Contextually Constrained Deep Networks for Scene Labeling

Taygun Kekec, Rémi Emonet, Elisa Fromont, Alain Trémeau, Christian Wolf

Contextually Constrained Deep Networks for Scene Labeling

Taygun Kekeç[1]
taygunkekec@gmail.com
Rémi Emonet[1]
remi.emonet@univ-st-etienne.fr
Elisa Fromont[1]
elisa.fromont@univ-st-etienne.fr
Alain Trémeau[1]
alain.tremeau@univ-st-etienne.fr
Christian Wolf[2]
christian.wolf@liris.cnrs.fr

[1] Université de Lyon, CNRS UMR 5516, Laboratoire Hubert-Curien Université de Saint-Etienne, F-42000, Saint-Etienne, France

[2] Université de Lyon, CNRS INSA-Lyon, LIRIS, UMR5205, F-69622, France

Abstract

Learning using deep learning architectures is a difficult problem: the complexity of the prediction model and the difficulty of solving non-convex optimization problems inherent to most learning algorithms can both lead to overfitting phenomena and bad local optima. To overcome these problems we would like to constraint parts of the network using some semantic context to 1) control its capacity while still allowing complex functions to be learned 2) obtain more meaningful layers. We first propose to learn a weak convolutional network which would provide us rough label maps over the neighborhood of a pixel. Then, we incorporate this weak learner in a bigger network. This iterative process aims at increasing the interpretability by constraining some feature maps to learn precise contextual information. Using Stanford and SIFT Flow scene labeling datasets, we show how this contextual knowledge improves accuracy of state-of-the-art architectures. The approach is generic and can be applied to similar networks where contextual cues are available at training time.

1 Introduction

Deep learning approaches, such as multi-layer neural networks, leverage the amount of available data to learn representations: instead of hand-crafting intermediate features, they are learned directly from the data. This is particularly relevant since there is no universal feature detector performing best for any given problem and these learned features have been shown to outperform hand-crafted features on many perception tasks.

Recent advances in deep learning methods allow them to scale to big vision datasets. For example, convolutional neural networks (CNNs) provide some amount of translation invariance and are perfectly adapted for spatial data such as images (and temporal data such as audio channels). From an optimization perspective, stochastic gradient descent and efficient back-propagation algorithms provide significant learning time improvements.

Here, we consider the task of semantic full scene labeling, in which an image is segmented into meaningful regions. However, a large part of the contributions can also be applied to related problems in which contextual information in addition to local appearance information is primordial such as, e.g., object detection and recognition. Contextual information often allows to disambiguate decisions where local information is not discriminant enough. It commonly comprises cyclic relationships between pixels, super-pixels or parts, which can be difficult to model. In principle, increasing the support of a classifier (the input patch size) can increase the amount of context taken into account for the decision. In practice, this places all the burden on the classifier, which needs to learn a highly complex prediction model from a limited amount of training data, often leading to poor performance.

In the approach we propose, we first learn a network to predict contextual information. We assume that the contextual information is obtainable from ground truth labels at training step. In parallel, we learn a second model for the original task assuming that clean contextual data is available. Finally, we combine these networks and perform a last training phase without using the contextual information.

As a summary, the contributions of this paper are: i) a generic way of integrating semantic context information when learning convolutional networks; ii) a new training procedure, which switches from a constrained but easy configuration without contextual noise to a realistic configuration, where the system learns to cope with noise in the contextual data; iii) an illustration of how such an approach improves learning (by avoiding bad optimum) leading to increased accuracy when applied to the challenging task of full scene labeling.

2 Related Work

For computer vision tasks, convolutional nets [10] have been gaining attention as a tool famous for its fast inference capabilities paving the way to many successful applications such as image classification [8], house digit classification [15] and human body part estimation [7]. The focus of this article is on improving such an architecture in two directions. First, due to the highly non-convexity of the target function, the optimization procedure usually gets stuck in local optima. For such cases, a popular strategy is to initialize the network with unsupervised pre-training to guide the optimization to a more reliable region in the weight space [3] and then train it with supervised information. This strategy has been proven handy for applications where vast amount of unlabeled data is available. Second, the more layers are added to a deep network, the more difficult it becomes to understand the semantics of the intermediate layers [21]. Interpretability of features can provide intuitions for architecture design and diagnostics for inference. For example, with a neural network trained for a bank credit approval application, interpretable features would provide reasons why one's credit application is not approved (salary feature may not be sufficiently activated). By forcing part of the network to capture some context information of our choice, we aim to improve the interpretability of the CNN.

