Rapport

BENAHMED Firdaws sotra

1 . Data pre-processing :

I have found an issue while trying to load all the images at once in memory. Therefore, instead of loading all images into memory, i've used TensorFlow's `ImageDataGenerator,` which processes images in batches as needed.

```
datagen = ImageDataGenerator(

    rescale=1./255, # Normalization by dividing by 255

    validation_split=0.3 # Will be used for question 2 30% validation 70% training

)
```

The `rescale=1./255` parameter handles the normalization required

2 . split the dataset :

By using the stratified holdout strategy split the dataset into 70% for training and 30% for validation (Built-in Data Splitting)

```
# Training generator - pulls from training split (70%)
train_generator = train_datagen.flow_from_directory(
    data_dir,
    target_size=(IMG_HEIGHT,
    IMG_WIDTH), batch_size=BATCH_SIZE,
    class_mode='categorical',
    classes=classes,
    subset='training', # Use the training split
    shuffle=True,
    seed=42 # For reproducibility
)

# Validation generator - pulls from validation split (30%)
validation_generator = train_datagen.flow_from_directory(
    data_dir,
    target_size=(IMG_HEIGHT,
    IMG_WIDTH), batch_size=BATCH_SIZE,
    class_mode='categorical',
    classes=classes,
    subset='validation', # Use the validation split
    shuffle=False,
    seed=42 # Same seed ensures proper stratification)
```

3 . importing architecture with ImageNet pretrained weights :

importing MobileNetV2

```
IMG_HEIGHT, IMG_WIDTH = 224, 224
base_model = MobileNetV2(
    input_shape=(IMG_HEIGHT, IMG_WIDTH, 3),
    include_top=False, #to exclude the classification layers at the top(original) apres we will
use them
    weights='imagenet',
    alpha=1.0 # This is the width multiplier (x1.0)
)
```

4 . Load the pretrained weights from ImageNet (you can use the kerascv library) :

we basically did it in the qst 3 when we set `weights='imagenet'` when creating the model. This parameter tells TensorFlow to automatically download and use pre-trained ImageNet weights.

5. Remove and replace the last fully connected layers by three fully connected layers composed of 1024, 512, and c (Number of classes) neurons :

```
NUM_CLASSES = 4
# First, get the output tensor from the base model
x = base_model.output
# Add a Global Average Pooling layer to reduce the spatial dimensions
x = layers.GlobalAveragePooling2D()(x)
# the fully connected layers
x = layers.Dense(1024, activation='relu')(x)
x = layers.Dense(512, activation='relu')(x)
x = layers.Dense(NUM_CLASSES, activation='softmax')(x) # softmax for multi-class
classification
model = Model(inputs=base_model.input, outputs=x)
```

6. Specify the total number of parameters of the new

architecture : total_params = model.count_params()

7. Print the new architecture (the layers associated to their number of parameters) : They are printed in the jupyter notebook .

8. Retrain the last (two convolutional layers + the last three fully connected), (one convolutional layers + the last three fully connected), and (the last three fully connected) using: the transfer learning strategy for 20 epochs with a batch size of 64 and the Adam optimizer with a learning rate of 0.001. (it is taking forever :(i have no results) :

step 1 : freezing the model (set all layers to non-trainable)
step 2 : we unfreeze the last N convolutional layers based on the configuration (3la hssab l qst)
step 3 : we Set the identified layers (that we found in step 2)to trainable .
step 4 : Build the full model with the custom classification layers

