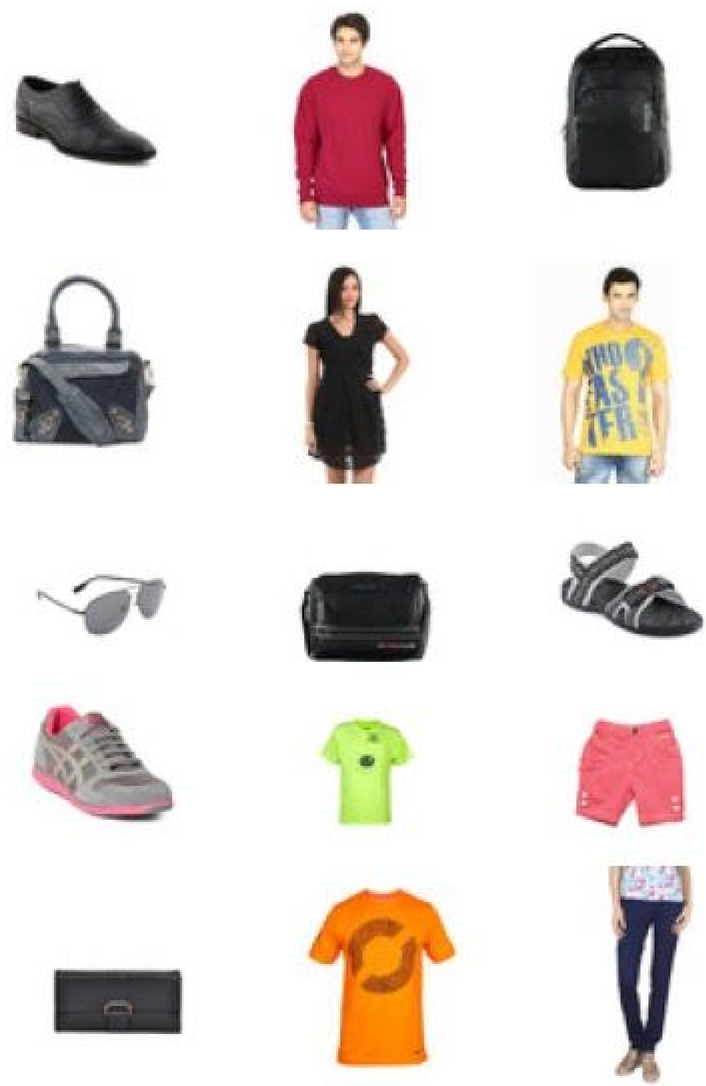
Fashion product image classification project

Final Report by Veronica Ferman, July 2020



The project delves into the exciting field of image classification using deep learning architectures in image recognition. The dataset is made up of professionally shot high-resolution fashion product images that have been cataloged with descriptive categories on their product characteristics.

Visual classification of commercial products is an important part of object detection and feature extraction in computer vision.The goal of the project is to create a classification model with a degree of accuracy high enough to be utilized by product suppliers, importers and exporters, product managers and inventory control professionals in the classification of products.

Examples of industry applications include:

- **Ecommerce**
  Ecommerce businesses store their products in e-catalogs also called product information management (PIM) systems that facilitate the search of products by consumers. Many businesses are still manually classifying products into their respective categories. The product classification can expedite this labor intensive process.

- **The fashion industry**
  In the fashion industry, automatically classifying product features facilitates the workflow required in the design and production of garments.

- **Retail**
  Product image classification can provide retailers with an understanding of the layout of goods on the shelf by uncovering anomalies in the product line, pricing and branding. For example, a product image classification system can process product data in real time to detect whether goods are present or absent on the shelf or if products next to each other belong to a different category and/or price groups.

  If the product classification identifies anomalies, managers can identify the cause, alert the merchandiser, and recommend solutions for the corresponding part of the supply chain.

- **International trade**

Inventory product classification is critical for import regulation compliance and for effective import operations. Suppliers can save precious time by automating the classification of products and outperform the competition.

- **Marketing**
  It is also fundamental in the creation of a well-planned marketing campaign that takes into account product categories, attributes and descriptions. It can also be an added advantage when pricing products for quick returns.

