Final Capstone - Project Report

CNN Classifier on Fashion Product Images

Problem statements

Fashion e-commerce accounted for roughly 29.5 percent of the total fashion retail sales in the United States (2020). Yet, one of the main problems they face is categorizing these apparels, such as clothing and accessories, from the images, especially when the categories provided by the brands are inconsistent.

By far, many businesses have developed and advanced their algorithms to classify products further into multi-categories, based on the product's color, texture or pattern, style, or look. The product features are informative for building a predictive model that customizes or promotes products regarding the customers' questionnaires (about their preferences in style, color, etc.), feedbacks, and ratings, on their past purchases. Some well-known e-commences specializing in customizing or matching fashion products to customers are *Stitch Fix*, *DAILYLOOK*, and *Nordstrom's Trunk Club* for fashion appeals; *Birchbox* for personal care products, to name a few.

In general, image classification poses an exciting computer vision puzzle and has gotten many deep learning researchers' attentions. Building a classification model for fashion product images would be an excellent start to dive into deep neural networks. This project aims to get myself hands-on experience dealing with the imbalanced dataset, building Convolutional Neural Networks (CNN) for image classification with **Keras API** and evaluating the model performances.

1. Data sources

Fashion Product Images (545.62 MB, 44k colored images of size 80 x 60 x 3)

File descriptions

- **styles.csv** contains 44446 rows and 10 columns.
  Columns/features: id, gender, masterCategory, subCategory, articleType, baseColour, season, year, usage, productDisplayName

- **44441 images** (*.jpg). Five images (id = 12347, 39401, 39403, 39410, 39425) are missing.

Take-away

- We focus on the three product categorical features: masterCategory, subcategory, articleType.
- The number of categories at each level are **7** (Figure 1), **45** (Figure 2), and **142** (Figure 3).
- The sample counts (y-axis, in logscale) at each categorical are highly imbalanced.
- What's the relation between different categorical levels? (See the sunburst plot in EDA section).

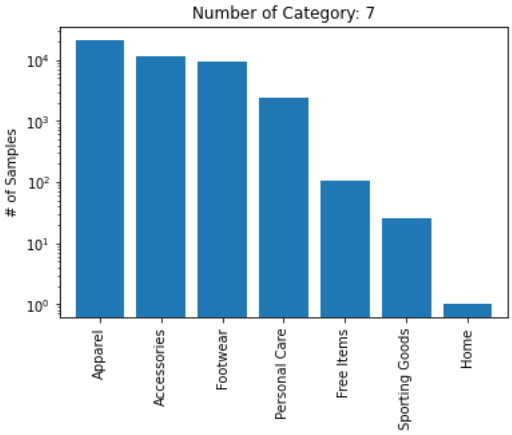

Figure 1: Item (image) counts at the categorical level 'masterCategory' in the default data frame (styles.csv).

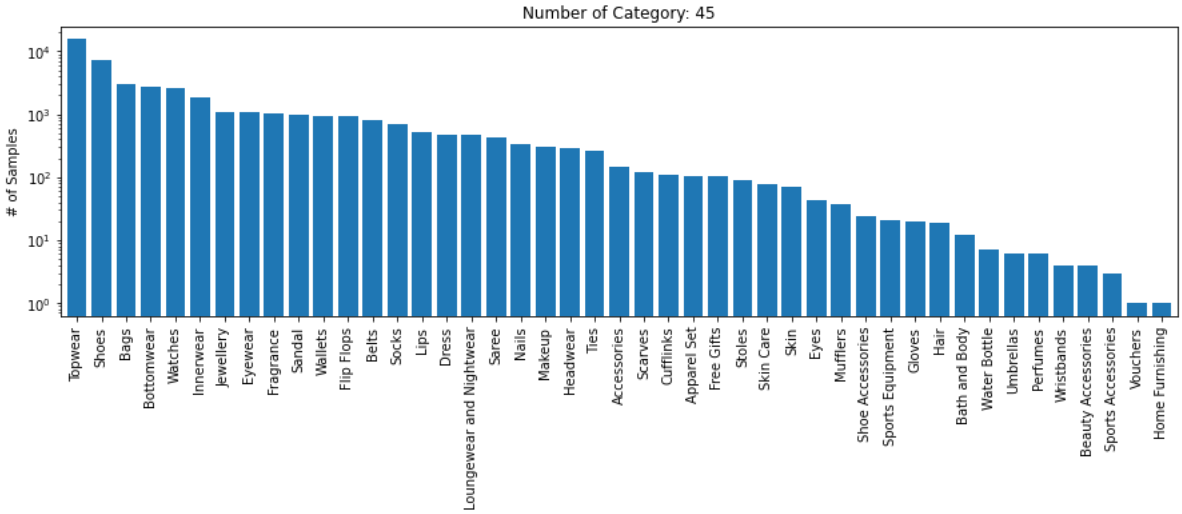

Figure 2: Item counts at the categorical level 'subCategory' in the default data frame (styles.csv).

