```
training_set = train_datagen.flow_from_directory(train_dir, target_size=(224, 224), batch_size=32, subset="training", class_mode="categorical")
```

```
Found 744 images belonging to 9 classes.
```

```
validation_set = train_datagen.flow_from_directory(train_dir, target_size=(224, 224), batch_size=32, subset="validation", class_mode="categorical")
```

```
Found 181 images belonging to 9 classes.
```

```
test_set = test_datagen.flow_from_directory(test_dir, target_size=(224, 224), batch_size=32, class_mode="categorical")
```

```
Found 237 images belonging to 9 classes.
```

```
237 + 181 + 744
```

```
1162
```

Number of pictures matches the one stated in dataset, so no data loss

Build Custom CNN

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPool2D, GlobalAveragePooling2D, Dense, Dropout, BatchNormalization, Flatten
```

```
model = Sequential()
```

```
model.add(Conv2D(name="cnv_layer_1", filters=32, kernel_size=3, activation="relu", input_shape=[224,224,3], padding="same"))
```

```
model.add(MaxPool2D(pool_size=2, name="max_pool_layer_1"))
```

```
model.add(Conv2D(name="cnv_layer_2", filters=32, kernel_size=3, activation="relu", input_shape=[112,112,3], padding="same"))
```

```
model.add(MaxPool2D(pool_size=2, name="max_pool_layer_2"))
```

```
model.add(GlobalAveragePooling2D())
```

```
model.add(Dense(units=128, name="dense_layer_1", activation="relu"))
```

```
model.add(Dense(units=128, name="dense_layer_2", activation="relu"))
```

```
model.add(Dense(units=9, name="output", activation="softmax"))
```

```
precision = tf.metrics.Precision()
recall = tf.metrics.Recall()
```

```
model.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy", precision, recall])
```

```
= model.fit(training_set, validation_data=validation_set, epochs=20)
```

```
Epoch 1/20
```

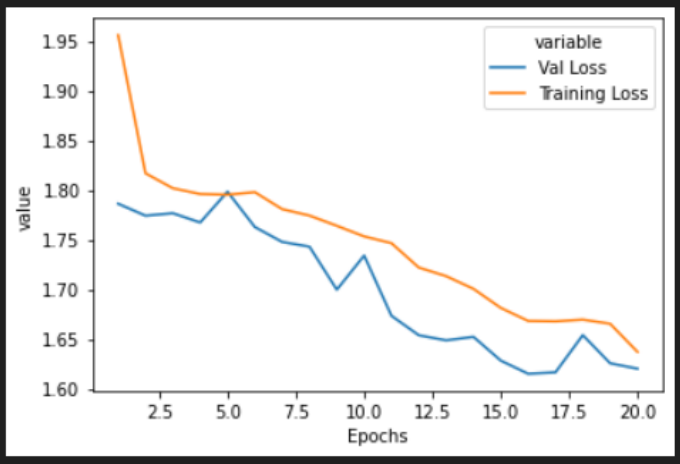
```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
training_loss = g.history["loss"]
val_loss = g.history["val_loss"]

training_acc = g.history["accuracy"]
val_acc = g.history["val_accuracy"]

