Assignment 2 – Deep Learning image classification

1. Performance:
   Training: loss: 0.2064; accuracy: 93.36%
   Testing: loss: 0.2567; accuracy: 91.93%

   Parameters and structure:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rescaling (Rescaling)        (None, 80, 60, 1)         0
_________________________________________________________________
conv2d (Conv2D)              (None, 78, 58, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 39, 29, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 37, 27, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 18, 13, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 11, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 5, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 5120)              0
_________________________________________________________________
dense (Dense)                (None, 13)                66573
=================================================================
Total params: 159,245
Trainable params: 159,245
Non-trainable params: 0
_________________________________________________________________
```

   Steps to replicate experiment:
   - I followed the steps given in the link https://towardsdatascience.com/image-classification-python-keras-tutorial-kaggle-challenge-45a6332a58b8 given in the assignment pdf.
   - I experimented with the 2d convolution and max pooling layers, and I found that more layers with increasing numbers of filters improved the accuracy of the model. I stuck with three pairs of 2d convolution and max pooling to keep the execution time in check.
   - I also experimented with the number of dense layers and the number of hidden layer neurons in each. I did not find a significant difference in any of my testing, so I just eliminated all the dense layers other than the output layer.

2. I generated additional training data to try to improve the model.
   Performance:
   Training: loss: 0.5917; accuracy: 0.8095
   Testing: loss: 238.5049; accuracy: 30.75%

Parameters and structure: Same as above

Steps to replicate:

- I followed the documentation at https://keras.io/api/preprocessing/image/#imagedatagenerator-class
- I experimented with various parameters in the image generator, but I found that the more non-default parameters I included, the worse the "improved" model performed. This is likely because the images' orientation has a part in the prediction – for example, all the shoe images are oriented horizontally with the toe pointing towards the right. This is standard for the dataset and for most retail locations in general, and thus changing it affected the accuracy of the model.
- I also found that using the image generator dramatically decreased the accuracy for the testing set. This is likely due to overfitting. For example, a horizontal flip essentially repeats the same image for a shirt.