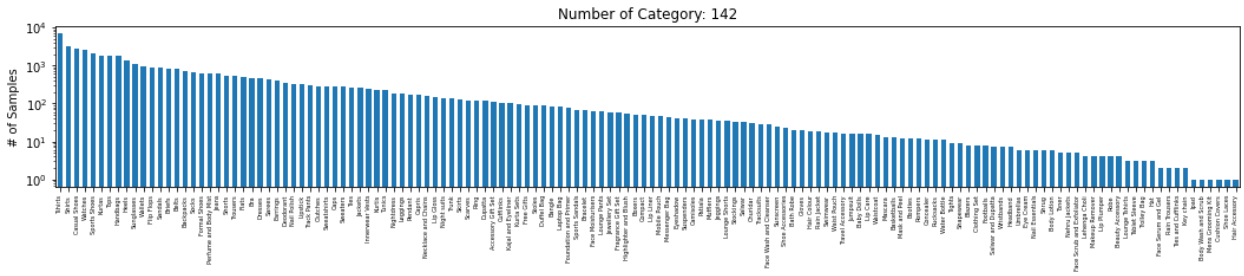

Figure 3: Item counts at the categorical level 'articleType' in the default data frame (styles.csv).

2. Data Wrangling and Feature Engineering

Here, we focus on data cleaning and feature engineering of **style.csv**. We keep the images unprocessed until we use Keras API to do the in-place/on-the-fly data augmentation when training the CNN model [Ref. 1].

2.1 Drop the Rows with Missing Images

We dropped the rows without the corresponding images (id = 12347, 39401, 39403, 39410, 39425).

2.2 Refine the Product Class for Image Classification

We defined a new categorical feature, `Class`, which combines the three hierarchical labels (`masterCategory`, `subCategory`, `articleType`). The **number of categories in** `Class` is **no larger than 35**.

Here is the procedure for getting the `Class`:

1) Initialize `Class` with the `masterCategory`.
2) Group by `subCategory`. If the sample count is greater than 105, we keep that `subCategory` in the `Class`.
3) Group by `articleType`. We kept the `articleType` **Tshirts** and **Shirts** in `Class` as they are two dominated products in **Topwear**.
4) Check the sample count in each `Class`.
   We dropped the labels whose sample count is less than 5 from `Class`, because those categories do not have enough samples (images) for training and prediction.
5) After data cleaning, the number of samples (rows) is 44440.

Take-away

- The size of refined DataFrame `style` is 44440-by-11.
- The new categorical feature `Class` has 31 product labels.
- However, about 105 samples listed as **Free Items** are mislabeled. We will address it in next section.

2.3 Assign Classes to the Images Under the Ambiguous Category - Free Items

How to relabel the ***Free Items***? Seeing Table 1, the column `productDisplayName` contains various product descriptions/keywords, such as “watch”, “Saree”, etc. We can reassign proper `Class` to *Free Items* based on these keywords,

1) For each mislabeled product, we compute the **Text Similarity** (cosine similarly) between its `productDisplayName` and the rest **30 product categorical names.**
2) Among the 30 similarity values, the maximal one should be the `Class` of the product. Replace the current Class (*Free Items*) with the predicted/matched category.
3) For items without matched category, we labeled them as *Unknow*. We will predict all *Unknow* items using our trained CNN model.

2.4 Export the Refined Spreadsheet

The refined spreadsheet (44440-by-11) was saved as **styles_cleaned.csv**.

*Table 1 shows subset of products listed as Free Items. There are 105 products in the Free Items class. Their **Class** could be reassigned based on the column **productDisplayName**.*

id	gender	masterCategory	subCategory	articleType	baseColour	season	year	usage	productDisplayName	Class
53598	Men	Free Items	Free Gifts	Free Gifts	Black	Winter	2016	Sports	Q&Q Men Black Digital Watch	Free Items
49696	Men	Free Items	Free Gifts	Ties	Blue	Summer	2012	Formal	Park Avenue Blue Patterned Tie	Free Items
53591	Men	Free Items	Free Gifts	Free Gifts	White	Winter	2016	Casual	Q&Q Men White Dial Watch	Free Items
45894	Men	Free Items	Free Gifts	Free Gifts	Black	Winter	2016	Casual	Maxima Men Black Dial Watch	Free Items
53596	Men	Free Items	Free Gifts	Free Gifts	Navy Blue	Winter	2016	Casual	Q&Q Men Navy Blue Digital Watch	Free Items
52676	Women	Free Items	Free Gifts	Free Gifts	White	Winter	2016	Casual	Morellato Women White Dial Watch	Free Items
29975	Women	Free Items	Free Gifts	Free Gifts	Black	Winter	2016	Casual	Aspen Women Black Dial Watch	Free Items
29981	Women	Free Items	Free Gifts	Free Gifts	Blue	Winter	2016	Casual	Aspen Women Blue Dial Watch	Free Items
51091	Women	Free Items	Free Gifts	Free Gifts	Green	Summer	2012	Ethnic	Satya Paul Green Saree	Free Items
54970	Women	Free Items	Free Gifts	Free Gifts	Pink	Spring	2017	Casual	Rasasi Women Body Lotion	Free Items

2.5 General Balance Measurement for Categorical Variable

Consider a categorical variable with ***C*** classes and each class i has sample count n_i, then its sample proportion (probability) can be written as

$$p_i = \frac{n_i}{N}, \qquad N = \sum_{i=1}^{C} n_i ,$$

where N is the total number of samples across all ***C*** classes. The entropy of the categorical variable is

$$H = -\sum_{i=1}^{C} p_i \log p_i .$$

Based on the **principle of maximum entropy**, the discrete proportion distribution yielding the maximal entropy is the **uniform distribution**, that is, each class has the equal number of samples,

$$n_i = \frac{N}{C}, \qquad p_i = \frac{1}{C}, \qquad H_{max} = \log C .$$

The measure of dataset balance can be defined as

$$Balence = \frac{H}{H_{max}} = -\frac{1}{\log C}\sum_{i=1}^{C} p_i \log p_i ,$$
$$= -\frac{1}{\log C}\sum_{i=1}^{C} \frac{n_i}{N}\log\frac{n_i}{N},$$

where $Balence$ is in range between 0 (highly imbalanced) and 1 (perfectly balanced).

