Predicting Under and Overfitting in Deep Neural Networks using Graph Smoothness

Carlos Eduardo Rosar Kos Lassance[1], Vincent Gripon[1] & Antonio Ortega[2]
[1]IMT Atlantique, Brest, France: firstname.lastname@imt-atlantique.fr
[2]University of Southern California, Los Angeles, CA: firstname.lastname@usc.edu

***Abstract*—Deep Neural Networks (DNNs) have become the state-of-the-art in many machine learning benchmarks. Important questions remain about understanding the reason for their performance and how to choose the right architecture for a given problem. In this paper we study the use of Graph Signal Processing (GSP) tools to predict under and overfitting without the need to rely on crossvalidation. Namely, we generate graphs using intermediate representations of training examples at different layers across the architecture, and we analyse the smoothness of label signals on these graphs. We perform experiments varying the hyperparameters of the network as well as the size of the dataset and show there exists a correlation between the smoothness gap across some layers and overall accuracy of the corresponding DNN.**

I. Introduction

Deep Neural Networks (DNNs) have become the state-of-the-art in many machine learning benchmarks ever since the AlexNet [6] won the ILSVRC-2012 competition. Due to the fact they rely on millions of trainable parameters, DNNs remain black-box methods. As a consequence, there is little understanding of the reasons for their generalization abilities and finding the best hyperparameters for a given problem requires to exhaustively search a lot of combinations.

It is often considered that architectures that are not optimal for a given problem suffer either from what is called underfitting or overfitting. Underfitting typically refers to conditions where increasing the number of trainable parameters in some parts of the architecture would lead to better generalization performance. To the contrary, overfitting refers to conditions where the network is containing too many parameters and despite being very efficient at classifying the training set, it fails when facing unseen examples.

To avoid underfitting and overfitting often boils down to performing crossvalidation, where a fraction of the training set – called validation set – is extracted to assess the generalization performance of a DNN trained on the remaining fraction of the training set. However, this requires to reduce the size of the training set, thus leading to globally poorer accuracy, and it does not guarantee the architecture will be the best for a set distinct from the validation one.

Graph signal processing (GSP) [7] is a recent framework which aims at extending classical harmonic analysis techniques to domains that can be defined by graphs. In previous work [1], [2], GSP has been used to analyse DNNs. In this paper, we are interested in showing that graph smoothness can be used to predict underfitting and overfitting in Deep Neural Networks.

II. Smoothness Gap

First we formally define what we call smoothness gap. We follow the notations from [2]. Let us consider M example inputs for each of the C classes which we then use to generate intermediate representations across a given trained DNN.

In the remaining of this work, we shall denote $y^\ell_{c,i}$ the intermediate representation of the i-th input of class c and at the output of layer ℓ, following Python's notation (where -1 means last layer and -2 penultimate layer). From a collection of signals $\left(y^\ell_{c,i}\right)_{c,i}$ we derive a similarity graph. As such, each signal is associated one-to-one with a vertex. We call label signal a binary vector $s_c \in \mathbb{R}^{CM}$, where all coordinates are zero but the ones mapped to signals belonging to that class.

We introduce $G_\ell = \left\langle V, A^\ell \right\rangle$ the weighted graph at layer ℓ where:

$$V = \{(c,i), 1 \leq c \leq C, 1 \leq i \leq M\}$$

$$A^\ell_{(c,i)(c\prime,i\prime)} = \cos(y^\ell_{c,i}, y^\ell_{c\prime,i\prime})$$

Note that we chose to use the cosine similarity after performing experiments with various other metrics such as Manhattan, Euclidean and/or Manahalobis [8], [4]. One of the main aspects that was important to us is that the cosine similarity is bounded by default.

We then introduce the k-nearest neighbor graph $G_{\ell,k} = \left\langle V, A^{\ell,k} \right\rangle$ associated with $G_\ell = \left\langle V, A^\ell \right\rangle$ where $A^{\ell,k}$ is obtained from A^ℓ by keeping only values that are among the k largest ones on their row or on their column. As a consequence, note that $A^{\ell,k}$ contains less than $2MCk$ nonzero elements. Choosing the correct value of k is a well known problem and the results can be quite sensitive to this decision.

Denote by $L^{\ell,k}$ the combinatorial Laplacian of the graph $G_{\ell,k}$ and $L^\top$ the transpose of the matrix L. We call smoothness of the label signal s_c on the graph $G_{\ell,k}$ the quantity $s_c^\top L^{\ell,k} s_c$. Smoothness of a label signal is a direct measure of how well the examples of this class are separated from the other classes. Since there are multiple classes, we are more interested in the global label smoothness $\sigma_\ell = \sum_c s_c^\top L^{\ell,k} s_c$. A global label smoothness of 0 indicates pairs of examples belonging to distinct classes are not connected through $G_{\ell,k}$ or are completely orthogonal.

In order to compare smoothness of a given signal on various graphs with possibly very different weightings, we choose to normalize smoothness by its maximum possible value. In our case, we rather use an upperbound which is $2MCk$.

