Noname manuscript No.
(will be inserted by the editor)

Handling Dropout Probability Estimation in Convolution Neural Networks Using Metaheuristics

Gustavo H. de Rosa, João P. Papa
Xin-S. Yang

Received: date / Accepted: date

Abstract Deep learning-based approaches have been paramount in recent years, mainly due to their outstanding results in several application domains, ranging from face and object recognition to handwritten digit identification. Convolutional Neural Networks (CNN) have attracted a considerable attention since they model the intrinsic and complex brain working mechanisms. However, one main shortcoming of such models concerns their overfitting problem, which prevents the network from predicting unseen data effectively. In this paper, we address this problem by means of properly selecting a regularization parameter known as Dropout in the context of CNNs using meta-heuristic-driven techniques. As far as we know, this is the first attempt to tackle this issue using this methodology. Additionally, we also take into account a default dropout parameter and a dropout-less CNN for comparison purposes. The results revealed that optimizing Dropout-based CNNs is worthwhile, mainly due to the easiness in finding suitable dropout probability values, without needing to set new parameters empirically.

Keywords Convolutional Neural Networks · Dropout · Meta-Heuristic Optimization

1 Introduction

One of the main computer vision problems concerns how to produce a good representation of the real world, thus allowing machine learning systems to understand

Gustavo H. de Rosa and João P. Papa
São Paulo State University
Bauru, SP 17033-360
Department of Computing, Brazil
E-mail: {gustavo.rosa, papa}@fc.unesp.br

Xin-S. Yang
Middlesex University
London, NW4 4BT
School of Science and Technology, U.K.
E-mail: x.yang@mdx.ac.uk

these descriptions for further detecting and classifying objects [2]. However, the problem still persists when faced with situations where there exist variations of luminosity in the environment, as well as different perspectives in the image acquisition process and problems related to rotation, translation and scale. Therefore, the great question to be answered by researchers in this area concerns how human beings and animals basically learn by looking around [16].

Standard machine learning approaches devise to settle the aforementioned situation by extracting feature vectors, feeding a classifier based on a training set, and thereafter classifying the remaining images. Thus, even though the feature learning problem has received a substantial attention in the last decades, a spotlight has been lit concerning the study of deep learning techniques [14, 1, 25, 6, 11]. As these methods are based on the hierarchical feature learning, we can establish an analogy with the human visual processing, i.e. with the information being compressed throughout the vision (eyes) and learning (visual cortex) procedure.

Notwithstanding that there exist several deep learning techniques in the literature, one of the most broadly used approaches concerns the Convolutional Neural Networks (CNN) [14]. These neural networks are formed by different stages and architectures, which are responsible for learning different information at each level (e.g., images and signals). The stages comprehend, basically, filtering operations known as convolution using different masks (kernels), followed by non-linear operations known as pooling for posterior image sub-sampling. Each set of these three operations is then executed once again in a new layer, but now with the sub-sampled image generated as an output of the previous layer, being used as an input to this new layer. Each of these layers creates a new feature vector, which is concatenated at the last layer to compose a high dimensional feature vector to represent the input image.

Convolutional Neural Networks have been widely applied in several applications, such as handwritten digit [24] and object recognition [15], as well as natural language processing [3]. One can observe that most applications are image processing- and computational vision-based, since some CNNs are naturally invariant to translation, rotation and scaling. However, training deep neural networks with a huge number of parameters can lead to some difficult issues such as overfitting and parameter tuning. The main idea behind the first issue is that during training, units can adapt to the weights drawn from the limited training data, being not that effective when collated with the testing data. Basically, in other words, when we deal with complex information, the units will accommodate themselves during the training step, and "memorize" the data instead of learning, thus resulting in a model that poorly predicts any new or unseen data.

Several attempts have been made to solve the overfitting problem, such as stopping the training as soon as its performance on the validation set starts to drop, or even introducing some types of regularization methods, such as soft-weight sharing [17]. On the other hand, a better alternative to address a regularization method would be averaging the predictions of all possible parameter configurations, weighting all the possibilities and checking which one would perform better. Nevertheless, this would require a huge computational effort, thus being only practical for small or simple models [30].

Recently, Srivastava et al. [26] proposed a technique to overcome these issues, acknowledged as Dropout. The pivotal concept about the term "dropout" refers to dropping out units from neural network layers; in other words, by provisionally

removing them out of the network, along with all their outgoing and incoming connections. Similarly, Wan et al. [27] proposed the *DropConnect*, where the authors dropped out connections instead of neurons, which means one can retain a neuron "partially", since original Dropout removes all connections at once. Finally, Iosifidis et al. [9] presented *DropELM*, in which Dropout and DropConnect are combined in the context of Extreme Learning Machines.

Nevertheless, one may find just a few recent works that face the problem of overfitting in CNNs using Dropout regularization. For instance, Wu et al. [29] introduced a max-pooling dropout method, while Dahl et al. [5] combined the use of ReLU (rectified linear unit) units with Dropout. Although all these techniques have obtained state-of-the-art results in a number of applications, their main drawback concerns fine-tuning the probability of dropping out a neuron, which is usually accomplished by hand.

