Static and Dynamic Autopsy of Deep Networks

Titouan Lorieul, Antoine Ghorra, Bernard Merialdo
Data science department
Eurecom
Biot, France
Firstname.Lastname@eurecom.fr

***Abstract*—Although deep learning has been a major breakthrough in the recent years, Deep Neural Networks (DNNs) are still the subject of intense research, and many issues remain on how to use them efficiently. In particular, training a Deep Network remains a difficult process, which requires extensive computation, and for which very precise care has to be taken to avoid overfitting, a high risk because of the extremely large number of parameters. The purpose of our work is to perform an autopsy of pre-trained Deep Networks, with the objective of collecting information about the values of the various parameters, and their possible relations and correlations. The motivation is that some of these observations could be later used as a priori knowledge to facilitate the training of new networks, by guiding the exploration of the parameter space into more probable areas. In this paper, we first present a static analysis of the AlexNet Deep Network by computing various statistics on the existing parameter values. Then, we perform a dynamic analysis by measuring the effect of certain modifications of those values on the performance of the network. For example, we show that quantizing the values of the parameters to a small adequate set of values leads to similar performance as the original network. These results suggest that pursuing such studies could lead to the design of improved training procedures for Deep Networks.**

I. Introduction

Deep learning is a branch of machine learning based on models that use multiple layers of processing to represent complex phenomena. They have recently achieved impressive performance, in particular in pattern recognition tasks like image and speech recognition [1], [2]. These models involve a large quantity of parameters, and one key factor of their success is the ability of training these models from large quantities of data. Still, the training phase remains a challenge, as the risk of overfitting the values of the parameters is very high. DCNN (Deep Convolutional Neural Nets) are specially well suited for image recognition tasks [3], as they efficiently use the spatial correlation of pixels, and progressively build more abstract representations of the visual content. It has also been shown that these networks are able to extract feature vectors that are more efficient than regular hand-crafted features [4].

A lot of research works have tried to understand what kind of calculation is performed by the Deep Networks and makes them so efficient. For image analysis networks, this has often been done through the visualization of the weights, or the feature maps at different layers [5]. Such analysis can sometimes provide hints on useful modifications of the network [6]. It can also help to understand what part of the images provoke a specific interpretation, and how the network can be fooled by ad-hoc examples [7].

In our work, we want to investigate the properties of the weights in trained networks, with the intention to derive information that can be useful for training new networks, or to optimize their storage. We focus on analyzing an existing Deep Network in both a static and a dynamic approach. In the static approach, we compute several statistics on the actual values of parameters, possibly separated by layer. In the dynamic approach, we are interested in the behaviour of the network while analyzing images. We study the values of the output of neurons per layer, and evaluate what is the sensitivity of the performance of the network when the values of some parameters are modified.

For our experiments, we chose to use AlexNet [1]. There are several reasons for that. First, it has now become one of the classical DNN. Second, an already trained, ready-to-use model of this network is provided with the Caffe [9] framework, which makes experimentations easy to conduct.

II. Static Analysis

In the first part of our work, we aim at getting some insights about the final values of the parameters in the trained network. Such statistics can be evaluated at the global scale, over the entire network, but also be focused on each layer, in particular taking into account the difference of structure between convolutional and fully-connected layers.

A. Global statistics

AlexNet consists of 5 convolutional layers followed by 3 fully connected layers. Table 1 shows the distribution of the number of parameters on the different layers. We can see that distribution of the number of parameters is greatly unbalanced between layers. The convolutional layers only account for a negligible amount in the total number of parameters of the network. Indeed, less than 4% of the parameters are contained in the convolutional layers and, thus, more than 96% of them are used for the fully connected layers. However those first layers are considered very important as they extract local discriminative visual features from the images while the last layers serve as a classifier.

Fig.1 shows the distributions of the parameters values for the convolutional layers and for the fully connected layers. As

978-1-4673-8695-1/16/$31.00

TABLE I
ALEXNET'S PARAMETER DISTRIBUTION THROUGH THE DIFFERENT LAYERS

Layer	Number of parameters	Ratio of parameters
conv1	34,944	0.06 %
conv2	307,456	0.50 %
conv3	885,120	1.45 %
conv4	663,936	1.09 %
conv5	442,624	0.73 %
fc6	37,752,832	61.93 %
fc7	16,777,216	27.52 %
fc8	4,097,000	6.72 %
Total	60,965,224	100%

TABLE II
SUMMARY OF THE CHARACTERISTICS OF THE FILTERS IN THE CONVOLUTIONAL LAYERS.

