An Artificial Intelligence Approach to the Clothing-sorting Problem

Pinto, Jorge

Abstract—The proposal of using Artificial Intelligence in a simple clothing sorting task is evaluated. To properly get reference points for comparison, two sets of data were used to train a neural network based in the googLeNet® model: a former one already provided for study, and a second one acquired via an executing script. Both groups were different in terms of the classes to be classified and the application they were intended to. Two classification datasets were generated prior training and learning rates oscillated between 0.01 and 0.008 between runs, while epochs were changed from 8 to 15 during analysis. Prediction tables were gathered in the tested scenarios and thoroughly compared against each other. The initial idea of a self-picking clothing robot was positively evaluated in terms of generated results, but false positives could exist if the input's format and features steer away from those used for training.

Index Terms—Artificial intelligence, inference, DIGITS workflow, clothing-sorting.

1 Introduction

Artificial Intelligence plays a major role in our daily activities. Many applications using this principle work in today's smart-phones to keep us up to date: stock-markets status, charted movies, etc. Human has been used to the idea that much can be told to a machine to do without the need to be actively involved in the control loop. As an example, studies on self-driving cars [1] have provided an invaluable amount of insight about the perceptive stimulae that drive such devices and whose principle can be extrapolated to other appliances [2].

Keeping in mind this transference of knowledge, the present project focuses in analyzing two separate classification networks: a first one trained over a set of supplied data (for which the requirements of inference time was less than 10ms and that of accuracy, higher or equal to 75%); and a second one being the proposal of an inference idea (i.e. clothing-recognition) based on the same framework while using externally acquired data.

The objective was to evaluate the results from choices made in the first part, and also check the viability of a self-picking clothing robot (using a set of acquired data) that would implement deep-learning network processes for the recognition of three different classes of clothing. Outcomes and their accuracy will provide an understanding of these approaches' advantages as well as downsides. Some researches already present in the literature which make use of other techniques, could be equally transferable to this case study [3]–[5].

2 Formulation

2.1 Data

The necessary data to carry out the project were of two different kinds, supplied and acquired. Formatting was the same for all images in terms of size and color, although some differences do exist:

1) **Supplied Data**: A set of images made up of candy boxes, bottles and those of an empty conveyor belt (see Fig. 1) was made available before training the model (see section 2.2). The photos' background was constant as per the application intended for the model; that is, no other scenario than goods transportation over the conveyor belt itself was planned.

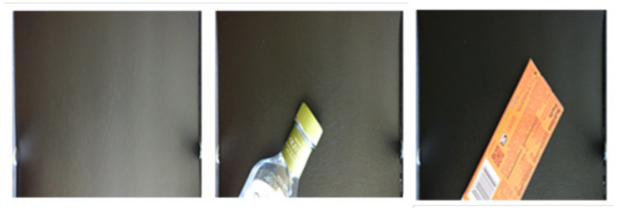


Fig. 1: Supplied data example (2018, Udacity)

2) **Acquired Data**: A set of images consisting on t-shirts, pants and socks. The photos' background was switched between bright and dark to allow the network more flexibility when detecting the objects. Section 3 provides more information in this regard.

2.2 Training Model

The architecture shown in Fig. 2 provides an representation of googLeNet's ® [6] model. It was named the winner of the ILSVRC 2014 competition as it achieved a top-5 error rate of 6.67% [7] over other competing structures, and although other algorithms as AlexNet® [8] could be used for results comparison, discussions in such regard are outside the scope of the current project.

On the other hand, the input's format for googLeNet® model is that of color images of size 256x256 (as so was the image format assumed for both the supplied and gathered data) thus making it the model to be trained in both scenarios. Moreover, due to the inherent complexity of the model, it was decided to keep it as it is for training. Generally, as stated by Le and Zoph in [9], "the process of manually

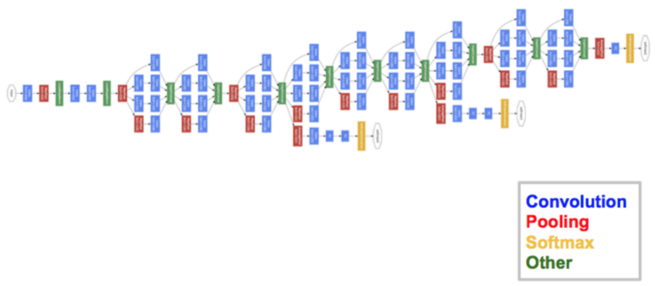


Fig. 2: googLeNet® Architecture [7]

designing machine learning models is difficult because the search space of all possible models can be combinatorially large a typical 10-layer network can have around 10^{10} candidate networks". Thus, the amount of time that could be required to experiment with different structures is not justifiable as it could impact the accuracy of the provided model.