The project chooses a deep learning technique for the automatic assignment of images to relevant categories. The image recognition model will be trained to recognize categories by building visual features that later can be applied to the categorization of new images that are fed to the model. Specifically, the project will employ a convolutional neural network model (CNN). The model will be scalable with the ability to handle large amounts of visual content as the needs of its users increase.

Different optimization techniques will be used to yield a classification system that is robust enough to address the needs of businesses requiring a product catalog to classify items in an efficient and automated manner.

The initial dataset that will be used to train the model can be found at:
kaggle.com/paramaggarwal/fashion-product-images-small

For project code, please visit:
github.com/vferman2000/Springboard/tree/master/FashionProductClassificationProject .

Dataset Description

The dataset consists of two files:

A styles.csv containing 44,446 items. The csv file was converted into a Pandas DataFrame named df_styles.

Each item in the DataFrame is identified by an id number and is classified by: gender,masterCategory, subCategory, articleType, baseColour, season, year, usage and productDisplayName.
A few rows contained an extra column in their productDisplayName which was omitted from the dataframe for uniformity.

	id	gender	masterCategory	subCategory	articleType	baseColour	season	year	usage	productDisplayName
0	15970	Men	Apparel	Topwear	Shirts	Navy Blue	Fall	2011.0	Casual	Turtle Check Men Navy Blue Shirt
1	39386	Men	Apparel	Bottomwear	Jeans	Blue	Summer	2012.0	Casual	Peter England Men Party Blue Jeans
2	59263	Women	Accessories	Watches	Watches	Silver	Winter	2016.0	Casual	Titan Women Silver Watch
3	21379	Men	Apparel	Bottomwear	Track Pants	Black	Fall	2011.0	Casual	Manchester United Men Solid Black Track Pants
4	53759	Men	Apparel	Topwear	Tshirts	Grey	Summer	2012.0	Casual	Puma Men Grey T-shirt

The **image** folder contains 44,441 jpg image files of fashion products. Each file is an 80x60 image in RGB (red, green and blue) channel. Each image is recognizable by its unique id that maps the jpg image to the df_styles file. Examples of the images can be seen below:

A new column “image” was added to the df_styles DataFrame containing the full name of the image in preparation for model training.

id	gender	masterCategory	subCategory	articleType	baseColour	season	year	usage	productDisplayName	image
7177	Men	Apparel	Topwear	Shirts	Blue	Fall	2011.0	Casual	Scullers Men Blue Check Shirt	7177.jpg
15207	Men	Apparel	Topwear	Sweaters	Black	Fall	2011.0	Casual	Arrow Sport Men Solid Black Sweaters	15207.jpg
26898	Women	Apparel	Topwear	Tops	Pink	Summer	2012.0	Casual	Jealous 21 Women Pink Top	26898.jpg
34133	Men	Accessories	Wallets	Wallets	Brown	Fall	2012.0	Smart Casual	Turtle Men Brown Wallet	34133.jpg
39456	Women	Apparel	Topwear	Kurtas	White	Summer	2012.0	Ethnic	Diva Women White Printed Kurta	39456.jpg

The master category is selected as the target class for the Convolutional Neural Network model. The model will be trained to predict the master categories to which the images belong.

There are seven categories in the masterCategory column: Apparel, Accessories, Footwear, Personal Care, Free Items, Sporting Goods and Home items. The masterCategory column will be the target variable in the convolutional network model. The number of class incidents in the dataset is as follows:

```
df_styles.masterCategory.value_counts()
```

```
Apparel           21400
Accessories       11289
Footwear           9222
Personal Care      2404
Free Items          105
Sporting Goods       25
Home                  1
Name: masterCategory, dtype: int64
```

Approach to handle imbalanced classes

As it can be seen in the table breakdown, the classes are not balanced and there are classes such as ‘Apparel’ and ‘Accessories’ that contain a large number of items while other ones, such as the Home category, contain one only one item. An unbalanced dataset can have the effect of generalizing well on majority classes but undermining minority classes. The objective of the model is to be scalable and to generalize well on unseen data across classes.

There are a few approaches available to minimize the impact that an imbalanced dataset has on the classification performance of the model. Methods such as oversampling of minority classes or undersampling of majority classes can be used for generalization

In the context of the present project, a ‘combination of minority classes’ approach was selected. The ‘Home’ and ‘Sporting Goods’ classes were added to the ‘Free Items’ class which contains a mixture of items.

