    ])

    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    return model

```

```

#####
# Data Generator - preprocess the data
#####

def create_data_generator(train_dir, valid_dir, test_dir, batch_size=32, image_size=(416, 416)):
    # Data augmentation for training
    train_datagen = ImageDataGenerator(
        rescale=1./255,
        horizontal_flip=True,
        rotation_range=20,
        zoom_range=0.1,
        brightness_range=[0.7, 1.3],
        width_shift_range=0.1,
        height_shift_range=0.1,
    )

    # Only rescaling for validation and test data
    val_test_datagen = ImageDataGenerator(rescale=1./255)

    train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=image_size,
        batch_size=batch_size,
        class_mode='categorical',
        shuffle=True,
    )

    val_generator = val_test_datagen.flow_from_directory(
        valid_dir,
        target_size=image_size,
        batch_size=batch_size,
        class_mode='categorical',
        shuffle=False,
    )

    test_generator = val_test_datagen.flow_from_directory(
        test_dir,
        target_size=image_size,
        batch_size=batch_size,
        class_mode='categorical',
        shuffle=False,
    )

    return train_generator, val_generator, test_generator

```

```
#####  
# Training the ResNet50 model  
#####
```

```
history = restNet50_model.fit(train_gen, epochs=10, validation_data=val_gen)  
print("Done")
```

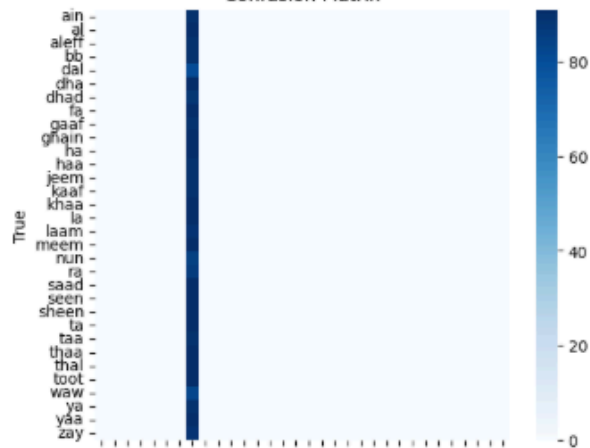
```
Epoch 1/10  
310/310 ————— 1738s 6s/step - accuracy: 0.0302 - loss: 3.4656 - val_accuracy: 0.0317 - val_loss: 3.4655  
Epoch 2/10  
310/310 ————— 1660s 5s/step - accuracy: 0.0323 - loss: 3.4655 - val_accuracy: 0.0317 - val_loss: 3.4654  
Epoch 3/10  
310/310 ————— 1706s 6s/step - accuracy: 0.0316 - loss: 3.4657 - val_accuracy: 0.0317 - val_loss: 3.4654  
Epoch 4/10  
310/310 ————— 1712s 6s/step - accuracy: 0.0324 - loss: 3.4654 - val_accuracy: 0.0317 - val_loss: 3.4654  
Epoch 5/10  
310/310 ————— 1734s 6s/step - accuracy: 0.0259 - loss: 3.4656 - val_accuracy: 0.0317 - val_loss: 3.4654  
Epoch 6/10  
310/310 ————— 1672s 5s/step - accuracy: 0.0310 - loss: 3.4658 - val_accuracy: 0.0317 - val_loss: 3.4654  
Epoch 7/10  
310/310 ————— 1691s 5s/step - accuracy: 0.0334 - loss: 3.4658 - val_accuracy: 0.0317 - val_loss: 3.4654  
Epoch 8/10  
310/310 ————— 1878s 6s/step - accuracy: 0.0279 - loss: 3.4658 - val_accuracy: 0.0317 - val_loss: 3.4654  
Epoch 9/10  
310/310 ————— 1824s 6s/step - accuracy: 0.0332 - loss: 3.4659 - val_accuracy: 0.0317 - val_loss: 3.4654  
Epoch 10/10  
310/310 ————— 1863s 6s/step - accuracy: 0.0268 - loss: 3.4655 - val_accuracy: 0.0317 - val_loss: 3.4654  
Done
```

classification Report:

	precision	recall	f1-score	support
ain	0.00	0.00	0.00	90
al	0.00	0.00	0.00	91
aleff	0.00	0.00	0.00	90
bb	0.00	0.00	0.00	90
dai	0.00	0.00	0.00	81
dha	0.00	0.00	0.00	91
dhad	0.00	0.00	0.00	88
fa	0.03	1.00	0.06	91
gaaf	0.00	0.00	0.00	90
ghain	0.00	0.00	0.00	91
ha	0.00	0.00	0.00	91
haa	0.00	0.00	0.00	90
jeem	0.00	0.00	0.00	90
kaaf	0.00	0.00	0.00	90
khaa	0.00	0.00	0.00	91
la	0.00	0.00	0.00	91
laam	0.00	0.00	0.00	89
meem	0.00	0.00	0.00	91
nun	0.00	0.00	0.00	85
ra	0.00	0.00	0.00	86
saad	0.00	0.00	0.00	91
seen	0.00	0.00	0.00	91
sheen	0.00	0.00	0.00	91
ta	0.00	0.00	0.00	91
taa	0.00	0.00	0.00	90
thaa	0.00	0.00	0.00	91
thal	0.00	0.00	0.00	91
toot	0.00	0.00	0.00	91
waw	0.00	0.00	0.00	83
ya	0.00	0.00	0.00	90
yaa	0.00	0.00	0.00	91
zay	0.00	0.00	0.00	89
accuracy			0.03	2867
macro avg	0.00	0.03	0.00	2867
weighted avg	0.00	0.03	0.00	2867

C:\Users\Administrator\anaconda3\envs\Practical\Lib\site-packages\sklearn\metrics_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this behavior.
_warn_prf(average, modifier, f'{metric.capitalize()} is', len(result))
C:\Users\Administrator\anaconda3\envs\Practical\Lib\site-packages\sklearn\metrics_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this behavior.
_warn_prf(average, modifier, f'{metric.capitalize()} is', len(result))
C:\Users\Administrator\anaconda3\envs\Practical\Lib\site-packages\sklearn\metrics_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this behavior.
_warn_prf(average, modifier, f'{metric.capitalize()} is', len(result))
C:\Users\Administrator\anaconda3\envs\Practical\Lib\site-packages\sklearn\metrics_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this behavior.
_warn_prf(average, modifier, f'{metric.capitalize()} is', len(result))

Confusion Matrix



