Fashion Product Image Classiffication with Convoluntion Neural Network

By Hongling Yang

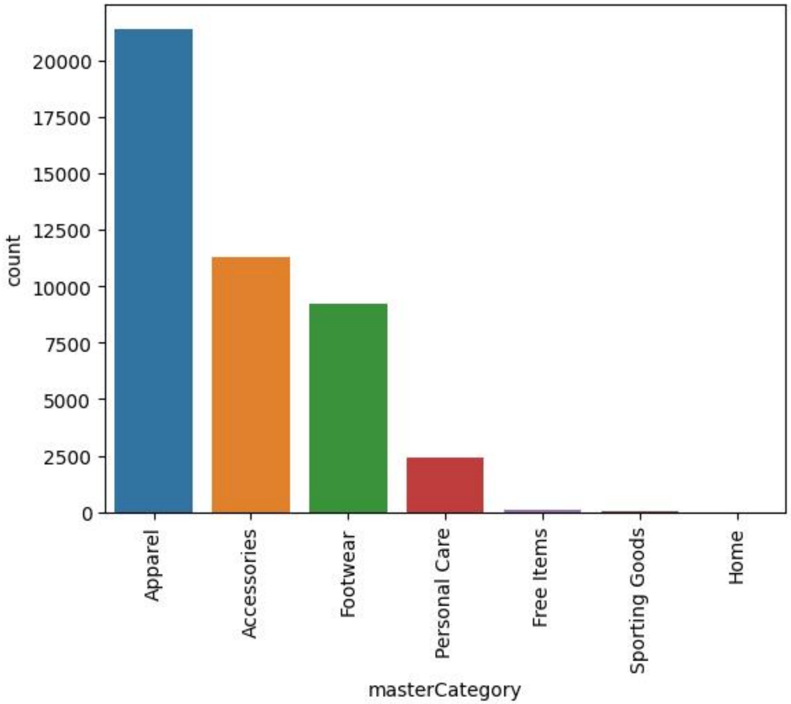
Objective

- Import fashion product image dataset and label product/images with matching master categories (the target feature) provided in an acompanying catalog file.
- Apply Transfering learning strategy and build a convoluntion neural network (CNN) to deep learn and extract information out of fashion product images. (supervised machine learning)
- Train and tune the CNN model which will acheive over 95% classification accuracy.
- Implement a similar appraoch (i.e., transfer learning, VGG16...) and build a recommendation system for fashion products/images. (unspupervised learning)
- Data Source (Kaggle):

(https://www.kaggle.com/datasets/paramaggarwal/fashion-product-images-dataset)

Distribution of Target Feature (Kaggle Dataset)

(1) Orginal Kaggle Dataset



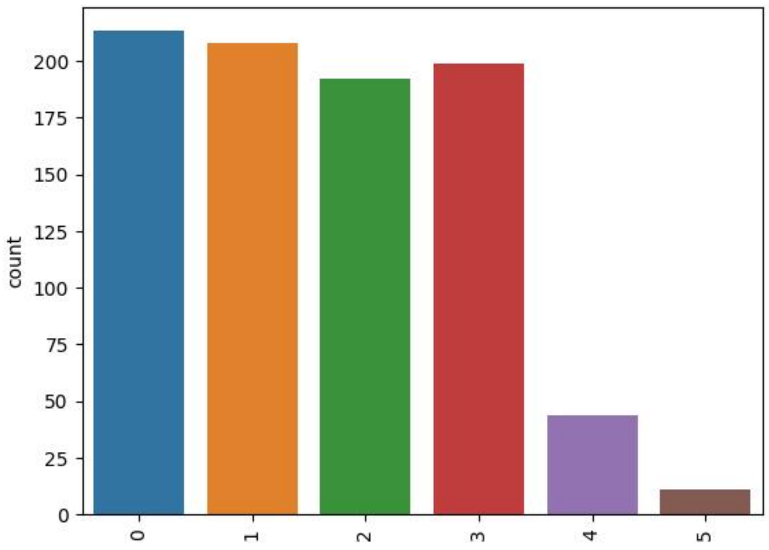
	masterCategory	Freq
Majority Classes	Apparel	21,397
	Accessories	11,274
	Footwear	9,219
	Personal Care	2,403
Minority Classes	Free Items	105
	Sporting Goods	25
	Home	1

