2016 23rd International Conference on Pattern Recognition (ICPR)
Cancún Center, Cancún, México, December 4-8, 2016

AutoMarkov DNNs for Object Classification

Cosmin Ţoca
cosmin.toca@gmail.com

Carmen Pătraşcu
carmen.patrascu@upb.ro

Mihai Ciuc
mihai.ciuc@upb.ro

University Politehnica of Bucharest
Faculty of Electronics, Telecommunications and Information Technology
Applied Electronics and Information Engineering Department
1-3, Iuliu Maniu Ave., Bucharest, Romania 061071

***Abstract*—Recent advances in the area of Deep Convolutional Neural Networks have led to steady progress, mainly observed in the field of object classification and localization. Extensive testing helped generate frameworks guaranteeing the initiation of successful network architectures. For this reason, the authors focus on bringing added value on specific nodes of a generic network configuration. We propose a novel type of convolutional layer based on Autobinomial Markov-Gibbs Random Fields (AutoMarkov Layer). Our choice is motivated by the fact that each neuron in a layer is only connected to a local region in the following layer. This property allows us to integrate Markov Random Fields into the structure of a neuron, to account for the probability of each particular pathway. Functional testing is performed on the MNIST, CIFAR-10 and CIFAR-100 datasets, showing clear improvements for correct classification scores on all the datasets mentioned regardless of the network architecture.**

I. Introduction

The technological advancement is currently underway and in a significant extent supported by recent focus on the area of Deep Learning. The ability of integrating large neural networks into mass market appliances is already making a profound impact on the user experience [1].

Deep Convolutional Neural Networks have led to a steady progress for object classification [2]–[5] but they are also successful when applied to object detection tasks [6]–[8] or computational photography [9]–[11], achieving the best results for all benchmarks and recent competitions [12]–[14]. The growing interest for this category of methods helped solve many issues related to convolutional neural networks, generating a list of steps that should be followed for building a successful network architecture.

This paper focuses on methods that improve the outcome of a convolutional neural network, without focusing on solving only specific problems. Based on information extracted from a state-of-the art review about how to select the best architecture and parameters for the network [2], [5], [15], we focus on adding prior knowledge in specific points of a generic network configuration.

There is a close relation between neural networks and probabilistic graphical models [16], both predicting effects derived from multiple causes. However, if a neural network predicts input to output relations, the probabilistic model focuses on analyzing how neurons interact and which connections are more likely. These structural differences represent the starting point for the idea developed in this paper.

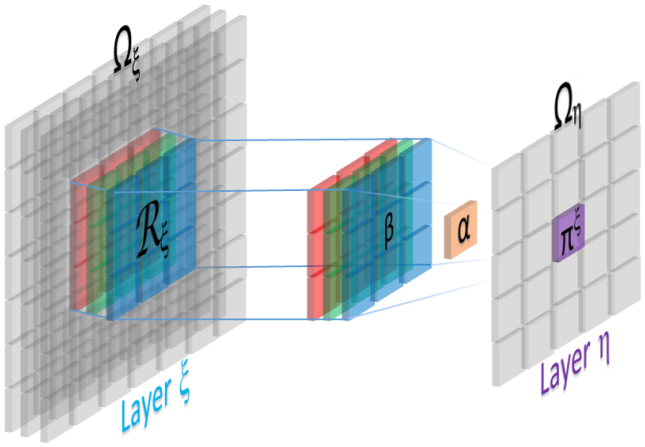

Fig. 1. **Local Property of AutoMarkov Layers.** We define $\mathcal{R}_\xi \in \Omega_\xi$ to be the receptive field, β a learnable filter, α an additive bias and $\pi_\xi \in \Omega_\eta$ the probability assigned to a local system.

In the current structures of convolutional neural networks each neuron is connected only to a local region in the previous layer. This property builds a base for integrating Markov Random Fields (MRFs) [17]–[19] into the structure of neurons in order to consider how likely a particular pathway is.

We propose a novel type of convolutional layer based on Autobinomial Markov-Gibbs Random Fields [20], [21] as shown in Fig. 2, which we call AutoMarkov Layer.

Markov Random Fields have found widespread use across image processing, in particular for texture classification [22]–[24], image segmentation [25], [26] and restoration [27], [28], but also in the field of machine learning, namely in object detection [29], [30], object recognition [31], [32], or in relation to neural networks [33], [34].

