Data Management Systems Project 2-Data Science/Scalability Challenge

Group: João Batanete, João Rainho, Inês Petronilho

Abstract - In this paper, we performed Image Recognition tests on a Food-101 dataset, which contains a large amount of different images of different types of food, as well as scalability tests using Apache Spark, on the same set. The tests were performed using various different algorithms and approaches. The main goal was to compare three different Image CLassification algorithms (Convolutional Neural Network, K-Nearest Neighbor and Support Vector Machine) in order to determine which one that yielded the best results, in terms of both accuracy, and computational speed. The final results of this experiment showed that, for the given conditions and dataset, the CNN algorithm presented the best results , with an accuracy of 26% and SVM is the one that presents the worst results with a validation accuracy of 5.7%. KNN yielded better results than the other two in terms of speed performance, but still had much worse precision than CNN's. The second part of the experiment refers to scalability tests performed on the same dataset, using Apache Spark to check how using multiple cores and machines affects performance.

Keywords - Deep learning; Food recognition; Convolutional neural network; K-Nearest Neighbor, Support Vector Machine

1 Introduction

Every day we are confronted with the most diverse images of food and we quickly associate the dish and the ingredients that make it up. Our memory associates some patterns between what we observe at a given moment and the memories we have. It is a process that takes time and is never fully perfected. But how do we apply this ability we have in the same way to a computer? While the human brain is extremely good at processing images, it is not quite so simple for a computer.

Machine learning is a subfield of computer science that studies the ability of computers and machines to "learn" and adapt their courses of action according to the information gathered. The array of techniques used to implement it, and its applications, are countless at this point, but this project focuses on techniques and algorithms related to Image Classification. This consists of being able to label a given image with a list of classes, such as saying that an image either has sushi or a hot dog. Generally speaking, the more diverse the data is (as in, how many different classes we have, and how similar images of different classes can be), the more complex the study becomes, and it will be increasingly more difficult to obtain an acceptable error margin.

Several algorithms have already been tested as specified in [2], [3] and [4]. But what we will do is compare the results of different techniques and evaluate the one that best fits the dataset used, [5].

We will present several models and methods of Image Classification, like Convolutional Neural Networks, k-nearest neighbors algorithm and Support Vector Machines, using Machine Learning libraries and the Python language. We shall present the results on each algorithm on a sample dataset containing 101 different types of food and try to determine which one performs best in this particular scenario.

In the scalability study, we weren't able to parallelize the algorithms themselves, mainly due to their difficulty of being described as a map/reduce problem and the black box nature of their implementation in the chosen libraries. Therefore, the study performed relates to performance experiments using Spark with several different combinations of cores and machines and compare the results.

The following section presents related work followed by general overview of the experiment, algorithms and technologies chosen. Section 4 has the results of our execution, in Section 5 we discuss the experimental results and conclusions of the study and finally we suggest some future work in this area.

2 Related Work

Image classification has been a very well studied problem in the field of machine learning.

Convolutional Neural Network approach is one of the most used techniques in this field. Kagaya et al [3] applied a convolutional neural network (CNN) to the tasks of detecting and recognizing food images. This algorithm showed significantly higher accuracy than traditional support vector machine-based methods.

D. Pishva et al. [6] created a bread detection system that distinguished between 73 types of breads, with a 95% hit rate. However, the photos used were taken so that the center of the bread coincided with the center of the photo and had a uniform background to easily distinguish the outlines of the bread.

S. Yang et al. [7] developed a recognition system capable of distinguishing various types of fast food. They defined 8 basic ingredients, such as meat, bread or cheese, and through their recognition and location, ranked the images in one of 61 categories, achieving a 28 % hit rate with Pittsburgh fast-food dataset [8].

H. Zhang et al. [9] propose a hybrid solution of two models: K-Nearest Neighbor and Support Vector Machine. They worked with a dataset with 102 different classes of objects and they achieved a correct classification rate of 59.05% at 15 training images per class, and 66.23% at 30 training images.

N.Kamarudin et al. [10] performed a study similar to ours, using the Caltech 101, which contains single object images of several categories. However, they did not include Convolution Neural Networks in their study.

L. Thai et al. [12] attempted to combine two other algorithms(Artificial Neural Networks and Support Vector Machines). They aimed to classify roman numeral symbols and achieved an 86% precision rate.