Take-away - the measure of balance for categorical features are summarized below:

Categorical Feature	Measure of Balance	Sample Distribution
masterCategory	0.671	Figure 1
subCategory	0.644	Figure 2
articleType	0.731	Figure 3
Class	0.815	Figure 5

Our new defined feature `Class` improved the labeling balance compared to the other three.

3 Exploratory Data Analysis

For multi-classification problems, it is essential to check the number of samples (images) in each class. Ideally, we preferred the representations to be about the same for every class (balanced dataset). Yet reality, the sample proportion would vary with classes.

We firstly examine the proportion of categorical labels in the raw data again (**styles.csv**), and the one after feature engineering (**styles_cleaned.csv**). Then, we randomly show subset of images for qualitatively checking how product varies within the same category (in notebook).

Proportion of the categorical tags/labels

The original data frame (**styles.csv**) labels each product image with three hierarchical levels (`master`, `subCategory`, and `articleType`). Figure 1, Figure 2 and Figure 3 show the sample count (in logscale) across the product labels within one of the categorical levels. The sunburst plot displays the hierarchical relation between the three levels.

Take-away 1:

- The number of samples for each Category is highly imbalanced.
- Four dominated masterCategory are 1) *Apparel*, 2) *Accessories*, 3) *Footwear* and 4) *Personal Care*.
- Around 50% of samples are *Apparel*, in which *Topwear* is the dominated subCategory.

In **styles_cleaned.csv** we added a new categorical feature, `Class`. The sunburst plot (Figure 4) shows the categorical proportions at the level of `Class` and `articleType`.

Take-away 2:

- The categorical feature has `Class` 30 classes (Figure 5).
- The number of samples `Class` is still imbalanced, but the measure of balance is improved from 0.73 to 0.81 (Sec. 2.5).
- Most common classes are Shoes, Tshirt, Topwear
- In contrast the small components like the Scarves, Cufflinks, Apparel Set and Sporting Goods are quite rare in our data.
- We might expect less accurate predictions on these minor classes.
- Whether the accuracy is the right score to measure the performance and validation?

*Figure 4: Proportion of categorical labels in the original data frame (styles.csv). The inner ring is the **masterCategory**; the middle one is **subcategory**; the outer is **articleType**.*

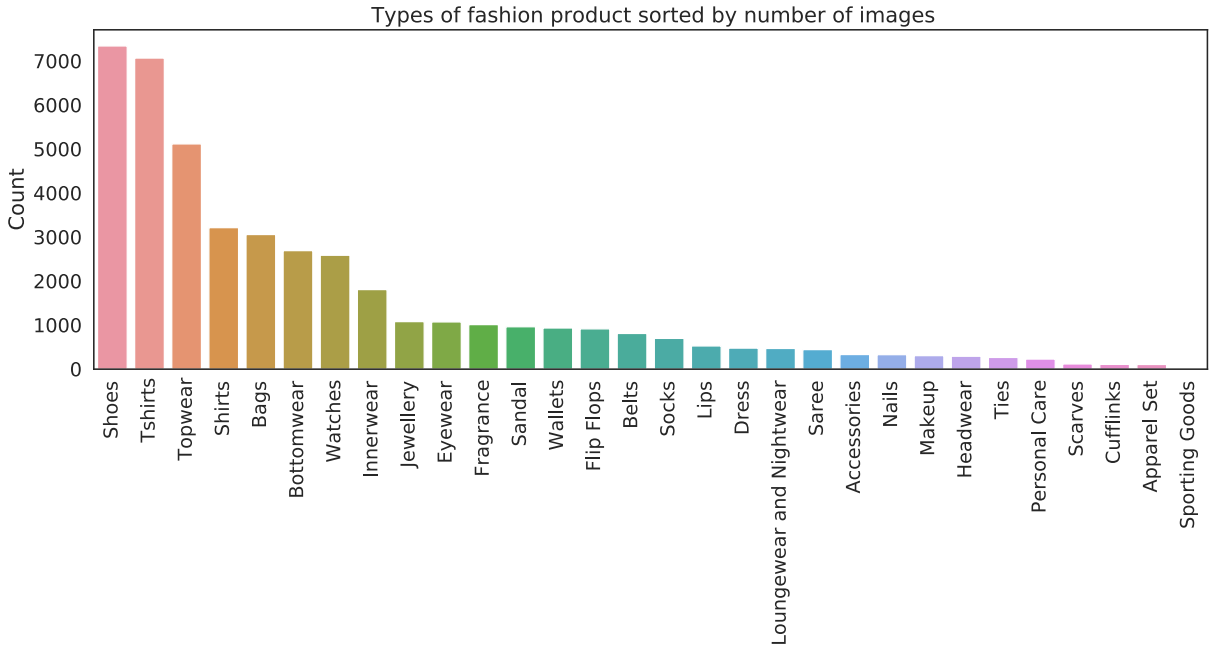

Figure 5: Item counts at the categorical level 'Class' in styles_cleaned.csv.

