This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Curriculum Dropout

Pietro Morerio[1], Jacopo Cavazza[1,2], Riccardo Volpi[1,2], René Vidal[3] and Vittorio Murino[1,4]

[1]Pattern Analysis & Computer Vision (PAVIS) – Istituto Italiano di Tecnologia – *Genova, 16163, Italy*
[2]Electrical, Electronics and Telecommunication Engineering and Naval Architecture Department (DITEN) – Università degli Studi di Genova – *Genova, 16145, Italy*
[3]Department of Biomedial Engineering – Johns Hopkins University – *Baltimore, MD 21218, USA*
[4]Computer Science Department – Università di Verona – *Verona, 37134, Italy*

`{pietro.morerio,jacopo.cavazza,riccardo.volpi,vittorio.murino}@iit.it, rvidal@cis.jhu.edu`

Abstract

Dropout is a very effective way of regularizing neural networks. Stochastically "dropping out" units with a certain probability discourages over-specific co-adaptations of feature detectors, preventing overfitting and improving network generalization. Besides, Dropout can be interpreted as an approximate model aggregation technique, where an exponential number of smaller networks are averaged in order to get a more powerful ensemble. In this paper, we show that using a fixed dropout probability during training is a suboptimal choice. We thus propose a time scheduling for the probability of retaining neurons in the network. This induces an adaptive regularization scheme that smoothly increases the difficulty of the optimization problem. This idea of "starting easy" and adaptively increasing the difficulty of the learning problem has its roots in curriculum learning and allows one to train better models. Indeed, we prove that our optimization strategy implements a very general curriculum scheme, by gradually adding noise to both the input and intermediate feature representations within the network architecture. Experiments on seven image classification datasets and different network architectures show that our method, named Curriculum Dropout, frequently yields to better generalization and, at worst, performs just as well as the standard Dropout method.

1. Introduction

Since [17], deep neural networks have become ubiquitous in most computer vision applications. The reason is generally ascribed to the powerful hierarchical feature representations directly learnt from data, which usually outperform classical hand-crafted feature descriptors.

As a drawback, deep neural networks are difficult to train because non-convex optimization and intensive computations for learning the network parameters. Relying on availability of both massive data and hardware resources, the aforementioned training challenges can be empirically tackled and deep architectures can be effectively trained in an end-to-end fashion, exploiting parallel GPU computation.

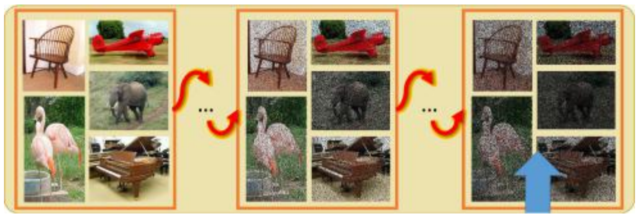
Figure 1. From left to right, during training (red arrows), our curriculum dropout gradually increases the amount of Bernoulli multiplicative noise, generating multiple partitions (orange boxes) within the ***dataset*** (yellow frame) and the ***feature representation*** layers (not shown here). Differently, the original dropout [13, 25] (blue arrow) mainly focuses on the hardest partition only, complicating the learning from the beginning and potentially damaging the network classification performance.

However, *overfitting* remains an issue. Indeed, such a gigantic number of parameters is likely to produce weights that are so specialized to the training examples that the network's generalization capability may be extremely poor.

The seminal work of [13] argues that overfitting occurs as the result of excessive co-adaptation of feature detectors which manage to perfectly explain the training data. This leads to overcomplicated models which unsatisfactory fit unseen testing data points. To address this issue, the Dropout algorithm was proposed and investigated in [13, 25] and is nowadays extensively used in training neural networks. The method consists in randomly suppressing neurons during training according to the values r sampled from a Bernoulli distribution. More specifically, if $r = 1$ that unit is kept unchanged, while if r=0 the unit is suppressed. The effect of suppressing a neuron is that the value of its output is set to zero during the forward pass of training, and its weights are not updated during the backward pass. One one forward-backward pass is completed, a new sample of r is drawn from each neuron, and another forward-backward pass is done and so on till convergence. At testing time, no neuron is suppressed and all activations are modulated by the mean value of the Bernoulli distribution. The resulting model is in fact often interpreted as an average of multiple models, and it is argued that this improves its generalization ability [13, 25].

Leveraging on the Dropout idea, many works have proposed variations of the original strategy [14, 22, 31, 30, 1, 20]. However, it is still unclear which variation improves the most with respect to the original dropout formulation [13, 25]. In many works (such as [22]) there is no real theoretical justification of the proposed approach other than favorable empirical results. Therefore, providing a sound justification still remains an open challenge. In addition, the lack of publicly available implementations (*e.g.*, [20]) make fair comparisons problematic.

The point of departure of our work is the intuition that the excessive co-adaptation of feature detectors, which leads to overfitting, are very unlikely to occur in the early epochs of training. Thus, Dropout seems unnecessary at the beginning of training. Inspired by these considerations, in this work we propose to dynamically increase the number of units that are suppressed as a function of the number of gradient updates. Specifically, we introduce a generalization of the dropout scheme consisting of a temporal scheduling - a *curriculum* - for the expected number of suppressed units. By adapting in time the parameter of the Bernoulli distribution used for sampling, we smoothly increase the suppression rate as training evolves, thereby improving the generalization of the model.

In summary, the main contributions of this paper are the following.

1. We address the problem of overfitting in deep neural networks by proposing a novel regularization strategy called Curriculum Dropout that dynamically increases the expected number of suppressed units in order to improve the generalization ability of the model.

