Using Sequential Context for Image Analysis

António R. C. Paiva[1], Elizabeth Jurrus[1,2] and Tolga Tasdizen[1,3]
[1] *Scientific Computing and Imaging Institute,* [2] *School of Computing, and*
[3] *Department of Electrical and Computer Engineering,*
University of Utah, Salt Lake City, UT, 84112
email: {arpaiva,liz,tolga}@sci.utah.edu

***Abstract*—This paper proposes the sequential context inference (SCI) algorithm for Markov random field (MRF) image analysis. This algorithm is designed primarily for fast inference on an MRF model, but its application requires also a specific modeling architecture. The architecture is composed of a sequence of stages, each modeling the conditional probability of the labels, conditioned on a neighborhood of the input image and output of the previous stage. By learning the model at each stage sequentially with regards to the true output labels, the stages learn different models which can cope with errors in the previous stage.**

***Keywords*-sequential context inference; Markov random fields; conditional random fields; neural networks.**

I. Introduction

Markov random field (MRF) models have been widely used in statistical image analysis [1]. MRF models characterize the joint statistics of the observed image (or its derived features) and the latent parameters of the vision processes. By considering the joint effect in the distribution by spatial neighbors, contextual information and other properties of early vision can be modeled in a convenient and consistent way, while using the Markov assumption to simplify the dependencies of the model.

Inference on the learned MRF models, however, is not straightforward. Inference implies finding the maximum *a posteriori* (MAP) of the model. In general, exact computation through exhaustive search is infeasible due to the combinatorial nature of the search. Algorithms that take advantage of the factorization of the distribution, like belief propagation (BP) [2], cannot be used in this case because they are limited to tree dependencies in the factor graph. The graphs associated with MRF models for images, however, have loops due to the interdependence between neighboring pixels. The typical solution is to resort to approximate computational methods such as loopy belief propagation [2], mean field annealing [1], relaxation labeling [3], iterated conditioned modes (ICM) [4], and fast graph cut methods [5], [6]. These algorithms have several drawbacks, but their two major limitations are their high computational complexity and the need to iterate until convergence, which prevents their use in time-critical applications. Moreover, some algorithms are not even guaranteed to converge [7].

In this paper, the sequential context inference (SCI) algorithm for modeling and inference in MRF image analysis is proposed. The key difference is the use of a sequential architecture of models. Each stage models the posterior distribution of the latent labels, conditioned on a neighborhood of the input image and estimated latent neighbors, utilizing the result of the previous stage as input in place of the values for the latent neighbors. By learning the model at each stage sequentially with regards to the *true output label*, the models learn to cope with errors in the previous stage.

Two conceptually related approaches are stacked graphical learning (SGL) [8] and Tu's auto-context [9]. Both of these methods build a dependency network of models that are used for inference, similar to the sequential architecture utilized by the SCI algorithm. SGL does inference using Gibbs sampling, since it was designed primarily to reduce the computation in Markov chains as the number of labels increases [8]. However, in image analysis, a more pressing problem is the intractability of inference in the presence of loops in the context estimation, which leads to the use of a different formulation for modeling and inference in the SCI algorithm. Auto-context is a boosting strategy that sequentially learns classifiers from local filters applied to the observed image and the result of the previous classifier. Because a classifier utilizes the result of the previous classifier, it effectively congregates information from an increasingly larger region in the input image, thus building context. However, the feature filters must be given a priori, or selected from a large filter bank of features. Thus, this approach is computationally expensive and may not achieve the best possible solution. In contrast, the models for the SCI algorithm learn the probability distribution, rather than a classifier, directly from image samples, implicitly finding the relevant features.