Considering the drawback of fine-tuning parameters, some recent works attempted to tackle this issue by means of meta-heuristic techniques. For example, Papa et. al [18,19,21] employed several meta-heuristics to calibrate Bernoulli Restricted Boltzmann Machines, Discriminative Restricted Boltzmann Machines and Deep Belief Networks parameters, respectively. Moreover, Rosa et. al have used meta-heuristics for optimizing CNNs [23] and Deep Belief Networks parameters [22].

In this paper, the problem of selecting a proper dropout probability in CNNs is modeled as a meta-heuristic-driven optimization task, in which agents encode the values of the probabilities in a search problem guided by the loss function over the validation set. As far as we are concerned, this is the first work that attempted to address the problem of proper selecting the dropout ratio in such a way. In order to validate the proposed approach, we employed Particle Swarm Optimization (PSO) [12], since it is a well-known and consistent meta-heuristic optimization technique, as well as Bat Algorithm (BA) [32], Cuckoo Search (CS) [33] and Firefly Algorithm (FA) [31], which are bio-inspired techniques that turned out to be a recent hotbed due to their good effectiveness in several applications.

Therefore, the main contributions of this paper are twofold: (i) to introduce meta-heuristic techniques in the context of Dropout regularization in CNNs, and (ii) to fill the lack of research regarding dropping out units in CNNs. The remainder of this paper is organized as follows. Sections 2 and 3 present some theoretical background with respect to CNNs and Dropout techniques, respectively. Section 4 discusses the methodology employed in this work, and Section 5 presents the experiments. Finally, Section 6 states conclusions and future works.

2 Convolutional Neural Networks

Convolutional Neural Networks can be seen as a representation of a bigger class of models based on the Hubel's and Wiesel's architecture, which was presented in a seminal study in 1962 concerning the primary cortex of cats. This research has identified, basically, two kinds of cells: (i) *simple cells*, which possess an analogous duty to the filter bank step, and (ii) the *complex cells*, which perform a similar job to the CNN sampling step.

The first model that simulated a computer-based convolutional neural network was the well-known "Neocognitron" [7], which implemented an unsupervised

training algorithm during the filter bank step, followed by a supervised training algorithm applied in the last layer. Later on, LeCun et al. [13] simplified this architecture by proposing the use of the Backpropagation algorithm to train the network in a supervised way. Thus, several applications that used CNN emerged in the subsequent decades.

Basically, a CNN can be understood as an N-layered data processing sequence. Thereby, given an input image[1], a CNN essentially extracts a high level representation of it, called *multispectral image*, whose pixel attributes are concatenated in a feature vector for later application of pattern recognition techniques. Figure 1 introduces the naïve architecture of a Convolutional Neural Network.

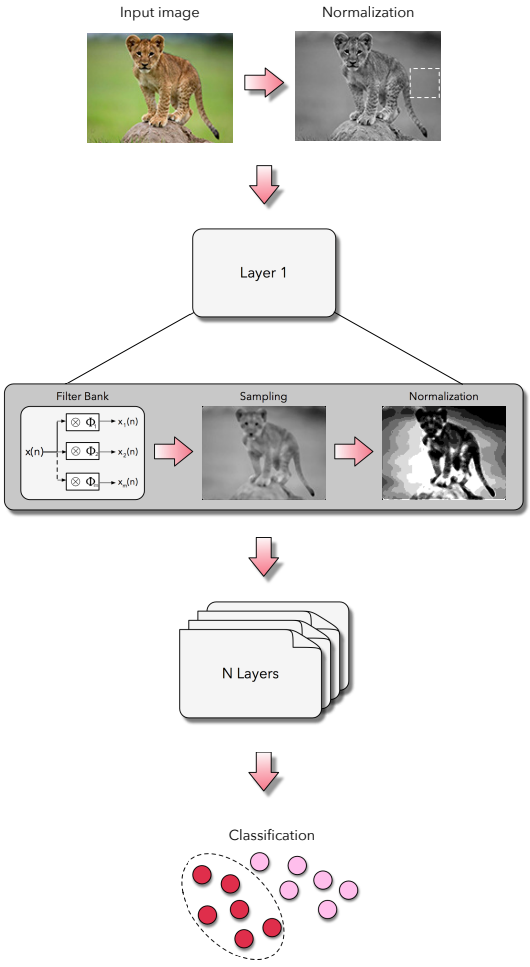


Fig. 1 A typical Convolutional Neural Network architecture.

[1] The same procedure can be extended to signal processing-based applications.

As aforementioned, each CNN layer is often composed of three operations, being the first one a convolution with a filter bank, followed by a sampling phase and then by a normalization step. As one can observe in Figure 1, there is still a possibility of a normalization operation in the beginning of the whole process. The next sections describe in more details each of these steps.

2.1 Filter Bank

Let $\hat{I} = (D_I, \mathbf{I})$ be a multispectral image such that $D_I \in n \times n$ is the image domain, and $\mathbf{I} = \{I_1(p), I_2(p), \ldots, I_m(p)\}$ corresponds to a pixel $p = (x_p, y_p) \in D_I$, and m stands for the number of bands. When $\hat{I}$ is a grey-scale image, for instance, we have that $m = 1$ and $\hat{I} = (D_I, I)$.

