First exploration of the potential of diverse training and voting for increasing the accuracy of CNNs

Julián Proenza, Yolanda González and Patricia Arguimbau
DMI - Universitat de les Illes Balears, Spain
{julian.proenza,yolanda.gonzalez,patricia.arguimbau}@uib.es

***Abstract*—Machine learning techniques are attracting a huge amount of interest from both industry and academia. For instance, Convolutional deep Neural Networks (CNNs) have recently enjoyed a notable success in image understanding. The automotive industry is already using image classifiers for Advanced Driver-Assistance Systems and in the development of the upcoming autonomous cars, which will have to guarantee high levels of reliability. The certification of systems based on machine learning is an open issue but it is clear that any improvement in the performance of image classifiers is to be welcomed. CNNs need to be trained to act as image classifiers. This training leads to slightly different classification capacity depending on some training parameters. In this paper we present a first exploration on the use of schemes based on voting on the results of several CNNs trained differently, as a means to increase the final classification performance, and thus the reliability, of this type of systems.**

I. Introduction

Machine learning is attracting a huge amount of interest from both industry and academia due to its high potential for solving complex problems in a more effective way than conventional methods based on task-specific algorithms. Convolutional deep Neural Networks (CNNs) are computing systems that use the machine learning principles and that have recently enjoyed a notable success in image classification or semantic segmentation, triggered by the significant improvement achieved in 2011 over the best performing classification methods described in the literature of the time [1].

The recent advances in machine learning and artificial intelligence (AI) in general have caught the attention of key sectors such as the automotive industry. One example is the use of image classifiers for Advanced Driver-Assistance Systems (ADAS) [2]. These classifiers are to be used as well in the unstoppable evolution towards level-5 fully autonomous cars, whose critically will make mandatory the use of fault tolerance.

The current trend is to mix information obtained from a wide variety of sensors (e.g. vision, radar, LIDAR, ultrasound) and it is accepted that advanced sensor fusion is especially important for robust performance in complex perception situations [3]. Moreover the natural redundancy of sensor information caused by the continuous sensor sampling (e.g. video) is used for boosting the efficacy. Nevertheless, the certification of the correct behavior of AI-based systems in general is an open issue that needs to be addressed by the critical-systems industry. Therefore, and as a first step in this direction, it is also clear that any kind of improvement in the performance of each sensor and its corresponding recognition algorithms will be welcomed, especially if the sensor is cost-effective such as is the case of vision when compared with LIDAR.

CNNs need to be trained before using them. For classification tasks and during the training phase the CNN receives a sequence of images from a database and with each image a tag that indicates the class the CNN should be able to identify in the image. As a consequence the CNN adjusts some internal coefficients, the so called weights. This training process leads to slightly different results depending on the actual set of images and their order in the sequence, among other hyperparameters of the CNN architecture. For instance, the same CNN architecture, once trained, could be able to classify a specific test image or not depending on said factors.

Some classical fault tolerance techniques exploit design diversity in order to tolerate design faults. In N-Version Programming (NVP), diverse versions of the same program are written by independent teams and then, in execution, segments of each version that are intended to generate the same results in the absence of faulty versions are executed and then a voting takes place on the results obtained by each version. Assuming design faults are independent and infrequent, NVPs voting is capable of producing the right result for each segment even in the presence of design faults in a minority of versions.

The slight diversity that is observed in different trainings of the same CNN architecture could be seen as a kind of design diversity and thus susceptible of being leveraged for improving the final accuracy by completing N trainings and voting on their results. The key requirement for this strategy to be effective is that the majority of the trainings have to be able to correctly classify the images that a minority is not able to classify. Only under those circumstances voting would be more effective than using a single training. In this paper we explore this possibility. In this initial approach to this subject the authors have chosen a 6 layer CNN trained over the MNIST dataset. This election allows reducing training times and takes advantage of a simple architecture and a well-known dataset.

