ADAPTIVE DATA AUGMENTATION FOR IMAGE CLASSIFICATION

Alhussein Fawzi**, Horst Samulowitz*†*, Deepak Turaga*†*, Pascal Frossard**

*EPFL, Switzerland & †IBM Watson Research Center, USA

ABSTRACT

Data augmentation is the process of generating samples by transforming training data, with the target of improving the accuracy and robustness of classifiers. In this paper, we propose a new automatic and adaptive algorithm for choosing the transformations of the samples used in data augmentation. Specifically, for each sample, our main idea is to seek a small transformation that yields *maximal* classification loss on the transformed sample. We employ a trust-region optimization strategy, which consists of solving a sequence of linear programs. Our data augmentation scheme is then integrated into a Stochastic Gradient Descent algorithm for training deep neural networks. We perform experiments on two datasets, and show that that the proposed scheme outperforms random data augmentation algorithms in terms of accuracy and robustness, while yielding comparable or superior results with respect to existing selective sampling approaches.

Index Terms— Data augmentation, transformation invariance, image robustness, trust-region optimization.

1. INTRODUCTION

In many classification problems, the available data is insufficient to train *accurate* and *robust* classifiers. To alleviate the relative scarcity of the data compared to the number of free parameters of a classifier, one popular approach is *data augmentation* (DA). Data augmentation consists in transforming the available samples into new samples using label-preserving transformations. For example, it is well known that sufficiently small affine transformations of the data preserve the label of an image. In [1], the importance of data augmentation is particularly outlined in order to train very large deep networks and improve the generalization error. Unfortunately, data augmentation is an art, as it involves many manual choices. Inappropriate choices of data augmentation schemes are likely to result in augmented samples that are not informative enough, which leads to no effect or detrimental effect on the accuracy and robustness of classifiers. The choice of the data augmentation strategy is therefore quite important to reach good accuracy and robustness properties, with a limited number of additional training samples.

In this paper, we focus on the problem of optimally choosing sample transformations from a transformation group for data augmentation. That is, we propose an automated and principled way for finding transformation parameters that lead to increased accuracy and robustness of classifiers. The key idea is to transform samples by small transformations that induce *maximal* loss to the current classifier. This *worst-case* data augmentation scheme leads to informative samples; i.e., when these are added to the training set, the classifier is significantly improved in terms of classification accuracy and robustness. We then propose a simple modification of the Stochastic Gradient Descent (SGD) algorithm to incorporate the proposed DA scheme in the training of deep neural networks classifiers. We show that, in cases where training data is insufficient, the proposed training algorithm yields significant improvements with respect to schemes using no augmentation, or random data augmentation.

Data augmentation has played an active role in achieving state-of-the-art results on many vision tasks. Very often, DA is performed by *randomly* generating transformations from a set of possible transformations (see e.g., [2, 3]). In [4], the authors propose a greedy strategy that selects the best transformation from a set of candidate transformations. While leading to impressive results, this strategy involves a significant number of classifier re-training steps, in addition to the necessity of hard-coding the parameters of candidate transformations, which can be computationally expensive when the number of candidate transformations is large. Data augmentation has also been shown to improve the classifiers' robustness to diverse sets of perturbations, such as additive adversarial noise [5, 6] or geometric transformations [7]. In [8], the authors propose a principled way for automatically computing elastic deformation fields for digits in a computationally efficient manner. The transformations considered in the paper are however specific to digits. The very recent work of [9] introduces an elegant way for data augmentation by randomly generating diffeomorphisms. The approach we follow in this paper is different, as we focus on *optimizing* the data generation process, while keeping the transformation group relatively small (e.g., affine transformations). Despite working with a small transformation group, the proposed approach gives results that are on par with the mentioned works, and leads in addition to significant improvements in terms of accuracy and robustness with respect to schemes that do not use data augmentation, or use random DA.