Let $\phi = (\mathcal{A}, W)$ be a filter with weights $W(q)$ associated with every *pixel* $q \in \mathcal{A}(p)$, where $\mathcal{A}(p)$ denotes a mask of size $L_\mathcal{A} \times L_\mathcal{A}$, centered at p, and $q \in \mathcal{A}(p)$ if, and only if, $\max\{|x_q - x_p|, |y_q - y_p|\} \leq (L_\mathcal{A} - 1)/2$. In case of multispectral filters, their weights can be depicted as vectors $\mathbf{W}_i(q) = \{w_{i,1}(q), w_{i,2}(q), \ldots, w_{i,m}(q)\}$ for each filter i, and a multispectral filter bank can be then defined as $\phi = \{\phi_1, \phi_2, \ldots, \phi_n\}$, where $\phi_i = (\mathcal{A}, \mathbf{W}_i)$, $i = \{1, 2, \ldots, n\}$.

Thus, the convolution between an input image $\hat{I}$ and a filter ϕ_i generates the band i of the filtered image $\hat{J} = (D_J, \mathbf{J})$, where $D_J \in D_I$ and $\mathbf{J} = \{J_1(p), J_2(p), \ldots, J_n(p)\}$, $\forall p \in D_J$:

$$J_i(p) = \sum_{\forall q \in \mathcal{A}(p)} \mathbf{I}(q) \otimes \mathbf{W}_i(q), \tag{1}$$

where $\otimes$ denotes the convolution operator. The weights of ϕ_i are usually generated from an uniform distribution, i.e., $U(0, 1)$, and afterwards normalized with mean zero and unitary norm.

2.2 Sampling

This operation is extremely important for a CNN, which provides translational invariance to the extracted features. Let $\mathcal{B}(p)$ be the sampling area of size $L_\mathcal{B} \times L_\mathcal{B}$ centered at p. Additionally, let $D_K = D_J/s$ be a regular sampling operation every s pixels. Therefore, the resulting sampling operation in the image $\hat{K} = (D_K, \mathbf{K})$ is defined as follows:

$$K_i(p) = \sqrt[\alpha]{\sum_{\forall q \in \mathcal{B}(p)} J_i(q)^\alpha}, \tag{2}$$

where $p \in D_K$ denotes every pixel of the new image, $i = \{1, 2, \ldots, n^2\}$, and α stands for the stride parameter, controlling the downsampling factor of the operation.

2.3 Normalization

The last operation of a CNN is its normalization, which is a widely employed mechanism in order to enhance its perfomance [4]. This operation is based on the

apparatus found on cortical neurons [8], being also defined under a squared-area $\mathcal{C}(p)$ of size $L_{\mathcal{C}} \times L_{\mathcal{C}}$ centered at pixel p, such as:

$$O_i(p) = \frac{K_i(p)}{\sum_{j=1}^{n} \sum_{\forall q \in \mathcal{C}(p)} K_j(q) K_i(q)}. \tag{3}$$

Thus, the above operation is accomplished for each pixel $p \in D_O \subset D_k$ of the resulting image $\hat{O} = (D_O, \mathbf{O})$.

3 Dropout Regularization

Considering the aforementioned CNN model, a Dropout-based CNN can be formulated as a new CNN layer. In this new formulation, $\mathbf{r}$ stands for the activation or dropout of the M neurons in a specific layer, where each variable r_j contains the value 1 (one) with probability $1 - p$, independent of other variables r_i, $i \neq j$. If r_j equals to 1 (one), the unit h_j is withheld, otherwise it is dropped from the network together with its connections.

Notice that probability p is independent of other units and $\mathbf{r}$ is sampled directly from a Bernoulli distribution [26], being resampled for every mini-batch during the learning process. Equation (4) describes this distribution:

$$r_j \sim Bernoulli(p), \forall j = \{1, 2, \ldots, M\}. \tag{4}$$

Therefore, during the training phase, let $y^{(L)}$ denote the vector of outputs at a layer L. The new vector of outputs $\tilde{y}^{(L)}$ considering dropout is formulated by Equation (5):

$$\tilde{y}^{(L)} = r y^{(L)}. \tag{5}$$

Finally, in the testing step, the weight matrix W needs to be scaled with ratio p in order to average all the 2^M possible dropped-out networks. This is the greatest contribution of the dropout regularization, as it only needs to test a single network. Equation (6) is responsible to illustrate this process.

$$W_{test}^{(L)} = p \mathbf{W}^{(L)}, \tag{6}$$

where $\mathbf{W}^{(L)}$ stands for the weight matrix at layer L.

4 Methodology

In this section, we present the methodology used to proper select the dropout ratio using Convolutional Neural Networks, as well as the employed datasets.

4.1 Modeling Dropout-based CNN Parameter Optimization

We propose to model the problem of selecting suitable dropout parameters considering CNN in the task of image classification. The learning step of a CNN has four parameters: the learning rate η, penalty parameter (momentum) α, weight decay λ and the dropout ratio p. As we are interested in fine-tuning the dropout parameter only, we fixed the 3-tuple (η, α, λ) and we played with p in order to minimize the loss function of the classified images over a validation set. After that, the selected parameter is thus applied to classify the unseen images of the test set.

