**Flow for Using a Pretrained Model**

**1. Dataset Setup**

* Dataset should be **available inside the project folder**.
* There are **two ways** to prepare the dataset:
  + **Manual labelling** (using tools like LabelImg, Roboflow, etc.)
  + **Folder structured dataset** (each class in a separate folder).

**2. Dataset preprocessing (train.py) :**

* Load the dataset.
* Preprocessing:
  + Resize all images to (224, 224).
  + Convert images to arrays.
  + Preprocess images using preprocess\_input (for MobileNetV2).
    - (Special case: For mask detection, crop only the face region.)
* One-hot encode the target labels.
* Train-test split the dataset.
* Data Augmentation:
  + Use ImageDataGenerator for random zooms, shifts, flips, etc.

**3. Model Building**

* **Load MobileNetV2 as base model**:

baseModel = MobileNetV2(weights="imagenet", include\_top=False, input\_tensor=Input(shape=(224, 224, 3)))

* + include\_top=False means we **remove the original output layer**.
* **Create the new head model**:

headModel = baseModel.output

headModel = AveragePooling2D(pool\_size=(7, 7))(headModel)

headModel = Flatten(name="flatten")(headModel)

headModel = Dense(128, activation="relu")(headModel)

headModel = Dropout(0.5)(headModel)

headModel = Dense(2, activation="softmax")(headModel)

* **Attach head to base model**:

model = Model(inputs=baseModel.input, outputs=headModel)

* **Freeze** the base model (don't train MobileNetV2 weights):

for layer in baseModel.layers:

layer.trainable = False

* **Compile** the model:

opt = Adam(learning\_rate=INIT\_LR)

model.compile(loss="binary\_crossentropy", optimizer=opt, metrics=["accuracy"])

* **Train** the model:

H = model.fit(aug.flow(trainX, trainY, batch\_size=BS),

steps\_per\_epoch=len(trainX) // BS,

validation\_data=(testX, testY),

validation\_steps=len(testX) // BS,

**4. Evaluate the Model**

* Predict on test data:

predIdxs = model.predict(testX, batch\_size=BS)

predIdxs = np.argmax(predIdxs, axis=1)

* Print the **classification report**:

print(classification\_report(testY.argmax(axis=1), predIdxs, target\_names=lb.classes\_))

**5. Save the Model**

* Save the trained model in .h5 format:

model.save(args["model"], save\_format="h5")

or simply:

model.save(args["model"])

**6. Application (application.py)**

* There could be **two application files**:
  + One for **video**.
  + One for **images**.
* If both exist, **use both** depending on your project need.

**Inside application.py:**

* Load:
  + **Face detector model** (to detect faces).
  + **Mask detector model** (the one we trained).
* Start video stream:
  + **Open the camera** or video file.
  + **While loop** to continuously:
    - Read frame.
    - **Detect faces**.
    - **Crop face**.
    - **Predict mask** or no mask.
    - **Draw rectangle and label** on the frame.

**Final Cleaned Flow :**  
  
Dataset ➔ Resizing ➔ Preprocessing ➔ Model Building ➔ Training ➔ Evaluation ➔ Save Model ➔ Application Deployment