3 Preparation of the experiment

3.1 Machine specifications

The machine used for the experiment contains the following specifications:

Processor: 4x Intel i7-5500U CPU(2.4 GHz, 4 Cores)

Memory: 8GB

Operating System: Ubuntu 16.04.2 LTS

3.2 Algorithms chosen in the experiment

3.2.1 Convolutional Neural Network

A Convolutional Neural Network is a popular deep learning algorithm for current visual recognition tasks, working like learnable local filters.

The first step is to perform some local filtering of the image, enhancing his edges. CNN do this by taking the neighborhood of each pixel and convolve it with a certain mask (set of weights), computing a linear combination of those pixels. For example, with a positive weight on the center pixel and negative weights on the surrounding pixels, we need to compute the difference between the center pixel and the surrounding.

What makes convolutional neural networks special is that the weights are shared, that is, they are the same for different pixels in the image (but different with respect to the position relative to the center pixel). This makes them a lot more suitable for Image processing specifically, as they are suitable for modeling animal perceptions(including vision).

CNN's are one of the most popular algorithms for Image Classification, and generally yield better results than the alternatives. They are however, relatively difficult to implement, as they require us to define the various layers of the network, and optimizing factors such as the learning rate, batch size and the number of passes can be time consuming.

3.2.2 K-Nearest Neighbor

This algorithm can be used for both classification and regression. It works by listing every neighbor to the current individual(in this case image) and using the class of every K-closest neighbor to classify it. In its simplest form, it uses a value of k=1, in which case we simply assume the class of the exact closest neighbor.

Increasing k values alone does not necessarily improve the accuracy of the algorithm. If we simply use a decision-voting system (that is, using whichever class is more prevalent in the k-neighbors) we may end up giving too much priority to less relevant neighbors(for example, the second closest neighbor should have more relevance to the classification than the fifth one). A common method for improving accuracy with k-neighbors is to give different weights to each of the neighbors in order to adjust their contribution to classifying the individual. In this project, we analyzed results both with uniform weights, and ranked weights, using 1/d to compute the weight of each neighbor, where d is the "distance" to the individual.

This algorithm is generally used when data is harder to model, and when it is easy to establish a distance between individuals. It is also relatively simple to implement.

3.2.3 Support Vector Machine

This algorithm consists in the division of the data in two subsets, and trying to categorize each individual into one. When we are unable to draw a hyperplane between the subsets due to overlapping, we perform a "jumbling" operation, in which we extend the space with an extra dimension, and do so recursively until we are able to.

In the case of studies involving more than two classes of individuals, this process is repeated recursively.

SVM's are widely used in a variety of use cases, including text processing, image classification and biology related fields. They are, however, relatively poor in terms of speed performance.

3.3 Technologies picked for the implementation of the experiment

3.3.1 Programming language

For this project, we decided to use the Python programming language. We made this decision based on our experience with the language, and the large amount of frameworks and documentation for Python in the field of Machine Learning, as well as previous work done in it.

3.3.2 Hadoop and Map Reduce

Hadoop has been around for more than 10 years and has proven to be one of the best solutions for processing large datasets. Hadoop uses MapReduce, which is a good solution for single-process calculations but not very efficient for use cases that require calculations and algorithms with multiple runs. MapReduce is a programming model which relies, at its core, on two functions(map and reduce). The first function, from a very general perspective, sorts data into queues(for example, sorting football players by their position in the field, one queue for each position) and the second one does a “summary operation” on each queue(such as determining the average number of goals for each position). This programming model is very common in functional programming languages(and also exists in more general ones, such as Python), however their purpose in the MapReduce framework is slightly different. By itself, the model does nothing to increase performance. It does however, make the overall operation much more easy to parallelize, which can, in many cases, mean a very large performance gain. This paradigm does have a few requirements to work properly however, and is not correct for every use case. If a given operation takes most of its time on a very precise bottleneck which isn’t trivial to compute in parallel, the performance gains might be marginal, and not worth the investment in more machines(in this case, it would be best to use vertical scaling). There have been many implementations of the model for several programming languages,Apache Hadoop being one of the most popular. Spark and Hadoop are often run together, even though Spark also allows the use of other file management systems.

3.3.3 Apache-Spark

Spark uses Cluster-computing to process information in parallel and exchange information as needed via messages, in order to perform tasks in a more efficient way than using a single machine. It has become increasingly popular in recent times due to the increasingly large amount of data that enterprise applications need to store and process. It also removes the problem of a single point of failure, as even if one cluster malfunctions, the remainder of the network will keep on working.

