A Training-time Analysis of Robustness in Feed-Forward Neural Networks

Cesare Alippi
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Milano, Italy
E-mail: alippi@elet.polimi

Daniele Sana, Fabio Scotti
Department of Information Technologies
University of Milan
Crema, Italy
E-mail: {sana,fscotti}@dti.unimi.it

Abstract—**The paper addresses the *analysis of robustness over training time* issue. Robustness is evaluated in the large, without assuming the small perturbation hypothesis, by means of Randomised Algorithms. We discovered that robustness is a strict property of the model -as it is accuracy- and, hence, it depends on the particular neural network family, application, training algorithm and training starting point. Complex neural networks are hence not necessarily more robust than less complex topologies. An early stopping algorithm is finally suggested which extends the one based on the test set inspection with robustness aspects**

I. Introduction

The robustness analysis goal is to estimate the variation in accuracy with respect to perturbations affecting a computational flow [1] [2] and, hence, quantify the model resilience to perturbations. In the neural network literature, robustness analysis has mainly focused on the impact of perturbations affecting weights and biases. A robustness analysis is beneficial both on theory and application sides since weights and biases constitute the "knowledge space" of a neural model: an accuracy index for a neural network augmented with a robustness index allow the researcher for a global and synthetic characterization of the neural network behaviour. A robustness analysis for the network weights has also an immediate impact on the physical realization of the neural network. In this context, perturbations affecting the network's weights abstract physical uncertainties induced by finite precision representations, deviation of parameters from nominal values and faults. Other physical phenomena abstracted by perturbations are fluctuations of the production parameters representing the weights in analog solutions, ageing effects or more complex and subtle uncertainties in mixed implementations.

Recent advances in the theory of robustness analysis allow researchers for removing strict hypotheses assumed in the related literature [7] [9] [11] [4] basically assuming a linearised analyses, the small perturbation hypothesis and/or given particular distributions for interim neural variables. [3] and [2] have demonstrated that a general robustness analysis can be applied to the very large class of Lebesgue measurable functions by means of a poly-time complexity algorithm based on Randomised Algorithms.

Since neural networks fully satisfy the Lebesgue measurability requirement, in this paper we adapt and apply such robustness analysis to study the evolution of the robustness index for perturbed weights over training time and investigate the relationships between neural network complexity (e.g., in terms of hidden units number), perturbations affecting the network weights and accuracy. An early stopping method based on test set investigation is then suggested which aims at identify a trade-off between neural network accuracy and weights robustness.

The structure of the paper is as follows. Section II introduces and provides an algorithm to estimates the robustness index for perturbations affecting the network weights. Section III analyses the evolution of such a robustness index during training time while the early-stopping method compromising robustness and accuracy is suggested in section IV.

II. A general robustness analysis

Randomized Algorithms -RAs- [3] [12] are here envisaged to transform the computationally intractable problem of evaluating the robustness in the large of a generic neural network with respect to generic, continuous perturbations affecting its weights, in a tractable problem solvable with a poly-time algorithm.

In the following we consider a feedforward neural network -not necessarily fully trained- implementing the $y = f(x, \theta)$ function where θ is a vector containing all the free parameters (weights) of the network.

A. A robustness index

A general, perturbation-size independent robustness analysis requires the evaluation of the loss in performance induced by a generic perturbation affecting the weights of a generic neural network. We denote by $y(x, \theta_\Delta)$ the mathematical description of the perturbed computation (i.e., obtained by perturbing the network's weights) and by $\Delta \in D \subseteq \Re^k$ a generic p-dimensional perturbation vector, a component for each independent perturbation affecting θ. The perturbation space D is characterized in stochastic terms by means of a probability density function pdf_Δ. The pdf_Δ abstracts, de facto, the effective sources of uncertainty affecting the network weights. For instance, if weights are represented as resistors in a fully analog implementation then a gaussian distribution nicely abstracts errors introduced by the production process.

0-7803-8359-1/04/$20.00 ©2004 IEEE

When such a distribution is unknown we can consider a uniform distribution for its conservative property.