3 Data Acquisition

As previously mentioned, two sets of data were used in this project: supplied and acquired. For the second one, the next considerations were taken:

1) The set of images were organized in three different classes (see section 2.1), as seen in Fig. 3.
2) All of the samples were collected using a web-cam executed via a Python data-capture script.
3) Over 400 samples were gathered and properly labeled for each class.
4) All of the samples were in color (i.e. no filter applied) and their size 640x480 pixels. This value was kept constant during the whole project.

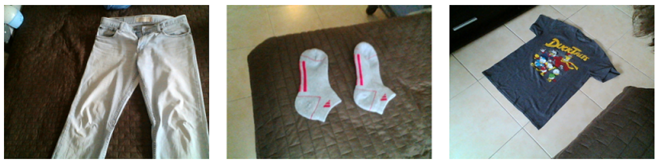


Fig. 3: Data independently collected

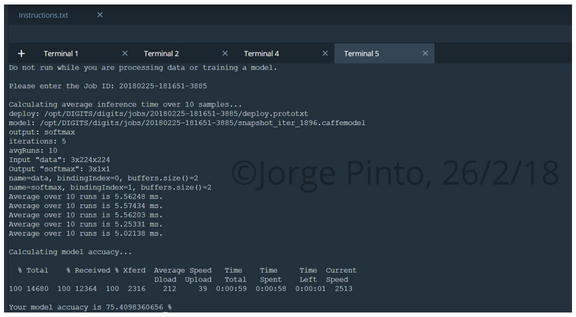
The method for taking the data was due to the uniformity of the capture in terms of pixels' size, color and required space in disk. Moreover, the reader would find practical to store the acquired images in the computer without the need to bridge them from a different device. Nevertheless, other methods for getting the data are also encouraged.

4 Results

For both sets of data, the provided GPU DIGITS workflow system (Udacity, 2018) was used. Inputs were loaded as classification type datasets; for the system to properly identify the number of contained classes, format name used was of the type WWW_XX.png, with WWW being the class and XX the associated number. From this section, information will be treated separately in order to differentiate one set from the other.

4.1 Supplied Data

Three runs were necessary to get the trained model passing the thresholds of inference time ($\leq$ 10 ms) and accuracy ($\geq$ 75%). Learning rate was progressively decreased to 0.007 and epochs, on the other hand, increased to 8. Both results and evolution of the model during training are shown in Figs 4 and 5 after



```
Do not run while you are processing data or training a model.

Please enter the Job ID: 20180225-181651-3885

Calculating average inference time over 10 samples...
deploy: /opt/DIGITS/digits/jobs/20180225-181651-3885/deploy.prototxt
model: /opt/DIGITS/digits/jobs/20180225-181651-3885/snapshot_iter_1896.caffemodel
output: softmax
iterations: 5
avgRuns: 10
Input "data": 3x224x224
Output "softmax": 3x1x1
name=data, bindingIndex=0, buffers.size()=2
name=softmax, bindingIndex=1, buffers.size()=2
Average over 10 runs is 5.56248 ms.
Average over 10 runs is 5.57434 ms.
Average over 10 runs is 5.56203 ms.
Average over 10 runs is 5.25331 ms.
Average over 10 runs is 5.02138 ms.

Calculating model accuacy...

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 14680  100 12364  100  2316    212     39  0:00:59  0:00:58  0:00:01  2513

Your model accuacy is 75.4098360656 %
```

Fig. 4: Inference time and accuracy results for the supplied dataset

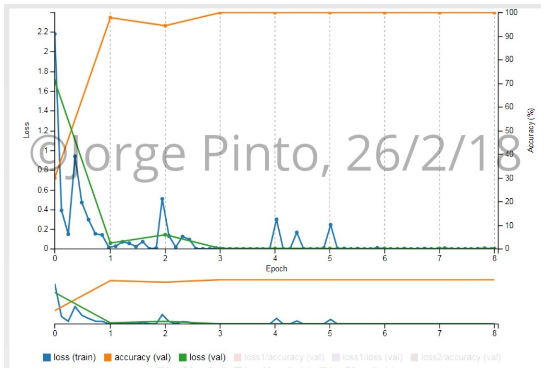


Fig. 5: Losses and Accuracy VS. Epochs over supplied dataset

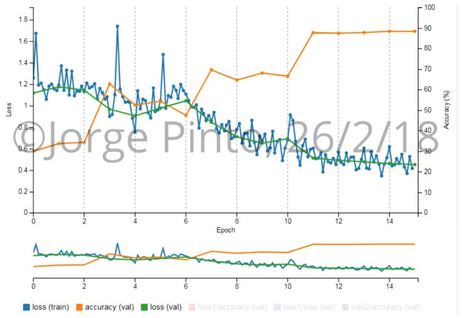
For this set, googLeNet's ® [6] model was trained straight-forward without the need of further pretrain. Although the tendency was oscillatory at the beginning, the reached accuracy provided enough insight to stop the process and move onto the other set of data.

