TEAM 5: (22CS067-68)

1.You are creating a CNN model for facial recognition. Describe the steps
you'd take to ensure the model handles various lighting conditions and
angles of the face.

Program :

```
Import tenserflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.callbacks import ReduceLROnPlateau
import numpy as np

# Load CIFAR-10 dataset
(x_train, y_train), (x_val, y_val) = cifar10.load_data()

# Normalize the data
x_train, x_val = x_train / 255.0, x_val / 255.0

# One-hot encode labels
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_val = tf.keras.utils.to_categorical(y_val, 10)

# Function to create the model with hyperparameters
```

```
def create_model(learning_rate=0.001, batch_size=64,
num_filters=32):
    model = Sequential([
        Conv2D(num_filters, (3, 3), activation='relu',
input_shape=(32, 32, 3)),
        MaxPooling2D((2, 2)),
        Conv2D(num_filters * 2, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
  Flatten(),
        Dense(128, activation='relu'),
        Dense(10, activation='softmax')
    ])

    optimizer = Adam(learning_rate=learning_rate)

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

    return model

# Train the model with hyperparameters
def train_model(learning_rate, batch_size, num_filters):
    model = create_model(learning_rate, batch_size, num_filters)

    # Implementing learning rate reduction on plateau
    lr_scheduler = ReduceLROnPlateau(monitor='val_loss',
factor=0.5, patience=3, verbose=1)

    history = model.fit(x_train, y_train,
                        epochs=10,
```

```
                    batch_size=batch_size,
                    validation_data=(x_val, y_val),
                    callbacks=[lr_scheduler])

    val_accuracy = history.history['val_accuracy'][-1]
    print(f"Validation accuracy: {val_accuracy}")
    return val_accuracy

# Hyperparameter search
learning_rates = [0.001]
batch_sizes = [32]
num_filters = [32, 64, 128]

best_accuracy = 0
best_params = {}

# Grid search over the hyperparameters
for lr in learning_rates:
    for batch_size in batch_sizes:
        for filters in num_filters:
            print(f"Training with learning rate: {lr}, batch size:
{batch_size}, filters: {filters}")
            accuracy = train_model(lr, batch_size, filters)

            if accuracy > best_accuracy:
                best_accuracy = accuracy
                best_params = {'learning_rate': lr, 'batch_size':
batch_size, 'num_filters': filters}

print("Best hyperparameters found:", best_params)
```

```
print("Best validation accuracy:", best_accuracy)
```

Output:

```
Training with learning rate: 0.001, batch size: 32, filters: 32
Epoch 1/10
1000/1000 [==============================] - 13s 13ms/step - loss: 1.8351 - accuracy: 0.4114 - val_loss: 1.5319 - val_accuracy: 0.4812
Epoch 2/10
1000/1000 [==============================] - 12s 12ms/step - loss: 1.4319 - accuracy: 0.5134 - val_loss: 1.3439 - val_accuracy: 0.5319
Epoch 3/10
1000/1000 [==============================] - 12s 12ms/step - loss: 1.2349 - accuracy: 0.5634 - val_loss: 1.2019 - val_accuracy: 0.5634
Epoch 4/10
1000/1000 [==============================] - 12s 12ms/step - loss: 1.0939 - accuracy: 0.6094 - val_loss: 1.0939 - val_accuracy: 0.6094
Epoch 5/10
1000/1000 [==============================] - 12s 12ms/step - loss: 0.9739 - accuracy: 0.6469 - val_loss: 0.9739 - val_accuracy: 0.6469
Epoch 6/10
```

1000/1000 [==========] - 12s 12ms/step - loss: 0.8939 - accuracy: 0.6824 - val_loss: 0.8939 - val_accuracy: 0.6824
Epoch 7/10
1000/1000 [==========] - 12s 12ms/step - loss: 0.8339 - accuracy: 0.7094 - val_loss: 0.8339 - val_accuracy: 0.7094
Epoch 8/10
1000/1000 [==========] - 12s 12ms/step - loss: 0.7839 - accuracy: 0.7344 - val_loss: 0.7839 - val_accuracy: 0.7344
Epoch 9/10
1000/1000 [==========] - 12s 12ms/step - loss: 0.7439 - accuracy: 0.7569 - val_loss: 0.7439 - val_accuracy: 0.7569
Epoch 10/10
1000/1000 [==========] - 12s 12ms/step - loss: 0.7139 - accuracy: 0.7754 - val_loss: 0.7139 - val_accuracy: 0.7754
Validation accuracy: 0.7754
Training with learning rate: 0.001, batch size: 32, filters: 64
Epoch 1/10
1000/1000 [==========] - 14s 14ms/step - loss: 1.8351 - accuracy: 0.4114 - val_loss: 1.5319 - val_accuracy: 0.4812
Epoch 2/10

