**TEAM 20: (22CS097-98)**

**1.While training a CNN for image classification, your validation accuracy is not improving. How would you go about tuning hyperparameters like**

**learning rate, batch size, and number of filters?**

**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