4.2 Acquired Data

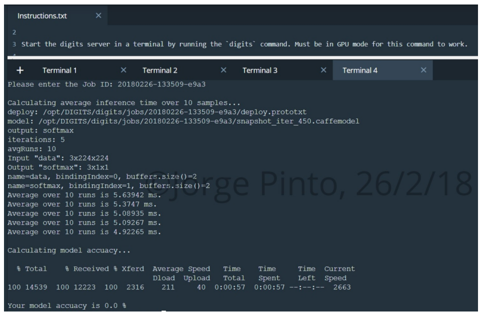
This set, on the other hand, required five different runs to get all classes identified. Inference time was always below the 10ms threshold in all cases. Evolution and results of the model at the third training attempt can be seen in Figs 6a and 6b.

Unlike the supplied data set, googLeNet's ® [6] model got pre-trained after the third run for this case scenario. At such point, prediction values as shown in Table 1a were gathered for a learning rate of 0.01 and 15 epochs. Measures were performed by means of a test data set (the same in both cases) passed to the DIGITS system (Udacity, 2018) in .list format.

Two new models were trained afterwards. The latest model accounted a learning rate of 0.009 and 15 epochs. Evolution and results of the model are shown in Figs 6d and 6c. The corresponding prediction values for this model are shown in Table 1b keeping the same test images.



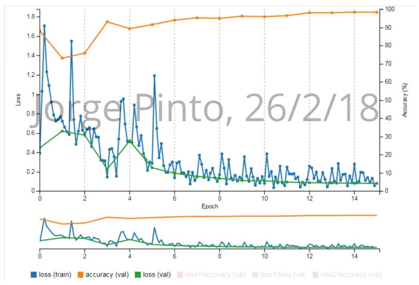
(a) Losses/Accuracy VS. Epochs before pretraining



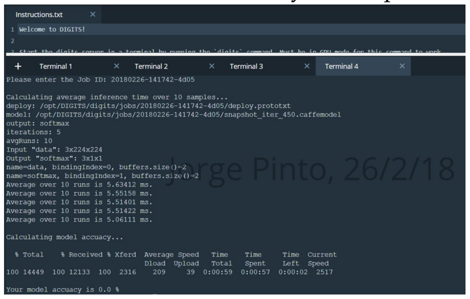
```
2
3 Start the digits server in a terminal by running the `digits` command. Must be in GPU mode for this command to work.
```

```
Please enter the Job ID: 20180226-133509-e9a3

Calculating average inference time over 10 samples...
deploy: /opt/DIGITS/digits/jobs/20180226-133509-e9a3/deploy.prototxt
model: /opt/DIGITS/digits/jobs/20180226-133509-e9a3/snapshot_iter_450.caffemodel
output: softmax
iterations: 5
avgRuns: 10
Input "data": 3x224x224
Output "softmax": 3x1x1
name=data, bindingIndex=0, buffers.size()=2
name=softmax, bindingIndex=1, buffers.size()=2
Average over 10 runs is 5.63942 ms.
Average over 10 runs is 5.3747 ms.
Average over 10 runs is 5.08935 ms.
Average over 10 runs is 5.09267 ms.
Average over 10 runs is 4.92265 ms.

Calculating model accuacy...

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 14539  100 12223  100  2316    211     40  0:00:57  0:00:57 --:--:--  2663

Your model accuacy is 0.0 %
```