2. We draw connections between the original dropout framework [13, 25] with regularization theory [8] and curriculum learning [2]. This provides an improved justification of (Curriculum) Dropout training, relating it to existing machine learning methods.

3. We complement our foundational analysis with a broad experimental validation, where we compare our Curriculum Dropout versus the original one [13, 25] and anti-Curriculum [22] paradigms, for (convolutional) neural network-based image classification. We evaluate the performance on standard datasets (MNIST [19, 26], SVHN [21], CIFAR-10/100 [16], Caltech-101/256 [9, 10]). As the results certify, the proposed method generally achieves a superior classification performance.

The remaining of paper is outlined as follows. Relevant related works are summarized in §2 and Curriculum Dropout is presented in §3 and §4, providing foundational interpretations. The experimental evaluation is carried out in §5. Conclusions and future work are presented in §6.

2. Related Work

As previously mentioned, dropout is introduced by Hinton et al. [13] and Sivrastava et al. [25]. Therein, the method is detailed and evaluated with different types of deep learning models (Multi-Layer Perceptrons, Convolutional Neural Networks, Restricted Boltzmann Machines) and datasets, confirming the effectiveness of this approach against overfitting. Since then, many works [29, 20, 30, 1, 31, 14, 28, 22] have investigated the topic.

Wan et al. [29] propose Drop-Connect, a more general version of Dropout. Instead of directly setting units to zero, only some of the network connections are suppressed. This generalization is proven to be better in performance but slower to train with respect to [13, 25]. Li et al. [20] introduce data-dependent and Evolutional-dropout for shallow and deep learning, respectively. These versions are based on sampling neurons form a multinomial distribution with different probabilities for different units. Results show faster training and sometimes better accuracies. Wang et al. [30] accelerate dropout. In their method, hidden units are dropped out using approximated sampling from a Gaussian distribution. Results show that [30] leads to fast convergence without deteriorating the accuracy. Bayer et al. [1] carry out a fine analysis, showing that dropout can be proficiently applied to Recurrent Neural Networks. Wu and Gu [31] analyze the effect of dropout on the convolutional layers of a CNN: they define a probabilistic weighted pooling, which effectively acts as a regularizer. Zhai and Zhang [33] investigate the idea of dropout once applied to matrix factorization. Ba and Frey [14] introduce a binary belief network which is overlaid on a neural network to selectively suppress hidden units. The two networks are jointly trained, making the overall process more computationally expensive. Wager et al. [28] apply Dropout on generalized linear models and approximately prove the equivalence between data-dependent L^2 regularization and dropout training with AdaGrad optimizer. Rennie et al. [22] propose to adjust the dropout rate, linearly decreasing the unit suppression rate during training, until the network experiences no dropout.

While some of the aforementioned methods can be applied in tandem, there is still a lack of understanding about which one is superior - this is also due to the lack of publicly released code (as happens in [20]). In this respect, [22] is the most similar to our work. A few papers do not go beyond a bare experimental evaluation of the proposed dropout variation [20, 1, 31, 14, 22], omitting to justify the soundness of their approach. Conversely, while some works are much more formal than ours [30, 28, 33], all of them rely on approximations to carry out their analysis which is biased towards shallow models (logistic [28] or linear regression [30, 28] and matrix factorization [33]). Differently, in our paper, in addition to its experimental effectiveness, we provide several natural justifications to corroborate the proposed dropout generalization for deep neural networks.

3. A Time Scheduling for the Dropout Rate

Deep Neural Networks display co-adaptations between units in terms of concurrent activations of highly organized clusters of neurons. During training, the latter specialize themselves in detecting certain details of the image to be classified, as shown by Zeiler and Fergus [32]. They visualize the high sensitivity of certain filters in different layers in detecting dogs, people's faces, wheels and more general ordered geometrical patterns [32, Fig. 2]. Moreover, such co-adaptations are highly generalizable across different datasets as proved by Torralba's work [34]. Indeed, the filter responses provided in the AlexNet within *conv1*, *pool2/5* and *fc7* layers are very similar [34, Fig. 5], despite the images used for the training are very different: objects from ImageNet versus scenes from Places datasets.

These arguments support the existence of some *positive* co-adaptations between neurons in the network. Nevertheless, as soon as the training keeps going, some co-adaptations can also be *negative* if excessively specific of the training images exploited for updating the gradients. Consequently, exaggerated co-adaptations between neurons weaken the network generalization capability, ultimately resulting in overfitting. To prevent it, Dropout [13, 25] precisely contrasts those negative co-adaptations.

The latter can be removed by randomly suppressing neurons of the architecture, restoring an improved situation where the neurons are more "independent". This empirically reflects into a better generalization capability [13, 25].

Network training is a dynamic process. Despite the previous interpretation is totally sound, the original Dropout algorithm cannot precisely accommodate for it. Indeed, the suppression of a neuron in a given layer is modeled by a Bernoulli(θ) random variable[1], $0 < \theta \leq 1$. Employing such distribution is very natural, since it statistically models binary activation/inhibition processes. In spite of that, it seems suboptimal that θ should be *fixed* during the whole training stage. With this operative choice, [13, 25] is actually treating the negative co-adaptations phenomena as uniformly distributed during the whole training time.

[1]To avoid confusion in our notation, please note that θ is the equivalent of p in [13, 25, 28], i.e the probability of *retaining* a neuron.