To measure the discrepancy between $y(x,\theta)$ and $y(x,\theta_\Delta)$ we consider a generic *loss function* $U(\cdot)$ we assume to be measurable according to Lebesgue with respect to D. A common example for U is the Mean Square Error -MSE- loss function but any other loss function can be considered instead:

$$U(x,\Delta) = \frac{1}{N_x}\sum_{i=1}^{N_x}\big(y(x_i) - y(x_i,\theta_\Delta)\big)^2 \tag{1}$$

and estimates the performance of the error-affected (perturbed) neural network (generalization ability of the perturbed neural model).

The impact of perturbations on the performance function can be quantified by introducing an index of robustness $\bar{\gamma}$ quantifying the impact of perturbations affecting weights on the neural network performance. We say that a neural network is robust at level $\bar{\gamma}$ in D, when $\bar{\gamma}$ is the minimum positive value for which

$$U(x,\Delta) \leq \bar{\gamma}, \forall\Delta \in D, \forall\gamma \geq \bar{\gamma} \tag{2}$$

Directly from the definition we have that neural network NN_1 is more robust than neural network NN_2 with perturbation defined in D iff $\bar{\gamma}_1 < \bar{\gamma}_2$ (the property holds independently from the topology of the two neural networks).

The main problem related with the determination of the robustness index $\bar{\gamma}$ is that we have to compute $U(x,\Delta)$ $\forall\Delta \in D$. The $\bar{\gamma}$-identification problem is therefore intractable from a computational point of view if we relax all assumptions made in the literature as we do. The problem can be solved by associating a dual probabilistic problem to (2).

We say that a neural network is robust at level $\bar{\gamma}$ in D, with confidence η, when $\bar{\gamma}$ is the minimum positive value for which:

$$Pr\big(U(\Delta) \leq \bar{\gamma}\big) \geq \eta, \text{holds } \forall\Delta \in D, \forall\gamma > \bar{\gamma} \tag{3}$$

In other words, not more than $100\eta\%$ of perturbations $\Delta \in D$ will generate a loss in performance larger than $\bar{\gamma}$.

Probabilistic and deterministic problems are "close" each other when we choose, as we do, η very close to 1. Note that $\bar{\gamma}$ depends only on the size of D and the neural network model.

The non-linearity with respect to Δ and the lack of a priori assumptions regarding the neural network do not allow computing (2) in a closed form for the general perturbation case. The analysis, which would imply testing $U(\Delta)$ in correspondence with a continuous perturbation space, can be solved by resorting to Randomized Algorithms.

B. Randomized algorithms and perturbation analysis

Denote by $p_\gamma = Pr\big(U(\Delta) \leq \gamma\big)$ the probability that the loss in performance induced by perturbations in D is below a given -but arbitrary- value γ. The unknown probability p_γ can be estimated by sampling D according to the pdf_Δ with N independent and identically distributed samples Δ_i. For each Δ_i we generate the triplet

$$\big\{\Delta_i, U(\Delta_i), I(\Delta_i)\big\}, i = 1\cdots N \tag{4}$$

where

$$I(\Delta_i) = \begin{cases} 1, & \text{if } U(\Delta_i) \leq \gamma \\ 0, & \text{if } U(\Delta_i) > \gamma, \end{cases} \tag{5}$$

The true probability p_γ can be estimated as:

$$\hat{p}_N = \frac{1}{N}\sum_{i=1}^{N} I(\Delta_i) \tag{6}$$

Of course, when N tends to infinity, $\hat{p}_N$ converges to p_γ. Conversely, on a finite data set of cardinality N the discrepancy between $\hat{p}_N$ and p_γ is $|p_\gamma - \hat{p}_N|$. By introducing an accuracy degree ε and a confidence level $1-\delta$ we require that inequality

$$Pr\big\{|p_r - \hat{p}_N| \leq \varepsilon\big\} \geq 1 - \delta \tag{7}$$

is satisfied for $\forall\gamma \geq 0$. The relationship holds by considering N satisfying the Chernoff inequality [6]

$$N \geq \frac{ln\frac{2}{\delta}}{2\varepsilon^2} \tag{8}$$

As an example, by considering a 5% in accuracy and 99% in confidence we have to extract 1060 samples from D.