The essence of these deep architectures is to gradually transform observations into high level abstract concepts in the highest layers, showing that increasing number of layers in a deep network makes it possible to learn abstract concepts better. An example is the work of Goodfellow et. al [5] which develops a multi-digit number recognition technique using a CNN with eight convolutional layers. Although, these deep networks excel in approximating underlying target function, training such networks is still difficult and special care must be taken to learn meaningful and accurate functions. If one does not have enough compu-

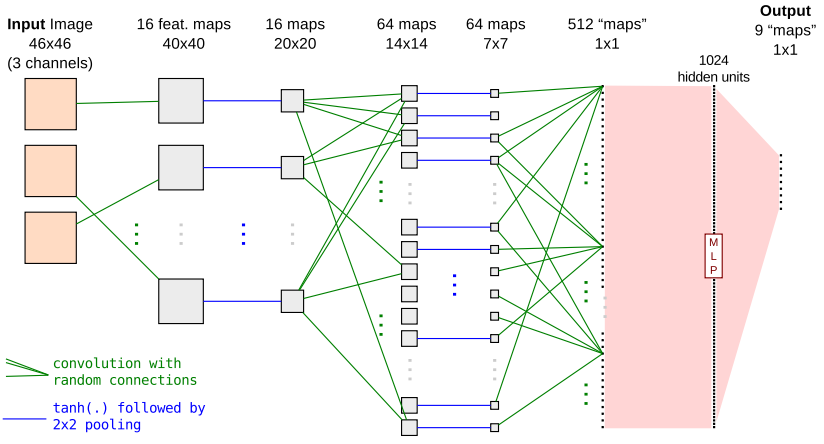

Figure 1: Plain single-scale convolutional architecture presented in [9] for scene labeling

tational resources and vast amount of data, the designer must provide extra topological or optimization constraints to effectively train the network. For example, in [9], Le et al. impose an orthonormal constraint to pre-initialize weights of a convolutional network to force the learned features to be more diverse. Then a a pooling step across multiple features is used to provide not only a translational invariance but also an invariance to more complex transformations.

The problem we are interested in this paper is scene labeling: given an image we wish to label each pixel with its object category. It is a joint formulation of the segmentation, detection and recognition problems. Some state-of-the-art approaches for scene labeling are based on graphical models such as Markov Random Fields (MRFs), Conditional Random Fields (CRFs) [19] or Bayesian networks. Inference in these models amounts to solving combinatorial problems, which in the case of high level contextual information are often non-submodular and intractable in the general case.

Instead, feedforward approaches formulate inference of labels as a local classification task so that inference will be extremely fast. They classify pixel labels with a pure discriminative approach by assuming that latent labels are independently and identically distributed. For example, Farabet et al. [4] uses a convolutional network to infer pixels labels (Figure 1). In this work, a multiscale approach is used to force the network to learn scale invariant features. The advantage of such a multiscale approach is to control the number of parameters (capacity). Local decisions resulting from the convNet are further corrected with global decision rules arising from a CRF which gives a greater spatial coherence between the estimated image labels. Unlike [4], we avoid any correction with a graphical model that would slow down the inference process. In a different fashion, we expect our Context Learner to learn about the spatial consistency and provide this information to the whole network.

For such feedforward approaches, one should select a patch big enough (i.e., with a large support size) to take large dependencies into account. At the same time, the number of parameters of the network must remain reasonable. Pinheiro tackles this problem by adding a recurrence structure to the convolutional network and operating on a larger support size [13]. To control the capacity, large pooling units are used that bring considerable loss in the image resolution in the upcoming layers. To cope with this loss of resolution, shifted

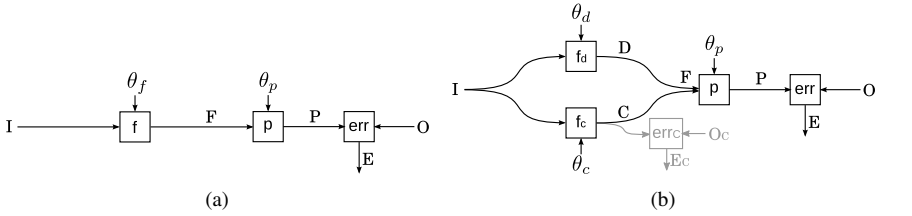

Figure 2: Functional representation of our feature learning approaches. (a) The target function is composed of a feature extraction function f and a prediction function p. (b) Our approach which distinguishes the learning of context features f_c and dependent features f_d.

versions of the input image are fed to the network at the expense of slower inference.