*Figure 6: The proportion of refined categorical labels. The inner ring shows the refined **Class** with 30 product labels; the outer ring shows the labels of **articleType**.*

4 Pre-processing and Training Data Development

Scale the Pixel Values - Each image is in size of 80-by-60 with 3 color channels, and the value is in the range of 0 to 255. With Keras *ImageDataGenerator* class, we assign the rescale factor 1.0/255 to transform the pixel values to range of 0-1, as preferred for neural network models. In addition, Keras API provides option to train with color (default setting) or grayscale images (color_mode="grayscale").

Image Augmentation - With ImageDataGenerator, we can easily augment the data by flipping, shifting, or rotating the image randomly. Here, we only perform horizontally flipping randomly on images. Most of the pictures are well centered and correctly orientated, so applying shifts or rotations would not improve the model performances.

Split Train-validation and Hold-on Sets - We split the data into a **training (72%)** and a **validation (18%)** and a **hold-on (10%)** sets. We pick the best model based on the validation accuracy and confirm the model performances with the hold-on set that, ideally, should have an accuracy about the same as the validation. We set a **batch size of 32**. Each of the train and validation datasets are divided into groups of 32 images.

5 CNN Model Architectures

The five basic layers in CNN includes

(1) input layer,
(2) **convolutional (Convo) layer**,
(3) **pooling layer**,
(4) **fully connected (FC) layer**, and
(5) output layer.

In addition, other operational layers usually appearing between the five basic layers are

(6) dropout layer,
(7) batch normalization (BatchNorm) layer, and
(8) the activation function (ReLU, Softmax).

Here, we set the CNN architecture as:

Input → Convo (ReLU) → BatchNorm → Convo (ReLU) → BatchNorm → Max Pooling 1 → Dropout (0.2)
→ Convo (ReLU) → BatchNorm → Convo (ReLU) → BatchNorm → Max Pooling 2 → Dropout (0.2)
→ FC layer (ReLU) → BatchNorm → Dropout (0.25)
→ Output (Softmax)

CNN model in Keras API:

```
## Build the model using the Keras API
chanDim = -1  # RGB Channel is last
model = tf.keras.models.Sequential()

# 1st CONV & RELU => BatchNorm => POOL layer set
model.add(Conv2D(32, (3, 3), activation='relu', padding='same',
                 input_shape=inputShape))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

# 2nd CONV & RELU => BatchNorm => POOL layer set
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

# first (and only) set of FC(fully connected) => RELU layers
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.25))

# softmax classifier
model.add(Dense(classes, activation='softmax'))
```