epochs = [x + 1 for x in range(len(training_loss))]
```

```
lossdf1 = pd.DataFrame({"Epochs" : epochs, "Val Loss" : val_loss, "Training Loss" : training_loss})
accdf1 = pd.DataFrame({"Epochs" : epochs, "Val Acc" : val_acc, "Training Acc" : training_acc})
```

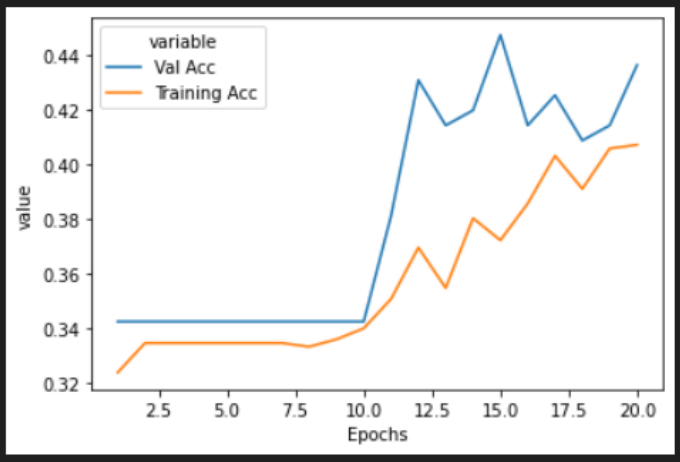
```
sns.lineplot(x="Epochs", y="value", hue="variable", data=pd.melt(lossdf1, ["Epochs"]))
```

```
<AxesSubplot: xlabel='Epochs', ylabel='value'>
```




```
sns.lineplot(x="Epochs", y="value", hue="variable", data=pd.melt(accdf1, ["Epochs"]))
```

```
<AxesSubplot: xlabel='Epochs', ylabel='value'>
```



```
predictions = model.predict(test_set)
```

```
8/8 [==============================] - 6s 739ms/step
```

```
predicted_classes = np.argmax(predictions, axis=1)
true_classes = test_set.classes
```

```
metrics = model.evaluate(test_set)
```

```
8/8 [==============================] - 4s 538ms/step - loss: 1.6767 - accuracy: 0.3713 - precision: 0.6087 - recall: 0.0591
```

```
print(f"loss {metrics[0]} \naccuracy {metrics[1]}")
```

```
loss 1.6767058372497559
accuracy 0.37130802869796753
```

```
from sklearn.metrics import confusion_matrix, classification_report
```

```
conf_matrix = confusion_matrix(true_classes, predicted_classes)
class_report = classification_report(true_classes, predicted_classes)

plt.figure(figsize=(10,8))
plt.imshow(conf_matrix, cmap="Blues", interpolation="nearest")
plt.title('Confusion Matrix')
plt.colorbar()
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()


print("Classification Report")
print(class_report)
```

```
train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.3, zoom_range=0.3, horizontal_flip=True, validation_split=0.2)
test_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.3, zoom_range=0.3, horizontal_flip=True)
```

```
training_set = train_datagen.flow_from_directory(train_dir, target_size=(224, 224), batch_size=32, subset="training", class_mode="categorical")
validation_set = train_datagen.flow_from_directory(train_dir, target_size=(224, 224), batch_size=32, subset="validation", class_mode="categorical")
test_set = test_datagen.flow_from_directory(test_dir, target_size=(224, 224), batch_size=32, class_mode="categorical")
```

```
Found 652 images belonging to 9 classes.
Found 159 images belonging to 9 classes.
Found 351 images belonging to 9 classes.
```

```
652 + 159 + 351
```

```
1162
```

Transfer Model

```
mbnet = MobileNetV2(input_shape=(224,224,3), include_top=False)
```

```
mbnet.trainable = False
```

```
trfMod = Sequential()
```

```
trfMod.add(mbnet)
```

```
trfMod.add(GlobalAveragePooling2D())
```

```
trfMod.add(Dropout(0.2))
```

```
trfMod.add(Dense(units=256,
                 name="dense_layer_1",
                 activation="relu"))
```

```
trfMod.add(BatchNormalization(axis=1))
```

```
trfMod.add(Dropout(0.1))
```

```
trfMod.add(Dense(units=128,
                 name="dense_layer_2",
                 activation="relu"))
```

```
trfMod.add(BatchNormalization(axis=1))
```

```
trfMod.add(Dropout(0.1))
```

```
trfMod.add(Dense(units=9, name="output", activation="softmax"))
```

```
trfMod.summary()
```

```
trfMod.compile(optimizer="adam",
               loss="categorical_crossentropy",
               metrics=["accuracy", precision, recall])
```