The final class breakdown is as follows:

```
[ ]  df_styles.masterCategory.value_counts()

     Apparel          21400
     Accessories      11289
     Footwear          9222
     Personal Care     2404
     Free Items         131
     Name: masterCategory, dtype: int64
```

Using a stratified approach to splitting data into train, validation and test subsets

To ensure that each of the five classes are equally represented in the train,validation and test subsets, the data was stratified by the master category column, so each subset has an equal proportion of classes.

A control subset was created and will be kept separate from the train, validation and test data. The control subset will be used in the final phase of the model evaluation as unseen data that will double check the performance of the model.

The final breakdown of the data subsets is:

```
The train subset has 31112 samples
The validation subset has 4444 samples
The test subset has 4445 samples
The control subset has 4445 samples
```

Building the initial convolutional neural network

Image segmentation with a convolutional neural network (CNN) involves feeding segments of an image as input to a convolutional neural network, which labels the pixels using a sliding pattern finder. The CNN scans the image, looking at a small “filter” of several pixels each time until it has mapped the entire image.

The CNN architecture chosen for the fashion image classification has a convolution + pooling architecture, followed by 2 fully connected dense layers. (3 convolutional layers and 2 dense layers).

- The input image is 80x60 in size with an RGB (red, green, blue) channel.
- The activation function for the convolutional layers is a ReLU ((Rectified Linear Units).
- The output of the convolutional layers are 2-D feature maps that identify where the features are found.
- A flatten layer to convert the 2-D feature map to a 1-D input vector as expected by the dense layer.
- The second dense layer or output layer will utilize a softmax activation function for the multiclass classification.

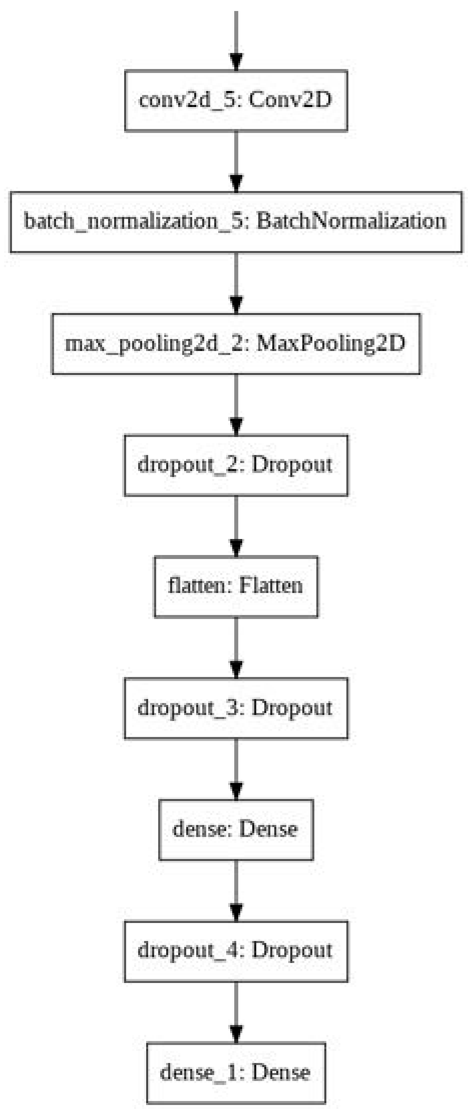
The hyperparameters used are:

- MaxPooling2d(2, 2) which extracts certain features from the image and reduces its height and the width. By using 2 pooling layers, the height and width are 1/4 of the original sizes.

- BatchNormalization which regularizes and makes the training of convolutional neural networks more efficient.

- Dropout to avoid overfitting.

Diagram showing last convolutional layer and dense layer structure:



An imageDataGenerator is used to map images from the image folder to the target classes in the df_train, df_validation and df_test DataFrames. The images are rescaled during the process.

Experiments performed to find best model

Different optimizers and learning rates combinations were used to find the best performing model capable of generalizing well on unseen data.

Some of the techniques tested include:

- Testing model with different optimizers to minimize the cost function and find the optimized value for weights. The model was optimized using Stochastic Gradient Descent, Adaptive Moment Estimation (Adam) and Root Mean Squared Propagation (RMSprop).