II. Diversity in the performance of CNNs

When a CNN, with an adequate internal architecture, is to be used as image classifier, a set of *training images* is input to the CNN together with the tags that identify the classes to which each of the images belong (e.g. what animal is shown in the image). During the training, the CNN learns the key features of each class of images by means of adjusting the weights. Then, once the training is finished, the classification accuracy

of the CNN is tested by inputting a set of test images (not used during the training) for each of which the CNN indicates the class it belongs to.

The fact is that depending on the way this training takes place, e.g. the specific set of images used for training or the order in which these images are processed in the CNN, the final capacity of the trained CNN for classifying the test images can be different. To the authors best knowledge there is no final research that has completely studied this phenomenon. In any case we see this diversity in the results of the training as a potential opportunity for using voting-based schemes for obtaining a final redundant system that is able to perform better than each one of the individual CNNs used to build it.

In this section we delve into the different sources of diversity in the training of a well known CNN using a widely used database of images. The CNN we have chosen is the one provided by Google's developers in their Tensorflow tutorials webpage [4]. This CNN is rather simple, which means it has a just 6 layers but still it is able to achieve a level of accuracy in the classification of images that is widely accepted as state-of-the-art nowadays.

On the other hand, the database we have used for training and testing the performance of the CNN is MNIST [5], which consists of 60000 training images and 10000 test images, each of them of a handwritten number from 0 to 9. This means the database provides 10 different classes that the CNN will have to be able to identify. In order to favor the generality in the classification capability of the CNN, the images correspond to numbers which are written by different persons and the training set is much larger than the test one. Note that the images of this database do not correspond to the kind of images that need to be processed in automotive applications, since we just want to carry out a first general study and thus we have simply chosen a widely known database.

The training of the CNN using MNIST consists of an iterative optimization algorithm to find the parameters and weights that minimize a loss function. In our case, the cross entropy loss function is used to estimate how badly the CNN are classifying the input images. In each iteration, as the entire dataset cannot be passed to the CNN at once, a number of training images is randomly selected to form a subset, called batch, with a fixed cardinality that can be decided by the user for the whole training. Each time, the loss function can be estimated comparing the predictions of the CNN with the image tags and the weights are adjusted to minimize the loss. The training phase can be configured to take a fixed number of iterations or o stop when the loss is zero or low enough.

Summarizing, in a first look at this working procedure we can identify multiple sources of diversity in the final performance of the trained CNN. Indeed, selecting a different subset of images for training, or a different batch size, or a different condition for stopping the training (either a previously fixed number of iterations or loss equal to zero or to other low-enough quantity) the CNN is likely to be able to identify some test images differently. Additionally, even if the previous three parameters are the same for two independent trainings, the mere random choosing of the specific images that are part of each batch will create diversity.

On the other hand, the architecture of the CNN is also designed by the user. This means that by changing the architecture we can also get different classification results, but this source of diversity has not been further explored in this paper.

In this rest of the paper, we are going to report on the experiments we have carried out so far in order to see whether these sources of diversity can be used to improve the classification capabilities of CNNs by building redundant systems.

III. Voting on diverse replicas of a CNN

Classical fault-tolerant systems have used replication and voting successfully in order to tolerate faults in a minority of the replicas. The usual example of this approach is Triple-Modular Redundancy (TMR) in which three replicas of the same hardware generate their outputs in parallel and there is a specific device called *voter* that is in charge of voting on the results of each replica and decide the output of the resulting system. The minimum number of versions for voting is 3, but it is also possible to use any higher odd number, which we will denote with N, leading to the so called N-Modular Redundancy (NMR). Typically it is fair to expect an increase of reliability when N is increased. When design faults are to be tolerated, the replicas need to exhibit design diversity. This is the approach used in NVP for tolerating software faults, which are by definition design faults.