We add two sets of convolutional and pooling layers, one with 32 filters and the other with 64. Having more layers does not improve the accuracy significantly but increases the training time. The number of trainable parameters is summarized in Figure 7.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 80, 60, 32)        896
_________________________________________________________________
batch_normalization (BatchNo (None, 80, 60, 32)        128
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 80, 60, 32)        9248
_________________________________________________________________
batch_normalization_1 (Batch (None, 80, 60, 32)        128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 40, 30, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 40, 30, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 40, 30, 64)        18496
_________________________________________________________________
batch_normalization_2 (Batch (None, 40, 30, 64)        256
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 40, 30, 64)        36928
_________________________________________________________________
batch_normalization_3 (Batch (None, 40, 30, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 20, 15, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 20, 15, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 19200)             0
_________________________________________________________________
dense (Dense)                (None, 512)               9830912
_________________________________________________________________
batch_normalization_4 (Batch (None, 512)               2048
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 30)                15390
=================================================================
Total params: 9,914,686
Trainable params: 9,913,278
Non-trainable params: 1,408
```

Figure 7: CNN model summary table shows the trainable and non-trainable parameters in each layer, given the image size (80, 60, 3). The two blue blocks indicate two sets of convolutional and pooling layers; the green block is the fully connected layer.

6 Model Prediction and Evaluation

6.1 Loss and Accuracy