C. Estimating $\bar{\gamma}$.

The dual probabilistic problem related to the identification of the robustness index $\bar{\gamma}$ can be solved with randomized algorithms and therefore with a polynomial complexity in the accuracy and the confidence degrees independently from the number of weights of the neural model network. In fact, by expanding the (7) we have that

$$Pr\Big\{|Pr(U(\Delta) \leq \gamma) - \frac{1}{N}\sum_i I(\Delta_i)| \leq \varepsilon\Big\} \geq 1 - \delta \tag{9}$$

If accuracy ε and confidence δ are small enough we can confuse p_γ and $\hat{p}_N$ by committing a small error. As a consequence, the dual probabilistic problem requiring $p_\gamma \geq \eta$ becomes $\hat{p}_N \geq \eta$.

The final algorithm, which allows us for testing the robustness degree $\bar{\gamma}$ of a neural network can be summed up as:

(1) Select ε and δ sufficiently small to have enough accuracy and confidence
(2) Extract a number of perturbations N from D, according to pdf_Δ, as suggested by (8).
(3) Estimate $\hat{p}_N = \hat{p}_N(\gamma)$ according to (6) $\forall\gamma$.
(4) Select the minimum value γ_η from the $\hat{p}_N = \hat{p}_N(\gamma)$ function so that $\hat{p}_N(\gamma_\eta) = 1$ is satisfied $\forall\gamma \geq \gamma_\eta$. γ_η is the estimate the of the robustness index $\bar{\gamma}$.

Note that with a simple algorithm we are able to estimate in polynomial time the robustness degree $\bar{\gamma}$ of a generic neural network.

III. THE $\bar{\gamma}(t)$ CURVE: ROBUSTNESS AND ACCURACY OVER TRAINING TIME

To study the evolution of the weights robustness over training time we have to compute the $\bar{\gamma}$ curve at the end of each training cycle, i.e., in correspondence with the $y = f(x, \theta(t))$ neural model. In turn, this requires a proper characterisation of the perturbation space D and its connection with the weights' space.

A. Generating the $\bar{\gamma}(t)$ curve

In the following we assume a multiplicative perturbation model by requiring that perturbation Δ_i affecting a generic weight of the neural network $\theta_i(\bar{t})$ at training time $\bar{t}$ is proportional to its weight magnitude

$$\theta_{\Delta,i}(\bar{t}) = \theta_i(\bar{t})(1 + \Delta_i), \quad \forall i = 1, \cdots n \tag{10}$$

where n is the cardinality of the θ vector. Since a uniform distribution is a rather severe perturbation and we wish to maximally and equally excite the knowledge space of the neural network, we consider the case where Δ_i is drawn from a symmetrical uniform distribution of extremes $[-\frac{T}{100}, \frac{T}{100}]$.

As such, a 5% perturbation affecting weights and biases composing vector θ implies that $T = 5$ and each weight/bias is affected by an independent perturbation extracted from the $[-0.05, 0.05]$ interval and applied to the nominal weight value according to the envisaged multiplicative perturbation model. We assume a 5% perturbation model in the following analysis. Given training time $\bar{t}$ we have to test the robustness of neural network $y(x, \theta(\bar{t}))$ by generating the $\hat{p}_N = \hat{p}_N(\gamma)$ curve as suggested by the algorithm delineated in the previous section. Figure 1 shows a typical pattern for a $\hat{p}_N(\gamma)$ curve and $\hat{\gamma}$. A unique run is sufficient to estimate $\hat{\gamma}$ with good accuracy provided that ϵ and δ are sufficiently small from the theory.

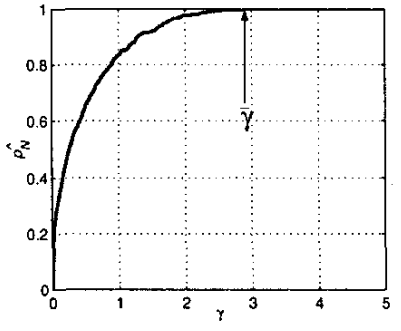

Fig. 1. A $\hat{p_N}(\gamma)$ curve.

