**Shoes, sandal, boot task**

**Hammad Ijaz**

**Code 1:**

In this code, I used the model DenseNet121 and modified it for a multi class classification. I added custom top layers according to my problem.

**The following layers are added:**

GlobalAveragePooling2D: This layer averages the spatial dimensions of the feature maps to convert the 4D tensor into a 2D tensor, retaining the depth (number of channels) information. It reduces the spatial dimensions to create a fixed-length feature vector that represents the entire image.

Dense(512, activation='relu'): A fully connected dense layer with 512 units and ReLU activation function. This layer performs complex transformations on the features extracted by the base model.

BatchNormalization: A normalization layer that helps stabilize and accelerate the training process.

Dropout(0.5): A dropout layer that randomly sets 50% of the activations to zero during training to reduce overfitting.

Another set of similar layers with 256 units, BatchNormalization, and Dropout.

Dense(num\_classes, activation='softmax'): The final dense layer with num\_classes units (in this case, num\_classes is set to 3 for a multi-class classification task) and a softmax activation function. The softmax function converts the model's output into class probabilities, representing the likelihood of each input image belonging to each class.

The loop iterates through each class name in the class\_names list, which represents the names of the classes for the labeled data. For each class name, it creates a class-specific directory inside the output\_dir to store the labeled data related to that class. The code creates these directories using os.makedirs() and ensures that existing directories are not overwritten by setting exist\_ok=True.

For example, if class\_names = ['cat', 'dog', 'bird'], the code will create three directories named "cat," "dog," and "bird" inside the /content/drive/MyDrive/LabeledData directory. These directories can then be used to store labeled data corresponding to each class separately, making it easy to organize and load data for machine learning tasks like image classification.

Then the for loop is taking all the images and resizing it in 224 by 224 pixels and converting them in numpy array.

**Code 2:**

I used a pre-trained model MobileNetV2. It moves the images to separate folders based on their predicted classes. I did not include the top layer of the model because it is used for the classification part and as we only need to sort the dataset we will not use that layer.

First I checked that if there are destination folders or not (shoe, sandal and boot). If there are no folders, it will create these folders and store the classified images in them.

In the for loop, first we ensured that the path is correct. It creates the file path for the each current image upto the folder. Then the next line loads the image and set the target size to 224x224 pixels. Then the image is converted into a numpy array because the model takes input in numpy array.

Since Deep learning models often expect input in batches, so this step adds an additional dimension to the array to represent a single sample as a batch of size one. Then I preprocessed the image to make it compatible with the MobileNetV2 model. This normalization is essential to ensure that the input data is within a suitable range and has the same preprocessing applied as during the original model's training on the ImageNet dataset.

After this for loop the images are ready to be further processed.