```
#####
# test data gen for resnet50
#####
def create_improved_data_generator(train_dir, valid_dir, test_dir, batch_size=32, image_size=(224, 224)):

    from tensorflow.keras.applications.resnet50 import preprocess_input

    # Data augmentation for training
    train_datagen = ImageDataGenerator(
        preprocessing_function=preprocess_input, # ImageNet preprocessing
        horizontal_flip=True,
        rotation_range=15,
        zoom_range=0.1,
        width_shift_range=0.1,
        height_shift_range=0.1,
        shear_range=0.1,
        brightness_range=[0.8, 1.2],

    )

    # Only preprocessing for validation and test data
    val_test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

    train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=image_size,
        batch_size=batch_size,
        class_mode='categorical',
        shuffle=True,
    )

    val_generator = val_test_datagen.flow_from_directory(
        valid_dir,
        target_size=image_size,
        batch_size=batch_size,
        class_mode='categorical',
        shuffle=False,
    )

    test_generator = val_test_datagen.flow_from_directory(
        test_dir,
        target_size=image_size,
        batch_size=batch_size,
        class_mode='categorical',
        shuffle=False,
    )

```

```
input_shape = (224,224,3)
restNet50_model = build_model_resnet50(input_shape, num_classes)

history = restNet50_model.fit(improved_train_gen, epochs=4, validation_data=improved_val_gen)
print("Done")
```

```
Epoch 1/4
620/620 ————— 558s 888ms/step - accuracy: 0.0350 - loss: 3.6595 - val_accuracy: 0.0408 - val_loss: 3.4418
Epoch 2/4
620/620 ————— 576s 930ms/step - accuracy: 0.0400 - loss: 3.4542 - val_accuracy: 0.0662 - val_loss: 3.4006
Epoch 3/4
620/620 ————— 545s 879ms/step - accuracy: 0.0552 - loss: 3.4150 - val_accuracy: 0.0951 - val_loss: 3.3203
Epoch 4/4
620/620 ————— 566s 912ms/step - accuracy: 0.0739 - loss: 3.3605 - val_accuracy: 0.1246 - val_loss: 3.2319
Done
```

```
# another 3 epochs
history = restNet50_model.fit(improved_train_gen, epochs=4, validation_data=improved_val_gen)
print("Done")
```

```
Epoch 1/4
620/620 ————— 551s 889ms/step - accuracy: 0.0876 - loss: 3.2984 - val_accuracy: 0.1549 - val_loss: 3.0985
Epoch 2/4
620/620 ————— 574s 925ms/step - accuracy: 0.1086 - loss: 3.2192 - val_accuracy: 0.1592 - val_loss: 3.0069
Epoch 3/4
339/620 ————— 4:09 890ms/step - accuracy: 0.1211 - loss: 3.1405
```

```
# evaluate the ResNet50 model
```