CNN model with the same architecture but trained with either color or grayscale images. The two model have similar performances.

Dataset		*Loss*	*Accuracy*
Color images (80 x 60 x 3)	Training	**0.196**	**0.935**
	Validation	0.285	0.913
	Hold-on (test)	**0.189**	**0.940**
Gray images (80 x 60 x 1)	Training	0.162	0.946
	Validation	**0.273**	**0.917**
	Hold-on (test)	0.191	0.938

6.2 Classification Report & Confusion Matrix

Classification report summarizes the standard evaluations for classification model, including **precision**, **recall** (or sensitivity), **f1-score**.

Confusion matrix shows the prediction results versus the actual labels, and the off-diagonal entries indicate the misclassified sample counts made by the classifier. This gives us insights into how accurately the model classified a class's inputs and which type of input the classifier is prone to fail.

A. Model trained with color images (validation sets)

The classifier has the least performance on **Scarves (#20)** and **Sporting Goods (#24)**.

- Note that the **Sporting Goods** has the least number of images (25 total; 19 used in training) across all classes, thus we expected our model made more error in **Sporting Goods** than in the rest.
- For **Scarves**, the prediction fails when photo shows a model wearing a scarf, or its texture/shape is like other products (Accessories, Jewellery, or socks), see Figure 8.

Label	*precision*	*recall*	*f1-score*	*support*
0	0.804	0.552	0.655	67
1	0.929	0.684	0.788	19
2	0.963	0.955	0.959	516
3	0.994	0.981	0.987	159
4	0.944	0.954	0.949	461
5	1.000	0.692	0.818	13
6	0.606	0.645	0.625	93
7	0.972	0.989	0.980	177
8	0.824	0.886	0.854	175
9	0.975	0.900	0.936	170
10	0.933	0.894	0.913	47
11	0.962	0.919	0.940	333
12	0.707	0.968	0.817	189
13	0.874	0.835	0.854	91

14	0.781	0.595	0.676	84
15	0.837	0.672	0.745	61
16	0.961	0.961	0.961	51
17	0.786	0.733	0.759	45
18	0.901	0.694	0.784	183
19	0.916	0.989	0.951	88
20	0.667	0.455	0.541	22
21	0.901	0.907	0.904	549
22	0.965	0.974	0.970	1346
23	0.925	0.954	0.939	130
24	0.500	0.333	0.400	3
25	1.000	0.983	0.991	58
26	0.791	0.899	0.842	911
27	0.941	0.875	0.907	1260
28	0.927	0.938	0.933	177