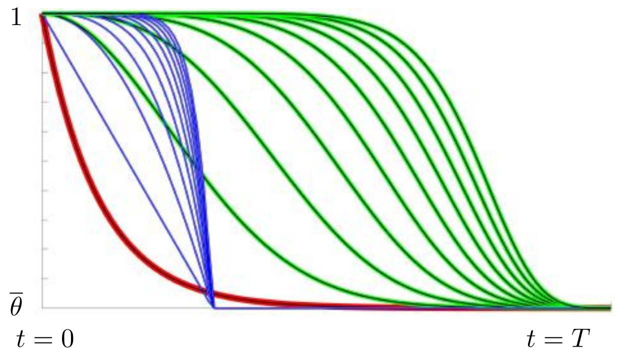

Figure 2. Curriculum functions. Eq. (1) (red), polynomial (blue) and exponential (green).

Differently, our intuition is that, *at the beginning of the training, if any co-adaptation between units is displayed, this should be preserved* as positively representing the self-organization of the network parameters towards their optimal configuration.

We can understand this by considering the random initialization of the network's weights. They are statistically independent and actually not co-adapted at all. Also, it is quite unnatural for a neural network with random weights to overfit the data. On the other hand, the risk of overdone co-adaptations increases as the training proceeds since the loss minimization can achieve a small objective value by overcomplicating the hierarchical representation learnt from data. This implies that *overfitting caused by excessive co-adaptations appears only after a while*.

Since a fixed parameter θ is not able to handle increasing levels of negative co-adaptations, in this work, we tackle this issue by proposing a temporal dependent $\theta(t)$ parameter. Here, t denotes the training time, measured in gradient updates $t \in \{0, 1, 2, \dots\}$. Since $\theta(t)$ models the probability for a given neuron to be retained, $D \cdot \theta(t)$ will count the average number of units which remain active over the total number D in a given layer. Intuitively, such quantity must be higher for the first gradient updates, then starting decreasing as soon as the training gears. In the late stages of training, such decrease should be stopped. We thus constrain $\theta(t)$ to be $\theta(t) \geq \overline{\theta}$ for any t, where $\overline{\theta}$ is a limit value, to be taken as $0.5 \leq \overline{\theta} \leq 0.9$ as prescribed by the original dropout scheme [25, §A.4] (the higher the layer hierarchy, the lower the retain probability).

Inspired by the previous considerations, we propose the following definition for a ***curriculum function*** $\theta(t)$ aimed at improving dropout training (as it will become clear in section 4, from now on we will often use the terms *curriculum* and *scheduling* interchangeably).

Definition 1. *Any function* $t \mapsto \theta(t)$ *such that* $\theta(0) = 1$

and $\lim_{t\to\infty} \theta(t) \searrow \overline{\theta}$ *is said to be a curriculum function to generalize the original dropout [13, 25] formulation with retain probability* $\overline{\theta}$.

Starting from the initial condition $\theta(0) = 1$ where no unit suppression is performed, dropout is gradually introduced in a way that $\theta(t) \geq \overline{\theta}$ for any t. Eventually (*i.e.* when t is big enough), the convergence $\theta(t) \to \overline{\theta}$ models the fact that we retrieve the original formulation of [13, 25] as a particular case of our curriculum.

Among the functions as in Def. 1, in our work we fix

$$\theta_{\text{curriculum}}(t) = (1 - \overline{\theta}) \exp(-\gamma t) + \overline{\theta}, \ \gamma > 0 \quad (1)$$

By considering Figure 2, we can provide intuitive and straightforward motivations regarding our choice.

The blue curves in Fig. 2 are polynomials of increasing degree $\delta = \{1, \ldots, 10\}$ (left to right). Despite fulfilling the initial constraint $\theta(0) = 1$, they have to be manually thresholded to impose $\theta(t) \to \overline{\theta}$ when $t \to \infty$. This introduces two more (undesired) parameters (δ and the threshold) with respect to [13, 25], where the only quantity to be selected is $\overline{\theta}$.

The very same argument discourages the replacement of the variable t by t^α in (1), (green curves in Fig. 2, $\alpha = \{2, \ldots, 10\}$, left to right). Moreover, by evaluating the area under the curve, we can intuitively measure how aggressively the green curves behave while delaying the dropping out scheme they eventually converge to (as $\theta(t) \to \overline{\theta}$). Precisely, that convergence is faster while moving to the green curves more on the left, being the fastest one achieved by our scheduling function (1) (red curve, Fig. 2).

One could still argue that the parameter $\gamma > 0$ is annoying since it requires cross validation. This is not necessary: in fact, γ can actually be *fixed* according to the following heuristics. Despite Def. 1 considers the limit of $\theta(t)$ for $t \to \infty$, such condition has to be operatively replaced by $t \approx T$, being T the total number of gradient updates needed for optimization. It is thus totally reasonable to assume that the order of magnitude of T is a priori known and fixed to be some power of 10 such as $10^4, 10^5$. Therefore, for a curriculum function as in Def. 1, we are interested in furthermore imposing $\theta(t) \approx \overline{\theta}$ when $t \approx T$. Actually, a rule of thumb such as

$$\gamma = 10/T \quad (2)$$

implies $|\theta_{\text{curriculum}}(T) - \overline{\theta}| < 10^{-4}$ and was used for all the experiments[2] in §5. Additionally, from Figure 2, we can grab some intuitions about the fact that the asymptotic convergence to $\overline{\theta}$ is indeed realized for a quite consistent part of the training and well before $t \approx T$. This means that during a big portion of the training, we are actually dropping out neurons as prescribed in [13, 25], addressing the overfitting issue. In addition to these arguments, we will provide complementary insights on our scheduled implementation for dropout training.

[2]Check the Supplementary Material where we proved that our approach is extremely robust with respect to different γ values.