Our strategy is not to choose an optimal support size to capture as much dependencies as possible. It is closer to the idea of iterative classification [16]. For example, in Tu et al.'s work [20], a weak classifier is first trained to classify each pixel independently of the context. Then, a new classifier is initialized with the result of the weak classifier and some uniformly distributed context cues. Subsequent iterations of the classifier learn to predict the values of context features in the image, showing that learning context brings greater accuracy. Shotton et al. [17] also propose a sequential schema using Randomized Decision Forests to incorporate semantic context to guide the classifier. In the proposed approach, bags of semantic textons act as a regional prior to maintain the coherence across a region in the image. These approaches show that providing contextual information to a classifier helps to prevent inconsistent classifications. This is also our aim in this paper.

3 Proposed approach

Following the intuition that context can help in learning classifiers, our approach is to first train a function that predicts some context information. Then, the context coming from this predictor, together with the input are used to learn a classifier for the original task. In this section, we introduce necessary notations and concepts, then we illustrate our approach with Convolutional Neural Networks (CNNs) and finally we show how learning is conducted.

3.1 Notations and CNNs

Classical feature learning – In the context of feature learning, the input processing is traditionally separated in two parts as illustrated in Figure 2a. The input I is first processed with a function $f(.)$, which has parameters θ_f and produces a set of features F. A predictor $p(.)$ having parameters θ_p takes the features F as input and produces a prediction.

At learning time, this output P is compared to the expected output O to produce an error using a loss function $\mathcal{L}$ that is often the quadratic error: $\mathcal{L}(P,O) = ||p(f(I,\theta_f),\theta_p) - O||^2$. When all the involved functions are differentiable functions, gradient descent can be used to minimize $\mathcal{L}(P,O)$ over a training set. The minimization process finds the (locally) optimal value for $\theta = (\theta_f, \theta_p)$. Unlike systems where the features are manually extracted and then a classifier is learned, here, the two sets of parameters (θ_f and θ_p) are learned jointly.

In practice, we learn from a training set S containing N samples: $\{(I_i, O_i)\}_{i=1..N}$. To cope with large training sets, stochastic gradient descent is often used. Once the parameters are

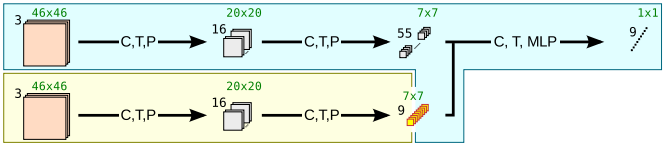

Figure 3: Implementation of the proposed approach. The architecture is a succession of convolution (C), element-wise hyperbolic tangent (T) and 2×2 pooling (P). The labels of 49 pixels (7x7) within the considered patch are considered as semantic context information. These labels are one-hot encoded into 9 feature maps. We first learn the lower yellow part of the network providing a set of ground truth labels.

learned, $p(f(I))$ is used as a prediction for the the input image I.

