Model Development Phase Template

Date	7 JULY 2024
Team ID	SWTID1720110768
Project Name	Covidvision: Advanced Covid-19 Detection From Lung X-rays With Deep Learning
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation And Evaluation Report:

The **initial model training code** for **CovidVision** sets up a CNN-based framework for detecting Covid-19 from lung X-rays, while the **model validation and evaluation report** details performance metrics like accuracy and F1 score to assess the model's effectiveness and compare it against existing methods.

Initial Model Training Code: Implements a CNN-based approach for Covid-19 detection from lung X-rays.

Model Validation and Evaluation Report: Analyzes performance metrics to gauge effectiveness and benchmark against existing solutions.

Data Preparation:

```
def __getitem__(self, index):
        batch_indices_range = self.indices[index * self.batch_size:(index + 1) * self.batch_size]
        batch_indices_paths = [self.file_paths[i] for i in batch_indices_range]
        batch_indices_labels = [self.labels[i] for i in batch_indices_range]
        return self.data_generator(batch_indices_paths, batch_indices_labels)
```

Model Architecture:

```
cnn = models.Sequential([
    layers.Conv2D(filters=32, kernel_size=(3, 3), input_shape=(128, 128, 3), activation='relu'),
    layers.MaxPool2D((2, 2)),

    layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu'),
    layers.MaxPool2D((2, 2)),

    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dense(2, activation='softmax')
])
```

Model Training:

```
cnn.fit(training, epochs=15)
```

Model Validation :

Compile the model

```
cnn.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Evaluation Report :

```
class Image_generator(tf.keras.utils.Sequence):
    def __init__(self, file_paths, labels, image_size, batch_size):
        self.file_paths = file_paths
        self.labels = labels
        self.image_size = image_size
        self.batch_size = batch_size
        self.indices = np.arange(len(file_paths))
        np.random.shuffle(self.indices)
```

```
def __len__(self):
    return int(len(self.file_paths) / self.batch_size)
```