The remainder of this paper is organized as follows. Sec. II gives a detailed overview of the proposed AutoMarkov Layers and briefly describes the forward and the backward propagation steps, arguing the use of Markov Random Fields as prior probability distributions in correlation with the Convolutional Neural Networks. In Sec. III we share information about

978-1-5090-4846-5/16/$31.00 ©2016 IEEE

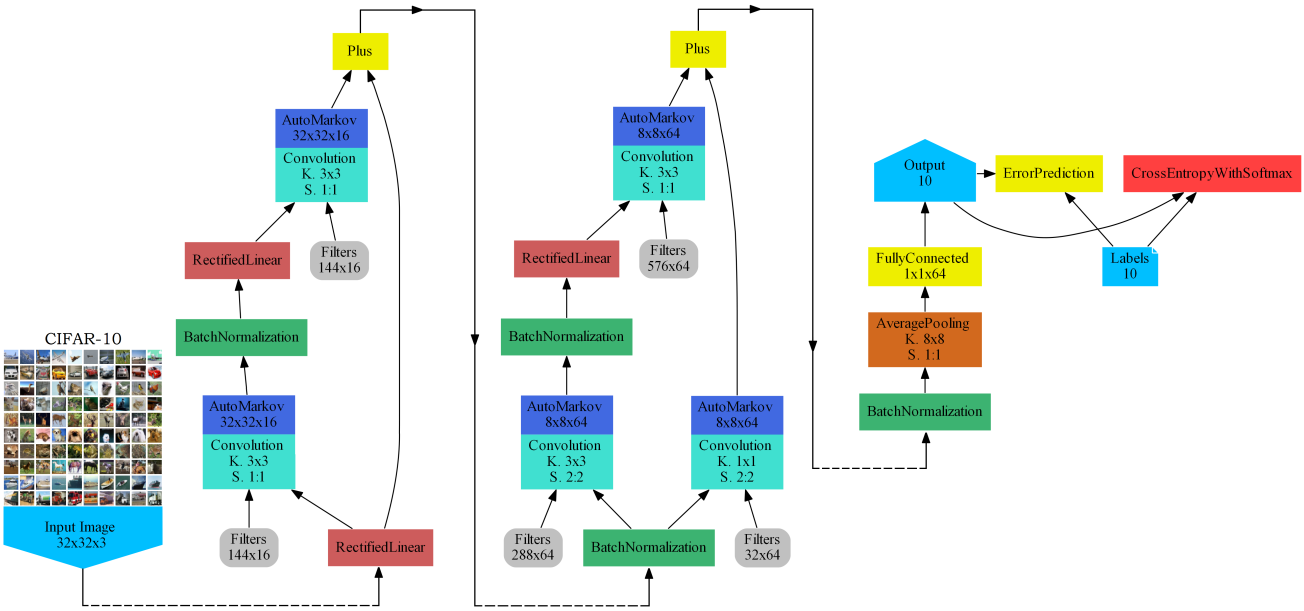

Fig. 2. **Potential Architecture of Deep AutoMarkov Residual Network.** AutoMarkov Layers compute the response of a neuron as being the local property of a Markov Random Field.

the implementation, code and tools. A detailed experimental evaluation of the results is performed in Sec. IV and the conclusions are drawn in Sec. V.

II. AutoMarkov Layers

Taking a closer look at individual structures of a Convolutional Neural Network, it can be inferred that the neurons in each convolutional layer respect the Markov Property, as described in Sec. II-A, in relation to adjacent layers. This fact allows us to provide the following probabilistic description by suitably adapting properties and definitions to meet distinguishing characteristics of the Convolutional Neural Networks.

A. Probabilistic Description

Markov Random Fields have been modeled to solve various computer vision tasks which can be posed either as energy minimization problems in image analysis, or as visual perception problems where different objects have to be distinguished using a set of discriminating features. Let us consider $X = \{X_\eta\}_{\forall \eta \in \Omega_\eta}$ a random process defined on a probability space $(\Omega_\xi, \mathcal{F}, \mathcal{P})$, where Ω_ξ is a finite set that refers to a set of neurons, ξ. Let $\mathcal{F}$ refer to all neural pathways connecting to the following layer $\Omega_\eta \ni \eta$ and let $\mathcal{P}$ be the set of probabilities assigned to each decision unit.

