LSDA Competition I Report

# iMaterialist Challenge (Fashion) at FGVC5

Team Bricksquad


| Christian Salbk | Juozas Miskinis | Isabela Blucher | Alexander Andersen |
|---|---|---|---|
| dml349 | xgs111 | rdz223 | ldp372 |


May 14, 2018

## 1 Motivation

Team Brick Squad is participating in the *iMaterialist Challenge (Fashion) at FGVC5* from the machine learning competition web site *Kaggle*. The goal of the competition is to come up with an algorithm, that can classify fashion elements in pictures, like a dress or a purse. The problem is a multilabel classification problem - meaning it is a supervised learning problem with labeled training data and one image can have multiple labels. The evaluation metric is the F1 score, which is a balanced measure combining precision and recall that works well with uneven datasets.

## 2 Data

The competition has training, validation and test sets which were downloaded using JSON files with image IDs and URLs. Table 1 shows the distribution of images and labels among these datasets.

| | Number of images | Number of unique labels |
|---|---|---|
| Training set | 1014544 | 228 |
| Validation set | 9897 | 225 |
| Test set | 39706 | 0 |

Table 1: Number of images and labels for each dataset

Some images are single labeled and some have up to 23 labels. We can see that our test set has no labels because our objective is to predict them with our model.

### 2.1 Preprocessing

For preprocessing the images were downloaded using an adapted version of the code from Kaggle[1]. Then the images were resized to $32 \times 32$ pixels and stored in numpy arrays with a coordinate for each color red, green and blue. The arrays where then flattened so that we separate the 3 color channels, and with that we get $32 \times 32 \times 3 = 3072$ columns that represent the features. Our final data structure is now an $n \times 3072$ matrix, with $n$ being the number of images. The Pillow library was used for this preprocessing step.

For preprocessing the labels, the JSON files were read and the labels were separated according to their imaged IDs, using a dictionary with the IDs as keys and the labels as values. After having them stored, we create a binary matrix $B$ with dimensions $n \times 228$ with n being the number of images and the columns representing all possible labels, so that $B_{ij} = 1$ if the $i$-th image has the label $j$ and $B_{ij} = 0$ on the contrary.

### 2.2 Feature extraction

We have implemented the Gray-Level Co-Occurence Matrix (GLCM) and 9 features derived from it. The GLCM tells us something about the spatial relationship of pixels in the images but tells us nothing about the colors of the images, which is likely relevant for clothes. Therefore will we look to extract more features in addition to GLCM derived features. Furthermore these GLCM derived features have not been used with the model as our feature set is not complete yet but also because no proper segmentation have been implemented. So far we have reduced the images and flattenend them for use as features. The 9 features are based on features found in a paper by Peter A. Freeborough and Nick C. Fox[4]. The features are as follows:

- Angular Second Moment
- Correlation
- Contrast
- Variance
- Inverse Difference Moment
- Sum Average
- Entropy
- Difference Variance
- Difference Entropy

While this is only 9 features, we will be able to extract more than that as we can define more than one GLCM per image by using different offset and angle values. By using only a single offset value with three angles (horizontal, vertical and diagonal) we can extract 27 features total per image. We plan to utilize feature selection to determine which features give the best signal.

## 3 Approach

We decided to work with the K-Nearest Neighbors algorithm since it is a well known and simple approach to start with. We use an adapted version of the algorithm that is built on top of the `scikit-learn` version, from a multilabel classification package for Python [3].

## 4 Results

By applying our preprocessing and approach to the dataset and computing the F1 score, we get an accuracy of 24% using 10000 instances of the training set. We tried using PCA to reduce the dimensionality of the dataset and see how our reults would change. So doing PCA with 40 and 50 components gave us F1 scores of 20% and 21%, which shows a slight decrease in accuracy.

## 5 Conclusions

It is possible to conclude that a simple adapted k-NN model for multilabel classification could be good for smaller problems, but our datasets are too big for this model to yield ideal results. Our position in the Kaggle leaderboard as of the last submission is 98th out of 145 teams with a score of 0.22743.

For the next round we would like to improve our classification by making use of a neural network, training it on the full dataset and making our preprocessing better applying the feature extraction devised in Section 2.2.

## 6 Individual Contributions

- Christian: feature extraction, report writing
- Juozas: configure Azure, implement knn, structuring Github
- Isabela: setup Github, download images, setup Overleaf, preprocess labels, format submission file, review report
- Alexander: preprocessing images, report writing

It is important to point out that our group had weekly meetings and we collaborated a lot in the project.

## References


[1] Kaggle: *iMaterialist downloader*
https://www.kaggle.com/nlecoy/imaterialist-downloader-util/code

[2] pyimagesearch: *k-NN classifier for image classification*
https://www.pyimagesearch.com/2016/08/08/k-nn-classifier-for-image-classification/

[3] scikit-multilearn: *skmultilearn.adapt package*
http://scikit.ml/api/api/skmultilearn.adapt.html?highlight=adapt#module-skmultilearn.adapt

[4] Peter A. Freeborough and Nick C. Fox: *MR Image Texture Analysis Applied to the Diagnosis and Tracking of Alzheimers Disease*. IEEE TRANSACTIONS ON MEDICAL IMAGING, June 1998.