Layer	Number of filters	Spatial extent	Depth
conv1	96	11×11	3
conv2	256	5×5	96
conv3	384	3×3	256
conv4	384	3×3	192
conv5	256	3×3	192

the figures illustrate, most of the values of the parameters are very small, these distributions both resemble to a Gaussian distribution centered in zero, with a smaller deviation for the fully connected layers. The standard deviation of the parameters of the convolutional layers equals to $1.69\ 10^{-2}$ while for the parameters of the fully-connected layers it equals to $5.68\ 10^{-3}$.

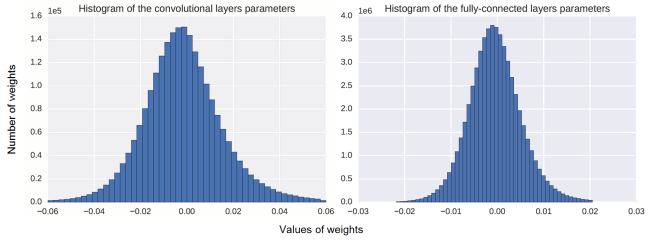

Fig. 1. Respective distribution of all the parameters of the convolutional layers and of the fully-connected layers.

B. Convolutional layer analysis

Convolutional layers are used to extract local discriminative visual features while keeping a certain amount of spatial information. Each of the convolutional layer is characterized by the number of filters it contains and by the size of the input patch of those filters. All these characteristics are summarized in Table 2. Except for the last layer, the number of filters is increasing with the depth of the layer in the network while the spatial extent decreases. As for the depth of the filters, it is equal to the number of filters of the preceding layer, except for the last two layers which are operating only on half of the filters of the preceding layers due to memory constraints.

In order to study the variability of these filters, we performed the following experiment on each convolutional layer. We consider each filter as a vector, and we perform a PCA analysis on the set of vectors for each layer. Then we look at the cumulated variance represented by the first k components of this PCA analysis as shown in equation 1, where the λ_i are the eigenvalues of the covariance matrix.

$$\alpha_k = \frac{\sum_{i<k} \lambda_i}{\sum_i \lambda_i} \quad (1)$$

Fig.2 shows the evolution of this ratio for the different layers. It can be noted that for the first layer, about half of the components are enough to represent most of the total variance. As the layers go deeper, the curves increase more slowly indicating a smaller correlation and a wider variety between the filters. This conforts the idea that, as we get deeper in the network, it detects higher-lever and more complex features. It also suggests that smaller number of filters is needed for the first layers and that this number should be increasing while we get deeper and deeper in the network.

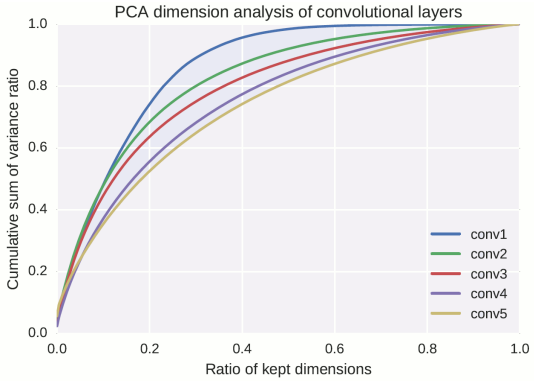

Fig. 2. Cumulative ratio of the variance ratio of the PCAs dimension kept

C. Fully-connected layers analysis

In AlexNet, the last three layers are fully connected layers, and are used to perform the final classification from the extracted features. The first two use the Rectified LinearUnit (ReLU) activation function, while the last one uses a softmax function. The ReLU, i.e. f (x) = max(0; x), is shown to allow a quicker learning of the network while enforcing sparsity. However, since the output of the network should be a probability distribution over the possible outcomes, the softmax is used for the last layer. As highlighted by Fig.3, the parameters of the fully-connected layers follow a kind of skew normal distribution, with a skewness increasing from the lowest fc6 layer to the last fc8 layer.

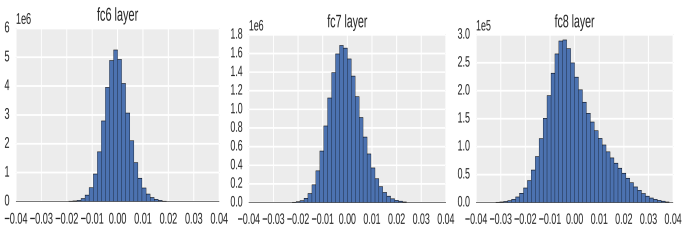

Fig. 3. Distribution of the parameters of the fully-connected layers

This raises the question of whether the difference between fc8 and the previous levels is due to a closer proximity to the output, or to the use of a different activation function. Further investigations will be needed to solve this issue.