Convolutional Neural Networks – A convolutional neural network (CNN) [10] is a special case of a neural network that respects the topological structure of the image by using multidimensional convolutions. Given an image, the network learns a set of filters. The CNN architecture is based on two main concepts: local receptive fields and weight sharing. Contrarily to other neural network architectures, neurons of a given layer are only connected to a subset of the neurons in the subsequent layer. This subset is called a local field or a receptive field. Since a convolution operation on an image can be implemented by sliding a convolutional kernel over the entire image, the weights of the filter are shared from one position to another which introduces some translational invariance into the network [13]. These two ideas reduce the overall complexity of the model, and allow CNNs to scale up to high dimensional inputs. A standard CNN is implemented using a set of operations. First, the input image is convolved with a set of 2D filters. Then, a point-wise non-linear squashing function is applied to a sparse linear combination of responses to introduce non linearity and allow the network to learn more complex function [14]. Finally, a pooling layer which down-samples the feature map is used to reduce the sensitivity of the network to input variations. After capturing much of the non-linearity using convolutional layers, a final classifier of arbitrary choice is used to classify the examples. An example of a convolutional network with three convolutional layers is depicted in Figure 1. The first two layers consist of 7×7 convolutional filters, a hyperbolic tangent squashing function and a 2×2 max pooling operations on each maps. The number of feature maps are generally determined empirically and the size of convolution filters are carefully selected to be coherent with the input image size. The last layer of this CNN has a final convolution and a standard multi-layer perceptron (MLP).

3.2 Proposed Augmented CNN

For a function composed of f and p (see Fig. 2), finding a good local optima may be troublesome especially with neural networks where a huge number of parameters is involved. Our goal is to maintain the representation power of such a network while guiding its learning. We thus propose to add some intermediate supervision, encouraging part of the learned features to capture some predetermined information.

To constrain the whole network, we propose to split the function f into two parts: f_d and f_c (Fig. 2b). Function f_c aims at predicting some context and it is learned with additional supervision (examples of the expected context). This increased supervision does not require more annotations: for instance, the scene labeling datasets are already densely annotated.

The dependent features function f_d computes additional features and is learned (jointly with p) conditionally on the context obtained from f_c. Learning f_d conditionally on f_c encourages the dependent features f_d to extract information that is complementary to f_c. Below, we describe the networks that constitutes f_c and f_d, based on the CNN from Fig. 1.

Context Learner – The aim of the first network is to produce the semantic context of a pixel. The weights of this network are the parameters of context the function $f_c(.)$. This network, called *Context Learner* is tightly supervised using ground truth annotations to learn the $f_c(.)$ function. It takes as input a training patch $\mathcal{X}_k$ of size $s \times s$ together with a set of labels $\mathcal{N}(x)$ around the target pixel x of the patch. It is also a convolutional network but with only two convolutional layers which aim at predicting not only the label y_k of the target patch pixel but also the label of the entire neighborhood context $\mathcal{N}(x)$.

Augmented Learner – Our full architecture consists of the above-mentioned *Context Learner* and a *Dependent Learner* (depicted in blue in Fig. 3). The Dependent learner is a typical CNN that is responsible for learning the $f_d(.)$ function. The augmented learner's prediction function $p(.)$ has the same capacity as the one learned from a plain CNN, but thanks to the internal separation of the network into two different entities, we expect it to be more accurate and more efficient than a plain convolutional network which would not explicitly learn some contextual features.

3.3 Learning phases in Augmented CNN

Our Augmented Learner architecture is trained in successive steps to predict the label of an input patch. We describe here the three phases of our learning strategy (see Fig. 2).

Learning context – In this step, we start from a random initialization θ_c^0 and learn θ_c^1 from some samples $S = \{(I_i, \mathbf{O_i})\}_{i=1..N}$ where the superscript j in θ_c^j indicates the training stage and $\mathbf{O_i}$ is a set of ground truth labels. The context learning step minimizes the following error function: $\mathcal{L}_c = \sum_{k=1}^{K} \left\| p_{soft}^k(f_c(I, \theta_c) - O_c^k) \right\|^2$ where K is the number of context pixels for a patch I_i, p_{soft}^k is the softmax prediction output for $k'th$ pixel and O^k is the ground-truth label of $k'th$ context pixel.

The context learner is trained with a semantic label map containing the ground truth labels of the pixels to predict. At the end of this training step, the feature maps that correspond to the output of the *Context Learner* will be specialized in modeling the neighboring context of the target pixel. As a standard CNN focuses only on learning the class of a given patch y_i, it is hard to infer what the last layers are actually learning. In contrast, our learner increases the interpretability of the whole network.