The paper is structured as follows. In Sec. 2, we delve into the main idea of the paper, namely the worst-case data augmentation process. We also derive intuitions that motivate this approach. In Sec. 3, we formalize the worst-case DA framework, and propose approximate algorithms for transformation computation, as well as an adapted training algorithm. Experimental results are provided in Sec. 4 showing the advantage of our adaptive data augmentation scheme, and we conclude in Sec. 5.

2. WHY WORST-CASE DATA AUGMENTATION?

To motivate the proposed procedure for data augmentation, we start with a simple example. Consider a simple classification task, where the goal is to classify between images representing a *vertical* line and a *horizontal* line. At training time, we only have access to two centered training samples represented in Fig. 1 (a,b). However, at test time, due to uncontrolled acquisition of images, the lines might not be perfectly centered. In particular, images can incur a horizontal or vertical translation. We define the linear classifier $f(x) = \sum_{i \in I_{\text{vert}}} x_i - \sum_{i \in I_{\text{hor}}} x_i$,, where I_{vert} and I_{hor} denote the indices of the pixels that are active in the vertical and horizontal training images (Fig. 1 (a,b)), respectively, and x is the column-reshaped image. Note that f is a perfectly accurate classifier on the training set, as $f(x) > 0$ for the vertical image (Fig 1 (a)), and $f(x) < 0$ for the horizontal image (Fig 1 (b)). It should be noted however that, without further augmentation of the data, this classifier is *not* robust to small translations of the data. Fig. 1 (c) shows the effect of vertical and horizontal shifts of the *vertical* line image (Fig. 1 (a)) on the value of f. Note that the value of f is robust to a large extent to vertical shifts, while being extremely unstable to horizontal shifts. Therefore, by adding horizontal shifts of Fig. 1 (a) to the training set, the classifier will get more robust to these transformations. On the other hand, adding vertical shifts to the training set will essentially have no impact on the decision. Conversely, adding vertical shifts of the horizontal line (Fig. 1 (b)) to the training set is certainly beneficial for boosting the robustness, but adding horizontal shifts will not change the decision function. This example highlights the importance of designing *adaptive* data augmentation strategies that are specific to the *dataset* and *classifier*. Moreover, it suggests a simple adaptive strategy for adding examples in the training set by adding transformations that *maximize the current classifier loss* (e.g., horizontal translations of vertical line images). In other words, this corresponds to searching for sufficiently small *worst-case transformations* that lead to images that are incorrectly captured by the (current) classifier; adding these *informative* training samples to the training set will lead to a change of the decision function and maximize the robustness. In the following sections, we formalize this idea in detail, and propose an efficient algorithm for data augmentation.

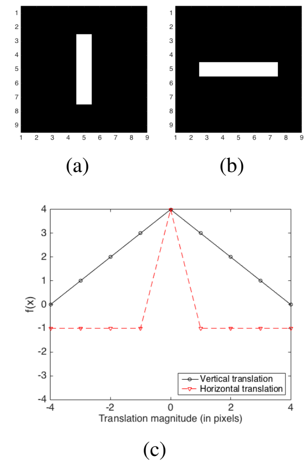

Fig. 1. (a): Class 1 image, (b): class -1 image, (c): score of the linear classifier when applying vertical and horizontal translations to the *vertical* line image (in (a)).

3. DATA AUGMENTATION ALGORITHM

In this section, we introduce a formalism for the worst-case data augmentation framework. Let $\mathcal{T}$ denote the set of possible transformations, which is assumed to have t degrees of freedom. For example, when $\mathcal{T}$ corresponds to two-dimensional translations, we have $t = 2$. For a given image I, and transformation $\theta \in \mathcal{T}$, we let I_θ be the image I transformed by θ. For an image I having label y, the worst-case DA can be described by

$$\text{(P)} \quad \max_{\theta \in \mathcal{T}} \ell(y, \Phi(I_\theta)) \text{ subject to } |\theta| \leq L,$$