(b) Inference time and accuracy before pretraining



(c) Losses/Accuracy VS. Epochs after pretraining



```
1 Welcome to DIGITS!
2
```

```
Please enter the Job ID: 20180226-141742-4d05

Calculating average inference time over 10 samples...
deploy: /opt/DIGITS/digits/jobs/20180226-141742-4d05/deploy.prototxt
model: /opt/DIGITS/digits/jobs/20180226-141742-4d05/snapshot_iter_450.caffemodel
output: softmax
iterations: 5
avgRuns: 10
Input "data": 3x224x224
Output "softmax": 3x1x1
name=data, bindingIndex=0, buffers.size()=2
name=softmax, bindingIndex=1, buffers.size()=2
Average over 10 runs is 5.63412 ms.
Average over 10 runs is 5.55158 ms.
Average over 10 runs is 5.51401 ms.
Average over 10 runs is 5.51422 ms.
Average over 10 runs is 5.06111 ms.

Calculating model accuacy...

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 14449  100 12133  100  2316    209     39  0:00:59  0:00:57  0:00:02  2517

Your model accuacy is 0.0 %
```

(d) Inference time and accuracy after pretraining

Fig. 6: Results gathered for the acquired data

TABLE 1: Predictions(%)

Label	T-shirts	Pants	Socks
Pants_10	64.05	31.05	4.9
Socks_207	0.74	21.69	77.58
T-shirts_127	81.57	17.1	1.33
Pants_174	10.74	55.75	33.51
Socks_58	3.24	34.87	61.89
T-shirts_154	63.63	28.87	4.49
Pants_321	9.85	55.3	34.85

(a) Before pretraining

Label	T-shirts	Pants	Socks
Pants_10	0.29	99.55	0.16
Socks_207	0.01	3.49	96.5
T-shirts_127	99.97	0.03	0.0
Pants_174	0.0	99.28	0.72
Socks_58	0.74	7.33	91.94
T-shirts_154	99.29	0.71	0.0
Pants_321	0.0	99.96	0.04

(b) After pretraining

5 Discussion

5.1 Supplied Data

For this set of data, googLeNet's ® [6] model served its purpose. It provided a good fit without the need of making a pretrained version of it during testing.

Three impressions can be pointed out from this experience:

1) Inference time was not impacted as learning rate or epochs were modified.
2) Accuracy reached an almost ”perfect” value (around 100%) as epochs were increased and learning rate decreased.
3) The effort to get the fit was much less than that for the acquired data.

The last can be drawn upon the uniformity of the images as all of them were taken on a conveyor belt (i.e. constant background) without important variations. Which means that, as no differences exists among the images besides the detected objects, there is little to no bias in the data provided to the model. In other words, the prediction percentages will reach values similar to those shown in Table 1b in a different input with the exact same characteristics are given to the model.

However, it should be kept in mind that the calculated fitting is expected to work for this sort of application. In case images from other backgrounds are placed for testing, it will not be possible to get the accuracy therein registered and a new complete model training would be required.

In such sense of ideas, this case provides insight regarding the type of environment it could be applied to: it will have to be a very controlled one, with high quality control standards (i.e., a food processing industry) were inference time rather than accuracy would mean the difference between improving the production level or loosing money.

5.2 Acquired Data

Unlike the earlier set, the acquired images set did have different background colors. A number of observations can be extracted from this experience:

1) Inference time was also not impacted as learning rate or epochs were modified, and the value ranges observed matches that of the supplied data.
2) Accuracy did reach an almost ”perfect” value (i.e. 99.7%) as epochs were increased and learning rate decreased, but the model had to be further pretrained to get a better fitting for the data

3) From the last affirmation, the effort to get such accuracy was more than that of the supplied data.

That been said, the lack of uniformity affected the model to the point that, in order to get a proper fitting, more than a simple training was required. Fig 6a and Table 1a provides a clear idea of the accuracy levels obtained in this stage of the analysis.

One could think though that the model is good enough at this point and that any set of images with different backgrounds can be given to it for classification. Which is why, for the sake of better understanding the reach of such affirmation, the already trained model got pretrained another time with hyperparameters of learning rate and epochs equal to 0.009 and 8. Losses VS. Accuracy behavior is shown in Fig. 7.

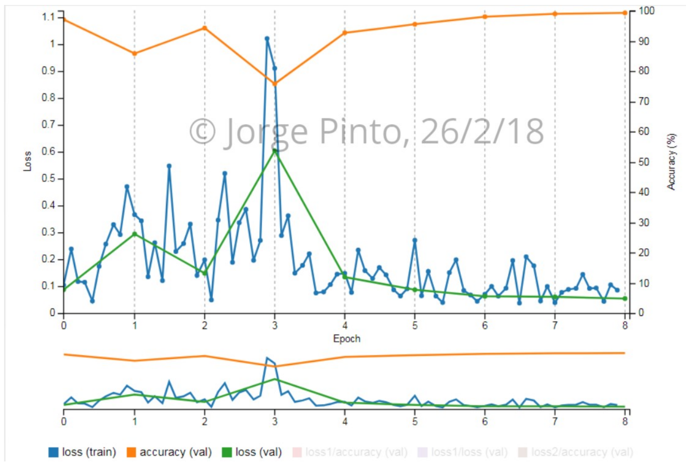


Fig. 7: Losses/Accuracy VS. Epochs extra training

Afterwards two sets of data were provided to the DIGITS workflow (Udacity, 2018) in .list format: one from the images collected and another one from a public source (i.e, Google search engine). Determined accuracies are shown in Table 2a and 2b for two sets of different images. For this, more than 400 images were randomly gathered and renamed according to the DIGITS workflow criteria (see section 4); examples of them are found in Fig 8.