29	0.998	0.988	0.993	515
accuracy			**0.911**	7993
macro avg	0.876	0.827	0.846	7993
weighted avg	0.915	0.911	0.911	7993

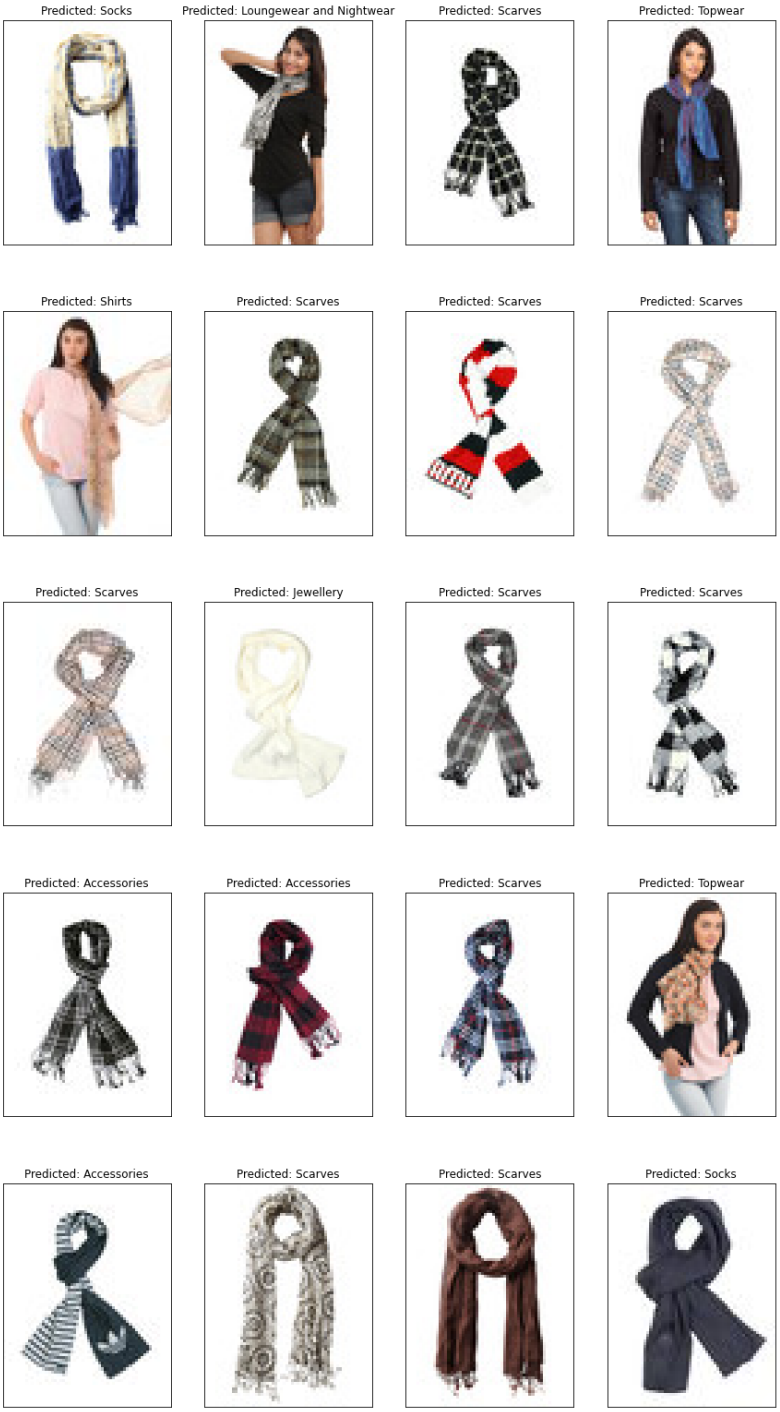

Figure 8: Predicted labels for "Scarves."

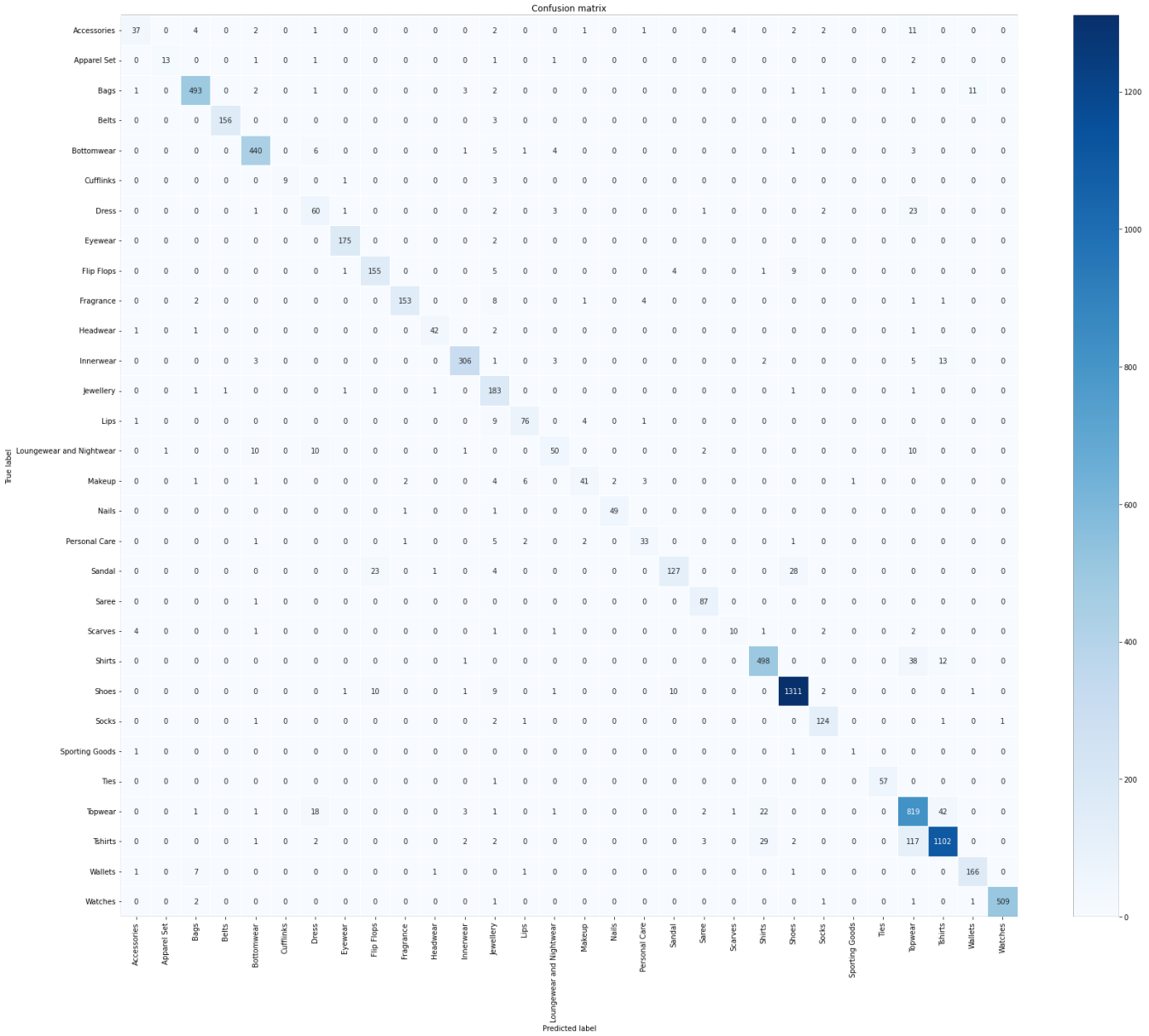

*Figure 9: Confusion matrix shows the true and label when testing the model on the **validation** sets. This model is a CNN classifier trained with the color images. The model is commonly confused with "Sandal" and "Shoes"; "Topwear" and "Shirts".*

Figure 10: Example of misclassified items. The model is commonly confused with "Sandal" and "Shoes"; "Topwear", "T-Shirts" and "Shirts". These items are sometimes ambiguous to classified with human eyes.

B. Model trained with grayscale images (validation sets)

The classifier has the least performance on **Sporting Goods (#24)** as there's not enough samples in the dataset. The performance on **Scarves (#20)** has improved compared to the model trained with color images.

Label	*precision*	*recall*	*f1-score*	*support*
0	0.915	0.642	0.754	67
1	0.895	0.895	0.895	19
2	0.982	0.932	0.956	516
3	0.975	1.000	0.988	159
4	0.935	0.970	0.952	461
5	0.923	0.923	0.923	13
6	0.683	0.742	0.711	93
7	0.989	0.989	0.989	177
8	0.751	0.949	0.838	175
9	0.851	0.976	0.910	170
10	0.936	0.936	0.936	47
11	0.961	0.892	0.925	333
12	0.950	0.910	0.930	189
13	0.810	0.890	0.848	91
14	0.867	0.619	0.722	84
15	0.826	0.623	0.710	61

16	0.926	0.980	0.952	51
17	0.833	0.667	0.741	45
18	0.888	0.694	0.779	183
19	0.946	0.989	0.967	88
20	0.708	0.773	0.739	22
21	0.875	0.929	0.901	549
22	0.968	0.972	0.970	1346
23	0.976	0.931	0.953	130
24	0.500	0.333	**0.400**	3
25	1.000	1.000	1.000	58
26	0.848	0.843	0.845	911
27	0.907	0.913	0.910	1260
28	0.896	0.977	0.935	177
29	0.998	0.992	0.995	515
accuracy			**0.917**	7993
macro avg	0.884	0.863	0.869	7993
weighted avg	0.919	0.917	0.916	7993

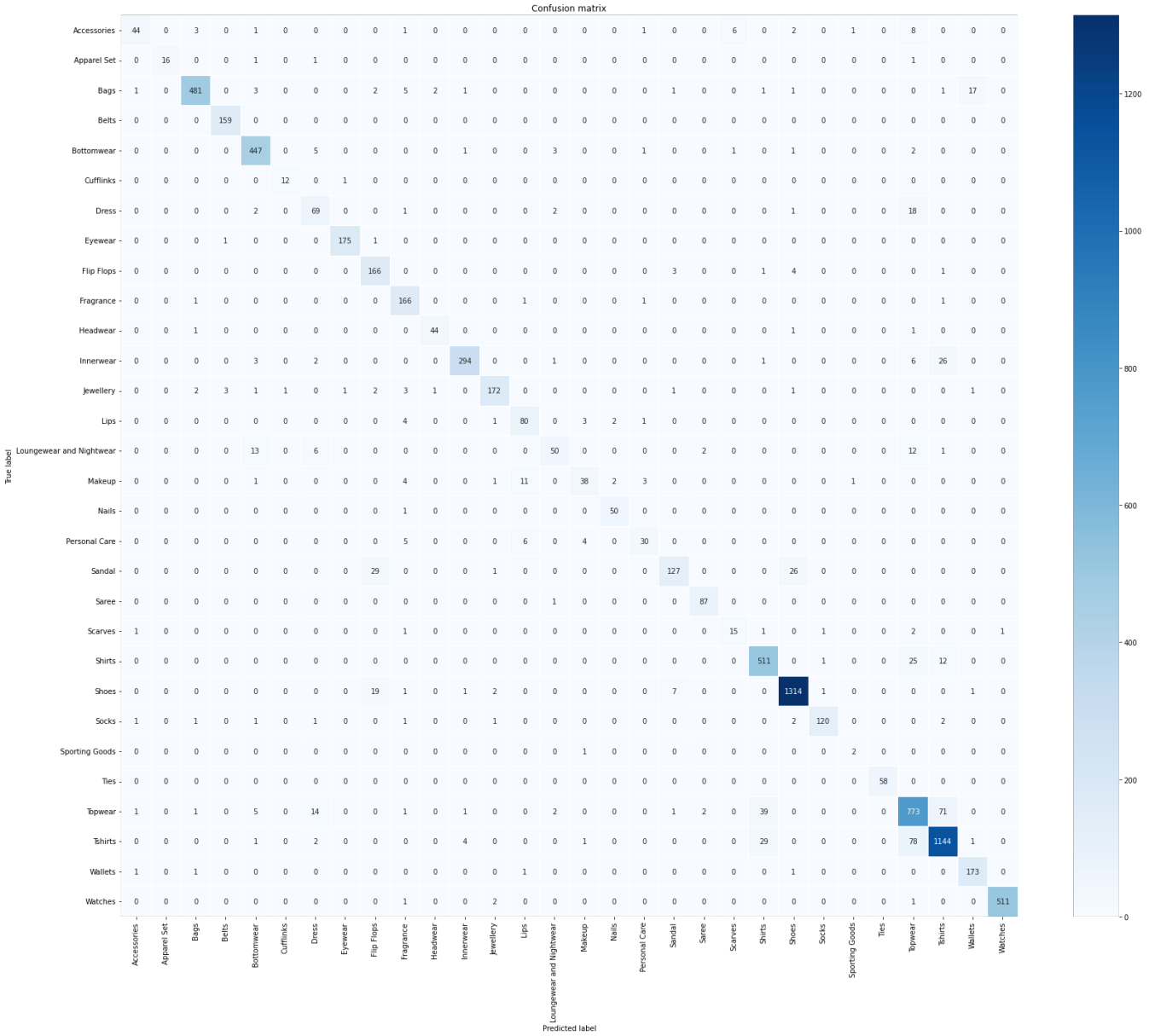

Figure 11: Confusion matrix shows the true and label when testing the model on the validation sets. This model is a CNN classifier trained with the grayscale images. Again, Sandal" and "Shoes"; "Topwear", "T- Shirts" and "Shirts" are ambiguous for the classifier.