4.2 Datasets

In regard to the parameter optimization experiment, we employed four datasets, as described below:

- MNIST dataset[2]: it is composed of images of handwritten digits. The original version contains a training set with $60,000$ images from digits '0'-'9', as well as a test set with $10,000$ images.
- Semeion Handwritten Digits dataset[3]: set of $1,593$ images from handwritten digits '0' - '9' written in two ways: the first time in a normal way (accurately) and the second time in a fast way (no accuracy). In the end, they were stretched with resolution of 16×16 in a grayscale of 256 values, and then each pixel was binarized.
- USPS dataset[4]: numeric data obtained from the scanning of handwritten digits from envelopes by the U.S. Postal Service. It is composed of $7,291$ training images and a test set with $2,007$ images. All the original digits were deslanted and size-normalized, resulting in 16×16 grayscale images.
- CIFAR-10 dataset[5]: is a subset image database from the "80 million tiny images" dataset, collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Composed of $60,000$ 32x32 colour images in 10 classes, with $6,000$ images per class. It is also divided into five training batches and one test batch, each one containing $10,000$ images. Therefore we have $50,000$ images for training purposes, and $10,000$ for testing duties.

Figure 2 displays some training examples from the above datasets.

4.3 CNN Architectures

In regard to the source-code, we used our own optimization library LibOPT [20][6] and the well-known Caffe library[7] [10], which is developed under GPGPU (General-Purpose computing on Graphics Processor Units) platform, thus providing more efficient implementations concerning CNNs. Also, in order to integrate LibOPT with

[2] http://yann.lecun.com/exdb/mnist/

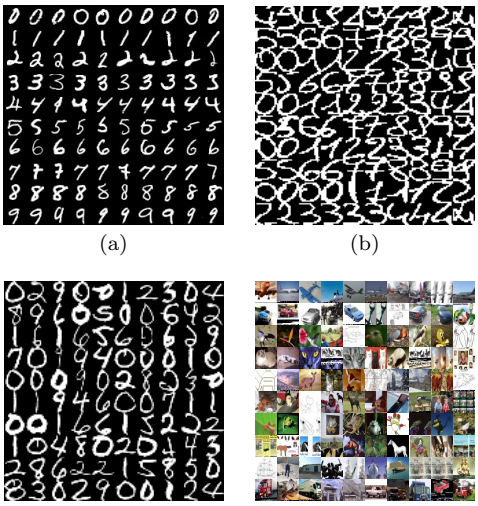
[3] `https://archive.ics.uci.edu/ml/datasets/Semeion+Handwritten+Digit`

[4] http://statweb.stanford.edu/ tibs/ElemStatLearn/datasets/zip.info.txt

[5] http://www.cs.toronto.edu/ kriz/cifar.html

[6] https://github.com/jppbsi/LibOPT

[7] http://caffe.berkeleyvision.org



(a) (b) (c) (d)

Fig. 2 Some training examples from: (a) MNIST, (b) Semeion, (c) USPS and (d) CIFAR-10 datasets.

Caffe, we developed a new library called LibOPT4Caffe[8]. Additionally, we considered two different CNN architectures to provide a deeper experimental analysis: one for CIFAR-10 dataset and another for MNIST, Semeion and USPS datasets. These architectures are the original ones proposed by Caffe examples, just with an extra dropout layer. Also, regarding Semeion and USPS datasets, we have used a kernel size of 3 instead of 5 for convolution layers due to the lower resolution of these datasets. Figure 3 illustrates the architectures used in this work. Note that "conv" stands for the convolution layer, "pool" for pooling, "relu" for rectified linear unit, "drop" for dropout and "ip" for inner product, which stands for a multiplication in the vector space.

4.4 Experimental Setup

In this work, we compared four distinct meta-heuristic optimization techniques against the default dropout ratio given by Caffe, and the standard Caffe architecture without dropout. The main idea to compare meta-heuristics against the default dropout ratio is to check wether it is better to perform or not an optimization prior to the experiments. In order to provide a statistical analysis by means of Wilcoxon signed-rank test [28], we conducted a cross-validation with 20 runs. As we working with a 1-dimensional search space, there is no need to employ a vastly number of agents nor iterations. Thus, we employed 7 agents over 10 iterations for convergence considering all techniques. Table 1 presents the native parameter configuration for each optimization technique. Notice these values have been empirically set.