In the work presented in this paper we use NMR and NVP as basic references. In both cases, the vote is a critical aspect of the resulting system operation. In NVP different kinds of voting algorithms are considered depending on the type of the output data [6]. For instance when bit-by-bit identical values are expected for the outputs of the versions, *exact match* is used. In contrast, when small differences are acceptable (e.g. for floating point numbers obtained from sensors' readings), *numeric match* is used, which consist on an average calculation that rules out values that differ too much from the rest.

In the case of the CNN considered for this work, once it is trained, its input is an image and its output is a list of the probabilities with which the input image corresponds to each of the 10 potential classes. Based on this type of output we propose two different voting algorithms.

Let p_i with $i \in \{0, \ldots, 9\}$ be the probability (calculated by the CNN) that the input image corresponds to the class of the number i. Considering the case in which there are 3 versions (same CNN architecture trained in three different ways) of the CNN, let ${p_i}^j$ with $j \in \{A, B, C\}$ be the value of p_i as calculated by the j version of the CNN, therefore $P^j = \{{p_i}^j\}_{i=0}^{9}$ will denote the set of the probabilities of each class calculated by the j version of the CNN .

Using *Voting Algorithm 1* (VA1) each version j calculates its own ${p_{\max}}^j = \max P^j$ and it chooses the class corresponding to that ${p_{\max}}^j$ as its output to the voter. Then the voter performs a simple majority voting and only in case two versions propose

the same class the voter will be able to produce a, hopefully correct, result. In contrast, using *Voting Algorithm 2* (VA2) each version j only calculates the set of values P^j as output to the voter. The voter then calculates a new set $P^\star = \{{p_i}^A + {p_i}^B + {p_i}^C\}_{i=0}^9$ and then ${p_{\max}}^\star = \max P^\star$. Then it chooses the class corresponding to that ${p_{\max}}^\star$ as final result. VA2 will potentially outperform VA1 in scenarios in which for the same j there are several ${p_i}^j$ with values close to the maximum one, but it cannot be anticipated which VA will yield a better accuracy in general, due to the wide variety of scenarios that are possible when considering the potential values of all ${p_i}^j$.

IV. Description of the conducted experiments

In the previous sections we have identified some parameters that can cause different results in the performance of image classifiers based on CNNs. First we discussed several sources of diversity in the training process, which we will call *training parameters*. These are the set of images actually used for training, the size of the used batch, the training stopping condition, the random selection of the specific images in each batch, and finally the architecture of the CNN. Second we have presented potential voting strategies, which are again likely to produce diverse results. Indeed we introduced two voting algorithms but it has to be noted that it is also possible to have different results by choosing different numbers of versions. As said above, the minimum number of versions for voting is 3, but it is also possible to use any higher odd number and typically expect a better reliability as a result. We will call *structural parameters* to the voting algorithm and the number of versions.

We have carried out a series of experiments in which we combine values of the training and structural parameters for generating different system configurations and for each of them diversity in the training process. More specifically, we have used some of the parameters above to generate different system configurations (i.e. the values of said parameters are the same for all versions) and then for each one of these configurations we have used other parameters for generating diversity in the training of the different versions.

The parameters that we have varied for generating configurations are those that it would have been unfair or unnatural to set differently for each version. For instance, if a certain value of the batch size used during the training is likely to produce better classification results, it would be unnatural to use a different value for some of the versions. Similarly, for some parameters such as the voting algorithm or the number of versions it simply makes no sense to consider different values for different versions. In contrast, the parameters we have set differently for different versions as a means to create some natural diversity in them are the specific set of images used for training and the actual order of the images in the batches.