- Adjusting the learning rate hyperparameter that controls how much the weights of the network are adjusted with respect to the loss gradient.

- An exponential decay function that reduces the learning rate by a consistent percentage rate over the training iterations.

- Using different numbers of iterations over the dataset (epochs) during training.

- An early stopping callback function that stops iterations when the loss metric has stopped improving during training.

- A parameter averaging across models using the top two models to create an ensemble model.

- An averaged prediction technique that takes the top two models predictions and averaged them to make improved predictions on unseen data.

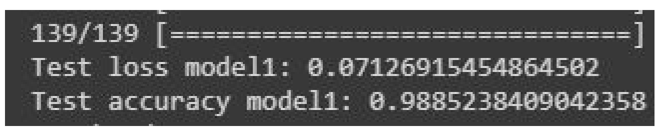
Results

The final model hyperparameters were selected based on their contribution to the image classifier performance, ability to generalize well and make accurate predictions on unseen data.

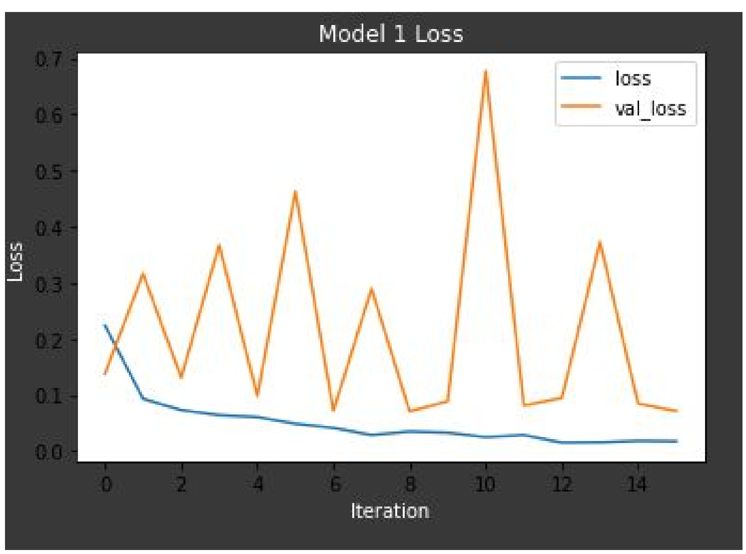
The convolutional neural network model was compiled and fitted on the training dataset using two separate hyperparameters combinations:

1) Model 1 used the Adam optimizer with an exponential decay function. The initial learning rate was set at 0.001, decay steps at 10000 and a decay rate of 0.9. Categorical_crossentropy is chosen as the loss function and accuracy as the metrics.

Model 1 resulted in a **98.85%** accuracy on the validation data and a loss of .**071**



Model 2 Loss Plot



The loss plot for model 1 shows spikes in the iterations. After running experiments multiple times, it was detected the spikes are the result of dataset outliers in the mini-batches used for Adam optimization. As previously discussed, it was decided to leave minority classes in the dataset.

The model scores a **99.12%** when making predictions on the test data. A column correct_label was created and assigned a 1 to predictions correctly labeled and a 0 to predictions that were incorrectly labeled when compared to the true labels.

```
4445/4445 [==============================] - 2252s 507ms/step
Model_1 prediction accuracy on test subset: 99.12260967379078
```

	id	image	masterCategory	predictions	correct_label
0	49120	49120.jpg	Personal Care	Personal Care	1
1	31996	31996.jpg	Accessories	Accessories	1
2	16535	16535.jpg	Accessories	Accessories	1
3	53441	53441.jpg	Footwear	Footwear	1
4	48318	48318.jpg	Accessories	Accessories	1

Some of the correctly classified images can be seen below:

Accessories | Accessories3440

Personal Care | Personal Care3008

Footwear | Footwear2933

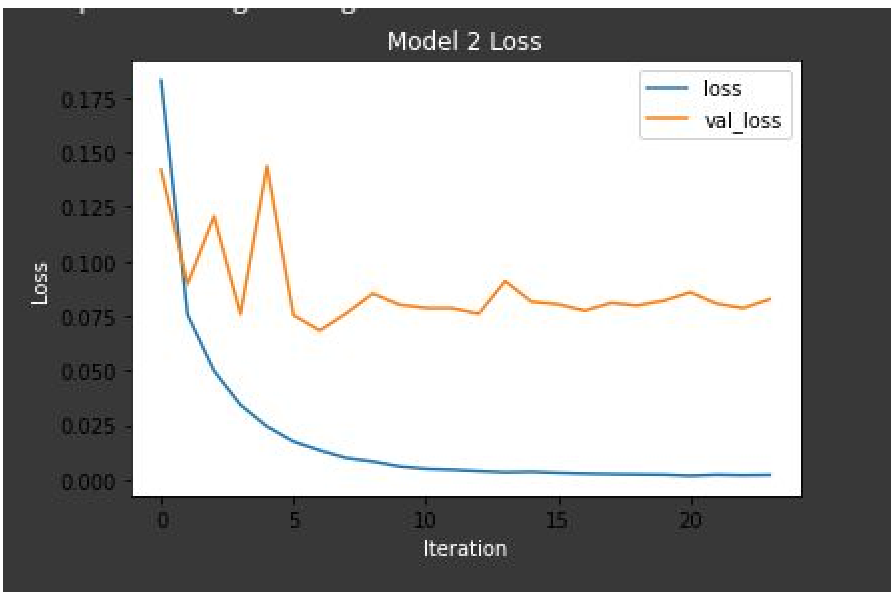
Apparel | Apparel2169

2) Model 2 used the stochastic gradient descent optimizer (SGD) with an exponential decay function. The initial learning rate was set at 0.01, decay steps at 10000 and a decay rate of 0.9. Categorical_crossentropy is chosen as the loss function and accuracy as the metrics.

Model 2 resulted in a **98.28%** accuracy on the validation data and a **.0827** loss.

```
139/139 [==============================]
Test loss model2: 0.08270454406738281
Test accuracy model2: 0.9828982949256897
```

Model 2 Loss Plot



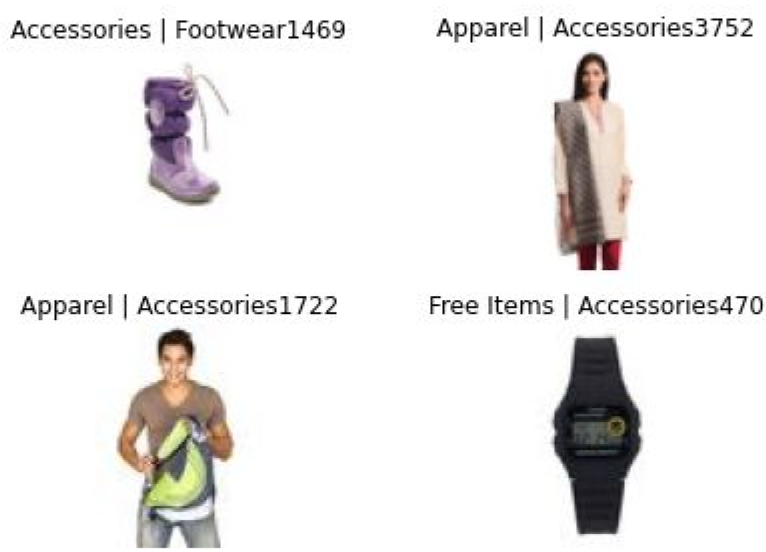
Model 2 scores a **98.85%** when making predictions on the test data. A column correct_label was created and assigned a 1 to predictions correctly labeled and a 0 to predictions that were incorrectly labeled when compared to the true labels.

```
4445/4445 [==============================] - 24s 5ms/step
Model_2 prediction accuracy on test subset: 98.85264341957254
```

	id	image	masterCategory	predictions	correct_label
0	49120	49120.jpg	Personal Care	Personal Care	1
1	31996	31996.jpg	Accessories	Accessories	1
2	16535	16535.jpg	Accessories	Accessories	1
3	53441	53441.jpg	Footwear	Footwear	1
4	48318	48318.jpg	Accessories	Accessories	1

Some of the misclassified images can be seen below:

Predicted labels | True labels



Some of the misclassified images have a model wearing or holding an accessory. The model also misclassified images found in both the free items and accessories categories, such as watches.

