Immaterialist Fashion - Report

Mick van Hulst Dennis Verheijden Roel van der burg Brian Westerweel
Joost Besseling

June 5, 2018

1 Introduction

This report describes the results for the first competition of the course Machine Learning in Practice. The source code for this project can be found by going to the public Github repository.

During the project each of the team members were assigned specific tasks. Do take note that our approach can be considered agile as we switched roles constantly and helped each other out where we could:

- **Mick van Hulst:** Exploratory data analysis, GCP, Batch generator, implementing ResNet
- **Dennis Verheijen:** Implementing networks, general code cleaning, weight determination
- **Roel van der Burg:** Exploratory data analysis, implementing ResNet
- **Brian Westerweel:** Implementing custom F1-measure, Object-detection, Report
- **Joost Besseling:** Object-detection, Ensemble algorithm

2 Problem statement

We have chosen the immaterialist fashion challenge on Kaggle. This challenge consisted of a data set of approximately 1.1 million images which had to be assigned one or multiple labels. In total there are 228 labels that could be assigned of which most don't make sense for the human mind.

3 Data set

Kaggle provided a training, validation and test set, combined they consist of approximately 1.1 million images. For each image there is a list containing one or more labels. As mentioned earlier for most of the labels it is unclear why images have a certain label (figure 1).

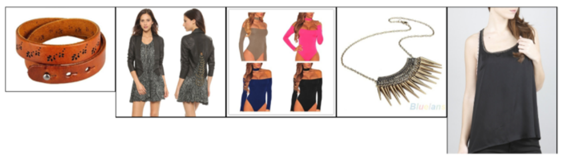

Figure 1: Example images for label 24.

When looking at the individual images label 24 could either be a shirt, necklace, skirt or bracelet. However, apparently there is a more general feature in this images that is represented by label 24 which we couldn't think of.

Another important characteristic of this data set is that it is quite imbalanced (figure 2). There are around 20 labels which occur more than a 100.000 times each while most of the other labels have roughly 30.000 occurrences.

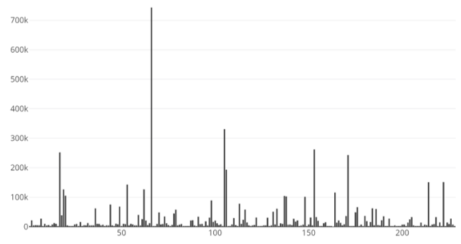

Figure 2: Data distribution, each value on the x-axis representing the label and on the y-axis are the occurences for the labels.

The imbalance in the data set posed the challenge of properly training our network by making sure that it doesn't label all images with the most occurring label. This can, for example, be done by setting weights to the different classes so incorrectly labeling a rare label is more heavily penalized than incorrectly labeling a very common label.

Data Preprocessing

The data for this challenge didn't need a lot of sanitizing. However, the images were of different sizes and because convolutional neural networks need a static input size we needed to make sure that all images that were fed to the network had the same size. This is accomplished by implementing a patch generator that cut same-sized patches from the source images.

It has to be noted that due to the gigantic disc size that was needed for the data we decided to download the images at 85% of the available quality.

4 Models

In contrast to the last project we decided to use pre-trained model for this challenge, instead of fully training the models ourselves.

The models used in this project are pre-trained using ImageNet [1]. ImageNet is a collection of over 100.000 concepts with average 1000 images per concept. Also, ImageNet is often used as a reference data set for setting convolutional neural network benchmarks.

For this project MobileNet, InceptionV3, Xception and ResNet50 are chosen because these model individually have a good performance for image classification problems. Therefore, we think that combining the models with ensemble learning methods will yield even better results. Besides the well-known models we created a model based on object detection.

In this way we ensure that the data knows all the possible labels it has to predict and the relation between the target data and the labels. By using the pre-trained weights for the different models they are already aware of the high-level abstraction for images. These abstractions have proven to work well on image classification tasks [2]. For fine-tuning the different models the last layers of each models are unfrozen. Unfreezing the last layers make it possible to train them on another data set. These partly unfrozen models are trained on the training data from the challenge.

Because the data was imbalanced to train the models on three data sets. The first one consists of all available training data. The other two data sets are mutually exclusive subsets of the total data set. One contains only the labels that occur more than 100.000 times, while the other contains the labels that have less than 100.000 occurrences.

For the loss of all models binary cross entropy is used, because the model has to output a probability for each label. To measure performance of the models a micro-averaged F1-score is used, because this also is the performance metric used by Kaggle. Hyper parameters are left as they were implemented by Keras.

4.1 Object detection

For the object detection model we first predicted objects for each image using a pre-trained ResNet50 [3] trained on the open image data set [4]. For each image we used the found objects as features for that image and used those features to train another neural network.

However, because the object detection network very often detected just a few objects. Because the object detection network resulted in very small features per image we found very early on that this network would not yield any noteworthy results. Therefore, this model has not been further developed.

4.2 ResNet50

The ResNet we used is based on Keras' version [3], where the last layer was unfrozen and fine-tuned to our target data set.

TODO: MORE EXPLANATION + RESULTS, ROEL HAS SUMBISSION

4.3 MobileNet

MobileNet again is based on a version of Keras [5]. In this case the last 5 layers are unfrozen and trained on the training images of this challenge.

TODO: MORE EXPLANATION + RESULTS, NOT SUBMITTED (At least with thresholds)

4.4 InceptionV3

The InceptionV3 [6] is a convolutional neural network developed by Google. Inception has a very big difference with other conventional neural networks because of how their layers are build.

In most convolutional networks a decision has to be made for each layer in which you can either do 5x5, 3x3 or 1x1 convolutions, max-pooling. Inception refuses to make the choice for one layer above the other and decided to do everything within one layer. This architecture makes the network very time expensive to train because it is basically brute-forcing the different options that are available.

TODO: RESULTS

4.5 Xception

5 Thresholding

TODO: HOW DID WE THRESHOLD

6 Ensembling

The data set is highly imbalanced. To counteract this we decided to train two special classifiers. The first classifier is only trained on the labels that occur at least 100 000 times. The other classifier is trained on the remaining labels. These two classifier are combined into one classifier, by simply . The imbalance for both classifiers is lower than the imbalance for the total data set.

We also trained multiple networks on all labels. We now want to combine these networks to give a single prediction on the test set. We will ensemble them with weighted soft voting.

To calculate the weights we generate the predictions for the validation set, and then use an optimization algorithm to find the optimal weights on this set. We have implemented a simple hill climbing algorithm and also PSO (Particle Swarm Optimization) to calculate the weights.

An interesting side effect of using this weighting algorithm is that we find which models work well, and which models don't work as well.

Results for:

simple average hill climbing pso harmonic mean

We can see that the results of the PSO algorithm are the most convincing. If we look at the weights that are generated by this strategy, we can see that the combined classifier is relatively important. This validates our idea that we could counteract the imbalance by training two seperate classifiers.

TODO: make a nice table of the ensembling results. And say something about these results.

7 Conclusion

TODO: WHAT CAN WE SAY?

8 Reflection

This section briefly discusses some things that stood out during this project.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[2] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, pages 512–519. IEEE, 2014.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[4] Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Shahab Kamali, Matteo Malloci, Jordi Pont-Tuset, Andreas Veit, Serge Belongie, Victor Gomes, Abhinav Gupta, Chen Sun, Gal Chechik, David Cai, Zheyun Feng, Dhyanesh Narayanan, and Kevin Murphy. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://storage.googleapis.com/openimages/web/index.html*, 2017.

[5] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[6] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.