where ℓ is the classifier's loss function, Φ is a feature extraction mapping, and L is a user-specified limit that upper bounds the entries of the vector $|\theta| \in \mathbb{R}^t$. Note that the constraint in (P) has the role of keeping the transformation sufficiently small, which is important to guarantee the label-preserving property of the transformation. The problem (P) is difficult to solve due to the nonconvexity of the objective function for typical classifiers, and approximations become necessary. For example, in convolutional neural network architectures, Φ is a composition of several elementary operations (e.g., convolution, nonlinearity, pooling) and ℓ is often set to the softmax loss. We therefore propose a generic *iterative trust-region* optimization [10] scheme, where a linear approximation is done at each iteration.[1] Specifically, by linearizing the objective function of (P) around the current iterate, we have for small enough $\Delta\theta$

$$\ell(y, \Phi(I_{\theta+\Delta\theta})) \approx \ell(y, \Phi(I_\theta)) + \nabla_\theta \ell(y, \Phi(I_\theta))^T \Delta\theta.$$

The gradient $\nabla_\theta \ell(y, \Phi(I_\theta))$ can be explicitly computed using the chain rule $\nabla_\theta \ell(y, \Phi(I_\theta)) = J_\theta(I_\theta)^T \nabla \ell(y, \Phi(\cdot))|_{I_\theta}$,

[1] The derivations and algorithms developed in this paper are not specific to deep networks.

Algorithm 1 Worst-case data augmentation

Inputs: image I, classification functions ℓ and Φ, transformation space $\mathcal{T}$, parameters ΔL, L.
Outputs: transformed image $I_{\hat{\theta}}$.
Initialize $\hat{\theta}_0 \leftarrow \mathbf{0}_{\mathcal{T}}$, $K \leftarrow \lfloor L/\Delta L \rfloor$.
for all $i \in \{1, \ldots, K\}$ **do**
Solve the linear program

$$\max_{\Delta\theta} \nabla \ell(y, \Phi(\cdot))|_{I_{\hat{\theta}_{i-1}}}^T J_\theta(I_{\hat{\theta}_{i-1}}) \Delta\theta \quad \text{subject to } |\Delta\theta| \leq \Delta L. \tag{2}$$

$\hat{\theta}_i \leftarrow \hat{\theta}_{i-1} + \Delta\theta$.
end for
Set the final estimate $\hat{\theta} \leftarrow \hat{\theta}_K$.

where $J_\theta \in \mathbb{R}^{d \times t}$ is the Jacobian matrix of the function $\theta \mapsto I_\theta$ (with d the number of pixels in I). Hence, the following linear program is considered to estimate the transformation parameter increment $\Delta\theta$:

$$\max_{\Delta\theta} \nabla \ell(y, \Phi(\cdot))|_{I_\theta}^T J_\theta(I_\theta) \Delta\theta \quad \text{subject to } |\Delta\theta| \leq \Delta L. \tag{1}$$

The parameter ΔL defines the size of the *trust region*, where the linear approximation holds. The approximate problem in Eq. (1) is now a linear program (LP) of size t, and can be solved in polynomial time using off-the-shelf linear solvers. Given that t is taken to be small in practice (e.g., $t = 6$ for affine transformations), the LP in Eq. (1) can be solved very efficiently.[2] Our full iterative trust-region optimization procedure is given in Algorithm 1. Note that $\lfloor L/\Delta L \rfloor$ iterations of Eq. (1) are solved, in order to guarantee that the final estimate $\hat{\theta}$ satisfies the constraint of problem (P), that is $|\hat{\theta}| \leq L$.

Given the trust-region data augmentation scheme in Algorithm 1, we propose a modification of the Stochastic Gradient Descent training procedure that incorporates data augmentation. The proposed Stochastic Gradient Descent with Data Augmentation (SGD-DA) is given in Algorithm 2. The algorithm follows the SGD procedure, but with an additional feature that permits to *transform* training points using Algorithm 1. At each iteration of the training algorithm, the current image is transformed with probability $p \in (0, 1)$ following Algorithm 1; that is, the training point is transformed by the *worst-case* transformation with respect to the current datapoint I, and current classifier parameters W, which is then added to the training set. This learning strategy has the benefit of automatically adapting to the current classifier, in order to generate points that are *informative* for the current estimated classifier. Note moreover that SGD-DA can be extended in a straightforward way to work with mini-batches at each iteration, instead of individual samples.