7 Prediction on the Unknow Items

The predations on the mislabeled images that were originally listed as "*Free Items.*"

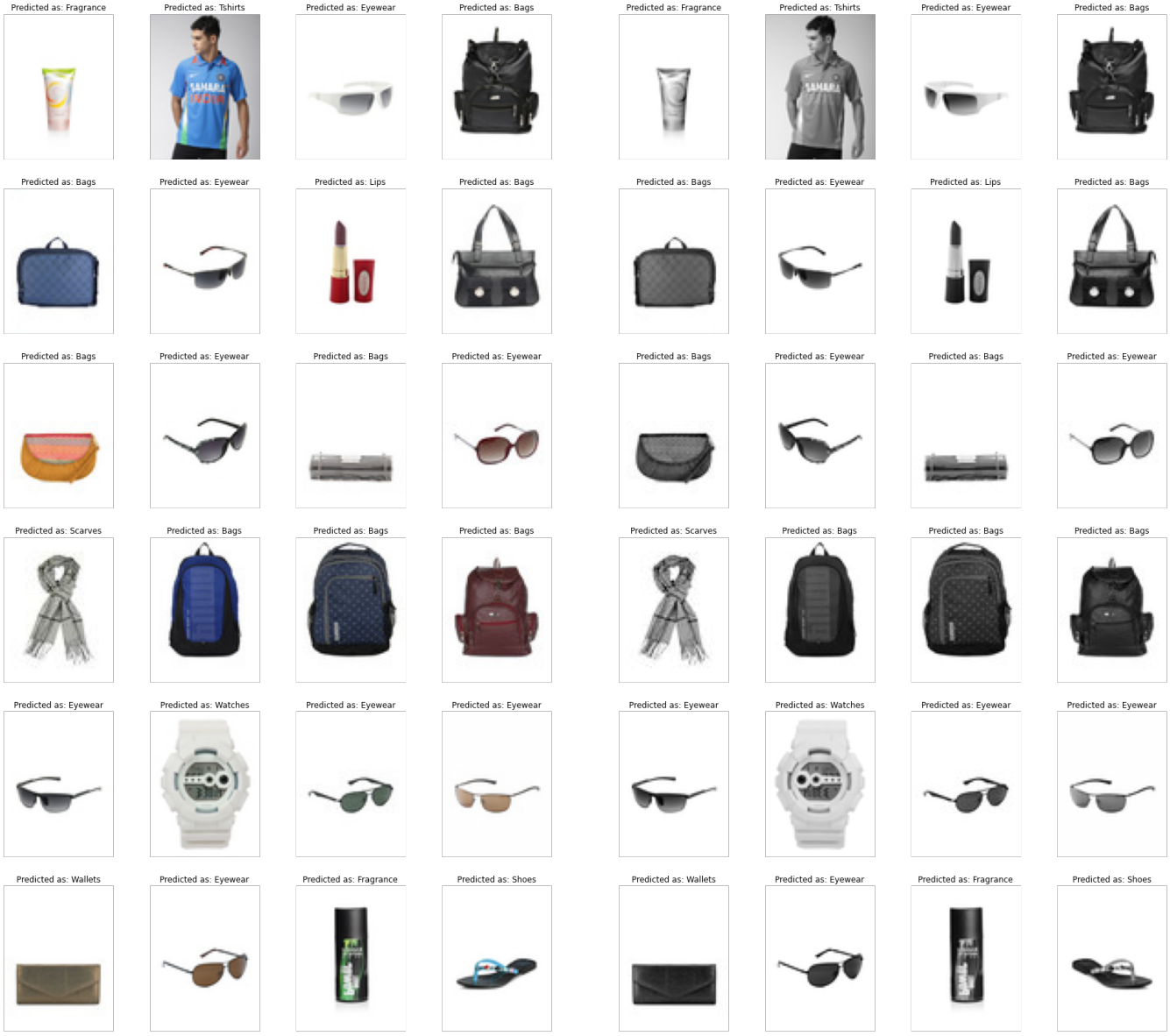

Figure 12: Perditions on unknown items. The text on the top of each image is the predicted label. The model trained with color image (left), or grayscale image (right) gives the same results.

8 Conclusion

We trained CNN model on classifying fashion product images (color and grayscale) into **30 classes**, and both reached the accuracy of 0.91 - 0.92 on the validation set. Considering the provided product labels are imbalanced and have high cardinality (too many unique values), we defined a new set of 30 classes based on the three categorical features in the original data. As expected, the model has the least performance in the class with fewer samples ("Sporting Goods," 25 images). Besides, the classifier is commonly confused when images belong to the same master category (ex: "Topwear", "T-shirt" and "Shirts" are in the same mastercategory "Topwear"), or high similarity in product shape (Sandal" and "Shoes"). This could be improved by providing images with higher resolution, data augmentation, or refining the image labels.

9 Future Work

- To improve the model classifying the fined details in products, one can use higher resolution images for training. Yet, this would cost more training time as trainable parameters will be scaled up with the input dimensions.
- Try fine tune the current CNN architectures. For instance, adding a pooling layer after each convolutional layer; for the fully connect layers, using two small dense layers (at size of 256 and 128) instead of a single large dense layer.
- There are countless CNN architectures or pre-trained models. One could use transfer learning to take advantage of those pre-trained models (ex: ResNet, VGG).
- Computer vision is the thrived research field, and new ideas/models are coming out time by time. The recent one capturing my eye is Vision Transformer (ViT) in 2021.

10 Reference

1. https://www.pyimagesearch.com/2019/07/08/keras-imagedatagenerator-and-data-augmentation/
2. Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020).