The Markov Property [35] says that, given a set of inputs $\mathcal{R}_\xi$, the variable η is conditionally independent to all variables in the random field (Ω_ξ) except the receptive field $(\mathcal{R}_\xi)$. Thus,

$$\mathcal{P}\Big(X_\eta = \gamma_\eta \mid X_{\Omega_\xi} = \gamma_{\Omega_\xi}\Big) = \mathcal{P}\Big(X_\eta = \gamma_\eta \mid X_{\mathcal{R}_\xi} = \gamma_{\mathcal{R}_\xi}\Big)$$

for all $\xi \in \Omega_\xi$, $\eta \in \Omega_\eta$ and $\gamma \in \mathcal{F}$.

This means that the response of a neuron η is directly influenced only by the values of the receptive neurons $\mathcal{R}_\xi$, which implies that the neurons inside a layer Ω_η are only connected to a compact subset of the previous layer.

The response (π^ξ) of a neuron ξ, as shown in Fig. 1, can be expressed as the local characteristic of a random field which is defined as the probability of assigning to each neuron η a certain probability given the values of all neurons in the receptive field. In other words $\pi^\xi = \mathcal{P}\Big(X_\eta = \gamma_\eta \mid X_{\mathcal{R}_\xi} = \gamma_{\mathcal{R}_\xi}\Big)$, where $\pi^\xi : \mathcal{F} \to [0, 1]$. Mapping the outputs in a probability space ensures that the learned filters β produce a stochastic response to a spatially local input.

A receptive field on Ω_ξ could be described as a family $\mathcal{R} = \{\mathcal{R}_\xi\}_{\forall \xi \in \Omega_\xi}$ of subsets of Ω_ξ, which extracts elementary visual features by enforcing a local connectivity between neurons of adjacent layers.

The local property of a Markov Random field can be interpreted in terms of energy (Eq. 1) and potential (Eq. 2), leading to the following definition:

$$\pi^\xi = \frac{\mathrm{e}^{-\sum_{\mathcal{R}_\xi} V_{\mathcal{R}}(\gamma)}}{\sum\limits_{\varphi \in \mathcal{F}} \mathrm{e}^{-\sum_{\mathcal{R}_\xi} V_{\mathcal{R}}(\varphi, \gamma)}}, \tag{1}$$

where $V_{\mathcal{R}}$ is the potential function and $\varphi \in \mathcal{F}$.

To get the probability associated to a neuron η being activated by $\mathcal{R}_\xi$, we need to define a potential function $V_{\mathcal{R}}$ in the receptive field. We refer to the auto-binomial model that was introduced by Besag [36] to describe various types of spatial processes, examining some stochastic models that occur in the texture of various physical materials.

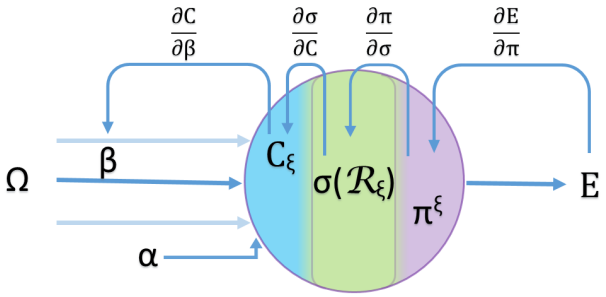

Fig. 3. **Forward and Backward Propagation.** Simplified diagram of a neuron ξ from an AutoMarkov Layer. The response π^{ξ} is an autobinomial function with respect to the result of a convolution C_{ξ} between a filter β and the receptive field R_{ξ}, where E stands for the error of the output given a target response T_{η} and α is a bias.

The potential at a certain state of the neurons is given by:

$$V_{\mathcal{R}} = \begin{cases} -\ln \binom{\Gamma_{\xi}}{\Gamma_{\eta}} + \gamma_{\xi} & \text{if } \dim(\Omega_{\xi}) = \dim(\mathcal{F}) \\ \\ \gamma_{\xi} \cdot \beta_{\xi} & \text{otherwise.} \end{cases} \tag{2}$$

It is worth mentioning that the product $\gamma_{\xi} \cdot \beta_{\xi}, \forall \xi \in \mathcal{R}$ will result in a convolution $C_{\xi,\beta}$ between the inputs $\gamma_{\xi} \in \mathcal{F}_{\xi}$ and the filters β_{ξ} to which we add a bias α_{ξ}. The potential is equal to zero outside the receptive field.