III. Dynamic analysis

In a second step, we perform a dynamic analysis of the network. By dynamic, we mean observations that are performed while the network is processing actual data. This of course involves mostly the feature maps, that is the output values of neurons in intermediate layers. Another objective is to evaluate the importance of specific parameter values for the global performance of the network. Intuitively, a minimal modification of a few parameters should not have a great impact on the global performance. Our goal in the dynamic analysis is to have some quantitative evaluation of this relationship.

A. DataSet

In the following experiments, we use a subset of the ILSVRC 2012 (ImageNet Large Scale Visual Recognition Competition) dataset. ILSVRC 2012 [8] is a reference dataset that contains 1.2 million images labeled with 1,000 classes and it is the dataset on which Caffe's AlexNet model has been trained. The subset we use consists of 20,580 images and 120 classes, and was chosen arbitrarily.

B. Output Sparsity

The main feature of the output values is that they are positive valued vectors due to the activation functions which are either ReLU or softmax. The next important aspect is that they are sparse (cf. Fig.4). Sparsity is defined as the ratio of zero values over the total number of values. All the layers which use ReLU as activation function are sparse as they should be. fc8 is the only layer that has a sparsity of zero which is normal as it uses softmax instead of ReLU to output probabilities and, thus, it cannot return a zero value. The fully-connected layers before fc8 achieve more than 80% sparsity which is interesting as these layers are sometimes used as feature vectors for the description of the visual content of an image.

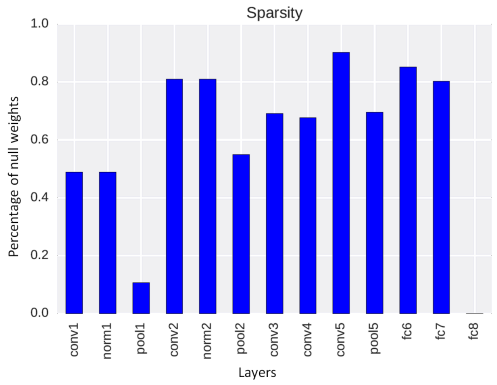

Fig. 4. Sparsity of weights per layer

C. Effect of parameter noise

To understand how sensitive to noise the parameters are, we artificially inject an additive normal noise to the values of the parameters. The standard deviation of that noise depends on the layer, as we feel that if a layer has a large standard deviation, a larger noise should be added than to a layer with small standard deviation. The value of the standard deviation of the noise is taken as a percentage of the standard deviation of the parameters of that layer. Fig.5 shows the effect of that noise on the accuracy of the network depending on the strength of that noise. Up to a 10% strength, the noise does not affect much the performance of the network but from 20% to 60% this error rate rapidly increases in a nearly linear relation and a high-valued slope.

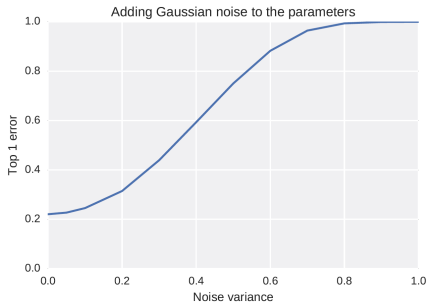

Fig. 5. Adding Gaussian noise to the parameters

In future experiments, we plan to refine these studies, for example to evaluate the effect of adding noise to a single layer, in order to see if some layers are more important than others.

D. Quantization

Quantization is another mechanism to modify the values of the parameters, while keeping the relation with the initial configuration. It may allow to compress the network by reducing its memory footprint and also to limit the parameters search space during training, thus resulting in a more efficient training. We studied two types of quantization: rounding quantization and K-means quantization.

Rounding quantization is simply done by reducing the number of digits used for the parameters by rounding their values up to a certain precision. As shown in Table 3, using only 4 to 5 decimals is sufficient to preserve the initial performance of the network. This suggests that it should be easy to reduce the storage used by the model by one half, depending on the encoding of real values. Note that this is true for testing purposes not training purposes [10].