It can be used for an array of computer science and engineering areas, whenever performance or parallelism is a priority. Spark in particular relies on the RDD(resilient distributed dataset) data structure, which gives it great fault-tolerance compared to similar software products. It was also created to accommodate the shortcomings of the MapReduce paradigm.

3.3.4 Scikit-learn(Sklearn)

This library was used to implement all algorithms, with the exception of the CNN’s. It was chosen due to its simplicity of use, and our previous experience with it in the course. The developers of CNN mentioned on their Github repository that it doesn’t support algorithms such as CNN due to its reliance on the machine’s GPU in the link mentioned in [1].

3.3.5 Tensorflow/tflearn

This library was used to implement the convolution neural network(CNN) classifier. Tensorflow is a widely used open-source Machine Learning library available for Python, Java, Go and C++. Tflearn is a higher level Python API built on top of Tensorflow which makes it easier to interact with Tensorflow for entry-level users and programmers.

This library was chosen over regular Tensorflow due to the group’s relative lack of experience with the API and machine learning in general.

3.4 Dataset: Food-101

Our dataset is Food-101 (available for download at http://www.vision.ee.ethz.ch/datasets/food-101/). It contains 101 000 real world images in total, divided by 101 classes of food, each one with 1000 images, including very diverse but also visually and semantically similar food classes such as Apple pie, Waffles, Sushi, Paella, Risotto, Omelette, Macarons to name a few.

The dataset contains 1000 images of each food type(101000 in total), taking 5GB of memory space.

For each class, 250 manually reviewed test images are provided as well as 750 training images for SVM and KNN algorithms. For the CNN algorithm, we use 100 test images and 900 training images.

On purpose, the training images were not cleaned, and thus still contain some amount of noise. This comes mostly in the form of intense colors and sometimes wrong labels. All images were rescaled to have a maximum side length of 512 pixels.

3.5 Scalability Challenge

3.5.1 Spark configuration

Spark was configured to run locally, via the pyspark shell. The group is still attemptings to configure it on multiple machines in order to perform more tests.

3.5.2 Changes to the source code

The source code was modified to process the conversion of the images into feature vectors and color histograms in parallel. The algorithms part was omitted, as the group was unable to set them up to run in parallel. Only the process of image conversion was tracked and run using Spark.

The program begins by reading the image files from disk and, whenever it reads 100 images, it applies a map-reduce operation on the list of images which converts the images (map) and joins them together in a new list (reduce). This process was done in batches of 100 images, as JVM simply ran out of memory without the split.

4 Results

4.1 K Nearest Neighbors

4.1.1 Uniform weights

K	Accuracy (%)	Time (s)
1	4.54	603
2	4.31	636
3	4.23	632
4	4.18	640
5	4.12	634

Figure 1: Results of the KNN Algorithm with uniform weights

We started our tests with the K Nearest Neighbors algorithm, with uniform weights. As can be seen in Figure 1, the highest value achieved is k = 1, with an accuracy of 4.54% and a runtime of 603 seconds. The following k-values show that there has been a worsening in accuracy.

Since the results only become worse from K=5 onwards, we decided to omit them.

4.1.2 Distance-based weights

To distance-based weights, the results were slightly different. Up to k = 200, with a higher value of k, we have better accuracy, but from this value, the accuracy begins to decrease slowly, As can be confirmed in Figure 2.

K	Accuracy (%)	Time (s)
1	4.54	556
2	4.54	535
3	4.69	551
4	4.93	571
5	5.15	574
10	5.78	705
15	6.19	648
20	6.27	669
25	6.3	686
30	6.67	711
100	7.02	718
200	7.12	714
300	6.97	713
400	6.92	718
500	6.86	722

Figure 2: Results of the KNN Algorithm with distance-based weights

In addition, it is possible to observe the increase of the execution time with the increase of the value of k.

4.2 Support Vector Machine

The precision of the results for the SVM algorithm is 0.0570 and the training and testing time was 3 hours and 23 minutes.