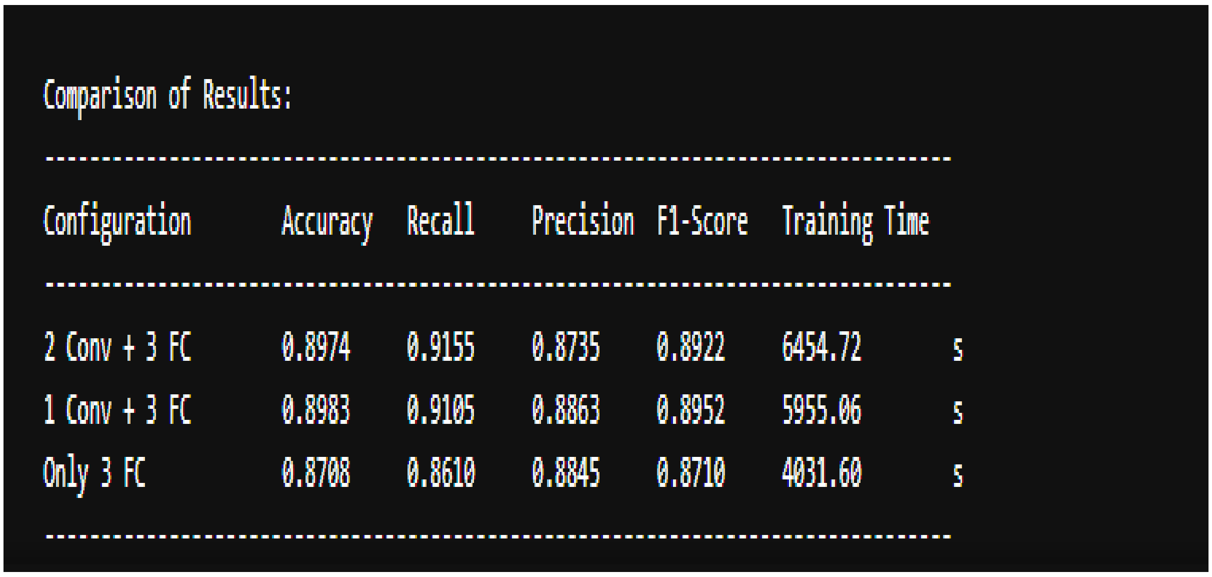
```
    x = base_model.output
    x = layers.GlobalAveragePooling2D()(x)
    x = layers.Dense(1024, activation='relu')(x)
    x = layers.Dense(512, activation='relu')(x)
    x = layers.Dense(NUM_CLASSES, activation='softmax')(x)
    model = Model(inputs=base_model.input, outputs=x)
    model.compile(
        optimizer=Adam(learning_rate=0.001),
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )
```

step 5 : Train the model

```
    history = model.fit(
        train_generator,
        epochs=10,
        validation_data=validation_generator,
        callbacks=callbacks,
        batch_size=64
    )
```

step 6:
Reset the generator + Get predictions + Calculate metrics + plotting

The results :

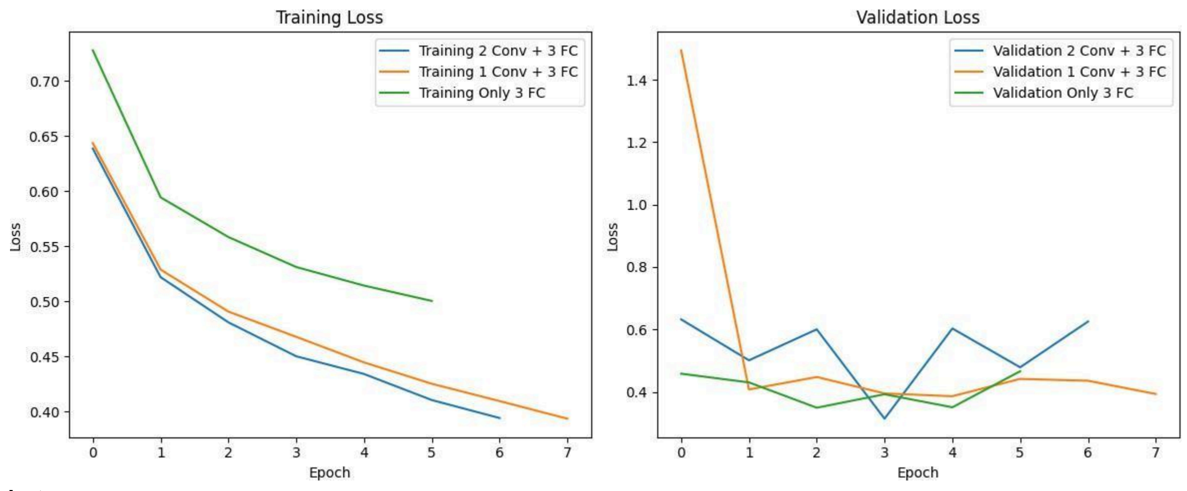


```
Comparison of Results:
--------------------------------------------------------------------------------
Configuration        Accuracy   Recall     Precision  F1-Score   Training Time
--------------------------------------------------------------------------------
2 Conv + 3 FC        0.8974     0.9155     0.8735     0.8922     6454.72       s
1 Conv + 3 FC        0.8983     0.9105     0.8863     0.8952     5955.06       s
Only 3 FC            0.8708     0.8610     0.8845     0.8710     4031.60       s
--------------------------------------------------------------------------------
```