By inspecting figure 1 the robustness index estimate $\bar{\gamma}$ is γ_η=3, since it is the smallest value for which $\hat{p}_N = 1$,$\forall \gamma \geq \bar{\gamma}$. In turn, this implies that $U(\Delta) \leq 3$, $\forall \delta \in D$, with high probability.

This procedure is then iterated at the end of each training time and generates the $\bar{\gamma}(t)$ curve which, de facto, represents the evolution of the robustness index over training time. Of course, the analysis can be carried out by using data coming from the training data set DS_{train}, the test data set DS_{test} or the validation set $DS_{validaton}$ according to the user needs.

B. The envisaged datasets

Examples here proposed have been obtained by processing two datasets DS_A and DS_B. DS_A is a regression problem with a dataset composed of uniformly selected input samples defined in the [0.1 1.9] interval. The y values come from the curve $y(x) = 4.26(e^{-x} - 4e^{-2x} + 3e^{-3x})$ suggested in [10]. The Train/Test/Validation sets cardinality are 25/25/100. Dataset DS_B is a 2-dimensional classification dataset suggested in [5], and refers to two non-linearly separable classes. In this case, the Train/Test/Validation sets cardinality are 200/25/100. Both applications refer to a non-linear regression problem; we consider neural families with a single hidden unit layer and a unique output neuron.

C. Studying the behaviour of the $\bar{\gamma}(t)$ curve

It must be pointed out that the experimented behaviour of the robustness index strongly depends on several experimental issues. Nevertheless, some general comments can be outlined which have general validity.

1) The $\bar{\gamma}(t)$ curve strongly depends on the available data samples, the neural network family (e.g., number of hidden neurons), the starting point of the training algorithm and the training algorithm itself: In different words, robustness w.r.t. a perturbation space D is a property of the model as it is accuracy; different models will experience a different behaviour in $\bar{\gamma}(t)$. Three examples are shown in Figure 5 for the DS_B case; all the experimental set up is fixed for the regression problem application but the starting training points are different.

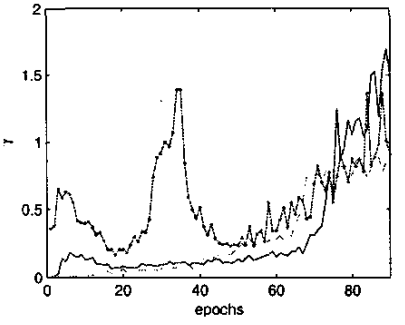

Fig. 2. Three $\bar{\gamma}(t)$ curves plotted over training time for the same feed-forward neural network family, considering different starting points.

2) The $\bar{\gamma}(t)$ curve experiences a rough behaviour independently from ϵ and δ: The not smooth behaviour of $\bar{\gamma}(t)$ over training time does not depend on the statistical fluctuation associated with randomisation. The same behaviour also arises when $\epsilon \rightarrow 0$ and $\delta \rightarrow 0$. We expect this behaviour to be associated with the not continuous nature of the training algorithm which, by starting from a point in the weight space, generates the next point reasonably "far" from the previous one.

3) Behaviour when $t \rightarrow \infty$: We observed that the behaviour of $\bar{\gamma}(t)$ over training time when $t \rightarrow \infty$ tends to stabilize to a monotonically increasing or decreasing curve.

4) The impact of the hidden units number on $\bar{\gamma}(t)$: We experimentally verified that the impact of the hidden units number on robustness strongly depends on the particular application. As such, it is not true in general that networks with a reduced number of hidden units are generally less robust than the ones possessing more degrees of freedom (i.e., that large networks provide always a sort of spatial redundancy) as pointed out also by other authors.