Smarter initialization for the network weights. The problem of optimizing deep neural networks is non-convex due to the non-linearities (ReLUs) and pooling steps. In spite of that, a few theoretical papers have investigated this issue under a sound mathematical perspective. For instance, under mild assumptions, Haeffele and Vidal [11] derive sufficient conditions to ensure that a local minimum is also a global one to guarantee that the former can be found when starting from *any* initialization. The same theory presented in [11] cannot be straightforwardly applied to the dropout case due to the pure deterministic framework of the theoretical analysis that is carried out. Therefore, it is still an open question whether all initializations are equivalent for the sake of a dropout training and, if not, which ones are preferable. Far from providing any theoretical insight in this flavor, we posit that Curriculum Dropout can be interpreted as a smarter initialization. Indeed, we implement a soft transition between a classical dropout-free training of a network versus the dropout one [13, 25]. Under this perspective, our curriculum seems equivalent to performing dropout training of a network whose weights have already been slightly optimized, evidently resulting in a better initialization for them.

As a naive approach, one can think to perform regular training for a certain amount of gradient updates and then apply dropout during the remaining ones. We call that *Switch-Curriculum*. This actually induces a discontinuity in the objective value which can damage the performance with respect to the smooth transition performed by our curriculum (1) - check Fig. 4.

Curriculum Dropout as adaptive regularization. Several connections [28, 29, 25, 33] have been established between Dropout and model training with noise addition [3, 23, 33]. The common trend discovered is that when an unregularized loss function is optimized to fit artificially corrupted data, this is actually *equivalent* to minimize the same loss augmented by a data dependent penalizing term. In both [28, Table 2.] for linear/logistic regression and [25, §9.1] for least squares, it is proved that Dropout induces a regularizer which is scaled[3] by $\theta(1 - \theta)$.

When $\theta = \overline{\theta}$, the impact of the regularization is just *fixed*, therefore rising potential over- and under-fitting issues [8]. But, for $\theta = \theta_{\text{curriculum}}(t)$, when t is small, the regularizer is set to zero ($\theta_{\text{curriculum}}(0) = 1$) and we *do not* perform any regularization at all. Indeed, the latter is simply not necessary: the network weights still have values which

[3]Please, check the Supplementary Material where we extended such result for a deep neural network, also allowing for a time-dependent $\theta(t)$.

are close to their random and statistically independent initialization. Hence, overfitting is unlikely to occur at early training steps. Differently, we should expect it to occur as soon as training proceeds: by using (1), the regularizer is now weighted by

$$\theta_{\text{curriculum}}(t)(1-\theta_{\text{curriculum}}(t)), \tag{3}$$

which is an increasing function of t. Therefore, the more the gradient updates t, the heavier the effect of the regularization. This is the reason why overfitting is better tackled by the proposed curriculum. Despite the overall idea of an adaptive selection of parameters is not novel for either regularization theory [12, 7, 4, 24, 6] or tuning of network hyper-parameters (e.g. learning rate, [5]), to the best of our knowledge, this is the first time that this concept of time-adaptive regularization is applied to deep neural networks.

Compendium. Let us conclude with some general comments. We posit that there is no overfitting at the beginning of the network training. Therefore, differently from [13, 25], we allow for a scheduled retain probability $\theta(t)$ which gradually drops neurons out. Among other plausible curriculum functions as in Def. 1, the proposed choice (1) introduces no additional parameter to be tuned and implicitly provides a smarter weight initialization for dropout training.

The superiority of (1) also relates to i) the smoothly increasingly amount of units suppressed and ii) the soft adaptive regularization performed to contrast overfitting.

Throughout these interpretations, we can retrieve a common idea of smoothly changing difficulty of the training which is applied to the network. This fact can be better understood by finding the connections with Curriculum Learning [2], as we explain in the next section.

4. Curriculum Learning, Curriculum Dropout

For the sake of clarity, let us remind the concept of curriculum learning [2]. Within a classical machine learning algorithm, all training examples are presented to the model in an unordered manner, frequently applying a random shuffling. Actually, this is very different from what happens for the human training process, that is education. Indeed, the latter is highly structured so that the level of difficulty of the concepts to learn is proportional to the *age* of the people, managing easier knowledge when babies and harder when adults. This "start small" paradigm will likely guide the learning process [2].

Following the same intuition, [2] proposes to subdivide the training examples based on their difficulty. Then, the learning is configured so that easier examples come first, eventually complicating them and processing the hardest ones at the end of the training. This concept is formalized by introducing a learning time $\lambda \in [0, 1]$, so that training begins at $\lambda = 0$ and ends at $\lambda = 1$. At time λ, $Q_\lambda(z)$ denotes the distribution which a training example z is drawn from. The notion of curriculum learning is formalized requiring that Q_λ ensures a sampling of examples z which are easier than the ones sampled from $Q_{\lambda+\varepsilon}$, $\varepsilon > 0$. Mathematically, this is formalized by assuming

$$Q_\lambda(z) \propto W_\lambda(z)P(z). \tag{4}$$

In (4), $P(z)$ is the target training distribution, accounting for all examples, both easy and hard ones. The sampling from P is corrected by the factor $0 \leq W_\lambda(z) \leq 1$ for any λ and z. The interpretation for $W_\lambda(z)$ is the measure of the difficulty of the training example z. The maximal complexity for a training example is fixed to 1 and reached at the end of the training, *i.e.* $W_1(z) = 1$, *i.e.* $Q_1(z) = P(z)$. The relationship

$$W_\lambda(z) \leq W_{\lambda+\varepsilon}(z) \tag{5}$$

represents the increased complexity of training examples from instant λ to $\lambda + \varepsilon$. Moreover, the weights $W_\lambda(z)$ must be chosen in such a way that

$$H(Q_\lambda) < H(Q_{\lambda+\varepsilon}), \tag{6}$$

where Shannon's entropy $H(Q_\lambda)$ models the fact that the quantity of information exploited by the model during training increases with respect to λ.