Learning dependent features – The goal of this part of the augmented learner is to learn the parameters (θ_d^2, θ_p^2) from a random initialization of (θ_d^0, θ_p^0) using some samples $S = \{(I_i, O_i)\}_{i=1..N}$ and from parameters θ_c^1 learned in the previous step. We minimize $\mathcal{L}$ while keeping θ_c^1 fixed. Fixing θ_c prevents harming the parameters of the context learner while learning θ_d^2. In Fig. 3, the parameters in the yellow region of the Augmented Learner are frozen during the back-propagation steps.

Learning θ_d^2 requires to use the features C of the context learner shown in Figure 2b. This can be either a ground truth label map or directly the predictions from the embedded context learner (previous step). We generate the context stochastically for learning $f_d(.)$ using a mixture of ground truth labels and context learner predictions. We replace the context learner predictions with some ground truth labels (it is also a $7 * 7$ label map) randomly following

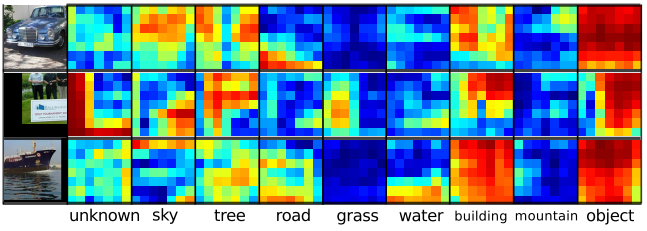

Figure 4: 3 input patches and their resulting feature maps produced by the context learner.

a Bernoulli distribution $Ber(x|\theta = \tau)$ where τ is generally chosen small. This is done both because a full label map would be too far from the actual context predictions and because it could result in trivially learning $f_d(.)$. On the other hand, introducing some ground truth context regularizes the $f_d(.)$ learning step. When the ground truth context is used we still mask a $3*3$ regions around the pixel of interest.

Fine tuning – In this step, we learn the final parameters $\theta^3 = (\theta_c^3, \theta_d^3, \theta_p^3)$ from some samples $S = \{(I_i, O_i)\}_{i=1..N}$. We start from an initial value of $(\theta_c^1, \theta_d^2, \theta_p^2)$, and we minimize $\mathcal{L}$. This idea of this overall refinement step is to weaken the level of supervision and allow both θ_f and θ_d to adjust to this sudden lack of possible ground truth contextual information which is obviously not present during the test step.

4 Experiments

Experimental setup – We selected Torch7, a scientific computing framework with wide support for machine learning algorithms [2] as our development environment. In order to ensure the modularity of our approach, we implemented several custom modules for the neural network package. Our approach has been tested on two scene labeling datasets: Stanford Background [6] and SIFT Flow [11]. The Stanford Background dataset contains 715 images of outdoor scenes having 9 classes. Each image has a resolution of 320x240 pixels. We randomly split the images to keep 80% of them for training and 20% for testing. From these images, we extract a total of 40 millions patches.

The SIFT Flow dataset contains 2688 256×256 manually labeled images. The dataset has 2488 training and 200 test images containing 33 classes of objects. From this we extract 160 millions patches. This dataset is more challenging than the former one because the training and test sets have different data distributions and the number of classes to predict is greater. In order to prevent overfitting in our network on such a large amount of patches, we arbitrarily use an early stopping strategy with a 10% holdout validation set [1]. For both datasets, 46×46 RGB patches are first converted to the YUV color space to separate the brightness and the color. The input size is a consequence of using pooling units with even sizes (2×2) and convolutional filters with odd sizes (7×7). These patches are then normalized to have a zero mean and a unit variance. The normalization of the Y channel is local for each patch while the U and V channels are normalized globally over all possible patches of the training set.

We report both the *pixel accuracy* measure which is the proportion of true positives over all pixels when classifying an image, and the *average class accuracy* where the average is computed with equal weights for each class. Note that in our experiments, the training sets

Table 1: Pixel and averaged per class accuracy of different methods for the Stanford and SIFT Flow datasets. $\tau = 0$ corresponds no ground truth context injection. Last columns show the number of parameters and the relative training time per sample.