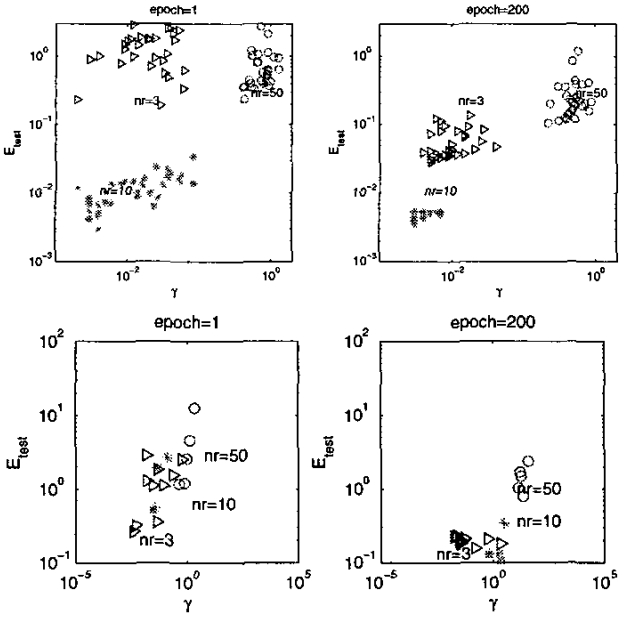

Fig. 3. Effects of the number of hidden units in the $\bar{\gamma}(t)$ and E_{test} plane (top: DS_A problem; bottom: DS_B problem)

The effect associated with the hidden units number on the envisaged data sets is given in figure 3. Different runs have been considered for the nr=3,10 and 50 hidden units cases. We can see that when the training epoch proceed clusters arise in the $\bar{\gamma}(t)$ and E_{test} plane; each cluster is associated with a different number of hidden units.

5) Patterns of γ are almost related to the model: Robustness is a strict property of the model -as it is accuracy- and, hence, it depends on the particular neural network family, application, training algorithm and training starting point. In our experiments we evaluated the robustness index during training time both on the test and the validation datasets. Results show a strong correlation between the two resulting patterns. Figure 4 shows an example of γ patterns produced by a 5-hidden-unit feed-forward neural network evaluated in test and validation using DS_B .

6) Typical trajectories are present in the $\bar{\gamma}(t)$ versus E_{test} plane: Considering the generalization/robustness plane, experiments enlighten four typical patterns that can be encountered. Patterns are related to presence/absence of the

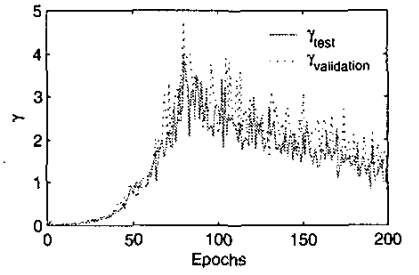

Fig. 4. Examples of $\bar{\gamma}$ values processed during training evaluated with test and validation data, for the same neural network.

overfitting phenomenon and if the robustness index $\bar{\gamma}$ tends to increase/descrease during time.

Figure 5 (top) schematizes the four distinct A, B, C and D patterns in the E_{test} versus $\bar{\gamma}$ plane. Figure 5 (bottom) shows the four basic patterns in real cases. In particular, it shows the trajectories during the training phase of four neural networks plotted in figure 3 (top) belonging in the 50-hidden-units cluster. Experiments show that basic patterns can be also combined giving rise to more complicated trajectories. Simpler network tend to generate simpler patterns since there is less probability to reveal overfitting. It means that the curves tend to be composed by pattern types A and B.