II. Sequential Context Inference

Consider an input image $X = \{x_i : i \in \Omega\}$, where x_i denotes the feature vector for the ith pixel, and Ω is the image lattice. The neighborhood of the ith pixel of X is $\mathbf{x}_i \equiv \{x_j : j \in \mathcal{N}_i^x\}$. For modeling, a supervised learning strategy will be utilized. Hence, to be able to learn the models, we will consider that a corresponding label image, denoted $Y = \{y_i : i \in \Omega, y_i \in \mathcal{L}\}$, where $\mathcal{L} = \{l_0, \ldots, l_{C-1}\}$ is the set of C possible labels, is available during training. Similarly, the neighborhood of the ith pixel in Y is denoted $\mathbf{y}_i \equiv \{y_j : j \in \mathcal{N}_i^y\}$, with $\mathcal{N}_i^y$ a neighborhood system such that $i \notin \mathcal{N}_i^y$. Clearly, the neighborhood systems $\mathcal{N}^x$ and $\mathcal{N}^y$, for X and Y, may be different. Specifically, note that the center pixel is required not to be included by $\mathcal{N}^y$.

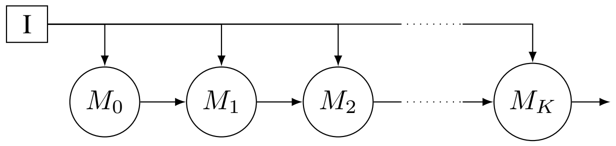


Fig. 1: Diagram of the sequential architecture utilized by the SCI algorithm. 'I' denotes the input image, and the M_i's are the models at each stage.

The goal of inference is to compute the latent image Y^* that achieves the MAP given X. That is,

$$\begin{aligned}&\max_Y P(Y|X)\\ &\quad= \max_Y P(y_i|Y\setminus\{y_i\},X)P(Y\setminus\{y_i\}|X)\\ &\quad= \max_{y_i}\left[\max_{Y\setminus\{y_i\}} P(y_i|\mathbf{x}_i,\mathbf{y}_i)P(Y\setminus\{y_i\}|X)\right],\end{aligned} \tag{1}$$

for any $i \in \Omega$, and using the Markov assumption with regards to the observed image and configuration neighbors in the last equality. This equation tells us that, given the optimum values of the neighbors $\mathbf{y}_i$ of the ith pixel in the underlying configuration, the most likely value of y_i can be obtained through local optimization. This result is the fundamental idea behind the ICM algorithm [4] and belief propagation [2]. The critical limitation, however, is that during inference, the optimum values of the neighbors are unknown. Hence, this approach cannot be applied directly. The SCI algorithm employs a multi-stage sequential architecture, depicted in Fig. 1, to circumvent this problem. Each stage models the probability distribution of the labels, given the observed input image and the probability output of the previous stage. Then, for inference, the sequential architecture is basically "played back," with the stages being applied sequentially. The key difference is that by learning *different models with regards to the true labels* the models learn to cope and correct errors from previous stages.

Thus, each stage of the sequential architecture learns the conditional distribution $P(y_i|\mathbf{x}_i,\mathbf{y}_i)$. The output of each stage is a distribution over the possible label values. Rather than computing the label assignment after each stage, formulating the optimization directly on these probabilities is advantageous because more information is preserved. For continuous optimization, the labels must be written directly as a vector of probabilities over the possible label values. This is similar to the assignment in mean field annealing [3]. To be unambiguous, each y_i can only be one of the vectors $[1,0,\ldots,0]$, $[0,1,0,\ldots,0]$,...,$[0,\ldots,0,1]$, corresponding to labels $l_0,l_1,\ldots,l_{C-1}$.