[2] In the experiments, we use the simplex algorithm in the MOSEK optimization toolbox [11].

Algorithm 2 Stochastic Gradient Descent with Data Augmentation (SGD-DA)

Inputs: Training samples $\mathscr{I}$, labels $\mathscr{Y}$, probability p, learning rate η, parameters ΔL, L, $\mathcal{T}$, loss ℓ.
Outputs: Classifier parameters W.
Initialize the classifier parameters W randomly.
while termination criterion not met **do**
Select a training point at random $I \in \mathscr{I}$, and let $y \in \mathcal{Y}$ be its associated label.
With probability p:
1. Let θ be the transformation obtained from Alg. 1.
2. Update: $W \leftarrow W - \eta \nabla_W \ell(y, \Phi_W(I_\theta))$.

Otherwise (i.e., with probability $1-p$), use the traditional update step: $W \leftarrow W - \eta \nabla_W \ell(y, \Phi_W(I))$.
end while

4. EXPERIMENTAL RESULTS

4.1. Experimental setup

We consider the transformation set $\mathcal{T}$ in our DA scheme to be the set of two-dimensional affine transformations ($t = 6$). We set the boundary condition parameter $L = 0.25$, the trust region parameter ΔL to 0.05 and the data transformation probability $p = 0.3$. In the following experiments, we evaluate our DA scheme in Algorithm 2 to train deep convolutional neural networks. Specifically, in a first training step, we train a neural network on the original dataset, using SGD. Then, using Algorithm 2, we *fine-tune* the network for a fixed number of extra epochs. The proposed scheme is compared to several DA algorithms. In particular, we compare it to a random DA scheme, which follows the same procedure as Algorithm 2 but applies a *random* transformation satisfying the constraint of (P): $|\theta| \leq L$. This comparison is particularly important, as random DA is commonly applied in practice. We perform quantitative comparison between the different methods in terms of test error, and robustness to transformations.

4.2. Experimental results

We first provide an evaluation of the proposed algorithm on the MNIST handwritten digits dataset [12]. The dataset is composed of $60,000$ training images, and $10,000$ test images. To make the problem more challenging, we consider that the number of available training data is limited; we sample randomly 500 digits from each class, which results in a dataset composed of $5,000$ images in total. The test set is however kept unchanged. We consider a Convolutional Neural Network (CNN) architecture containing three successive layers of convolution, pooling and rectified linear units operations. Table 1 shows the test errors of the proposed method (Algorithm 2), as well as competing methods. The classifier fine-tuned with the proposed method outperforms the approach with no DA, as well as the random DA. This lends

	Test error (%)
Without DA	1.84
Random (affine)	1.58
InfiMNIST [8]	1.04
AlignMNIST [9]	0.84
POP [13]	1.52
Proposed (affine)	1.03

Table 1. Test error on the MNIST-500 dataset

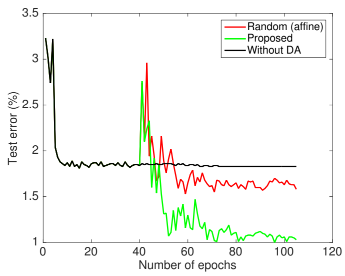

Fig. 2. Evolution of the test error with the epochs for i) random, ii) proposed and iii) no DA. For random and proposed, transformed samples are added starting from epoch 40.

credence to the intuitive idea that optimizing over the set of transformations leads to better results, compared to a scheme that chooses the parameters in a random fashion. To further study the difference between the two approaches, Fig. 2 illustrates the evolution of the test error with respect to the number of epochs used to train the neural network. While the random scheme stabilizes quite quickly around an error rate of 1.5%, the *adaptive* approach constantly improves its classification error as it seeks for the optimal transformation with respect to the current classifier. We further analyze the influence of parameter p on the test error in Fig. 3. For this experiment, all networks are fine tuned for 20 epochs. It can be seen that, for our method, larger p implies a lower classification error. In other words, adding more samples to the training set largely improves the error rate. Compared to the random scheme, the proposed method requires much less transformed samples in order to achieve low classification error.