In order to prove that our scheduled dropout fulfills this definition, for simplicity, we will consider it as applied to the input layer only. This is not restrictive since the same considerations apply to any intermediate layer, by considering that each layer trains the feature representation used as input by the subsequent one.

As the images exploited for training, consider the partitions in the dataset including all the (original) clean data and all the possible ways of corrupting them through the Bernoulli multiplicative noise (see Fig. 1). Let π denote the probability of sampling an uncorrupted d-dimensional image within an image dataset (nothing more than a uniform distribution over the available training examples). Let us fix the gradient update t. The case of sampling a dropped-out z is equivalent to sampling the corresponding uncorrupted image z_0 from π and then overlapping it with a binary mask b (of size d), where each entry of b is zero with probability $1 - \theta(t)$. By mapping b to the number i of its zeros,

$$\mathbb{P}[z] = \mathbb{P}[z_0, i] = \binom{d}{i}(1-\theta(t))^i\theta(t)^{d-i} \cdot \pi(z_0). \tag{7}$$

Indeed, $(1 - \theta(t))^i\theta(t)^{d-i}$ is the probability of sampling *one* binary mask b with i zeros and $\binom{d}{i}$ accounts for all the possible combinations. Re-parameterizing the training time $t = \lambda T$, we get

$$Q_\lambda(z) = \binom{d}{i}(1-\theta(\lambda T))^i\theta(\lambda T)^{d-i} \cdot \pi(z_0). \tag{8}$$

By defining $P(z) = Q_1(z)$ and

$$W_\lambda(z) = \frac{1}{P(z)} \binom{d}{i} (1 - \theta(\lambda T))^i \theta(\lambda T)^{d-i} \cdot \pi(z_0), \quad (9)$$

we can easily prove (refer to the Supplementary Material for the complete proof) that the definition in [2] is fulfilled by the choice (8) for curriculum learning distribution $Q_\lambda(z)$.

To conclude, we give an additional interpretation to Curriculum Dropout. At $\lambda = 0$, $\theta(0) = 1$ and no entry of z_0 is set to zero. This clearly corresponds to the easiest available example, since the learning starts at $t = 0$ by considering all possible available visual information. When θ start decreasing to $\theta(\lambda T) \approx 0.99$, only 1% of z_0 is suppressed (on average) and still almost all the information of the original dataset $\mathcal{Z}_0$ is available for training the network. But, as λ grows, $\theta(\lambda T)$ decreases and a bigger number of entries are set to zero. This complicates the task, requiring an improved effort from the model to capitalize from the reduced uncorrupted information which is available at that stage of the training process.

After all, this connection between Dropout and Curriculum Learning was possible thanks to our generalization through Def. 1. Consequently, the original Dropout [13, 25] can be interpreted as considering the single specific value $\overline{\lambda}$ such that $\theta(\overline{\lambda}T) = \overline{\theta}$, being $\overline{\theta}$ the constant retain probability on [13, 25]. This means that, as previously found for the adaptive regularization (see §3), the level of difficulty $W_{\overline{\lambda}}(z)$ of the training examples z is fixed in the original Dropout. This encounters the concrete risk of either oversimplifying or overcomplicating the learning, with detrimental effects on the model's generalization capability. Hence, the proposed method allows to setup a progressive curriculum $Q_\lambda(z)$, complicating the examples z in a smooth and adaptive manner, as opposed to [13, 25], where such complication is fixed to equal the maximal one from the very beginning (Fig. 1).

To conclude, let us note that the aforementioned work [22] proposes a linear *increase* of the retain probability. According to equations (4-6) this implements what [2] calls an anti-curriculum: this is shown to perform slightly better or worse than the no-curriculum strategy [2] and always worse than any curriculum implementation. Our experiments confirm this finding.

5. Experiments

In this Section, we applied Curriculum Dropout to neural networks for image classification problems on different datasets, using Convolutional Neural Network (CNN) architectures and Multi-Layer Perceptrons (MLPs)[4]. In particular, we used two different CNN architectures: LeNet [18] and a deeper one (conv-maxpool-conv-maxpool-conv-maxpool-fc-fc-softmax), further called CNN-1 and CNN-2, respectively. In the following, we detail the datasets used and the network architectures adopted in each case.

MNIST [19] - A dataset of grayscale images of handwritten digits (from 0 to 9), of resolution 28 $\times$ 28. Training and test sets contain 60.000 and 10.000 images, respectively. For this dataset, we used a three-layer MLP, with 2.000 units in each hidden layer, and CNN-1.

Double MNIST - This is a static version of [26], generated by superimposing two random images of two digits (either distinct or equal), in order to generate 64 $\times$ 64 images. The total amount of images are 70.000, with 55 total classes (10 unique digits classes + $\binom{10}{2} = 45$ unsorted couples of digits) . Training and test sets contain 60.000 and 10.000 images, respectively. Training set's images were generated using MNIST training images, and test set's images were generated using MNIST test images. We used CNN-2.

SVHN [21] - Real world RGB images of street view house numbering. We used the cropped 32 $\times$ 32 images representing a single digit (from 0 to 9). We exploited a subset of the dataset, consisting in 6.000 images for training and 1.000 images for testing, randomly selected. We used CNN-2 also in this case.