For optimization over C labels, the kth stage of the sequential architecture learns the vector function,

$$\begin{aligned}y_i^k &= M_k(\mathbf{x}_i,\mathbf{y}_i^{k-1})\\ &= \begin{bmatrix}P_k(y_i^*=[1,0,\ldots,0]|\mathbf{x}_i,\mathbf{y}_i^{k-1})\\ \vdots\\ P_k(y_i^*=[0,\ldots,0,1]|\mathbf{x}_i,\mathbf{y}_i^{k-1})\end{bmatrix},\end{aligned} \tag{2}$$

Algorithm 1 Inference on the SCI algorithm.

```
• Initialization: compute/extract the observed image input vectors $\{\mathbf{x}_i : i \in \Omega\}$.
• For each stage $k = 0,1,\ldots,K$,
    – Compute,
          $y_i^k = M_k(\mathbf{x}_i,\mathbf{y}_i^{k-1})$.
    – If $k < K$, form the vectors with all the probabilities from the configuration neighbors of each pixel, $\{\mathbf{y}_i^k : i \in \Omega\}$.
• Assign each $y_i^K$ to the label with maximum probability.
```

with $\mathbf{y}_i^0 = \emptyset$, meaning that no context is utilized, corresponding to

$$y_i^0 = M_0(\mathbf{x}_i) = \begin{bmatrix}P_0(y_i^*=[1,0,\ldots,0]|\mathbf{x}_i)\\ \vdots\\ P_0(y_i^*=[0,\ldots,0,1]|\mathbf{x}_i)\end{bmatrix}, \tag{3}$$

where y_i^* is the *true* label of the ith pixel. In practice, only $C-1$ probabilities are needed because the probabilities over the possible labels must sum to one. To map back to labels of $\mathcal{L}$, the pixel are assigned to the label with corresponding largest probability.

The models are learned sequentially, meaning that the model for the first stage is learned first, then the model for second stage, and so forth. Sequential training must be utilized because the output of the $(k-1)$ stage is needed as input to train the subsequent model at the kth stage, as shown in (2).

To learn the model at each stage, in this paper, we utilize multi-layer perceptron (MLP) artificial neural networks (ANN) [10]. MLP-ANNs have been shown to be universal function approximators, if the networks have at least one hidden layer with enough processing elements (PEs) [11], [12]. Note that the SCI algorithm is not specifically coupled with MLPs or, even more generally, neural networks. In fact, *any* universal function approximator can be utilized. The MLP is chosen here because it is very compact and computationally efficient once the parameters of the network have been learned.

Two important design choices are the number of stages and the configuration of each model. The number of stages can be determined by evaluating the improvement in classification accuracy after each stage. Training is stopped if the improvement is considered minimal compared to the effort of training another model. The choice of the configuration of each model involves the usual trade-offs in adaptive systems' training. In the case of neural networks specifically, one must choose the number of hidden PEs and learning rate by ensuring that the system is converging and is not overfitting the data based on the results of the first stage.

The primary advantage of the SCI algorithm is its ease of inference. Given the input neighborhood context features $\mathbf{x}_i$ and neighborhood context in the estimated latent image obtained from the previous stage, the models evaluate the output probability directly. Inference is then simply a matter of applying the models sequentially, as depicted in Fig. 1 and outlined in Algorithm 1. Note that training the SCI algorithm

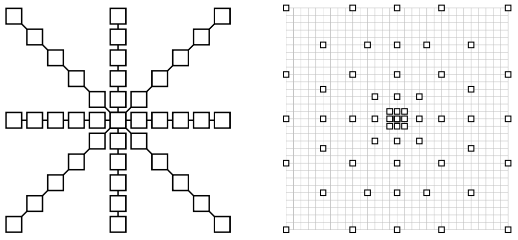


Fig. 2: Stenctil neighborhoods used for the (left) texture and (right) horse segmentation.

follows the same procedure as in Algorithm 1, with the computing step corresponding to training of each model.