The set of inputs of a certain layer is $\Omega_{\xi} = \mathbb{R}^{m \times n \times d}$, the connections space is $\mathcal{F} : \mathbb{N} \to \mathbb{R}$, where $m, n, d \in \mathbb{N}$ and Γ_{η} represents the number of activations given by β_{ξ} in relation to the total number of inputs Γ_{ξ}. A larger difference between the two numbers Γ_{ξ} and Γ_{η} implies an increase to the overall certainty.

By replacing Eq. 2 in Eq. 1, one gets the probability assigned to a local system:

$$\pi^{\xi} = \binom{\Gamma_{\xi}}{\Gamma_{\eta}} \sigma^{\Gamma_{\eta}} (1-\sigma)^{\Gamma_{\xi} - \Gamma_{\eta}} , \tag{3}$$

where $\sigma = \sigma(\mathcal{R}_{\xi})$ is a real-valued and differentiable logistic function of $\mathcal{C}_{\xi,\beta}$:

$$\sigma(\mathcal{R}_{\xi}) = \frac{e^{\mathcal{C}_{\xi,\beta}}}{1 + e^{\mathcal{C}_{\xi,\beta}}} , \tag{4}$$

where $\mathcal{C}_{\xi,\beta} = \langle \mathcal{F}_{\xi}, \beta \rangle + \alpha_{\xi}$ and $\langle \mathcal{F}_{\xi}, \beta \rangle$ is the convolution of two matrices $\mathcal{F}_{\xi}$ representing the connections to the input layer Ω_{ξ} and β which is a learnable filter.

Detailed explanations of the Markov random fields and Gibbs distribution are found in Pierre Bremaud's book [20] and the proof of Eq. 3 is found in literature as the Markov-Gibbs Equivalence or Hammersley-Clifford Theorem [37].

By using the Markovian probabilistic model, we intend to transform the response of each convolutional layer into a probability value that takes into account both the receptive field and the certainty induced by systematic variations in the connections between adjacent layers.

B. Forward Propagation

After computing the values of the previous layer nodes, one can generate the activation value of the propagation's output as being the probability π^{ξ} assigned to a local system, as seen in Eq. 3. The local system of a certain neuron in the current layer is composed by the receptive field of the previous layer, a learnable filter β that may or may not be shared across the entire visual field and a bias term α_{ξ}, as depicted in Fig. 2.

It should be mentioned that the forward step is shaped through a binomial distribution, modeling Γ_{ξ} independent success-failure experiments, each of them yielding success with probability $\sigma(\mathcal{R}_{\xi})$. The sigmoid function $\sigma(\mathcal{R}_{\xi})$ incorporates the convolution between the receptive field in Ω_{ξ} and the kernel field β, which allows to separately forward the convolution followed by the sigmoid and ending with the binomial mapping, as shown in Fig. 3.

C. Backward Propagation

In order to propagate the error $E = \frac{1}{2}(T_{\eta} - \pi^{\xi})^2$ back through an AutoMarkov node, the derivative of the error function with respect to the weights $\partial E / \partial \beta_{\xi,\eta}$ needs to be computed. Note that T_{η} represents the target output.

The derivative of the error function is easy to solve by using the chain rule that may be written, in Leibniz's notation, in the following way:

$$\frac{\partial E}{\partial \beta_{\xi,\eta}} = \frac{\partial E}{\partial \pi^{\xi}} \cdot \frac{\partial \pi^{\xi}}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial \mathcal{C}_{\xi,\beta}} \cdot \frac{\partial \mathcal{C}_{\xi,\beta}}{\partial \beta_{\xi,\eta}} . \tag{5}$$

The derivative of the error with respect to the network output gives $\partial E / \partial \pi^{\xi} = -(T_{\eta} - \pi^{\xi})$, while the derivative of the total network input with respect to the weights results in $\partial \mathcal{C}_{\xi,\beta} / \partial \beta_{\xi,\eta} = \mathcal{F}_{\xi}$, which are similar consequences to those of a regular convolutional layer.

From now on we focus our attention on the two middle terms of the Eq. 5.

The derivative of the sigmoid function σ with respect to $\mathcal{C}_{\xi,\beta}$ has been often used as typical activation function in many neural networks but has not been so successful as rectified linear unit. As the sigmoid always has a positive derivative $\partial \sigma / \partial \mathcal{C}_{\xi,\beta} = \sigma(1-\sigma)$ the slope of the error function provides a descent direction which can be followed.