***

- Scaled Rating = Raw Rating/log (Rating Counts)
- Scaled Tating is right -skewed,

Distribution of Target Feature (Study Sample)

(1) Test Set of Study Sample



Study Sample (4,329 Images)		
	masterCategory	**Freq**
Majority Classes	Apparel	1,000
	Accessories	1,000
	Footwear	1,000
	Personal Care	1,000
Minority Classes	Free Items**	267
	Sporting Goods**	62
	Home***	0

** Including 162 new ‘Free Items’ and 37 ‘Sproting Goods” generated by ImageDataGenerator()

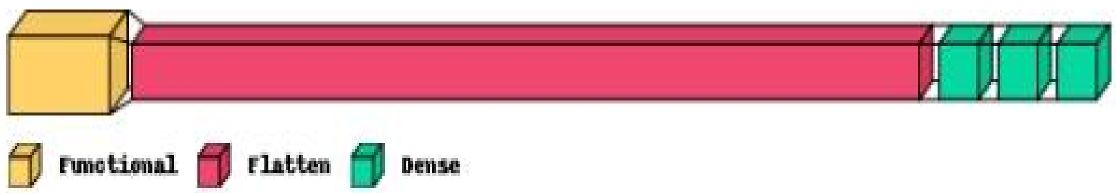
***masterCategory ‘Home” is removed from classification

CNN Model Wihtout Hyperparameter Tuning

Un-tuned Model Topology

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 7, 7, 512)	14714688
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 50)	1254450
dense_1 (Dense)	(None, 20)	1020
dense_2 (Dense)	(None, 7)	147

Total params: 15,970,305
Trainable params: 1,255,617
Non-trainable params: 14,714,688



- **Squential Model:**

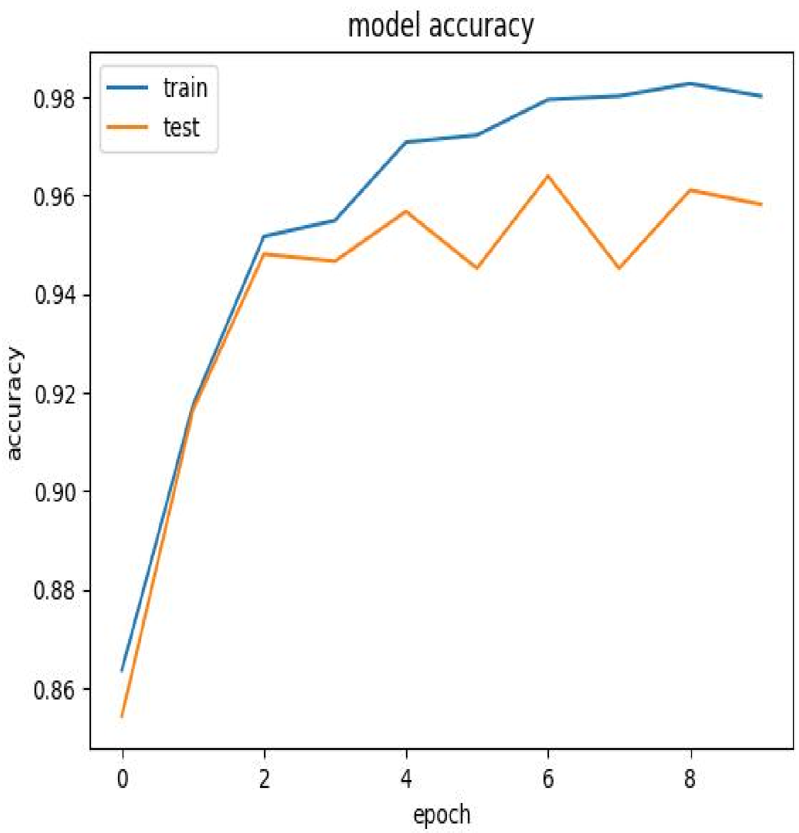
  VGG16 (wihtout top dense layers) + dense layer (neurons = 50) + dense layer (neurons=20) + output layer (neuron=7)