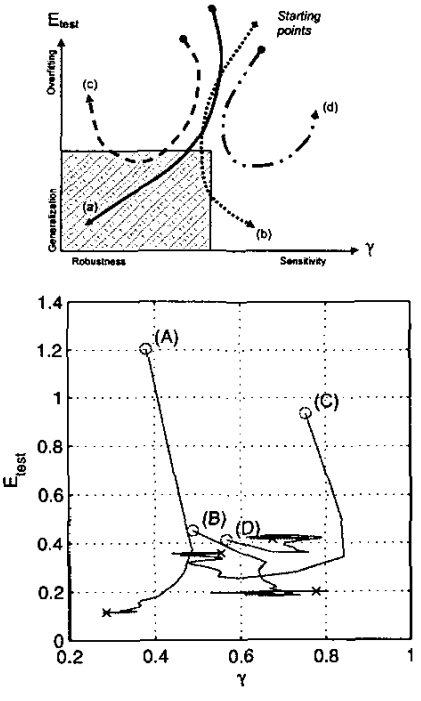

Fig. 5. Top: schema of neural networks's trajectories during training in the robustness/generalization plane. Overfitting is present in patterns C and D (not in pattern A and B). γ is increasing during training in pattern B ad D (not in A and C). Bottom: real examples of A, D, C and D patterns in the robustness/generalization plane. Circles represent the starting points of the trajectories.

IV. A PROPOSAL FOR A ROBUSTNESS/ACCURACY-BASED EARLY-STOPPING METHOD

Since robustness and accuracy are apparently independent high-level model properties, it is interesting to identify an early stopping method compromising accuracy and robustness.

Very interestingly, a *little* variation in selecting the stopping time $\bar{t}$ can produce even *large* variations in robustness for the final neural network. An example of such a situation in given in figure 6 where the evolution of E_{test} and γ is plotted for a 5 hidden neurons neural network trained on the DS_B dataset. Circles represent the $\epsilon = 0.05$-equivalent points w.r.t. E_{test}, i.e., the set of neural network configurations generated during training whose test error is below the minimum E_{test} plus ϵ. The triangle point identifies the global minimum of E_{test} over the training trajectory. Such model is generally selected as the optimal one.

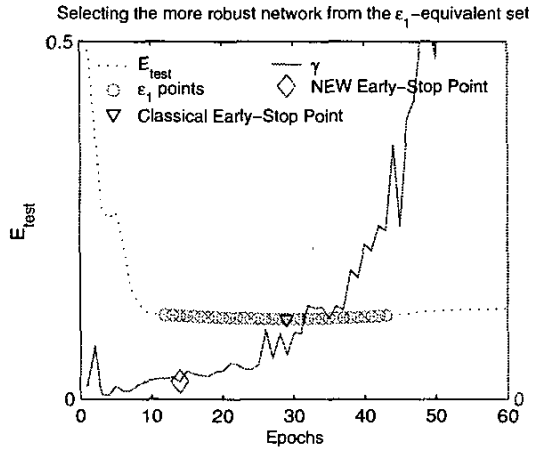

Fig. 6. The proposed method to stop the training.

Hence, by monitoring only the behavior of $E_{test}(t)$ it may be difficult to guarantee a satisfactory generalization/robustness compromise. In addition, since $E_{test}(t)$ and the $\bar{\gamma}(t)$ curves are not enough correlated, the $\bar{\gamma}(t)$ curve by itself does not allow the user for identifying an effective stopping point solving the compromise.

We therefore wish to suggest an early stop condition which tackles, at the same time, generalization and robustness issues. We assume that robustness and generalization estimates over the test set are in line with the estimates over the validation one and, indirectly, that the number of data samples is large enough.

The straightforward early stopping algorithm con be summarized as:

(1) Select, during training, those networks satisfying inequality $E_{test}(t) \leq E_{test}(t_1) + \epsilon$ where $E_{test}(t_1)$is the minimum test error found during training (classical early-stopping point). Insert such networks in the I set.

(2) Select from I the model characterized by the smallest $\bar{\gamma}$ value; such model is the one solving the accuracy/robustness compromise.

We applied the proposed method to the case plotted in figure 6. We considered only the ϵ-equivalent set of models (the ones identified by circles) and we identified the new stopping point. It corresponds to the the most robust neural (with lower γ) network belonging to the ϵ-equivalent set (the parallelogram in figure).

