**Product Pattern Classification**

The objective is to classify each product image to one of 13 product patterns. A robust solution can be approached by using deep learning image classifier.

**Steps Taken:**

1. **Data exploration:**

* Given data set contained around unbalanced 3300 images across 13 classes. One of the class ‘Predicted’ had zero images in training data.
* Using class explanation URL, understood pattern of each class. Observed that training data had missing ‘embroidery’ class images. Scraped around 100 images for ‘embroidery’ class.
* Cleaned up the images in each class to train the model on correct images pertaining to particular class. This step of preparing data was crucial in improving accuracy of model.
* Used Google Colab to train models.

**2)**  **Tensorflow training in Google Colab:**

* For training, splitted data into train (90%) and validation (10%).
* First, built a 3 convolutional layer network in tensorflow with raw images (trained on unbalanced data while cleaning images for classes). Accuracy for with this network was very poor (20%).
* After cleaning up images, trained network again. With this data, accuracy improved above 40%. The training and validation loss was not reducing, signifying that training data is not sufficient. Model had poor bias and poor variance.

**Improving model performance:**

* In order to improve bias of the network, tried training bigger model (4 layers of conv, different configurations of max pooling and conv layer), different hyperparameters and optimization techniques.
* To reduce the variance of the model, used data augmentation techniques. With understanding of the training and test data, it was clear that all data augmentation methods can not be used. First method used was center scaling of the images to focus on the pattern of the product. Used 90%, 80% and 70% scaling of images. The other method used was horizontal flip augmentation for images.



Original Image Center scale 90% Center scale 80% Center scale 70%

After data augmentation, number of training images increased to 16046 with more than 1000 images per class. For training a good deep learning model, it is expected to have around 2000 images per class.

With data augmentation, tried tensorflow training with 3 conv layers. But faced memory issues in Google colab. Tried using less batch size but it was taking very long time to train model and faced frequent disconnection while training in Google Colab.

So, I have done further training with augmented data in Keras.

**3) Keras training in Google Colab:**

* In order to avoid memory issue, I tried first training smaller network with 3 conv layers for 30 epochs. It showed improvement in validation accuracy (60%).
* Next training phase, tried 4 conv layers in network. It improved validation accuracy but loss still didn’t reduce after certain epochs. It was understood that this is due to overfitting.
* To avoid overfitting, included **dropout layer** in the network and also added **batch normalization** to improve training time and validation accuracy.

Final trained model had 4 conv layers, 2 FC layers, Batch normalization and dropout layers with adam optimizer. Model had been trained for 50 epochs. Training model history with accuracy (75%) and loss plots can be seen in the attached image files.

**4) Nexts steps to improve model:**

* 1. Model can be trained for more epochs (100-150).
  2. More data can be added to improve on variance and reduce validation error.
  3. Try to train more deep network with more conv layers and different parameters like filter size, strides.

**References:**

1. Stanford CS class [CS231n: Convolutional Neural Networks for Visual Recognition](http://cs231n.stanford.edu/) notes.
2. [AlexNet](http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks)
3. [VGG](http://www.robots.ox.ac.uk/~vgg/research/very_deep/)
4. [Deep Learning textbook - Goodfellow](http://www.deeplearningbook.org/)
5. [Deep learning specialization - Andrew Ng](https://www.coursera.org/specializations/deep-learning)