Architecture	Stanford Dataset		SIFT Flow Dataset		number of	train
	Pixel Acc.	Class Acc.	Pixel Acc.	Class Acc.	# param.	speed
ContextL	54.19	45.12	42.52	9.89	4.4k	0.75x
ConvNet	69.72	66.24	48.02	44.04	700k	1x
AugL ($\tau = 0$)	72.06	67.22	48.93	44.53	701k	1.1x
AugL ($\tau = 0.05$)	71.97	66.16	49.39	44.54	701k	1.1x
msContextL	55.39	50.06	44.71	10.20	4.4k	2.1x
msConvNet	75.67	67.1	69.93	45.65	1224k	2.70x
msAugL ($\tau = 0$)	76.05	68.01	70.88	44.82	1225k	2.85x
msAugL ($\tau = 0.05$)	76.36	68.52	70.42	45.80	1225k	2.85x

are not expanded with artificial transformations. Such an procedure would increase the classification accuracy by few percents at the expense of slower training. We balance our training set over the classes which means that the training procedure will try to maximize the class accuracy. The results reported in this section are thus computed with our implementation of the Convolutional Network (*ConvNet*) presented in [4] to have a fair comparison with our architecture.

Architecture details – The context learner implementation (Fig. 3) transforms a 46×46 patch into a 7×7 context output. In the first layer, it has sixteen 7×7 filters and then 2×2 pooling operations for each feature map. Its second layer is composed of K filters (each of size 7×7) each encoding the context of a specific class followed by a 2×2 pooling operation. This layer has thus K output maps, where K corresponds to the number of classes (9 in Stanford Background, 33 in SIFT Flow). For all experiments in this work, we use the hyperbolic tangent activation function.

The supervision strategy of the context learner training depends on its filter and pooling sizes. In order to correctly subsample the ground truth labels of the dense 46×46 patch, we compute the receptive field of each neuron of the output 7×7 context map. We then use the label at the center of this receptive field (this corresponds to the precomputed pixel coordinates $\{11, 15, 19, 23, 27, 31, 35\}$).

We consider both a plain single scale convolutional net and a multiscale version (all the reported results which with the *ms* prefix are multiscale versions). In the multiscale case, the *Context Learner* is trained to classify a grid of pixels for each scale. An image pyramid with a scale ratios of 1x, 2x and 4x (with wider support) is used and the parameters of the network are shared between the 3 scales. The total number of parameters of the multiscale Context Learner is thus the same as the single scale version. From a computational perspective, our approach increases the number of parameters by less than 1% compared to the ConvNet (Table 1). The only parameter increase in our architecture is due to the first two layers of the context learner. However, these layers have less maps than the later convolutional and classifier layers.

Our Augmented Learner ("AugL" in the table) has 64 feature maps at the end of its second layer with K of them coming from the Context Learner. The third layer has a final 7×7 convolution resulting in 256 1x1 maps (in the multiscale implementation it is 256 maps for each scale, for a total of 768 maps) that are connected to an MLP with 1024 hidden units.

(a) msConvnet (b) msAugLearner (c) GroundTruth

Figure 5: Raw image labeling of the multiscale ConvNet, our multiscale augmented learner and ground truth labels.

We experiment with an AugL that does not use any true context label injection corresponding to $\tau = 0$ and another AugL that has an injection parameter $\tau = 0.05$.

Intermediate results: context learner – The classification accuracies obtained from the context learner ("ContextL" in the table) are given in Table 1 for both datasets. In Fig. 4, we show the responses of our context learner maps for some input patches. The second row shows strong responses for the object, tree and building classes. For the second and third rows, although the context learner outputs a strong response for the object class, due its the relative simplicity, it is not able to provide an accurate classification (e.g., for the building class). Nevertheless, this network is useful for the augmented learner and it's training time is negligible: the time per sample is lower (see Table 1) and it converges faster than the Convnet.

Classification accuracy results – Table 1 shows the classification results obtained with the different approaches. Overall, we observe that our method provides better results for both the Stanford and the SIFT Flow datasets. For Stanford dataset, another state of the art technique is reported by Munoz et al. [12]. They reported their pixel accuracy as 76.9 and class accuracy as 66.2 without a deep learning architecture. With our technique, we were able to obtain much higher class accuracy.

While the accuracy gain varies between singlescale and multiscale implementations, we observe that our approach consistently improves both pixel and class accuracies. The gain on single-scale experiments are higher compared to multiscale implementations. This brings us to the empirical conclusion that contextual cues obtained implicitly through appearance cues of large support size provides valuable contextual information.