We can perform a similar calculation to determine how the probability assigned to a local system π^{ξ} changes with respect to variations of $\sigma(\mathcal{R}_{\xi})$:

$$\frac{\partial \pi^{\xi}}{\partial \sigma} = \pi^{\xi} \cdot \frac{\Gamma_{\eta} - \sigma \cdot \Gamma_{\xi}}{\sigma(1-\sigma)} . \tag{6}$$

It easily follows that Eq. 5 can be significantly simplified, as the multiplication between the two middle terms can be reduced by a common factor:

$$\frac{\partial E}{\partial \beta_{\xi,\eta}} = \mathcal{F}_{\xi} \cdot (\sigma \Gamma_{\xi} - \Gamma_{\eta}) \cdot \pi^{\xi} (T_{\eta} - \pi^{\xi}) . \tag{7}$$

We have shown how to integrate AutoMarkov layers in a multi-layer convolutional neural network, specifying that the computation of the error terms must proceed backwards

through the whole network, beginning with the output layer and ending with the first hidden layer.

III. Implementation Details

We have selected Microsoft's Computational Network Toolkit (CNTK) [38] as baseline for code integration. The source code and the network description files required for training reproduction are available on GitHub[1], as well as are the results of several tested models. As the current integration of the proposed layer architecture is available only on GPU, a video graphics card which supports CUDA is necessary.

As the development tendency of convolutional neural networks is to reach a unique vocabulary, accessibility from Network Description Language (NDL) is also provided for the developed AutoMarkov Layers, offering a flexible and simplified user interface for specifying the network topology, components and parameters including logging of various statistics during processing.

Installation guidelines, a general introduction to computational networks and the core algorithms are found on the repository's wiki page.

IV. Performance Evaluation

This section focuses on the evaluation of the proposed method by measuring the improvements brought by replacing the regular Convolutional Layers with AutoMarkov Layers.

To quantify the effect of the proposed architecture on different models, we perform an ablation study on CIFAR-10 that is an established computer-vision dataset used for object recognition, representing a great starting point towards future applications. Our aim is to assess and analyze the performance gain if the standard convolutions are replaced with the probabilistic model proposed in the present article by keeping the same network architecture (in terms of hyperparameters) and the same configuration for the learning method (momentum, learning rate and batch size) as advised by the authors. Each model has been trained from scratch, for one hour, then each training result has been tested on the entire testing dataset and the classification error has been retained in Tab. I.

TABLE I
Comparison on CIFAR-10. Comparison between standard Convolution Layers and the proposed AutoMarkov Layers within different network models.

	CIFAR-10 Testing Error - Top 1st	
	Regular Convolutions	AutoMarkov Layers
ConvNet [2]	35.76%	32.49%
AlexNet [2]	27.48%	25.92%
BatchConv [15]	26.49%	25.80%
ResNet-20 [5]	18.67%	17.58%

The results show improvements up to 3.2% in terms of correct classification on the preliminary epochs of training.

As shown in Tab. I, there are multiple models covered in this testing scenario, as the proposed architecture can be used in combination with any convolutional neural network. However, the main focus lies on an extensive performance testing of Deep Residual Networks in combination with AutoMarkov Layers. Deep Residual Networks, authored by Kaiming He et al. [5], have recently achieved state-of-the-art results in image classification and detection. For treating the underfitting issues that appear beyond a certain depth even if batch normalization is used, ResNets add skip connections that bypass a few convolution layers at a time. They are standard feed-forward convolutional networks where each shortcut generates a residual block in which the convolution layers predict a residual that is added to the block's input.

The functional testing in the present work is performed on MNIST [39], CIFAR-10 [14] and CIFAR-100 [14] datasets. In this case the models have been trained from scratch, for ten hours, then each training output has been evaluated on the corresponding testing dataset and the classification error has been retained in Tab. II, Tab. III and Tab. IV.

TABLE II
Classification on MNIST. Comparison between standard Convolution Layers and the proposed AutoMarkov Layers within ResNet models on 28-by-28 gray level images divided into 10 classes.

	MNIST Testing Error - Top 1st	
	Regular Convolutions	AutoMarkov Layers
ResNet-56 [5]	0.59%	0.43%

TABLE III
Classification on CIFAR-10. Comparison between standard Convolution Layers and the proposed AutoMarkov Layers within ResNet models on 32-by-32 color images divided into 10 classes.