CIFAR-10 and **CIFAR-100** [16] - These datasets collect 32 $\times$ 32 tiny RGB natural images, reporting 6000 and 600 elements per each of the 10 or 100 classes, respectively. In both datasets, training and test sets contain 50.000 and 10.000 images, respectively. We used CNN-1 for both datasets.

Caltech-101 [9] - 300 $\times$ 200 resolution RGB images of 101 classes. For each of them, a variable size of instances is available: from 30 to 800. To have a balanced dataset, we used 20 and 10 images per class for training and testing, respectively. Images were reshaped to 128×128 pixels. We used CNN-2 again here.

Caltech-256 [10] - 31000 RGB images for 256 total classes. For each class, we used 50 and 20 images for training and testing, respectively. Images were reshaped to 128 $\times$ 128 pixels. We used CNN-2.

For training CNN-1, CNN-2 and MLP, we exploited a cross-entropy cost function with Adam optimizer [15] and a momentum term of 0.95, as suggested in [25]. We used mini-batches of 128 images and fixed the learning rate to be 10^{-4}. Please refer to the Supplementary Material for additional details regarding the architectures and the hyper-parameters.

We applied curriculum dropout using the function (1) where γ is picked using the heuristics (2) and $\overline{\theta}$ is fixed as follows. For both CNN-1 and CNN-2, the retain probability for the input layer was set to $\overline{\theta}_{\text{input}} = 0.9$, selecting $\overline{\theta}_{\text{conv}} = 0.75$ and $\overline{\theta}_{\text{fc}} = 0.5$ for convolutional and fully connected layers, respectively. For the MLP, $\overline{\theta}_{\text{input}} = 0.8$ and

[4]Code available at https://github.com/pmorerio/curriculum-dropout.

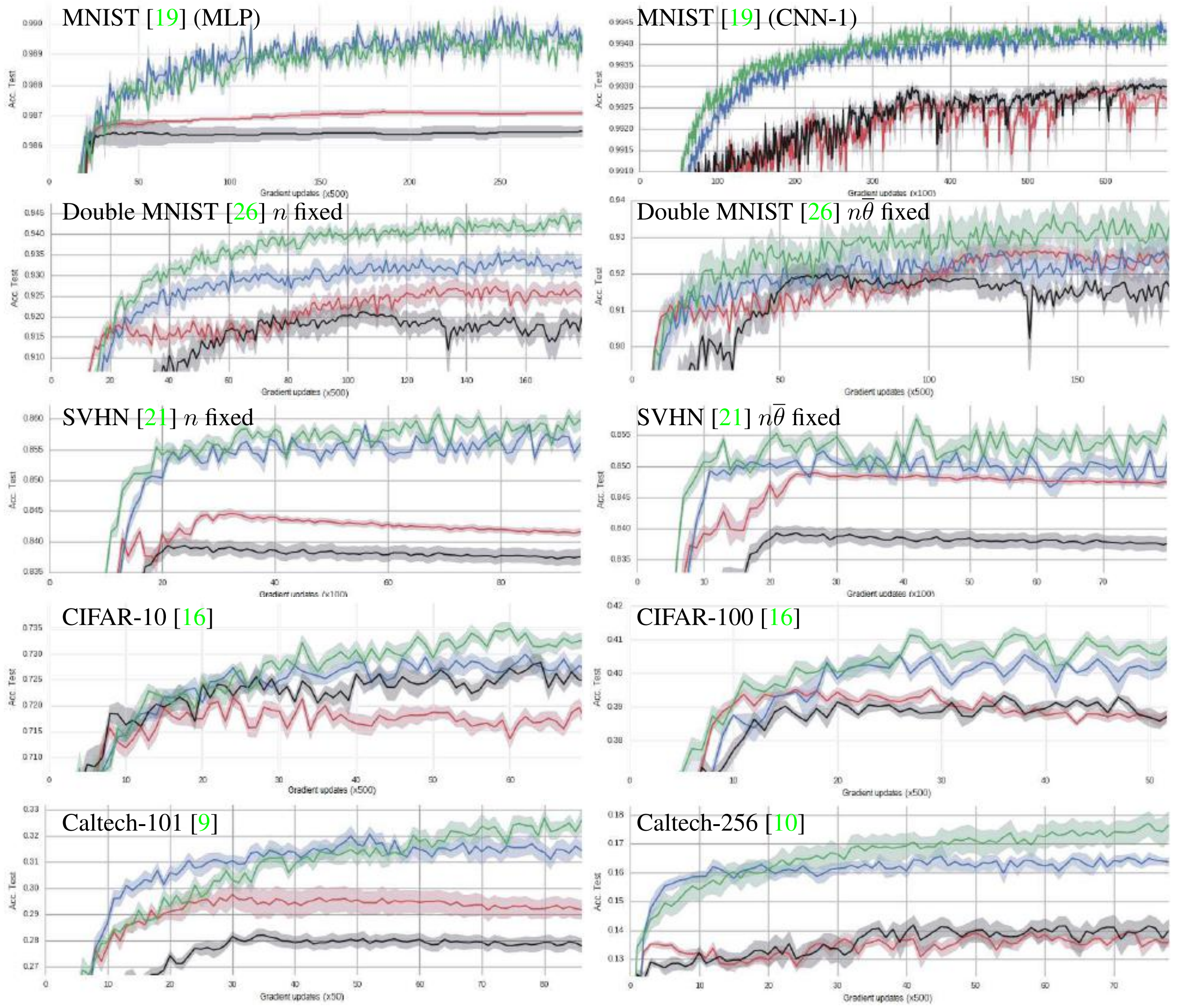

Figure 3. Curriculum Dropout (green) compared with regular Dropout [13, 25] (blue), anti-Curriculum (red) and a regular training of a network with no units suppression (black). For all cases, we plot mean test accuracy (averaged over 10 different re-trainings) as a function of gradient updates. Shadows represent standard deviation errors. Best viewed in colors.