[8] https://github.com/gugarosa/LibOPT4Caffe

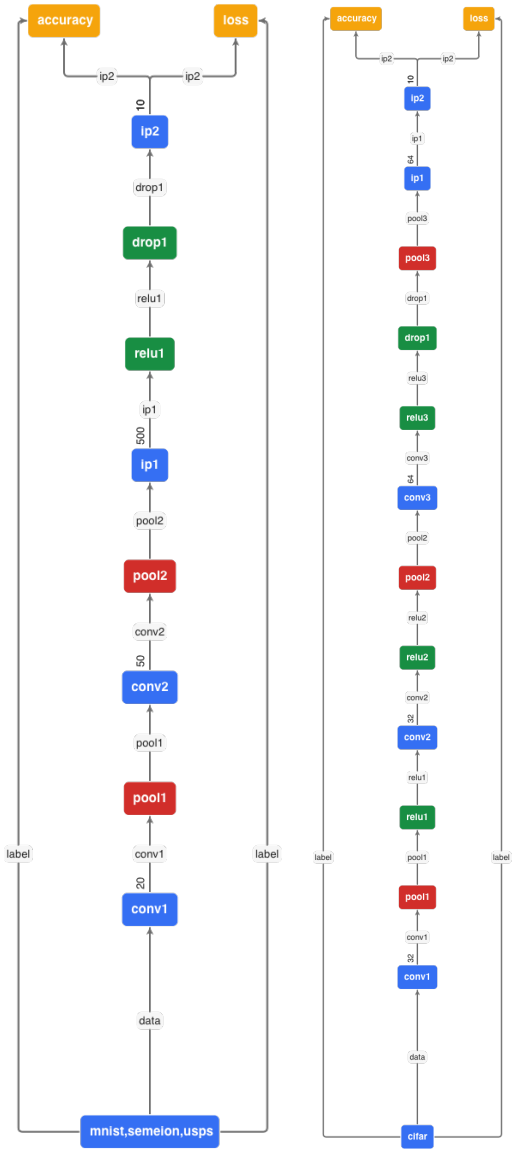


Fig. 3 CNN architectures for: (a) MNIST, Semeion and USPS, and (b) CIFAR-10 datasets. The picture was generated with the following tool: http://yanglei.me/gen_proto/.

All datasets were split into training, validation and testing sets. Table 2 describes the amount of images, as well as the number of the batch size (parenthesis) employed for each set.

Finally, we have set each CNN parameter according to Table 3. Note that all these parameters are the default ones provided by Caffe library. The only exception

Technique	Parameters
BA	$f_{min} = 0$, $f_{max} = 2$, $A = 0.5$, $rand = 0.5$
CS	$\beta = 1.5$, $p = 0.25$, $\alpha = 0.8$
FA	$\gamma = 1.0$, $\beta_0 = 1.0$, $\alpha = 0.2$
PSO	$c_1 = 1.7$, $c_2 = 1.7$, $w = 0.7$

Table 1 Meta-heuristic parameter configuration.

Dataset	# Training set	# Validation set	# Testing set
CIFAR-10	20000 (100)	30000 (100)	10000 (100)
MNIST	20000 (64)	40000 (100)	10000 (100)
Semeion	200 (2)	400 (400)	993 (993)
USPS	2406 (32)	4885 (977)	2007 (2007)

Table 2 Dataset configuration.

holds for Semeion dataset due to its small amount of images, being trained in a slower pace with a learning rate ten times smaller, i.e., 0.001,

Dataset	η	α	λ	p	# Iterations
CIFAR-10	0.001	0.9	0.004	[0, 1]	4000
MNIST	0.01	0.9	0.0005	[0, 1]	10000
Semeion	0.001	0.9	0.0005	[0, 1]	10000
USPS	0.01	0.9	0.0005	[0, 1]	10000

Table 3 CNN parameter configuration.

5 Experimental Results

This section aims at presenting the experimental results concerning CNN dropout parameter fine-tuning. We compared four optimization methods against the default dropout-less and the default dropout ratio provided by Caffe. The most accurate results, according to Wilcoxon signed-rank test, are in bold.

Tables 7, 4, 5 and 6 present the average accuracies and average hyper-parameters found over MNIST, Semeion, USPS and CIFAR-10 datasets, respectively. As one can observe, the accuracy rates between different datasets are inconsistent, mainly due to their distinct natures, i.e., amount of images and features, odd acquisition processes, among others.

Regarding MNIST dataset, it is possible to state that a dropout network was once again better than the standard one. All the meta-heuristic techniques, except BA, were able to find the best results among the standard dropout ratio provided by Caffe, even using a smaller probability for dropping off the units. If we take a closer look at Figure 4, it is possible to observe that after 4000 epochs, the no-dropout network stalls at around 0.01% of error, while the dropout networks still introduce some kind of noise, preventing the network from overfitting and improving its classification rate. Note that "Best Dropout" concerns the p value found in Table 8.

		Hyper-parameters			
	Accuracy (%)	η	α	λ	p
Caffe	99.07%	0.01	0.9	0.0005	0
Dropout Caffe	**99.18%**	0.01	0.9	0.0005	0.5
BA	99.13%	0.01	0.9	0.0005	0.4988
CS	**99.14%**	0.01	0.9	0.0005	0.4883
FA	**99.16%**	0.01	0.9	0.0005	0.4630
PSO	**99.17%**	0.01	0.9	0.0005	0.4559

Table 4 Average accuracies and average hyper-parameters over the test set considering MNIST dataset.

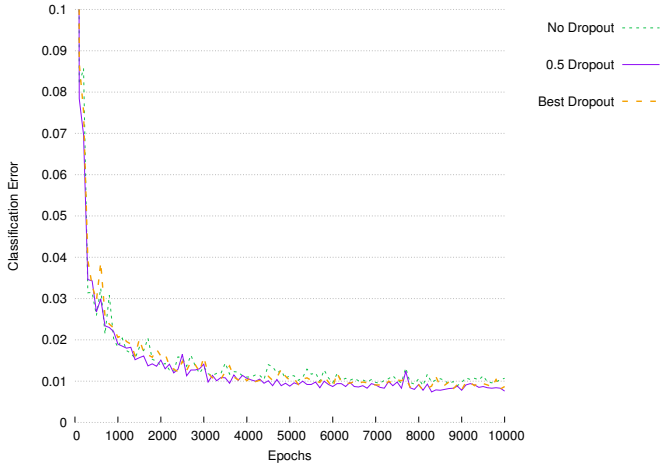


Fig. 4 Classification error over MNIST testing set.