	CIFAR-10 Testing Error - Top 1st	
	Regular Convolutions	AutoMarkov Layers
ResNet-20 [5]	8.26%	7.83%
ResNet-56 [5]	6.43%	5.55%

TABLE IV
Classification on CIFAR-100. Comparison between standard Convolution Layers and the proposed AutoMarkov Layers within ResNet models on 32-by-32 color images divided into 100 classes

	CIFAR-100 Testing Error - Top 1st	
	Regular Convolutions	AutoMarkov Layers
ResNet-56 [5]	28.29%	27.61%

By testing the proposed neuronal architecture within a configuration of ResNet with 56 layers on the MNIST database of handwritten digits has been observed an improvement of 0.16% leading to a correct classification rate of 99.57%.

In the case of the same model, the proposed convolutions led to an improvement from 93.57% to 94.45%, corresponding to 3rd position on CIFAR-10, while in the case of a shallow model composed of 20 convolutional layers, it has been measured an improvement of 0.45%.

If the number of classes is much bigger, as in the case of CIFAR-100, the improvements brought by the proposed convolutional layers have been increased up to 0.68%.

[1] git clone - -recursive -b AutoMarkov https://github.com/ctoca/cntk.git

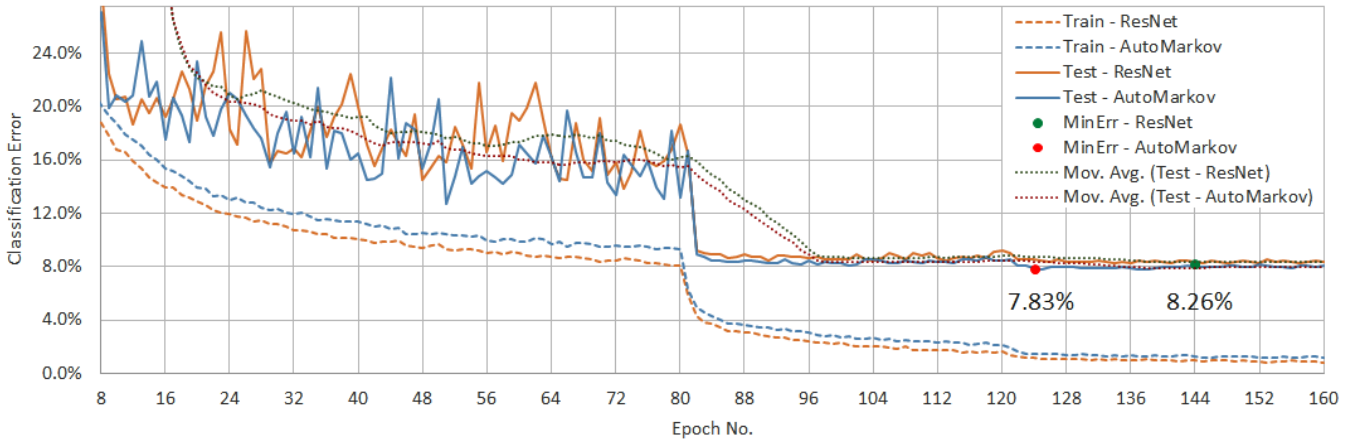

Fig. 4. **Training and Testing Error.** Both AutoMarkov and ResNet have 20 layers. The dotted lines overlapping the test results represent a moving average with a step of twenty.