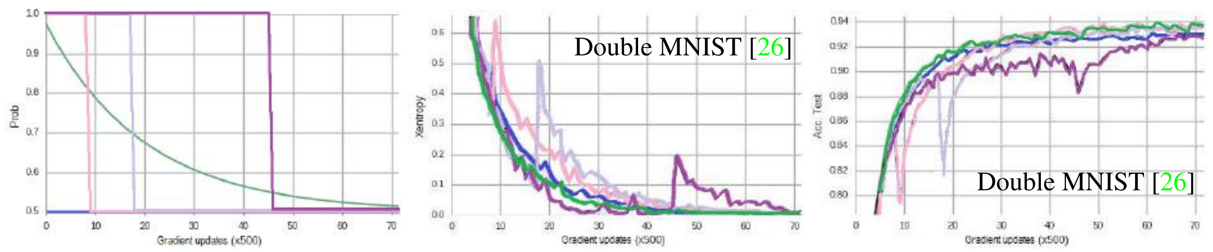

Figure 4. Switch-Curriculum. We compare the Curriculum (green) and the regular Dropout (blue) with three cases where we switch from regular to dropout training i) at the beginning (pink) ii) in the middle (violet), iii) almost at the end (purple) of the learning. From left to right, curriculum functions, cross-entropy loss and test accuracy curves.

$\overline{\theta}_{\text{hidden}} = 0.5$. In all cases, we adopted the recommended values [25, §A.4].

Before reporting our results, let us emphasize that our aim is to improve the standard dropout framework [13, 25], not to compete for the state-of-the art performance in image classification tasks. For this reason, we did not use engineering tricks such as data augmentation or any particular pre-processing, and neither we tried more complex (or deeper) network architectures.

In Fig. 3, we qualitatively compared Curriculum Dropout (green) versus the original Dropout [13, 25] (blue), anti-Curriculum Dropout (red) and an unregularized, *i.e.*

Dataset	Architecture	Configuration (n or $n\overline{\theta}$ fixed)	Classes	Unregularized network	Dropout [13, 25]	Anti-Curriculum	*Curriculum Dropout* (percent boost [27] over Dropout [13, 25])
MNIST [19]	MLP	n	10	98.67	**+0.38**	+0.04	*+0.36* (-5.3%)
	CNN-1	n		99.25	+0.15	-0.05	***+0.18 (20.0%)***
Double MNIST	CNN-2	n	55	92.48	+1.42	+0.73	***+2.35 (65.5%)***
	CNN-2	$n\overline{\theta}$			+0.87	+0.53	***+1.11 (27.6%)***
SVHN [21]	CNN-2	n	10	84.63	+2.35	+1.17	***+2.65 (12.8%)***
	CNN-2	$n\overline{\theta}$			+1.59	+1.51	***+2.06 (29.6%)***
CIFAR-10 [16]	CNN-1	n	10	73.06	+0.22	-0.68	***+0.62 (182%)***
CIFAR-100 [16]	CNN-1	n	100	39.70	+1.01	+0.01	***+1.66 (64.4%)***
Caltech-101 [9]	CNN-2	n	101	28.56	+4.21	+1.57	***+4.72 (12.1%)***
Caltech-256 [10]	CNN-2	n	256	14.39	+2.36	-0.22	***+3.23 (36.9%)***

Table 1. Comparison of the proposed scheduling versus [13, 25] in terms of percentage accuracy improvement.

no Dropout, training of a network (black). Since CNN-1, CNN-2 and MLP are trained from scratch, in order to ensure a more robust experimental evaluation, we have repeated the weight optimization 10 times for all the cases. Hence, in Fig. 3, we report the mean accuracy value curves, representing with shadows the standard deviation errors.

Additionally, we report in Table 1 the percentage accuracy improvements of Dropout [13, 25], anti-Curriculum Dropout [22] and Curriculum Dropout (proposed) versus a baseline network where no neuron is suppressed. To do that, we selected the average of the 10 highest mean accuracies obtained by each paradigm during each trial; then we averaged them over the 10 runs. We accommodated the metric of [27] to measure the boost in accuracy over [13, 25]. Also, we reproduced for two datasets the cases of fixed layer size n or fixed $n\overline{\theta}$ as in [25, §7.3]. Here the network layers' size n is preliminary increased by a factor $1/\overline{\theta}$, since on average a fraction $\overline{\theta}$ of the units is dropped out. However, we notice that those bigger architectures tend to overfit the data.

Switch-Curriculum. Figure 4 shows the results obtained on Double MNIST dataset by scheduling the dropout with a step function, *i.e.* no suppression is performed until a certain *switch-epoch* is reached (§3). Precisely, we switched at 10-20-50 epochs. This curriculum is similar to the one induced by the polynomial functions of Figure 2: in fact, both curves have a similar shape and share the drawback of a threshold to be introduced. Yet, Switch-Curriculum shows an additional shortcoming: as highlighted by the spikes of both training and test accuracies, the sudden change in the network connections, induced by the sharp shift in the retain probabilities, makes the network lose some of the concepts learned up to that moment. While early switches are able to recover quickly to good performances, late ones are deleterious. Moreover, we were not able to find any heuristic rule for the *switch-epoch*, which would then be a parameter to be validated. This makes Switch-Curriculum a less powerful option compared to a smoothly-scheduled curriculum.