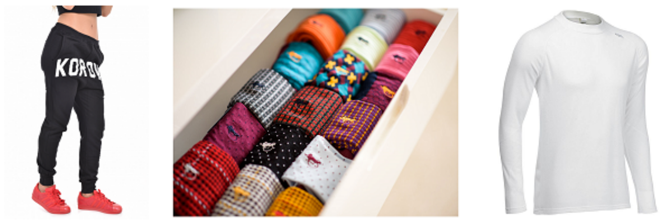


Fig. 8: Examples of public images from Google® search engine

As the reader can infer, it is very difficult to get appropriate accuracies for images whose format, background color and even content is different from what the model was trained with in the first place. In this case, inference failed providing only around a 14% accuracy for inputs that were not included in the training set.

Finally, it is important to point out that the model by itself is biased to the provided input in any scenario. That is, if the list passed to the workflow as a test comes from input used to train the same model, the likely predicted accuracy will be high if compared with data coming from a totally different source. In addition, if a self-picking clothing robot were to be constructed based on this model, it will encounter a lot of issues if any of the clothing is not properly placed (i.e, extended over a surface) as seen with public images. Therefore, increasing the size of the training set to over 1000 images placed in any possible way could help the robot to properly classify the piece, with an increased inference time due to the higher number of possibilities it has to check for.

TABLE 2: Predictions(%)

Label	T-shirts	Pants	Socks
Pants_10	3.5	96.46	0.04
Socks_207	0.01	4.15	95.85
T-shirts_127	100	0	0
Pants_174	0.01	99.59	0.4
Socks_58	2.3	20.55	77.15
T-shirts_154	99.08	0.16	0.76
Pants_321	0.01	99.98	0.0

(a) For the collected images set

Label	T-shirts	Pants	Socks
Pants_25	76.49	23.51	0.0
Socks_400	99.92	0.08	0.0
T-shirts_96	29.65	70.34	0.0
Pants_154	2.01	32.74	65.25
Socks_235	35.2	64.23	0.57
T-shirts_333	18.84	81.16	0.0
Pants_291	1.32	98.49	0.19

(b) For the public images set

6 Conclusion and Future Work

Two sets of inputs were used to train googLeNet's ® [6] model. Uniformity of the data in general proved to have a inverse effect on the computational effort required to train a network: to less uniformity, more effort is needed. Predictions were also calculated, being positive for both supplied and acquired data. However, to better understand the accuracy provided by the network, inputs different from those contained in the sets are preferred.

The acquired data set (the one to be used by the proposed robot) would work in an invariant system where the clothes are properly extended over a flat surface. The contrary however would create many false positives. For future work, a higher size of training data (including images with different features) could be acquired and tested to check the performance of the googLeNet's ® [6] model and verify the fact that better data does provide better predictions.

References

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[2] B. A. Wallach, H. A. Koselka, and D. L. Gollaher, "Autonomous vacuum cleaner," Aug. 9 2005. US Patent 6,925,679.

[3] I. Mariolis, G. Peleka, A. Kargakos, and S. Malassiotis, "Pose and category recognition of highly deformable objects using deep learning," in *Advanced Robotics (ICAR), 2015 International Conference on*, pp. 655–662, IEEE, 2015.

[4] P. Monsó, G. Alenyà, and C. Torras, "Pomdp approach to robotized clothes separation," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 1324–1329, IEEE, 2012.

[5] L. Sun, G. Aragon-Camarasa, S. Rogers, R. Stolkin, and J. P. Siebert, "Single-shot clothing category recognition in free-configurations with application to autonomous clothes sorting," *arXiv preprint arXiv:1707.07157*, 2017.

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, *et al.*, "Going deeper with convolutions," Cvpr, 2015.

[7] S. Das, "Cnns architectures: Lenet, alexnet, vgg, googlenet, resnet and more," 2017.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[9] Q. L. . B. Zoph, "Using machine learning to explore neural network architecture," 2017.