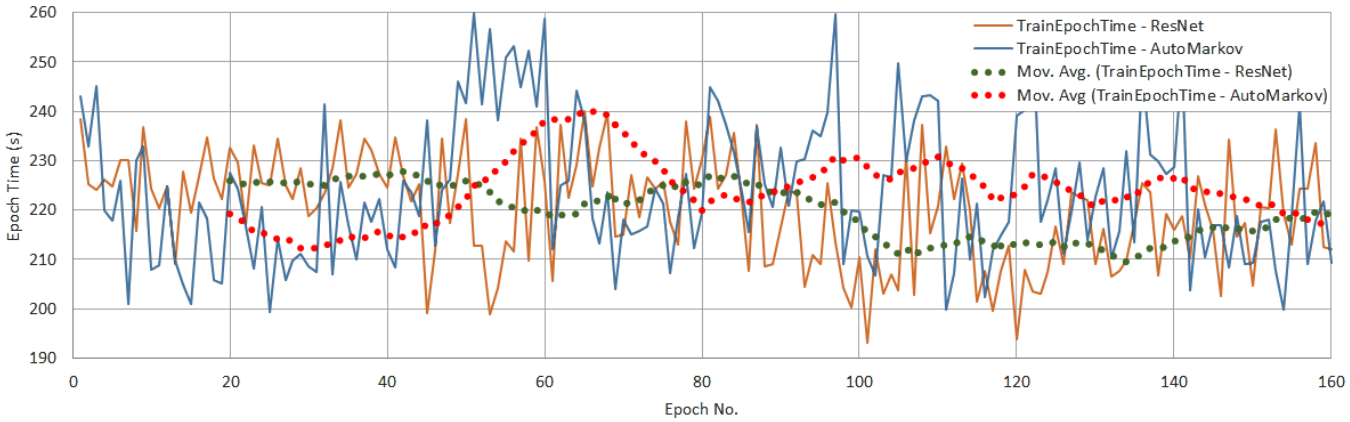

Fig. 5. **Training Time.** Comparison between ResNet-20 and AutoMarkov Layers shows that no critical overhead is added to the epochs' training time during the computation of forward and backward steps.

The results shown in this section demonstrate the effectiveness of the proposed layers regardless of network type or size.

Analyzing the behavior of the classification errors over multiple epochs (observed in Fig. 4) it can be noticed that AutoMarkov has a higher error rate than ResNet during the training process, but presents an improvement of nearly 0.5%, reaching 92.17% correct classification during the testing stage.

These results prove that the proposed AutoMarkov Layers improve the network generalization making it less prone to overfitting.

Regarding the training time, Fig. 5 shows that the computation of forward and backward steps of AutoMarkov Layers does not add a critically overhead to the epochs training time.

On average, training an epoch of a neural network with added AutoMarkov Layers took 223s compared to the original model (ResNet-20) that took 220s on CIFAR-10. This means that we process about 227 images/s on a NVIDIA GeForce GTX 980 Ti graphics card, with 2 images/s less than the baseline version.

Deep neural networks usually have a large number of parameters detailing the complicated relationships between their inputs and outputs. With limited training data, these relationships will be the result of sampling noise leading to overfitting. If the network is just large enough to provide an adequate fit, it is unlikely for the model to overfit the training data, but the computation of a network's depth for specific scenarios represents an extremely difficult problem.

All test results and network description files required for training reproduction for each AutoMarkov architecture are available on GitHub.

V. Conclusions

This article presents a novel type of convolutional layer based on Autobinomial Markov-Gibbs Random Fields, called the AutoMarkov Layer. The proposal exploits the standardized architectural deep network configurations developed in recent years and adds prior knowledge to the neurons by integrating a probabilistic model which focuses on their interaction and probabilities associated to particular pathways. Improvements of up to 1% have been observed for correct classification by replacing the Standard Convolutional Layers with the AutoMarkov Layers.

Acknowledgement: This work was equally supported by the Joint Applied Research Projects Intelligent System for Automatic Assistance of Cervical Cancer Diagnosis, grant number: PN-II-PT-PCCA-2013-4-0202, funded by Executive Unit for Higher Education, Research, Development and Innovation Funding (UEFISCDI) and by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Ministry of European Funds through the Financial Agreement POSDRU/159/1.5/S/134398.

REFERENCES

[1] J. Howard, "The business impact of deep learning," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD. ACM, 2013, pp. 1135–1135.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[6] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *Advances in Neural Information Processing Systems 26*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., 2013, pp. 2553–2561.

[7] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," *CoRR*, vol. abs/1312.2249, 2013.

[8] W. Ouyang, P. Luo, X. Zeng, S. Qiu, Y. Tian, H. Li, S. Yang, Z. Wang, Y. Xiong, C. Qian, Z. Zhu, R. Wang, C. C. Loy, X. Wang, and X. Tang, "Deepid-net: multi-stage and deformable deep convolutional neural networks for object detection," *CoRR*, vol. abs/1409.3505, 2014.

[9] Z. Yan, H. Zhang, B. Wang, S. Paris, and Y. Yu, "Automatic photo adjustment using deep learning," *CoRR*, vol. abs/1412.7725, 2014.

[10] J. Xu, L. Jin, L. Liang, Z. Feng, and D. Xie, "A new humanlike facial attractiveness predictor with cascaded fine-tuning deep learning model," *CoRR*, vol. abs/1511.02465, 2015.

[11] J. Li, C. Xiong, L. Liu, X. Shu, and S. Yan, "Deep face beautification," in *Proceedings of the 23rd ACM International Conference on Multimedia*. ACM, 2015, pp. 793–794.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[13] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[14] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[16] A. B. Patel, T. Nguyen, and R. G. Baraniuk, "A probabilistic theory of deep learning," *arXiv preprint arXiv:1504.00641*, 2015.