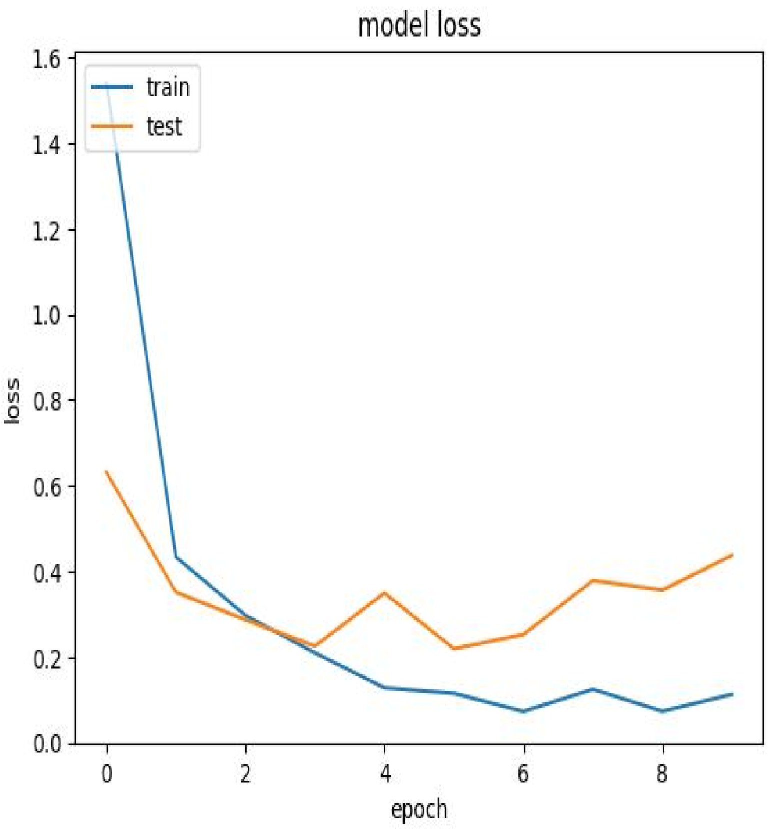
Classification Report

	precision	recall	f1-score
Apparel	0.99	0.98	0.98
Accessories	0.92	0.94	0.93
Footwear	0.99	0.99	0.99
Personal Care	0.99	0.98	0.99
Free Items	0.67	0.70	0.69
Sporting Goods	0.92	1.00	0.96
accuracy			0.96
macro avg	0.92	0.93	0.92
weighted avg	0.96	0.96	0.96

- Train/test split at 80/20,
- Optimizer = ‘adam’, epochs=10, batch_size=10, learning rate=0.001

Un-tuned CNN Model Performance





- test data accuracy: 0.959

Hyperparameter Tuning

- 3)Tuning three hyaperparameters:

  - the number of neurons in the first dense layer
  - the learning rate of the optimizer
  - the best number of epochs

- The tuning is done in two steps.

Hyperparameter Tuning

Step 1. Set epochs =5 and tuned the number of neurons and the learning rate (RandomSearch in Keras Tuner module).

- Search spacing :
  {‘neurons’ : (20, 40, 50), ‘learning rate’ : (0.01, 0.001, 0.0001) }
- Searching Result:
  {‘neurons’=20, ‘learning rate’ = 0.001}

Step 2. Set {‘neurons’=20, ‘learning rate’ = 0.001} and tune ‘epochs’

- Set ‘epochs’ = 50 and trace the model the accuracy score on test set.
- Searching Result: {‘epochs’=20}

Best Hyperparameters.