4.3 Convolutional Neural Network

Epoch	Loss	Acc	Val_loss	Val_Acc	Time
1	4.23	0.063	4.05	0.094	404
5	3.71	0.1396	3.41	0.210	410
10	3.55	0.183	3.23	0.239	486
15	3.46	0.201	3.16	0.250	422
20	3.48	0.205	3.14	0.256	534
25	3.26	0.228	3.09	0.268	472
30	3.35	0.215	3.13	0.261	503
35	3.24	0.232	3.10	0.269	409
40	3.22	0.229	3.13	0.265	413
45	3.20	0.254	3.15	0.270	426
50	3.30	0.229	3.14	0.270	380
55	3.24	0.250	3.11	0.281	505
60	3.37	0.228	3.16	0.279	495
65	3.19	0.246	3.11	0.280	422
70	3.52	0.226	3.15	0.274	423
75	3.24	0.236	3.15	0.277	410
80	3.20	0.243	3.17	0.275	429
85	3.24	0.228	3.15	0.272	362
90	3.17	0.253	3.18	0.266	361
95	3.18	0.248	3.16	0.272	361
100	3.13	0.246	3.20	0.267	360

Figure 3: Results of the CNN Algorithm

Finally, the cnn algorithm showed encouraging results, with accuracy reaching almost 27%, which is much higher than the previous ones. The runtime with Epoch = 35 was 409 seconds, also the best of the three.

4.4 Scalability challenge

For the second part of the project were carried out scalability tests to load the food images, to distribute the work by Spark. Tests were run for one, two, and four cores, and the results showed what the group expected: a decrease in runtime with increasing number of cores.

Machines	Cores	Time(seconds)
1	1	3539
1	2	2832
1	4	2119

Figure 4: Results of scalability challenge

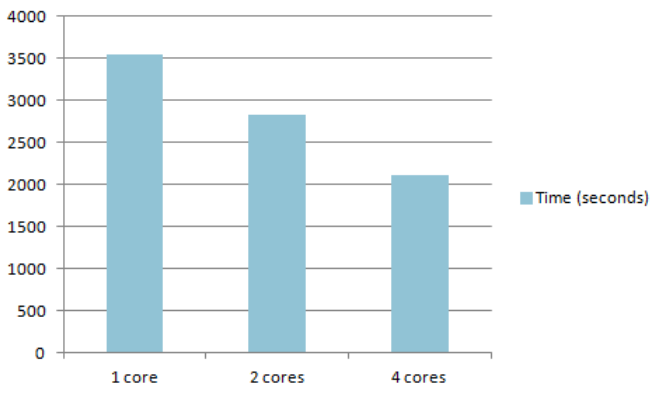

Figure 5: Comparison of results of scalability challenge

Tests for multiple machines were not performed due to some problems running the program across multiple clusters.

5 Discussion and conclusions

5.1 Data science challenge

While the data science results might seem underwhelming at first sight(approximately 26% validation accuracy on the best case), one needs to remember that the dataset contains 101 different classes. This would mean that a simple random based algorithm would, on average, only have 1/101 accuracy. Therefore, for a dataset of this size and variety, this is actually a relatively good result, since it means the algorithm is actually learning from the dataset.

As expected, CNN clearly performs better than either SVM and KNN. The results also seem to indicate that 55 epoch was the proper value, since validation accuracy peaked by that step, leading us to assume

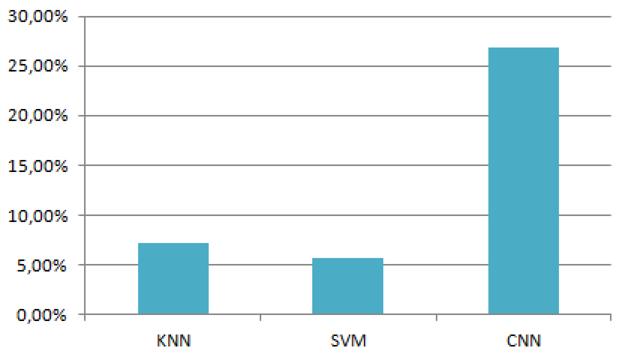

Figure 6: Comparison of accuracy between the 3 algorithms

that approximately 28.1% is the most that could easily be obtained without changing the model, or with more training data. However, accuracy and validation accuracy both remained relatively stable, which indicates that the model didn't enter overfitting stage, it simply couldn't adjust itself anymore with the given dataset.