In Table I we show the results of the experiments. Each line corresponds to a configuration of the training parameters and it is identified with a number. Looking at the columns now, we have considered the values 3 and 5 for the number of versions (N) generated in each case. Each one of these versions has been trained in some cases with all the 60000 images available in the database and in some other cases with a subset of 60000 / N images. The specific images in each subset have been chosen randomly to prevent intentionally biased results. For each number of training images, two batch sizes have been chosen (50 and 100 images per batch). Finally for each one of the previous combinations, two training stopping conditions have been used: loss equal to zero and loss $< 10^{-6}$. With each one of these 16 training configurations, each one of the corresponding N versions has been trained 10 times, since in each training the specific images in each batch per iteration are chosen randomly and, thus, the learning result will be slightly different. Then each one of the $(10 \times N)$ trained versions has been evaluated using the 10000 test images and the corresponding accuracy (ratio between the number of images correctly classified and the 10000 images of the test) has been calculated. The minimum, average and maximum values of this accuracy considering the 10 different trainings for each version and configuration are shown in the column called *Simplex Accuracy* (we use the term *simplex* as a synonym of non-redundant). Taking one of the 10 training of each version, 10 NMR configurations are built and for each one of them, both the use of VA1 and VA2 have been considered. The columns *VA1 Accuracy* and *VA2 Accuracy* show the minimum, average and maximum accuracy obtained when voting (using VA1 or VA2, respectively) on the values proposed by each one of the N versions.

V. Discussion of the results

In a first analysis of the results in Table I it is easy to notice several general trends. Regarding the simplex (non-redundant) CNNs, the accuracy is better for trainings that use all the 60000 training images of the database than for those that use a subset of them. This is not surprising since the basic principle of learning in CNNs is to accumulate as much and diverse experience as possible and thus it is intuitive to see that training with more images has to be better. This tendency is also clear when considering the voting setups (columns *VA1 Accuracy* and *VA2 Accuracy*). Concerning the size of the batch, there are cases in which choosing 50 leads to slightly better results but also cases in which the opposite occurs. For the stopping condition, it seems the situation is more consistent and the zero loss condition leads always to better results. This is coherent with the fact that this condition forces the CNN to keep training for a much larger number of iterations. For this reason, Table I only shows the accuracy values for the zero loss condition in the case of the voting configurations.

When comparing the accuracy of the simplex configuration with those of the voting ones, again some general tendencies appear. First, the average accuracy is always better for voting configurations than for the simplex one. Additionally, this average is again better when using VA2 than when using VA1, which means that indeed VA2 is able to properly deal with complex classification scenarios in which the different versions propose similar probabilities for several classes.

TABLE I: Accuracy of simplex and redundant CNN systems for 16 training configurations

Conf.	Number of Versions	Training Images	Batch Size	Loss	Simplex Accuracy			VA1 Accuracy			VA2 Accuracy		
					Min	Average	Max	Min	Average	Max	Min	Average	Max
1	3	60000	50	0	0.9909	0.992633452	0.9939	0.9927	0.9934	0.994	0.9927	0.99358	0.9941
2	3	60000	50	$< 10^{-6}$	0.987	0.990553459	0.9921						
3	3	60000	100	0	0.9915	0.992286791	0.9936	0.9926	0.99299	0.9936	0.9926	0.9932	0.9939
4	3	60000	100	$< 10^{-6}$	0.9905	0.99177679	0.993						
5	3	20000	50	0	0.9847	0.987453467	0.9891	0.9884	0.98954	0.9902	0.9891	0.99005	0.991
6	3	20000	50	$< 10^{-6}$	0.985	0.986953459	0.9885						
7	3	20000	100	0	0.9844	0.987206803	0.9885	0.9888	0.98938	0.9898	0.9896	0.99009	0.9906
8	3	20000	100	$< 10^{-6}$	0.985	0.986706787	0.9882						
9	5	60000	50	0	0.9913	0.992438127	0.9938	0.9896	0.99357	0.9931	0.9932	0.99369	0.9943
10	5	60000	50	$< 10^{-6}$	0.9852	0.99054413	0.9923						
11	5	60000	100	0	0.9911	0.992240002	0.9936	0.9927	0.99316	0.9936	0.9928	0.99333	0.9936
12	5	60000	100	$< 10^{-6}$	0.9901	0.991709995	0.9935						
13	5	12000	50	0	0.9835	0.985193998	0.9871	0.9864	0.98728	0.9877	0.9872	0.9881	0.9886
14	5	12000	50	$< 10^{-6}$	0.9816	0.984100004	0.9861						
15	5	12000	100	0	0.9821	0.985220129	0.9868	0.9865	0.98763	0.9888	0.9875	0.98831	0.9892
16	5	12000	100	$< 10^{-6}$	0.982	0.984350128	0.9864						