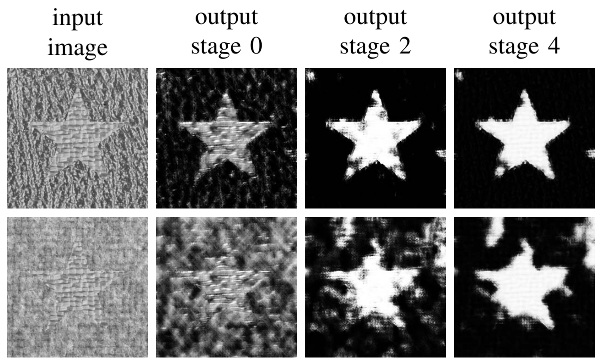
III. Experimental Results

To illustrate the SCI algorithm, results are shown for texture segmentation and segmentation of the Weizmann horse dataset [13]. In the first case, the dataset comprised 20 images, each with two textures: a background texture, and a foreground texture delimited by a star-shape (cf. Fig. 3(a)). Each image has size 256×256 pixels. The images were obtained from all pairwise combinations of a set of 4 background and 5 foreground textures. The Weizmann horse dataset consists of 328 grayscale horse images with different sizes. In both cases the images were randomly divided into training and testing sets of equal size.

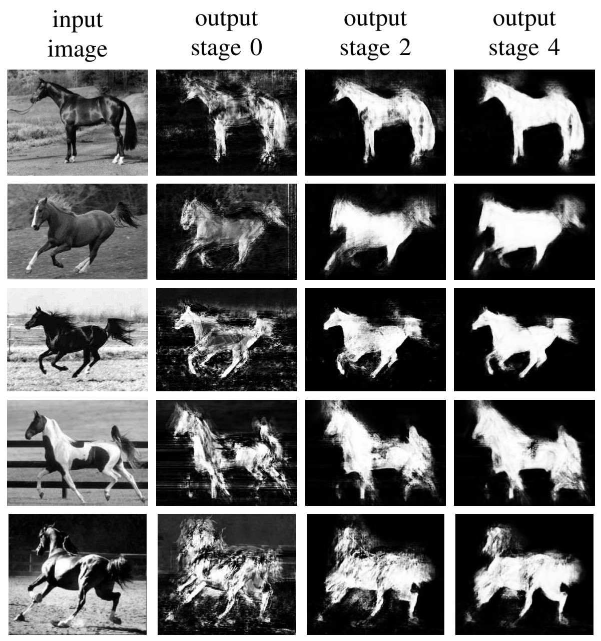
In both cases, each model of the sequential architecture was learned using an MLP-ANN. For texture segmentation, we utilized an MLP with one hidden layer of 20 PEs, trained using back-propagation with a stepsize of 0.001 and a momentum parameter of 0.5. For horse segmentation, we used 30 PEs and stepsize 0.0001. Five Monte Carlo instances were used per network to minimize problems with local optima. Stopping of training was decided by measuring the performance on a cross-validation set of 20% of the training images. The remaining images were used for testing. The input vectors were formed by sampling the input image and output of the previous stage with the stencil neighborhood given in Fig. 2, without the center pixel in the latter case.

The inference results on two texture segmentation test images are shown in Fig. 3(a). The second image shown is one of the worst cases encountered, where it is clearly noticeable that the labeling based on the output of the first model, i.e., stage 0, would yield very poor results. However, as one advances through the stages, the probability image progressively approaches the true segmentation. At the end of the last stage, the star shape is easily discernible, which is a remarkable improvement to the output of the first model. These observations can be verified quantitatively from the average F-value curves shown in Fig. 4(left), which quantify the performance for different classification thresholds. It is clearly noticeable the systematic improvement of the F-value with each inference stage. Moreover, in this case, applying the SCI algorithm largely compensates for the high sensitivity to the threshold value observed in the initial stages.

