MODEL BUILDING

Train The Model

Date	04 November 2022
Team ID	PNT2022TMID13480
Project Name	Emerging methods for the early detection of forest fires

Model training is the phase in the data science development lifecycle where practitioners try to fit the best combination of weights and bias to a machine learning algorithm to minimize a loss function over the prediction range.

- ✞ At this point, we have training data and a fully configured neural network to train.
- ✞ All that is left is to pass the data to the model for the training process to commence, a process that is completed by iterating on the training data.
- ✞ Training begins by calling the fit ( ) method.

The arguments are the batch size as you are using "adam" (bath gradient descent and epochs: no: of times the model should get trained).

- **steps_per_epoch:**

  It specifies the total number of steps taken from the generator as soon as one epoch is finished and the next epoch has started.
  We can calculate the value of steps_per_epoch as the total number of samples in your training folder divided by the batch size.

- **Epochs:**

  It an integer and number of epochs we want to train our model for.

- **Validation_data:**

  It can be either input and targets list generator inputs, targets, and sample_weights list which can be used to evaluate.

The loss and metrics for any model after any epoch has ended.

- **Validation_steps:**

  Only if the validation_data is a generator then only this argument can be used.
  It specifies the total number of steps taken from the generator before it is stopped at every epoch and its value is calculated as the total number of validation data points in your dataset divided by the validation batch size.

Your model will likely perform better when trained on all of the available data than just the subset used to estimate the performance of the model. This is why we prefer to train the final model on all available data.
We can continuously train a machine learning model in multiple ways. Incremental training - training the model with new data as the data comes in. Batch training - training the model once a significant amount of new data is available.

Train the model

```
model.fit_generator(x_train,steps_per_epoch=14,epochs=
10,validation_da ta=x_test,validation_steps=4)
```

```
Epoch 1/10
14/14 [==============================] - 173s
12s/step - loss: 2.1876
- accuracy: 0.7592 - val_loss: 0.0665 - val_accuracy:
0.9917 Epoch 2/10
14/14 [==============================] - 27s
2s/step - loss: 0.4629 - accuracy:
0.8761 - val_loss: 0.1060 -
val_accuracy: 0.9669 Epoch 3/10
14/14 [==============================] - 30s
2s/step - loss: 0.2542 - accuracy:
0.9106 - val_loss: 0.1746 -
val_accuracy: 0.9256 Epoch 4/10
```

```
14/14 [==============================] - 27s
2s/step - loss: 0.1984 - accuracy:
0.9266 - val_loss: 0.0767 -
val_accuracy: 0.9752 Epoch 5/10
14/14 [==============================] - 30s
2s/step - loss: 0.2195 - accuracy:
0.9174 - val_loss: 0.0795 -
val_accuracy: 0.9669 Epoch 6/10
14/14 [==============================] - 28s
2s/step - loss: 0.1656 - accuracy:
0.9312 - val_loss: 0.0541 -
val_accuracy: 0.9752 Epoch 7/10
14/14 [==============================] - 26s
2s/step - loss: 0.1576 - accuracy:
0.9404 - val_loss: 0.0618 -
val_accuracy: 0.9752 Epoch 8/10
14/14 [==============================] - 28s
2s/step - loss: 0.1690 - accuracy:
0.9289 - val_loss: 0.0498 -
val_accuracy: 0.9752 Epoch 9/10
 14/14   0.18 - [==============================]
 - 26s 58
 2s/step - loss:
accuracy: 0.9381 - val_loss: 0.0726 -
 val_accuracy: 0.9752
Epoch 10/10
14/14                                          0.18 -
 [==============================] - 27s 35
 2s/step - loss:
```

```
accuracy: 0.9151 - val_loss: 0.0846 -
 val_accuracy: 0.9504
<keras.callbacks.History at 0x7ff6344410d0>
```