Predictions and accuracy on the control group

On the control subgroup, held as unseen data, the accuracy of the predictions was:

Model 1 **98.76%**

```
139/139 [==============================] - 19s 133ms/step
Model_1 prediction accuracy  on the control subset: 0.9876265466816648
```

Model 2 **98.58%**

```
139/139 [==============================] - 19s 135ms/step
Model_2 prediction accuracy on the control subset: 0.9858267716535433
```

Averaging two model predictions for better model performance

```
finalpred=(pred_1+pred_2)/2

def acc(y, pred):
     return np.equal(control_generator.classes, np.argmax(finalpred, axis=1)).mean()

print("Accuracy of averaged predictions: " + str(acc(control_generator.classes, finalpred)))
```

```
Accuracy of averaged predictions: 0.9903262092238471
```

When making predictions using the averaging of predictions technique, the accuracy of the predictions reached **99.03%** on the control group.

Based on the predictions performed on the test and control datasets, the model generalizes well on unseen data.

The average prediction technique increased the accuracy of the predictions on unseen data.

Classification Report

The following tables show the classification report for the averaged model. The first table shows the precision, recall, f1-score, accuracy for each class.

	Labels
0	Accessories
1	Apparel
2	Footwear
3	Free Items
4	Personal Care

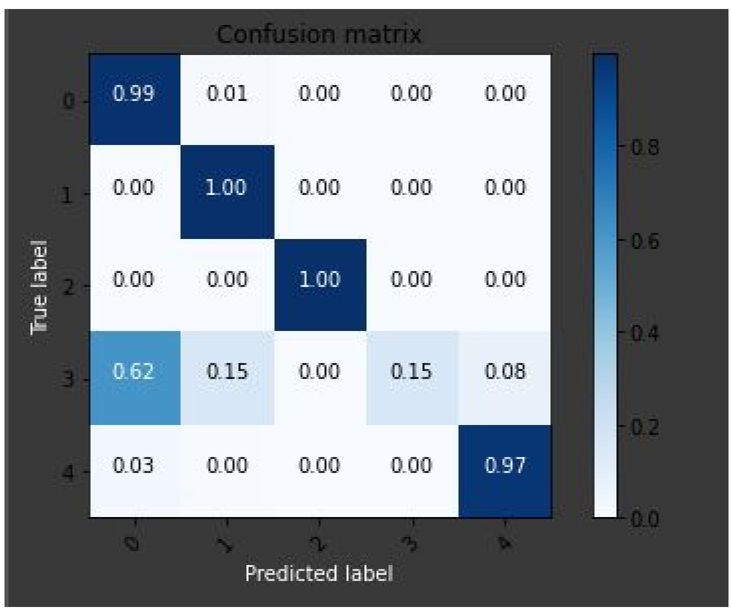
```
Classification Report of averaged model:
              precision    recall  f1-score   support

           0       0.98      0.99      0.98      1129
           1       0.99      1.00      0.99      2140
           2       1.00      1.00      1.00       923
           3       1.00      0.15      0.27        13
           4       0.99      0.97      0.98       240

    accuracy                           0.99      4445
   macro avg       0.99      0.82      0.84      4445
weighted avg       0.99      0.99      0.99      4445
```

The classification report shows the high level of precision, recall and F1 score among the majority classes of Accessories, Apparel and Footwear and the Personal Care minority class.

The averaged prediction model was successful in identifying the classes between 97% to 100% of the time and they were correctly labeled between 98% and 100% percent of the time.

The report shows that the minority class ‘Free Items’ had a very low recall and F1 scores. As stated before, the free items class contains a mixture of items that do not fit into one product category. In the future, this class could be relabeled at a data assessment stage.

Normalized Classification Matrix



The normalized classification matrix showing the model’s high level of performance in image classification.

Conclusion

Convolutional Neural Networks are extremely powerful models for product image classification. The industry applications of a well optimized model range from retail stores, ecommerce sites, fashion firms to trade companies.

It is not an understatement to say that deep learning architectures such as CNNs are in the middle of a digital transformation where images can be automatically detected and classified for pricing, design, branding, trade and content creation.

To maximize the power of deep learning architectures, the project has shown the importance of having a dataset with enough class representation. Different techniques could be utilized to address data imbalances, including the decision to combine minority classes and in some cases, omit outliers.

The result of multiple experiments is a model with a 99% percent accuracy on unseen data that generalizes well.