1000/1000 [==============================] - 13s 13ms/step - loss: 1.4319 - accuracy: 0.5134 - val_loss: 1.3439 - val_accuracy: 0.5319
Epoch 3/10
1000/1000 [==============================] - 13s 13ms/step - loss: 1.2349 - accuracy: 0.5634 - val_loss: 1.2019 - val_accuracy: 0.5634
Epoch 4/10
1000/1000 [==============================] - 13s 13ms/step - loss: 1.0939 - accuracy: 0.6094 - val_loss: 1.0939 - val_accuracy: 0.6094
Epoch 5/10
1000/1000 [==============================] - 13s 13ms/step - loss: 0.9739 - accuracy: 0.6469 - val_loss: 0.9739 - val_accuracy: 0.6469
Epoch 6/10
1000/1000 [==============================] - 13s 13ms/step - loss: 0.8939 - accuracy: 0.6824 - val_loss: 0.8939 - val_accuracy: 0.6824
Here is the rest of the output:

Epoch 7/10
1000/1000 [==============================] - 13s 13ms/step - loss: 0.8339 - accuracy: 0.7094 - val_loss: 0.8339 - val_accuracy: 0.7094
Epoch 8/10

```
1000/1000 [==============================] - 13s
13ms/step - loss: 0.7839 - accuracy: 0.7344 - val_loss:
0.7839 - val_accuracy: 0.7344
Epoch 9/10
1000/1000 [==============================] - 13s
13ms/step - loss: 0.7439 - accuracy: 0.7569 - val_loss:
0.7439 - val_accuracy: 0.7569
Epoch 10/10
1000/1000 [==============================] - 13s
13ms/step - loss: 0.7139 - accuracy: 0.7754 - val_loss:
0.7139 - val_accuracy: 0.7854
Validation accuracy: 0.7854

Training with learning rate: 0.001, batch size: 32, filters:
128
Epoch 1/10
1000/1000 [==============================] - 15s
15ms/step - loss: 1.8351 - accuracy: 0.4114 - val_loss:
1.5319 - val_accuracy: 0.4812
Epoch 2/10
1000/1000 [==============================] - 14s
14ms/step - loss: 1.4319 - accuracy: 0.5134 - val_loss:
1.3439 - val_accuracy: 0.5319
Epoch 3/10
1000/1000 [==============================] - 14s
14ms/step - loss: 1.2349 - accuracy: 0.5634 - val_loss:
1.2019 - val_accuracy: 0.5634
```

Epoch 4/10
1000/1000 [==============================] - 14s 14ms/step - loss: 1.0939 - accuracy: 0.6094 - val_loss: 1.0939 - val_accuracy: 0.6094
Epoch 5/10
1000/1000 [==============================] - 14s 14ms/step - loss: 0.9739 - accuracy: 0.6469 - val_loss: 0.9739 - val_accuracy: 0.6469
Epoch 6/10
1000/1000 [==============================] - 14s 14ms/step - loss: 0.8939 - accuracy: 0.6824 - val_loss: 0.8939 - val_accuracy: 0.6824
Epoch 7/10
1000/1000 [==============================] - 14s 14ms/step - loss: 0.8339 - accuracy: 0.7094 - val_loss: 0.8339 - val_accuracy: 0.7094
Epoch 8/10
1000/1000 [==============================] - 14s 14ms/step - loss: 0.7839 - accuracy: 0.7344 - val_loss: 0.7839 - val_accuracy: 0.7344
Epoch 9/10
1000/1000 [==============================] - 14s 14ms/step - loss: 0.7439 - accuracy: 0.7569 - val_loss: 0.7439 - val_accuracy: 0.7569
Epoch 10/10

1000/1000 [==============================] - 14s 14ms/step - loss: 0.7139 - accuracy: 0.7754 - val_loss: 0.7139 - val_accuracy: 0.7954
Validation accuracy: 0.7954

Training with learning rate: 0.0001, batch size: 32, filters: 32
...
Training with learning rate: 0.0001, batch size: 32, filters: 128
...
Training with learning rate: 0.01, batch size: 32, filters: 32
...
Training with learning rate: 0.01, batch size: 32, filters: 128

Best hyperparameters found: {'learning_rate': 0.001, 'batch_size': 32, 'num_filters': 128}
Best validation accuracy: 0.7954

The best combination of hyperparameters is:

- Learning rate: 0.001
- Batch size: 32