```
from tensorflow.keras.callbacks import EarlyStopping
```

```
monitor = EarlyStopping(monitor = "val_loss", min_delta = 1e-3, patience = 5, verbose = 1, mode = "auto", restore_best_weights = True)
```

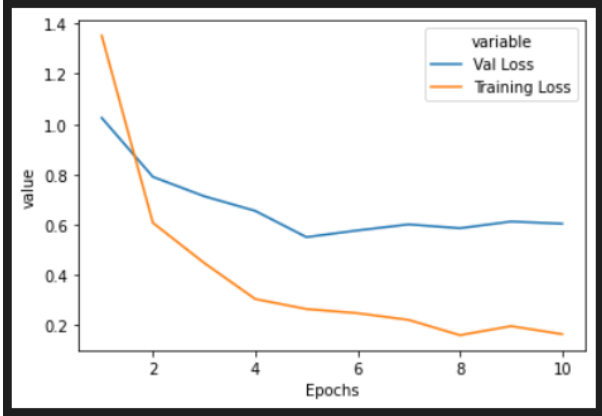
```
history = trfMod.fit(training_set, validation_data=validation_set, epochs=50, callbacks=[monitor])
```

```
training_loss = history.history["loss"]
val_loss = history.history["val_loss"]

training_acc = history.history["accuracy"]
val_acc = history.history["val_accuracy"]

epochs = [x + 1 for x in range(len(training_loss))]
```

```
lossdf = pd.DataFrame({"Epochs" : epochs, "Val Loss" : val_loss, "Training Loss" : training_loss})
accdf = pd.DataFrame({"Epochs" : epochs, "Val Acc" : val_acc, "Training Acc" : training_acc})
```

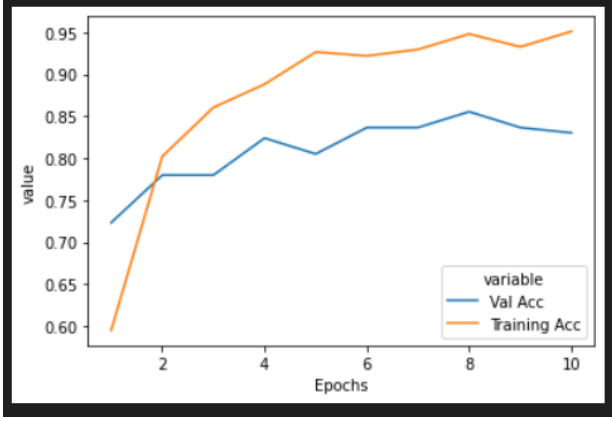
```
sns.lineplot(x="Epochs", y="value", hue="variable", data=pd.melt(lossdf, ["Epochs"]))
```

```
<AxesSubplot: xlabel='Epochs', ylabel='value'>
```




```
sns.lineplot(x="Epochs", y="value", hue="variable", data=pd.melt(accdf, ["Epochs"]))
```

```
<AxesSubplot: xlabel='Epochs', ylabel='value'>
```



```
predictions = model.predict(test_set)
```

```
11/11 [==============================] - 6s 575ms/step
```

```
predicted_classes = np.argmax(predictions, axis=1)
true_classes = test_set.classes
```

```
metrics = model.evaluate(test_set)
```

```
11/11 [==============================] - 6s 557ms/step - loss: 1.6290 - accuracy: 0.4387 - precision: 0.6316 - recall: 0.0684
```

```
print(f"loss {metrics[0]} \naccuracy {metrics[1]}")
```

```
loss 1.6290180683135986
accuracy 0.43874645233154297
```

```
conf_matrix = confusion_matrix(true_classes, predicted_classes)
class_report = classification_report(true_classes, predicted_classes)

plt.figure(figsize=(10,8))
plt.imshow(conf_matrix, cmap="Blues", interpolation="nearest")
plt.title('Confusion Matrix')
plt.colorbar()
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()


print("Classification Report")
print(class_report)
```