While the results for the other two algorithms were poor compared to CNN's, KNN surprisingly performed better than SVM's as long as uniform weights weren't used, which had a big impact on the algorithm's accuracy. Increasing K on KNN with uniform weights is actually detrimental to the algorithms precision.

In terms of speed performance, SVM performed extremely poorly as well, as expected, since the algorithm is generally very slow for big datasets, and is rarely used in scenarios that value speed over accuracy. KNN is faster than CNN in a sense, as one only needs to perform one training epoch, with a proper value for K(around 200 seems to be the algorithm's "peek"). CNN takes a very long time when one considers the total time taken for the 55 training epochs. Therefore, we conclude that KNN could be a consideration when learning speed performance is a more important factor than raw accuracy.

It is interesting to note that, even with distance-based weights, KNN starts becoming worse close to the K=200 mark. This can be explained by the fact that using individuals which are different enough from the individual in the voting process(albeit with a very small weight) can actually be detrimental.

5.2 Scalability Challenge

The times seem to scale in a non linear way(in terms of the number of cores used). This could be attributed to the extra overhead in the scheduling and communication between processes involved when there are more cores.

The group will soon attempt to configure Spark to work on more than one machine, and analyse results accordingly.

6 Future Work

In future work, one could perform scalability tests using a tool similar to Spark and Hadoop, and try and measure the performance difference. However, since the best performing algorithm tested is clearly CNN, which relies heavily on the machine's GPU, the performance difference may not be as significant as one would expect. We could also repeat the experience while altering different parameters of CNN, namely by experimenting the algorithm with a different number of layers, neurons, a different pooling strategy or other parameters, in order to study how the performance is affected and perhaps come up with a better solution.

In terms of improvements to be done to the models tested and further studies related to them, one could try different neural networks, with different layers, as well as a different input/output model, such as based on the image features. While KNN and SVM both performed very poorly compared to CNN's, it would also be possible to test them under different circumstances, such as by performing simple image transformations beforehand.

KNN specifically could be experimented with using negative weights after a certain distance to the individual, which could wield marginally better results.

Scalability tests can and will be eventually performed using more than one machine.

References

[1] https://github.com/scikit-learn/scikit-learn/issues/3166, accessed on 21/05/2017

[2] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101–mining discriminative components with random forests," in Computer Vision– ECCV 2014. Springer, 2014, pp. 446–461.

[3] H. Kagaya, K. Aizawa, M. Ogawa, "Food Detection and Recognition Using Convolutional Neural Network", ACM Multimedia, 2014, Florida

[4] G. Ciocca, P. Napoletano, R. Schettini, "Food recognition for dietary monitoring: a new dataset, experiments and results", IEEE Journal of Biomedica and Health Informatics, 2017, pp. 588-598

[5] D.Lima, "Uso de Bibliotecas SciPy para Classificação de Imagens (UECFOOD100)"

[6] D. Pishva, A. Kawai, K. Hirakawa, K. Yamamori e T. Shiino, "Bread Recognition Using Color Distribution Analysis," IEICE Trans. on Information and Systems, vol. 84, n^o 12, p. 1651–1659, 2001

[7] S. Yang, M. Chen, D. Pomerleau e R. Sukthankar, "Food recognition using statistics of pairwise local features," Proc. of IEEE Computer Vision and Pattern Recognition, 2010

[8] M. Chen, K. Dhingra, W. Wu, L. Yang e R. Sukthankar, "PFID: Pittsburgh fast-food image dataset," Proc. of IEEE International Conference on Image Processing, p. 289–292, 2009

[9] H. Zhang, A. C. Berg, M. Maire, J. Malik, "SVM-KNN: Discriminative nearest neighbor classification for visual category recognition", "Computer Vision and Pattern Recognition", IEEE, p. 2126-2136, 2006

[10] N. S. Kamarudin, M. Makhtar, S.A. Fadzli, M. Mohamad, F. S. Mohamad, M. F. A. Kadir, "COMPARISON OF IMAGE CLASSIFICATION TECHNIQUES USING CALTECH 101 DATASET", "Journal of Theoretical and Applied Information Technology", 10th January 2015

[11] http://cs231n.github.io/convolutional-networks/, "Convolutional Neural Networks for Visual Recognition", accessed on 01/06/2017

[12] L. H. Thai, T.S. Hai, N.T. Thuy, "Image Classification using Support Vector Machine and Artificial Neural Network", I.J. Information Technology and Computer Science, 2012