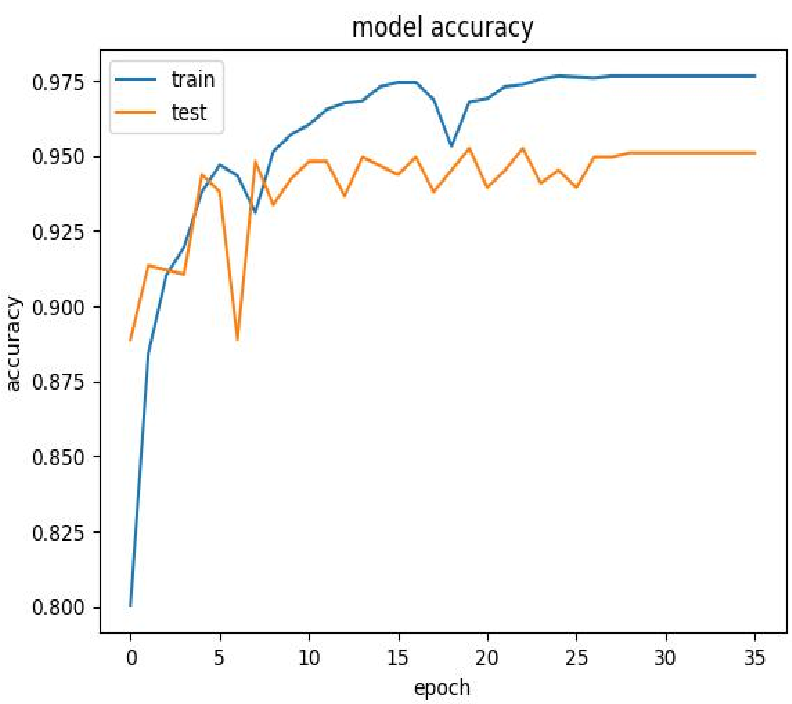
{'neurons': 20, 'learning_rate': 0.001, 'epochs': 36}

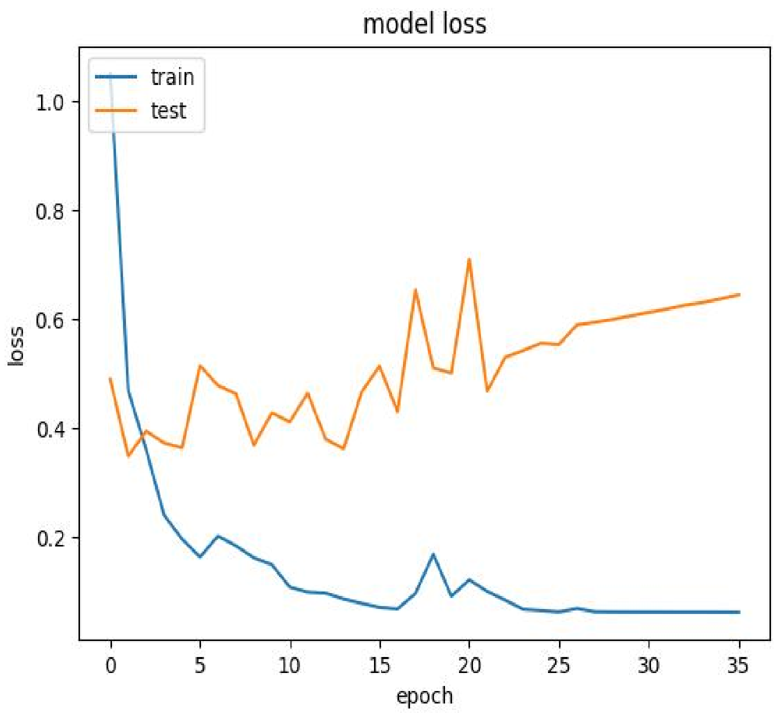
Tuned Model Classification Report

	precision	recall	f1-score
Apparel	0.99	0.98	0.98
Accessories	0.90	0.94	0.92
Footwear	1.00	0.99	0.99
Personal Care	0.99	0.98	0.99
Free Items	0.55	0.66	0.60
Sporting Goods	0.00	0.00	0.00
accuracy			0.94
macro avg	0.74	0.76	0.75
weighted avg	0.94	0.94	0.94