		Hyper-parameters			
	Accuracy (%)	η	α	λ	p
Caffe	97.62%	0.001	0.9	0.0005	0
Dropout Caffe	**98.14%**	0.001	0.9	0.0005	0.5
BA	**98.30%**	0.001	0.9	0.0005	0.7532
CS	**98.19%**	0.001	0.9	0.0005	0.7546
FA	**98.23%**	0.001	0.9	0.0005	0.7786
PSO	**97.66%**	0.001	0.9	0.0005	0.8263

Table 5 Average accuracies and average hyper-parameters over the test set considering Semeion dataset.

Considering Semeion dataset, once again dropout was capable of obtaining the best results. Now, all meta-heuristic techniques were also able to obtain the best results, together with our baseline, i.e., Dropout by Caffe being BA the one which achieved the highest accuracy over the test set. However, the statistical evaluation pointed out all dropout-based approaches can be considered similar to each other, although the lowest results were obtained by the baseline provided by Caffe. An interesting point concerns the values found for the probability parameter p by the meta-heuristic techniques, which were similar to each other, though being different from the one obtained by the baseline.

Glancing Figure 5, it is possible to observe the best dropout network was the worst one at the beginning of the epochs. After around 5000 epochs, it starts improving its performance and overcomes the other networks. This is mainly due

to the limited amount of images Semeion dataset has, which creates a significant overfitting and stalls the training after a certain number of epochs.

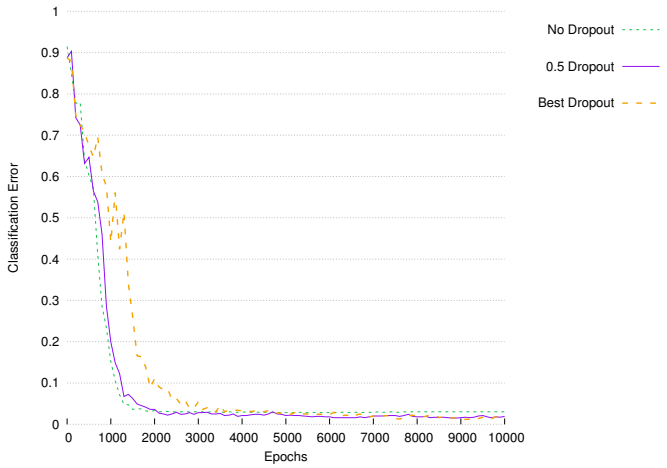


Fig. 5 Classification error over Semeion testing set.

		Hyper-parameters			
	Accuracy (%)	η	α	λ	p
Caffe	95.80%	0.01	0.9	0.0005	0
Dropout Caffe	96.21%	0.01	0.9	0.0005	0.5
BA	**96.43%**	0.01	0.9	0.0005	0.7688
CS	96.28%	0.01	0.9	0.0005	0.6870
FA	**96.38%**	0.01	0.9	0.0005	0.7319
PSO	**96.37%**	0.01	0.9	0.0005	0.8031

Table 6 Average accuracies and average hyper-parameters over the test set considering USPS dataset.

BA, FA and PSO were able to obtain the best results considering USPS dataset. One can observe that their accuracy rates were slightly better than the standard dropout-less and dropout ratio provided by Caffe. As the USPS dataset poses a greater challenge than Semeion one, but still lacks a huge amount of images for deep learning, we can observe in Figure 6 the search for the optimum dropout ratio performed nicely since the initial epochs. The best dropout network was able to provide a lower classification error almost all the times.

As one can observe, only the default dropout ratio ($p = 0.5$) played a big role on CIFAR-10 dataset, while all the meta-heuristic techniques failed in finding the best parameter for this dataset. However, it is still important to highlight that p was found over the validation set, which is less likely to burden the classification rates. Nevertheless, as already expected, dropout was capable of slightly improving the recognition rate of this architecture. Figure 7 illustrates the classification error over the testing set during all training epochs.

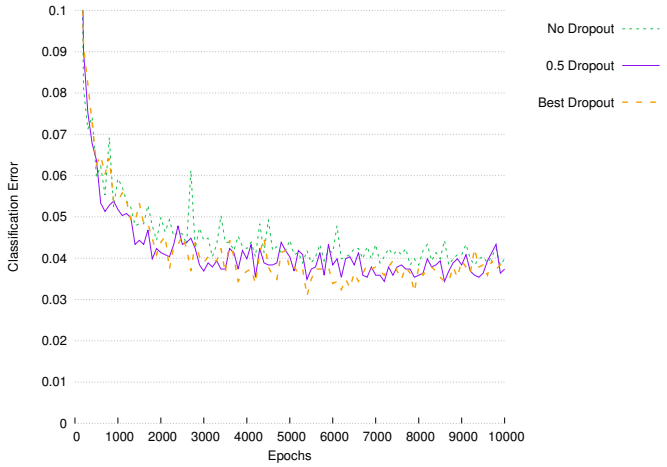


Fig. 6 Classification error over USPS testing set.