In [2], the authors suggest that the smoothness gap in the last layers of the network is a good proxy to how underfitted/overfitted the considered DNN is. As a consequence, we use the measure:

$$\delta_s = \frac{\sigma_{-3} - \sigma_{-2}}{2MCk} \ .$$

III. Experiments

A. Details

For experiments, we use PreActResNet18-inspired DNNs [3] trained on a portion of the CIFAR-10 dataset [5]. To estimate label smoothness, we sample 50 examples from each class to generate our graphs ($M = 50$ and $C = 10$). We repeat this sampling 10 times. The results reported are the mean label smoothness over the 10 graphs. To artificially create underfitting and overfitting conditions, we proceed as follows:

- **Overfitting:** we use only a portion of the training set ranging from 21% to 99% by 2% increments. Smaller values result in highly overfitted DNNs,
- **Underfitting:** convolutional layers come with a hyperparameter which is the number of feature maps. In order to easily vary the number of trainable parameters without changing the global architecture, we thus vary the number of feature maps. In the chosen architecture, the number of feature maps on the first convolutional layer determines all the others. We thus vary it from 5 to 64, its default value.

We considered various values of k $(10, 20, M = 50, MC = 500)$. Most consistent results were obtained with a value of 20. Using 10 would incur on a lot of points being concentrated with approximately zero smoothness. This is not surprising as it tends to select only the very nearest neighbors. Using M or MC leads to a lot of noise in the measures as there are many more pairwise distances to take into consideration.

B. Results

In Figure 1 we show that by varying the size of the dataset we can generate highly overfitted DNNs. Moreover, there is a correlation between generalization abilities reported by the test accuracy score and the smoothness gap δ_s. We stressed this fact by computing a linear regression and obtained a R^2 coefficient of 68%.

We also obtained a strong correlation between the test accuracy and the smoothness gap δ_s in the case of underfitted DNNs. We computed a linear regression and obtained a R^2 coefficient of 84%.

These results show a very high predictability of the test error given the smoothness gap δ_s. This is very interesting as the computation of the smoothness gap does not require any knowledge about the test set. Note that for the overfitted condition, the architecture contains exactly the same number of parameters each time. To the contrary, for the underfitted condition, the performance of the network could be directly derived from the number of parameters ($R^2 = 50\%$). In order to be sure that our measure was not only an indirect measure of the size of the penultimate layer, we performed additional experiments where the number of feature maps at each layer is changed independently, resulting in almost no correlation between global number of parameters and test accuracy ($R^2 = 14\%$), while still maintaining the correlation between the smoothness gap and test accuracy ($R^2 = 67\%$). Results are depicted in Figure 2.

IV. Conclusion

In this paper we have shown via experiments that there exists a strong correlation between smoothness gap and generalization abilities in deep neural networks. Future work includes further testing and understanding why the training set smoothness is correlated withs the test set accuracy, using this measure explicitly when performing hyperparameter search, and studying how to use this measure during the training phase.

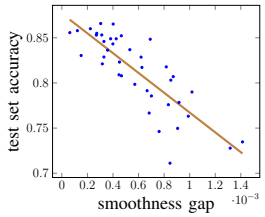

Figure 1. Results generated by varying the size of the dataset.

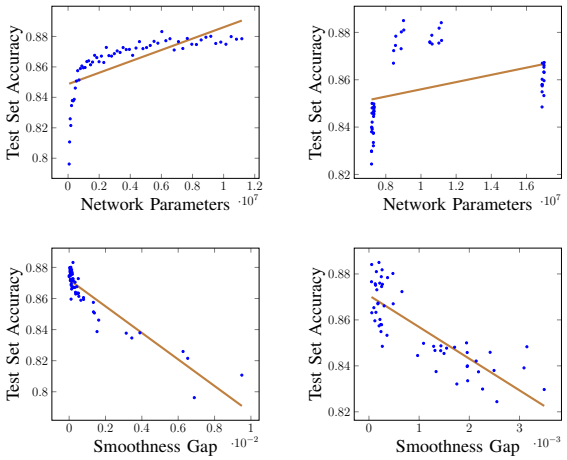

Figure 2. Results generated by varying the size of the DNNs. The right column shows results where the size of feature maps is changed independently, while the left column the feature maps depend on the size of the first convolutional layer. While the upper row shows the correlation between the number of trainable parameters and the network performance, while the lower row shows the correlation between the smoothness gap and network performance. The brown lines are linear regressions.

References

[1] R. Anirudh, J. J. Thiagarajan, R. Sridhar, and T. Bremer. MARGIN: Uncovering Deep Neural Networks using Graph Signal Analysis. *ArXiv e-prints*, November 2017.

[2] V. Gripon, A. Ortega, and B. Girault. An Inside Look at Deep Neural Networks using Graph Signal Processing. *to appear in ITA 2018*.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[4] Eamonn Keogh and Abdullah Mueen. Curse of dimensionality. In *Encyclopedia of Machine Learning and Data Mining*, pages 314–315. Springer, 2017.

[5] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

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

[7] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing. *arXiv preprint arXiv:1712.00468*, 2017.

[8] Pourya Zadeh, Reshad Hosseini, and Suvrit Sra. Geometric mean metric learning. In *International Conference on Machine Learning*, pages 2464–2471, 2016.