```
from google.colab import drive
```

```
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
# import tensorflow as tf
# from tensorflow.keras.applications import ResNet50
# from tensorflow.keras.layers import GlobalMaxPooling2D
# from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.preprocessing import image
```

```
img=image.load_img('/content/drive/MyDrive/Data Science Project/Dressing Guide System/images/10000.jpg',target_size=(224,224))
```

```
img_array=image.img_to_array(img)
```

```
img_array
```

```
array([[[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.]],

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.]],

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.]],

       ...,

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.]],

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.]],

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.]]], dtype=float32)
```

```
img_array.shape
```

```
(224, 224, 3)
```

```
import numpy as np
expanded_img_array=np.expand_dims(img_array,axis=0)
print(expanded_img_array.shape)
```

```
(1, 224, 224, 3)
```

```
expanded_img_array #4-D image array
```

```
array([[[[255., 255., 255.],
         [255., 255., 255.],
         [255., 255., 255.],
         ...,
         [255., 255., 255.],
         [255., 255., 255.],
         [255., 255., 255.]],

        [[255., 255., 255.],
         [255., 255., 255.],
         [255., 255., 255.],
         ...,
         [255., 255., 255.],
         [255., 255., 255.],
         [255., 255., 255.]],

        [[255., 255., 255.],
         [255., 255., 255.],
         [255., 255., 255.],
         ...,
         [255., 255., 255.],
         [255., 255., 255.],
         [255., 255., 255.]],

        ...,

        [[255., 255., 255.],
         [255., 255., 255.],
         [255., 255., 255.],
         ...,
         [255., 255., 255.],
         [255., 255., 255.],
         [255., 255., 255.]],

        [[255., 255., 255.],
         [255., 255., 255.],
         [255., 255., 255.],
         ...,
         [255., 255., 255.],
         [255., 255., 255.],
         [255., 255., 255.]],

        [[255., 255., 255.],
         [255., 255., 255.],
         [255., 255., 255.],
         ...,
         [255., 255., 255.],
         [255., 255., 255.],
         [255., 255., 255.]]]], dtype=float32)
```

```
from tensorflow.keras.applications.resnet50 import ResNet50,preprocess_input
```

```
import tensorflow
from tensorflow.keras.preprocessing import image
from tensorflow.keras.layers import GlobalMaxPooling2D
```

```
model = ResNet50(weights='imagenet',include_top=False,input_shape=(224,224,3))
model.trainable = False

model = tensorflow.keras.Sequential([
    model,
    GlobalMaxPooling2D()
])
```

```
preprocessed_img=preprocess_input(expanded_img_array)
print(preprocessed_img.shape)
preprocessed_img
```

```
(1, 224, 224, 3)
array([[[[151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ],
         ...,
         [151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ]],

        [[151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ],
         ...,
         [151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ],
```

```
         [151.061  , 138.22101, 131.32   ]],

        [[151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ],
         ...,
         [151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ]],

        ...,

        [[151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ],
         ...,
         [151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ]],

        [[151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ],
         ...,
         [151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ]],

        [[151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ],
         ...,
         [151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ],
         [151.061  , 138.22101, 131.32   ]]], dtype=float32)
```

```
model.predict(preprocessed_img)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
array([[ 0.        ,  4.4787416 ,  0.43626547, ...,  3.1709626 ,
         6.93157   , 17.540562  ]], dtype=float32)
```

```
model.predict(preprocessed_img).flatten()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 179ms/step
array([ 0.        ,  4.4787416 ,  0.43626547, ...,  3.1709626 ,
        6.93157   , 17.540562  ], dtype=float32)
```

```
model.predict(preprocessed_img).flatten().shape
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 144ms/step
(2048,)
```

```
from numpy.linalg import norm
norm(model.predict(preprocessed_img).flatten())
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 160ms/step
254.24007
```

```
model.predict(preprocessed_img).flatten()/norm(model.predict(preprocessed_img).flatten())
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 210ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 205ms/step
array([0.        , 0.01761619, 0.00171596, ..., 0.01247232, 0.02726388,
       0.06899212], dtype=float32)
```

```
np.sqrt(np.dot(model.predict(preprocessed_img).flatten(),model.predict(preprocessed_img).flatten()))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 144ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step
254.24007
```

```
Start coding or generate with AI.
```