		Hyper-parameters			
	Accuracy (%)	η	α	λ	p
Caffe	71.47%	0.001	0.9	0.004	0
Dropout Caffe	**72.08%**	0.001	0.9	0.004	0.5
BA	71.43%	0.001	0.9	0.004	0.6430
CS	71.16%	0.001	0.9	0.004	0.6270
FA	71.52%	0.001	0.9	0.004	0.6629
PSO	71.55%	0.001	0.9	0.004	0.6655

Table 7 Average accuracies and average hyper-parameters over the test set considering CIFAR-10 dataset.

		Best Hyper-parameters				
	Accuracy (%)	η	α	λ	p	Technique
CIFAR-10	71.69%	0.001	0.9	0.004	0.6177	BA
MNIST	99.12%	0.01	0.9	0.0005	0.5100	PSO
Semeion	98.51%	0.001	0.9	0.0005	0.8037	BA
USPS	96.43%	0.01	0.9	0.0005	0.8284	FA

Table 8 Average accuracies and best hyper-parameters over the test set considering all datasets.

Comprehending our initial experiments, each dataset had a best accuracy rate value for a particular dropout probability p. As one can observe in Table 8, BA, PSO, BA and FA were the meta-heuristic techniques which found the best p over CIFAR-10, MNIST, Semeion and USPS datasets, respectively. Therefore, we performed another round of experiments using these values. Additionally, Table 8 presents, for each dataset, the average accuracies using the best hyper-parameter p. It is important to highlight that, for Semeion dataset, BA was capable of finding a more suitable dropout probability, leading to a significantly higher classification rate than the one observed in Table 5.

Nevertheless, the main drawback concerning meta-heuristic techniques consists in its longer time to attain the final output. As we are evaluating a distinct CNN every time we need to update our particle's fitness, this procedure will take a longer time to converge and find a suitable dropout parameter. BA, CS and PSO are composed by m initial evaluations + (m agents $\times$ t iterations) + final evaluation

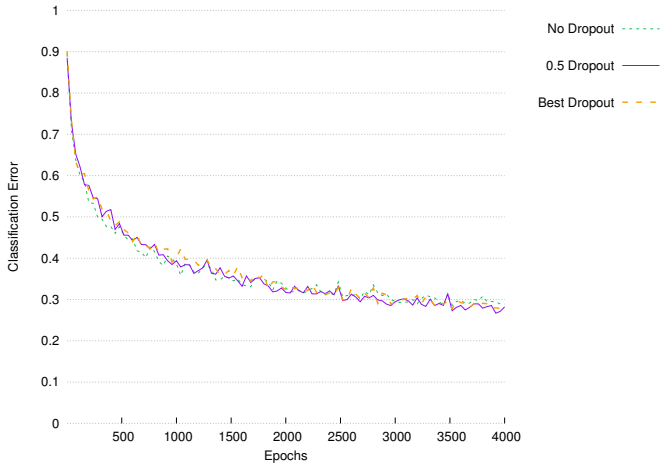


Fig. 7 Classification error over CIFAR-10 testing set.

with best parameter found, while FA does not have the m initial evaluations. As we are working with $m = 7$ and $t = 10$, Table 9 describes the number of calls to the CNN learning procedure to give us an idea about the computational burden of each technique.

Technique	# calls
Caffe	1
Dropout Caffe	1
BA	78
CS	78
FA	71
PSO	78

Table 9 Number of CNN evaluations for each technique.

6 Conclusions

In this paper, we dealt with the problem of proper selecting dropout parameters concerning CNNs by means of Particle Swarm Optimization, Bat Algorithm, Cuckoo Search and Firefly Algorithm. The experiments were carried out over four public datasets in the context of image classification.

The experimental section comprised different CNN architectures, as well as images with different resolutions and distinct training set sizes. The results obtained by the meta-heuristic-based dropout CNNs were compared against a standard dropout ratio and a dropout-less network, and showed to be very promising, since meta-heuristic CNNs were able to obtain the suitable dropout parameters in almost all datasets. On the other hand, such task requires a higher computational load than no-optimized CNNs, as each particle's fitness update needs to be evaluated under a CNN architecture, thus, taking a longer time to find a suitable

dropout parameter and achieve its output. Since the meta-heuristic-based techniques use a fitness function based on the CNN's outputs, each iteration of the optimization process requires a full training of the CNN, which turns out to be a costly process. It is usually expected that more iterations and agents would provide better results, but such process comes at the price of a higher computational burden.

In regard to future works, we intend to investigate the proposed approach with other meta-heuristic techniques, as well as to select feasible parameters using meta-heuristics for Dropconnect regularization in the context of CNNs.