- Train/test split at 80/20,
- Optimizer = ‘adam’, epochs=10, batch_size=10, learning rate=0.001

Hyperparameter-tuned CNN Model Performance





- test accuracy: 0.945

Why Tuning Made Worse

- Model perforamnce is wrose after tuning than before tuning

- **Possible Explanations:**
  - Small search space with a Randomsearch engine. With only a few trials (=6) of Randomsearch. The optimal configuration may never be reached;
  - When tuning, models under different configurations are trained with ‘epochs’ = 5

- **Possible Solutions:**
  - Expand search space and tune with an adpative searching engine (e.g., Hyperhand, BayesianOptimization).
  - Increase searching trials and ‘epochs’ = 5

Item-Based Recommendation System

➢**Feature Extraxction**:

- Implement a pre-trained VGG16 model (including top dense layers)
- return 1,000 new hiden features.
- Different from the VGG16 base model in classification which does not have top dense layers.

➢**Closeness Meaure**:

- Metrc: the cosine similarity scores

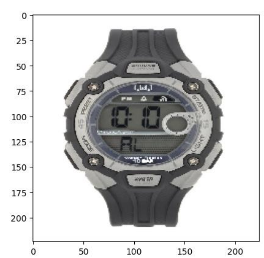
➢**Recommendations:**

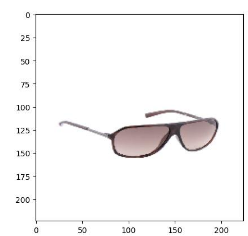
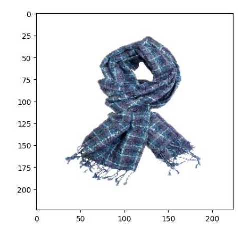
- For a given product/image, the 4,328 similarity scores with other products/images are ranked. Products/Images with the top 5 highest scores are recoomeded

Visual Evaluation of Recommendation System



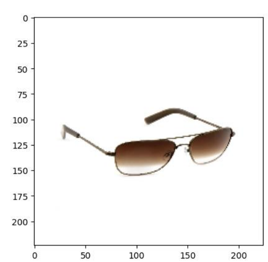
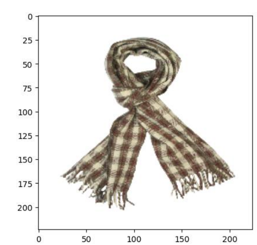

Orginal



Recommendations



Conclusion

- The CNN model we bulit in this study can classify fashion products/images with a good accuracy (~96%).
- Applying transfer learnig with a pre-trained classification model such as VGG16 saves resource and time. Model is built and trained fast with good satisfactory performance.
- Transfer learning can also leverage feature extraction, which makes building Recommendation system easy and fast
- Hyperparameter tuning is important in model configuration. Apply efficient search engines with decent search spaces and adequate numer of trails is necessary.
- Randomsearch engine may not work well when resources are limited

What is Next

- *Utilize Google Colab or Kaggle Notebooks and test the CNN model on the entire 44K+ Kaggle dataset;*
- *Make the model more robust, modular and portable for easy adoption by other image classification tasks.*
- *Experiment with tuning other hyperparameters, such as Optimizers ('adam', 'sgd', or 'rmsprop'?), the number of dense layers, and training bacth size…*
- *Experiment with adding a dropout layer between two top dense layers to constrain the network from over-learning certain features.*
- *Experiemnt with other per-trianed models (such as Xception, ResNet50, InceptionV3, etc. in Keras applications) as the base model.*
- *Conitnue learning about deep Neural network in regression.*