Qualitative segmentation results. Some labeling results from the Stanford dataset are shown in Figure 5. Our approach yields results that are more visually coherent than those obtained with the plain ConvNet architecture. For the second and third rows, our architecture correctly classifies the most important regions of the image whereas the baseline (ConvNet)

fails at maintaining coherent results. In the first row, we illustrate a challenging traffic scene. We see that in this scenario, both approaches are insufficient to come up with a consistent global labeling. This can be arbitrarily corrected with an external CRF. However, one interesting result is that our approach detects the traffic sign correctly while the baseline CNN cannot. These figures show that, roughly initializing the context maps and lumping them to a full architecture to let them evolve further under the new task (only classifying the target pixel of a patch) teaches the full network how to take some spatial consistency into account.

5 Conclusions

We have presented a new deep learning architecture based on convolutional layers. The architecture is trained in multiple distinct steps. We show that the iterative learning strategy maintains the capacity of the network while improving both the interpretability of the network layers and the accuracy compared to a good state-of-the-art convnet architecture. This method is applied to a scene labeling problem but the idea is general enough to be applied to other computer vision problems where contextual cues are important to classify an object. Further works include testing more systematic label injection procedures.

Acknowledgement

This work has been supported by the ANR project SoLStiCe (ANR-13-BS02-0002-01).

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006. ISBN 0387310738.

[2] Ronan Collobert, Clément Farabet, and Koray Kavukcuoglu. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

[3] Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. Why does unsupervised pre-training help deep learning? In *Proceedings of AISTATS*, pages 201–208, 2010.

[4] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning Hierarchical Features for Scene Labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35 (8):1915–1929, 2013. ISSN 0162-8828. doi: 10.1109/TPAMI.2012.231.

[5] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. In *International Conference on Learning Representations*, 2014.

[6] Stephen Gould, Richard Fulton, and Daphne Koller. Decomposing a scene into geometric and semantically consistent regions. In *ICCV*, pages 1–8, 2009.

[7] Mingyuan Jiu, Christian Wolf, Graham W. Taylor, and Atilla Baskurt. Human body part estimation from depth images via spatially-constrained deep learning . *Pattern Recognition Letters*, December 2014.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105, 2012.

[9] Quoc V. Le, Jiquan Ngiam, Zhenghao Chen, Daniel Chia, Pang Wei Koh, and Andrew Y. Ng. Tiled convolutional neural networks. In *In Neural Information Processing Systems (NIPS)*, pages 1279–1287, 2010.

[10] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[11] Ce Liu, J. Yuen, and A Torralba. Nonparametric scene parsing via label transfer. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(12):2368–2382, Dec 2011. ISSN 0162-8828.

[12] Daniel Munoz, J. Andrew (Drew) Bagnell, and Martial Hebert. Stacked hierarchical labeling. In *ECCV*, September 2010.

[13] Ronan Collobert Pedro H. O. Pinherio. Recurrent convolutional neural networks for scene parsing. In *International Conference of Machine Learning*. 2014.

[14] Raul Rojas. *Neural Networks - A Systematic Introduction*. Springer-Verlag, 1996.

[15] P. Sermanet, S. Chintala, and Y. Lecun. Convolutional neural networks applied to house numbers digit classification. In *International Conference on Pattern Recognition (ICPR)*, pages 3288–3291, 2012.

[16] Roman Shapovalov, Dmitry Vetrov, and Pushmeet Kohli. Spatial inference machines. *CVPR*, 0:2985–2992, 2013. ISSN 1063-6919.

[17] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, pages 1–8, June 2008. ISBN 978-1-4244-2242-5.

[18] P.Y. Simard, D. Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Seventh International Conference on Document Analysis and Recognition*, pages 958–963, 2003. doi: 10.1109/ICDAR.2003.1227801.

[19] Joseph Tighe and Svetlana Lazebnik. Superparsing: Scalable nonparametric image parsing with superpixels. In *ECCV*, pages 352–365, 2010. ISBN 3-642-15554-5, 978-3-642-15554-3.

[20] Zhuowen Tu and Xiang Bai. Auto-context and its application to high-level vision tasks and 3d brain image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(10):1744–1757, Oct 2010. ISSN 0162-8828.

[21] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Advances in Neural Information Processing Systems*, 2013.