Discussion. The proposed Curriculum Dropout, implemented through the scheduling function (1), improves the generalization performance of [13, 25] in almost all cases. As the only exception, in MNIST [19] with MLP, the scheduling is just equivalent to the original dropout framework [13, 25]. Our guess is that the simpler the learning task, the less effective Curriculum Learning. After all, for a task which is relatively easy itself, there is less need for "starting easy". This is in any case done at no additional cost nor training time requirements.

As expected, anti-Curriculum was improved by a more significant gap by our scheduling. Also, sometimes, an anti-Curriculum strategy even performs worse than a non-regularized network (*e.g.*, Caltech 256 [10]). This is coherent with the findings of [2] and with our discussion in §4 concerning Annealed Dropout [22], of which anti-Curriculum represents a generalization. In addition, while neither regular nor Curriculum Dropout ever need early stopping, anti-Curriculum often does.

6. Conclusions and Future Work

In this paper we have propose a scheduling for dropout training applied to deep neural networks. By softly increasing the amount of units to be suppressed layerwise, we achieve an adaptive regularization and provide a better smooth initialization for weight optimization. This allows us to implement a mathematically sound curriculum [2] and justifies the proposed generalization of [13, 25].

Through a broad experimental evaluation on 7 image classification tasks, the proposed Curriculum Dropout have proved to be more effective than both the original Dropout [13, 25] and the Annealed [22], the latter being an example of anti-Curriculum [2] and therefore achieving an inferior performance to our more disciplined approach in ease dropout training. Globally, we always outperform the original Dropout [13, 25] using various architectures, and we improve the idea of [22] by margin.

We have tested Curriculum Dropout on image classification tasks only. However, our guess is that, as standard Dropout, our method is very general and thus applicable to different domains. As a future work, we will apply our scheduling to other computer vision tasks, also extending it for the case of inter-neural connection inhibitions [29] and Recurrent Neural Networks.

Acknowledgment

We gratefully acknowledge the support of NVIDIA Corporation with the donation of one Tesla K40 GPU used for

part of this research.

References

[1] J. Bayer, C. Osendorfer, and N. Chen. On fast dropout and its applicability to recurrent networks. In *CoRR:1311.0701*, 2013. 2, 3

[2] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML*, 2009. 2, 5, 6, 8

[3] C. M. Bishop. Training with noise is equivalent to Tikhonov regularization. *Neural Comput.*, 7(1):108–116, 1995. 4

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, 2011. 5

[5] G. Caglar, S. Jose, M. Marcin, and Y. Bengio. A robust adaptive stochastic gradient method for deep learning. In *CoRR:1703.00788*, 2017. 5

[6] J. Cavazza and V. Murino. Active Regression with Adaptive Huber loss. In *CoRR:1606.01568*, 2016. 5

[7] K. Crammer, A. Kulesza, and M. Dredze. Adaptive regularization of weight vectors. In *NIPS*. 2009. 5

[8] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13(1), 2000. 2, 4

[9] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPR workshop*, 2004. 2, 6, 7, 8

[10] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. In *Technical Report 7694, California Institute of Technology*, 2007. 2, 6, 7, 8

[11] B. D. Haeffele and R. Vidal. Global optimality in tensor factorization, deep learning, and beyond. In *CoRR:1506.07540*, 2015. 4

[12] L. Hansen and C. Rasmussen. Pruning from adaptive regularization. *Neural Computation*, 6(6):1222–1231, 1994. 5

[13] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. In *CoRR:1207.0580*, 2012. 1, 2, 3, 4, 5, 6, 7, 8

[14] B. Jimmy and B. Frey. Adaptive dropout for training deep neural networks. In *NIPS*, 2016. 2, 3

[15] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *CoRR:1412.6980*, 2014. 6

[16] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009. 2, 6, 7, 8

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*. 2012. 1

[18] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Backpropagation applied to handwritten zip code recognition. In *Neural Computation*, pages 541–551, 1989. 6

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE, 86(11):22782324*, 2009. 2, 6, 7, 8

[20] Z. G. Li and T. Boqing Yang. Improved dropout for shallow and deep learning. In *NIPS*, 2016. 2, 3

[21] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop*, 2011. 2, 6, 7, 8

[22] S. J. Rennie, V. Goel, and S. Thomas. Annealed dropout training of deep networks. In *Proceedings onf the IEEE Workshop on SLT*, pages 159–164, 2014. 2, 3, 6, 8

[23] S. Rifai, X. Glorot, B. Yoshua, and P. Vincent. Adding noise to the input of a model trained with a regularized objective. In *CoRR:1104.3250*, 2011. 4

[24] M. Soltanolkotabi, E. Elhamifar, and E. J. Candès. Robust subspace clustering. In *CoRR:1301.2603*, 2013. 5

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014. 1, 2, 3, 4, 5, 6, 7, 8

[26] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised Learning of Video Representations using LSTMs. In *CoRR:1502.04681*, 2015. 2, 6, 7

[27] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *CVPR*, 2011. 8

[28] S. Wager, S. Wang, and P. S. Liang. Dropout training as adaptive regularization. In *NIPS*. 2013. 2, 3, 4

[29] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *ICML*, 2013. 2, 4, 8

[30] S. Wang and C. D. Manning. Fast dropout training. In *ICML*, pages 118–126, 2013. 2, 3

[31] H. Wu and X. Gu. Towards dropout training for convolutional neural networks. *Neural Networks*, 71:1–10. 2, 3

[32] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 3

[33] S. Zhai and Z. M. Zhang. Dropout training of matrix factorization and autoencoders for link prediction in sparse graphs. In *CoRR:1512.04483*, 2015. 2, 3, 4

[34] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NIPS*. 2014. 3