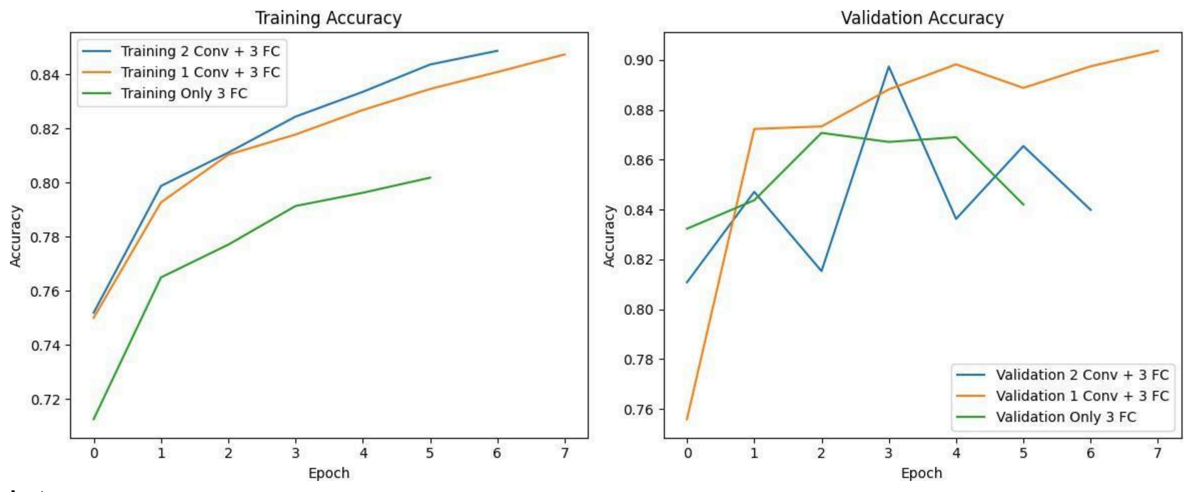
The 1 Conv + 3 FC configuration achieves the highest accuracy and F1-score it provides the best classification.

The 2 Conv + 3 FC configuration has the highest recall and sensitivity making it better at identifying positive cases.
The Only 3 FC configuration, good precision makes fewer false positive predictions.

Loss curves :



Notes :
In the training loss graph the Only 3 FC shows a higher loss compared to the others .
in the validation loss graph the 1 Conv + 3FC is the most stable and lowest validation loss
Accuracy curves :



Notes :
in the training accuracy 2 Conv + 3 FC and 1 Conv + 3 FC achieves higher final training accuracy.

in the validation accuracy 1 Conv + 3 FC is kinda stable compared to the other and it reaches the 0.90 accuracy in the end .

important remark :
there is an overfitting in the 2 Conv + 3 FC the model is good in the training set (both loss and accuracy) but it performs poorly in the validation set (we can see this in both graph loss and accuracy)
The Only 3 FC is underperforming as we can see in the graph .

Finally :

the best performing and balanced model and the best in generalization is the 1 Conv + 3 FC (good accuracy , F1-score , precision and it also comparing all these benefits it has less time compared to 2 Conv + 3 FC) .