Compliance with Ethical Standards

This study was funded by FAPESP (grants number #2015/25739-4, #2014/12236-1 and #2014/16250-9) and by CNPq (grant number #306166/2014-3). Additionally, Gustavo Rosa declares that he has no conflict of interest, João Papa declares that he has no conflict of interest, and Xin-She Yang declares that he has no conflict of interest. Notice this article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Arel, I., Rose, D., Karnowski, T.: Deep machine learning - a new frontier in artificial intelligence research [research frontier]. Computational Intelligence Magazine, IEEE **5**(4), 13–18 (2010)
2. Bishop, C.: Neural networks for pattern recognition. Oxford University Press (1995)
3. Collobert, R., Weston, J.: A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of the 25th International Conference on Machine Learning, pp. 160–167. ACM (2008)
4. Cox, D., Pinto, N.: Beyond simple features: A large-scale feature search approach to unconstrained face recognition. In: Proceedings of the IEEE International Conference on Automatic Face Gesture Recognition and Workshops, pp. 8–15 (2011)
5. Dahl, G.E., Sainath, T.N., Hinton, G.E.: Improving deep neural networks for lvcsr using rectified linear units and dropout. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 8609–8613. IEEE (2013)
6. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Learning hierarchical features for scene labeling. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**, 1915–1929 (2013)
7. Fukushima, K., Miyake, S.: Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. Pattern Recognition **15**(6), 455–469 (1982)
8. Geisler, W., Albrecht, D.: Cortical neurons: isolation of contrast gain control. Vision Research **32**(8), 1409–1410 (1992)
9. Iosifidis, A., Tefas, A., Pitas, I.: Dropelm. Neurocomputing **162**(C), 57–66 (2015)
10. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., T., T.D.: Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
11. Kavukcuoglu, K., Sermanet, P., lan Boureau, Y., Gregor, K., Mathieu, M., Cun, Y.L.: Learning convolutional feature hierarchies for visual recognition. In: Advances in Neural Information Processing Systems 23, pp. 1090–1098. Curran Associates, Inc. (2010)
12. Kennedy, J., Eberhart, R.: Swarm Intelligence. M. Kaufman (2001)
13. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Computation **1**(4), 541–551 (1989)
14. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)

15. LeCun, Y., Huang, F.J., Bottou, L.: Learning methods for generic object recognition with invariance to pose and lighting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 97–104 (2004)
16. LeCun, Y., Kavukcuoglu, K., Farabet, C.: Convolutional networks and applications in vision. In: Proceedings of 2010 IEEE International Symposium on Circuits and Systems, pp. 253–256 (2010)
17. Nowlan, S.J., Hinton, G.E.: Simplifying neural networks by soft weight-sharing. Neural Computation **4**(4), 473–493 (1992). DOI 10.1162/neco.1992.4.4.473
18. Papa, J.P., Rosa, G.H., Costa, K.A.P., Marana, A.N., Scheirer, W., Cox, D.D.: On the model selection of bernoulli restricted boltzmann machines through harmony search. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1449–1450. ACM (2015)
19. Papa, J.P., Rosa, G.H., Marana, A.N., Scheirer, W., Cox, D.D.: Model selection for discriminative restricted boltzmann machines through meta-heuristic techniques. Journal of Computational Science **9**, 14–18 (2015)
20. Papa, J.P., Rosa, G.H., Rodrigues, D., Yang, X.S.: Libopt: An open-source platform for fast prototyping soft optimization techniques. ArXiv e-prints (2017). URL `http://adsabs.harvard.edu/abs/2017arXiv170405174P`
21. Papa, J.P., Scheirer, W., Cox, D.D.: Fine-tuning deep belief networks using harmony search. Applied Soft Computing **46**, 875 – 885 (2016)
22. Rosa, G., Papa, J., Costa, K., Passos, L., Pereira, C., Yang, X.S.: Learning parameters in deep belief networks through firefly algorithm. In: Artificial Neural Networks in Pattern Recognition: 7th IAPR TC3 Workshop, Proceedings, pp. 138–149. Springer International Publishing (2016)
23. Rosa, G., Papa, J., Marana, A., Scheirer, W., Cox, D.: Fine-tuning convolutional neural networks using harmony search. In: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 20th Iberoamerican Congress, Proceedings, pp. 683–690. Springer International Publishing (2015)
24. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: Proceedings of the Seventh International Conference on Document Analysis and Recognition, pp. 958–963. IEEE Computer Society (2003)
25. Socher, R., Huang, E.H., Pennin, J., Manning, C.D., Ng, A.Y.: Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In: Advances in Neural Information Processing Systems 24, pp. 801–809. Curran Associates, Inc. (2011)
26. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research **15**(1), 1929–1958 (2014)
27. Wan, L., Zeiler, M., Zhang, S., LeCun, Y., Fergus, R.: Regularization of neural networks using dropconnect. In: International Conference on Machine learning, ICML '13 (2013)
28. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin **1**(6), 80–83 (1945)
29. Wu, H., Gu, X.: Towards dropout training for convolutional neural networks. Neural Networks **71**, 1–10 (2015)
30. Xiong, H.Y., Barash, Y., Frey, B.J.: Bayesian prediction of tissue-regulated splicing using rna sequence and cellular context. Bioinformatics **27**(18), 2554–2562 (2011). DOI 10.1093/bioinformatics/btr444
31. Yang, X.S.: Firefly algorithm, stochastic test functions and design optimisation. International Journal Bio-Inspired Computing **2**(2), 78–84 (2010)
32. Yang, X.S.: A new metaheuristic bat-inspired algorithm. In: Nature inspired cooperative strategies for optimization (NICSO 2010), pp. 65–74. Springer (2010)
33. Yang, X.S., Deb, S.: Engineering optimisation by cuckoo search. International Journal of Mathematical Modelling and Numerical Optimisation **1**(4), 330–343 (2010)