Let us now compare the results of the proposed method with respect to the classical early-stopping approach by plotting the generalization and robustness indexes of the neural network stopped in t_1 (new early-stopping method) and t_2 (classical early-stopping method). Figure 7 shows the comparison. In the lower part of the figure, we plot the ϵ-equivalent performance of the neural network over E_{test} and the associated validation performance against the robustness index. DS_{test} performance are represented as circles, the validation ones $DS_{validation}$ as dots. It is immediate to see that the ϵ-equivalent networks experience different values in $\bar{\gamma}$, even when test and validation errors are comparable. In this case, the application of the new stopping criterion leads to a more robust neural network.

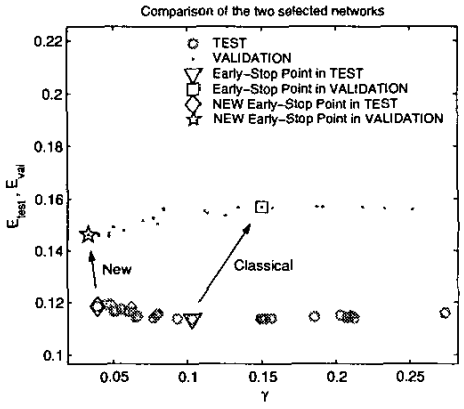

Fig. 7. Comparison of the two early-stopping methods

Others experiments clearly show that small variations in the perturbation space (few percentile) do not significantly change the observed behaviour and, hence, it seems that the suggested stopping method is effective. Nevertheless, new experiments are necessary to better understand the relationship between robustness and accuracy of a feed-forward neural network during its training.

V. CONCLUSIONS

The paper investigates the evolution of the robustness index over training time and its relationships with the neural network accuracy and complexity. It is found that robustness is only weakly related to accuracy and, as such, it must be intended as an independent property. Classical early stopping methods (which solely aim at accuracy) can be improved by considering a tradeoff between accuracy and robustness. Interesting preliminary results show the feasibility of the hint.

REFERENCES

[1] C. Alippi, "Application-Level Robustness and Redundancy in Linear Systems", *IEEE Transactions on Circuits and Systems - 1:Fundamental Theory and Applications*, Vol.49, No.7, 2002

[2] C. Alippi, "Selecting Accurate, Robust and Minimal Feedforward Neural Networks", *IEEE Transactions on Circuits and Systems - 1:Fundamental Theory and Applications*, Vol.49, No.12, 2002.

[3] C. Alippi, "Randomized Algorithms: A System-Level, Poly-Time analysis of robust computation", *IEEE Transactions on Computers*, Vol. 51, No. 7, 2002.

[4] C. Alippi, V. Piuri, M. Sami, "Sensitivity in Errors in Artificial Neural Networks: A Behavioural Approach". *IEEE Transactions on Circuits and Systems - 1:Fundamental Theory and Applications*, Vol.42, No.6, 1995.

[5] F.Blayo, Y. Cheneval, et.al."Enhanced Learning for Evolutive Neural Architecture", in *Deliverable R3-B4-P TaskB4: Benchmarks*, from Esprit Research Project Number 6891, 1995

[6] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations", Ann. Math. Stat. 23, 1952.

[7] C. Dundar, K. Rose, "The Effects of Quantization on Multilayer Neural Networks", *IEEE Transactions on Neural Networks*, Vol. 6, Pagg.1446-1451, 1995.

[8] P.Koopman, Embedded systems Design Issues (the Rest of the Story), *Proceedings of the IEEE-ICCD*, 1996

[9] J. Holt, J. Hwang, "Finite Precision Error Analysis of Neural Network Hardware Implementations", *IEEE Transactions on Computers*, Vol. 42, Pagg. 281-290, 1993.

[10] K. Hornik, M. Stinchombe, and H. White, "Multilayer feedforward networks are universal approximators," in *Neural Networks*, vol. 2, 1989.

[11] M. Stevenson, R. Winter, B. Widrow, "Sensitivity of Feedforward Neural Networks to Weigths Errors", *IEEE Transaction on Neural Networks*, Vol. 1, No. 1.

[12] M. Vidyasagar, "An Overview of Computational Learning Theory and its Applications to Neural Network Training", Identification, Adaption, Learning, NATO ASI Series F, Vol. 153, Pagg. 400-422, 1996.