Voting with 5 versions is also better than voting with 3. This can be seen comparing configurations 1 and 3 with configurations 9 and 11. However the tendency is not reflected in the comparison between configurations 5 and 7 with configurations 13 and 15. A reasonable explanation is that in 13 and 15 much less images are used for training each version. This makes us think that the higher the accuracy achieved by each of the individual versions is, the higher the improved accuracy of the voting configuration will be.

So far the observations are mostly positive and consistent with what it happens in classical NMR configurations designed to tolerate hardware faults. Unfortunately other aspects of the table are not so encouraging. First the improvements achieved are not huge. And second it is possible to see in Table I that the maximum accuracy that is achievable by a simplex configuration is in most of the cases higher than the minimum it can be achieved with any voting configuration. This means that there is no guarantee that voting is actually going to give us a better performance than taking the *best possible* training of a simplex version. A relevant aspect to add to this discussion is that both the design of the optimal CNN architecture and the tuning of its parameters to obtain optimal performance results are still open issues in machine learning. Therefore the common user, once the architecture is fixed, does not have definitive tools for making sure she has obtained the best out of the training. Thus the presented voting approaches could be seen as a way to ensure that the performance of the resulting system is going to be above the average that can be obtained with a simplex version. Additionally using N versions we can have tolerance to hardware faults in case each version is executed in a different computer.

Even if these results are not conclusive in terms of showing an overwhelming advantage of voting over the simplex configuration, they allow reflecting on the need of training optimizing methods. The next steps of this work should be to assess the voting schemes with images that are likely to produce higher accuracies (e.g. of traffic signals) and then see the effects of combining voting on different versions with the voting on results obtained in time (e.g. out of a video).

VI. Conclusions

There are still too many parameters in the training of CNNs that are tuned using best-effort approaches. This does not seem the most appropriate way for obtaining the kind of optimal results that are expected when targeting critical applications. In this context, the voting techniques presented in this paper offer the possibility of achieving a better performance in average without having to put an extra effort in understanding the internals of the CNN. These techniques are compatible with any improvement that simplex CNNs could experience in their performance. This paper represents a first step towards a better understanding of the potential of diverse training as a means to improve the performance of CNNs.

Acknowledgements

This work is supported in part by the Spanish Agencia Estatal de Investigación (AEI) and in part by FEDER funding through grant TEC2015-70313-R (AEI/FEDER, UE).

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[2] M. Teichmann, M. Weber, J. M. Zöllner, R. Cipolla, and R. Urtasun, "Multinet: Real-time joint semantic reasoning for autonomous driving," *CoRR*, vol. abs/1612.07695, 2016.

[3] Á. González, M. Á. Garrido, D. F. Llorca, M. Gavilán, J. P. Fernández, P. F. Alcantarilla, I. Parra, F. Herranz, L. M. Bergasa, M. Á. Sotelo *et al.*, "Automatic traffic signs and panels inspection system using computer vision," *IEEE Transactions on intelligent transportation systems*, vol. 12, no. 2, pp. 485–499, 2011.

[4] Tensorflow tutorials. [Online]. Available: https://www.tensorflow.org/tutorials/layers

[5] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *AT&T Labs [Online].*, vol. 2, 2010.

[6] S. V. Makam, "Design study of a fault-tolerant computer system to execute N-Version Software," Ph.D. dissertation, Computer Science Department. University of California, Los Angeles (UCLA), December 1982.