[17] J. Moussouris, "Gibbs and markov random systems with constraints," *Journal of Statistical Physics*, vol. 10, no. 1, pp. 11–33, 1974.

[18] F. Preston, "Gibbs states on countable sets," *Cambridge University Press*, 1974.

[19] W. Woods, "Two-dimensional discrete markovian fields," *IEEE Transactions on Information Theory*, vol. 18, no. 2, pp. 721–741, 1972.

[20] P. Bremaud, *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*, ser. Texts in Applied Mathematics. Springer, 1999.

[21] C. Ţoca, M. Ciuc, and C. Pătraşcu, "Normalized autobinomial markov channels for pedestrian detection," in *Proceedings of the British Machine Vision Conference (BMVC)*, X. Xie, M. W. Jones, and G. K. L. Tam, Eds. BMVA Press, September 2015, pp. 175.1–175.13.

[22] R. Chellappa and S. Chatterjee, "Classification of textures using gaussian markov random fields," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 33, no. 4, pp. 959–963, 1985.

[23] F. Cohen, Z. Fan, and M. Patel, "Classification of rotated and scaled textured images using gaussian markov random field models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 2, pp. 192–202, 1991.

[24] M. Berthod, Z. Kato, S. Yu, and J. Zerubia, "Bayesian image classification using markov random fields," *Image and Vision Computing*, vol. 14, no. 4, pp. 285 – 295, 1996.

[25] B. Manjunath and R. Chellappa, "Unsupervised texture segmentation using markov random field models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 5, pp. 478–482, 1991.

[26] K. Held, E. Kops, B. Krause, W. Wells, R. Kikinis, and H.-W. Muller-Gartner, "Markov random field segmentation of brain mr images," *Medical Imaging, IEEE Transactions on*, vol. 16, no. 6, pp. 878–886, 1997.

[27] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions and the bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721–741, 1984.

[28] J. Zhang, "The mean field theory in em procedures for blind markov random field image restoration," *Image Processing, IEEE Transactions on*, vol. 2, no. 1, pp. 27–40, 1993.

[29] C. Ţoca, C. Pătraşcu, and M. Ciuc, "Performance testing and functional limitations of normalized autobinomial markov channels," in *Intelligent Computer Communication and Processing (ICCP), 2015 IEEE International Conference on*, 2015, pp. 401–405.

[30] J. Winn and J. Shotton, *Markov Random Fields for Object Detection*. MIT Press, 2011, ch. 25, pp. 389–404.

[31] P. Vernaza and D. D. Lee, "Scalable real-time object recognition and segmentation via cascaded, discriminative markov random fields," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 3102–3107.

[32] A. Ghosh, A. Mondal, and S. Ghosh, "Moving object detection using markov random field and distributed differential evolution," *Applied Soft Computing*, vol. 15, pp. 121–136, 2014.

[33] C. Li and M. Wand, "Combining markov random fields and convolutional neural networks for image synthesis," *CoRR*, vol. abs/1601.04589, 2016.

[34] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr, "Conditional random fields as recurrent neural networks," *CoRR*, vol. abs/1502.03240, 2015.

[35] W. Feller, *Introduction to Probability Theory and Its Applications*. Wiley, 1971, ch. 2, pp. 9–20.

[36] J. Besag, "Spatial interaction and the statistical analysis of lattice systems," *Journal of the Royal Statistical Society*, vol. 36, no. 2, pp. 192–236, 1974.

[37] J. M. Hammersley and P. Clifford, "Markov field on finite graphs and lattices," *Unpublished*, 1971.

[38] A. Agarwal, E. Akchurin, C. Basoglu, G. Chen, S. Cyphers, J. Droppo, A. Eversole, B. Guenter, M. Hillebrand, R. Hoens, X. Huang, Z. Huang, V. Ivanov, A. Kamenev, P. Kranen, O. Kuchaiev, W. Manousek, A. May, B. Mitra, O. Nano, G. Navarro, A. Orlov, H. Parthasarathi, B. Peng, M. Radmilac, A. Reznichenko, F. Seide, M. L. Seltzer, M. Slaney, A. Stolcke, H. Wang, Y. Wang, K. Yao, D. Yu, Y. Zhang, and G. Zweig, "An introduction to computational networks and the computational network toolkit," Microsoft Technical Report MSR-TR-2014-112, Tech. Rep., 2014.

[39] 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.