With K-means scalar quantization, we learn a codebook that minimizes the L2 distances between the real values and the codewords. In our experiments, we learn one specific codebook for each layer. So, for each layer, we accumulate all the parameters values which appear in this layer, we perform a K-means clustering over this set, and we replace each parameter value by the value of the closest centroid in the final clustering. This amounts to using only K different values for the parameters, while these values are chosen adequately by

TABLE III
EFFECT OF ROUNDING ON THE ACCURACY OF THE NETWORK. (THE NUMBER OF DECIMALS KEPT IS THE NUMBER OF DIGITS AFTER THE DECIMAL POINT, NOT THE NUMBER OF SIGNIFICANT OR NON-ZERO DIGITS)

Number of decimals kept	Top 1-error
all	22.06 %
5	22.06 %
4	22.07 %
3	22.25 %
2	27.45%
1	100%

the clustering algorithm. Fig.6 shows the evolution of the error rate for various sizes of the K-means. It is noticeable that for K=32, the error rate is as low as the non-quantized network. This suggests that a precise value of the parameters is not so important, but probably that what is more important is their relationship. This information could also be used to reduce the size of the model drastically, as floating point values would be replaced by small range integer values.

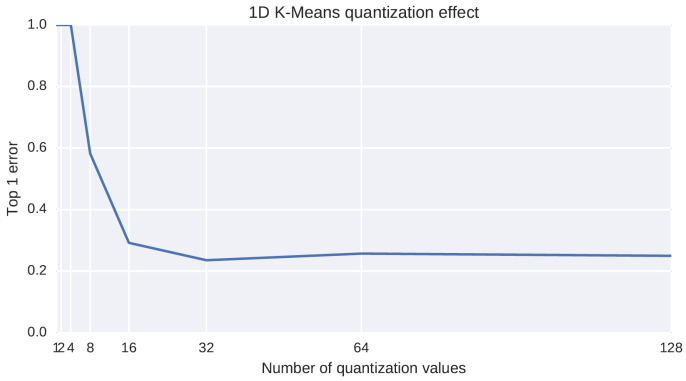

Fig. 6. Effects of 1D K-Means quantization on the accuracy of the network.

We also studied the impact of vector quantization of the filters, by performing a clustering of the filters for each layer, and replacing each filter by the centroid of the cluster it belongs to. Here we focus on the convolutional layers. Fig.7 shows the evolution of the error when the number of clusters increase.

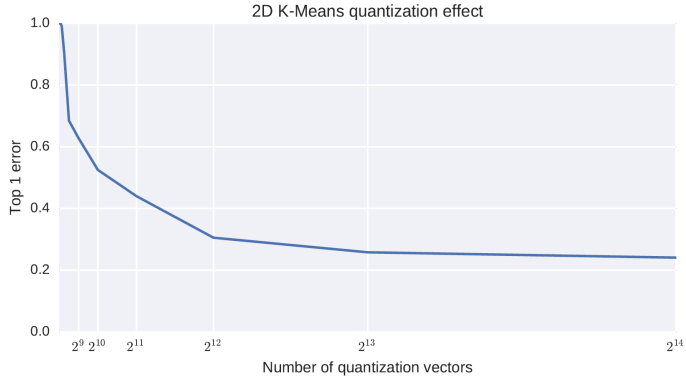

Fig. 7. Effects of 2D K-Means quantization on the accuracy of the network.

Not surprisingly, the 2-D K-means clustering method as shown in the Figure above requires a very large number of clusters to get an error rate comparable to the full network.

IV. CONCLUSION

In this paper, we presented a first attempt at the autopsy of Deep Networks by analyzing AlexNet. We performed a static analysis showing the parameter values distributions and a dynamic analysis evaluating the sensitivity of the final performance to these values. For example, we found that a quantization of the parameter values into a small adequate set does not degrade the performance of the network. We believe that extending this type of studies to other networks, and having a deeper analysis of the distributions, can lead to useful indications on how to improve the training phase of new networks, by taking these indications are heuristics to guide the training. This would allow to speed-up the training phase while reducing the risks of overfitting.

REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[3] Yann LeCun, Koray Kavukcuoglu, Clément Farabet, et al., "Convolutional networks and applications in vision.," in *ISCAS*, 2010, pp. 253–256.

[4] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson, "Cnn features off-the-shelf: An astounding baseline for recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2014.

[5] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson, "How transferable are features in deep neural networks?," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., pp. 3320–3328. Curran Associates, Inc., 2014.

[6] Matthew D. Zeiler and Rob Fergus, "Visualizing and understanding convolutional networks.," *Computer visionECCV 2014*, 2014.

[7] Anh Nguyen, Jason Yosinski, and Jeff Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," 2015.

[8] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al., "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[9] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.

[10] Vincent Vanhoucke, Andrew Senior, and Mark Z Mao, "Improving the speed of neural networks on cpus," in *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*, 2011, vol. 1.