The proposed algorithm is also compared to state-of-the-art algorithms in [8, 9, 13] in Table 1. As can be seen, our results are on par with existing methods that either consider digit-specific transformations or much larger transformation groups (e.g., diffeomorphisms in [9]). Conversely, the proposed method exploits a relatively simple and generic transformation group (affine), and maximizes the impact of transformed samples to improve classification results.

We now conduct similar experiments on the more challenging Small-NORB dataset [14], which contains 3D objects representing 5 categories. Similarly to the previous experiment, we train a CNN classifier, and perform fine-tuning using random and the proposed DA scheme. The results are

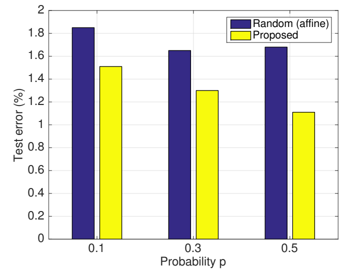

Fig. 3. Error rates for different choices of p.

	Test error (%)
Without DA	6.80
Random (affine)	6.49
[2]	4.71
Proposed (affine)	4.02

Table 2. Test error on the Small-NORB dataset

	MNIST-500	Small-NORB
Without DA	1.62	0.26
Random (affine)	1.68	0.29
Proposed (affine)	1.98	0.36

Table 3. Manitest invariance scores for the two datasets

shown in Table 2. Our adaptive DA scheme results in a significant performance boost compared to the classifier that is trained without DA, as well as random DA, and recent reported results on this dataset in [2].

We finally assess the robustness to transformations of our learned classifier in Table 3 by reporting the *Manitest* invariance scores [7] for similarity transformations. It can be seen that, for both datasets, the scores significantly increase after data augmentation using the proposed approach. Hence, our DA scheme not only results in higher accuracy, but also leads to larger robustness to transformations, which can be crucial in real-world applications where images are perturbed by unknown transformations.

5. CONCLUSION

We proposed a novel DA approach where small transformations are sought to maximize the classifier's loss. The problem is formalized as a constrained optimization problem, and solved using a trust-region approach with an iterative linearization scheme. Experimental results on two datasets have shown that this simple scheme yields results that are on-par (or superior) to state-of-the-art methods. In future work, we plan to build on this framework to provide DA strategies that can handle very large datasets (potentially containing millions of images) by providing transformations that are common to a large number of samples in the training set.

6. REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1106–1114.

[2] D. Cireşan, U. Meier, J. Masci, L. Gambardella, and J. Schmidhuber, "High-performance neural networks for visual object classification," *arXiv preprint arXiv:1102.0183*, 2011.

[3] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3642–3649.

[4] M. Paulin, J. Revaud, Z. Harchaoui, F. Perronnin, and C. Schmid, "Transformation pursuit for image classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 3646–3653.

[5] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations (ICLR)*, 2014.

[6] S-M. Moosavi Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: a simple and accurate method to fool deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[7] A. Fawzi and P. Frossard, "Manitest: Are classifiers really invariant?," in *British Machine Vision Conference (BMVC)*, 2015, pp. 106.1–106.13.

[8] G. Loosli, S. Canu, and L. Bottou, "Training invariant support vector machines using selective sampling," pp. 301–320. MIT Press, Cambridge, MA., 2007.

[9] S. Hauberg, O. Freifeld, A. Larsen, J. Fisher III, and L. Hansen, "Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation," *arXiv preprint arXiv:1510.02795*, 2015.

[10] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint, *Trust-region Methods*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[11] MOSEK ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 7.1 (Revision 28).*, 2015.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[13] Y. Amit and A. Trouvé, "Pop: Patchwork of parts models for object recognition," *International Journal of Computer Vision*, vol. 75, no. 2, pp. 267–282, 2007.

[14] Y. LeCun, F.J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004, pp. 97–104.