The test results on the horse segmentation experiment are shown in Fig. 3(b). Visually, the results are comparable to Tu's auto-context [9]. Note that in our case the complexity increases with the context because the feature filters are learnt from data. In auto-context, however, the filters are given and can have very large support, but one needs very large sets of filters to ensure a reasonable basis. In spite of that, our results are remarkable considering that we utilized only 49 features, 25 from the input image and 24 from the configuration neighbors, whereas Tu [9] selects from a set of around 12000, 8000 from the input image and 4000 from the previous classification output. This indicates that feature filters should be learned from data to ensure that all available information is utilized and they accurately fit the application. Again, the averaged F-value curves shown in Fig. 4(right) verify our observations quantitatively. Our best F-value in the test set is 0.834 (0.823 at zero threshold) compared to below 0.84 using auto-context [9]. Finally, these results highlight once more the importance of sequentially applying the different models. As shown in the images, the first inference stages detect primarily the contours

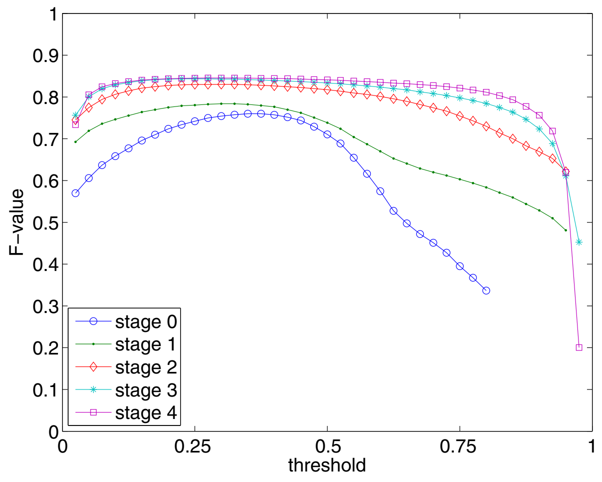


(a) Texture segmentation.



(b) Horse segmentation.

Fig. 3: Testing segmentation results using the SCI algorithm.



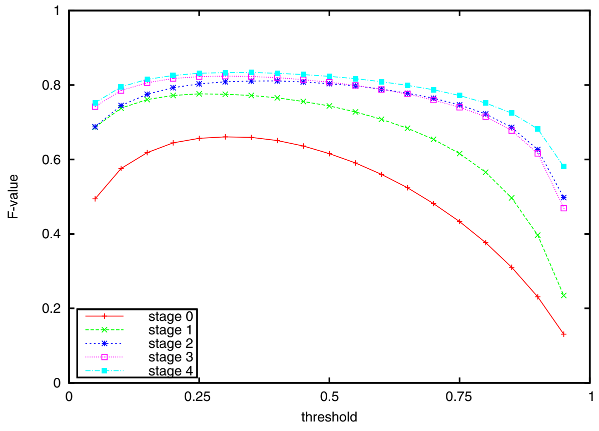


Fig. 4: F-value curves of the classification obtained from the output of stage 4, averaged over all test images, for the (left) texture and (right) horse segmentation.

of the horses which are easier to infer locally but fails to discern the body of the horse because the local information is confusing. However, using the context information from the output of the previous models in addition to the input image, the later stages are able to fill in the segmentation. Note that the context information increases implicitly with each stage because of the context on the output of the previous model.[1]

Learning the sequential architecture is a slow process due to the use of back-propagation. Training all the models of the sequential architecture took several hours for the texture segmentation experiment, and almost 2 weeks for the horse segmentation experiment. (Note that, in the latter case, the training data comprises more than 10 million training feature vectors.) On the other hand, inference is very fast taking less than 0.25 seconds per image, compared to the 40 seconds reported by Tu [9].

[1]This is similar in principle to the fact that a 3×3 filter applied twice has a 5×5 support.

IV. Conclusion

This paper presents the SCI algorithm as a general approach for MRF image modeling and inference. The main advantage of the SCI algorithm is that it allows very fast inference by using a determined number of stages. Thus, the SCI algorithm represents a major improvement in inference speed compared to iterative MRF inference algorithms, such as loopy BP [2], mean field inference [1], and relaxation labelling [3]. This characteristic allows the SCI algorithm to be utilized in applications where, currently, the computational complexity and the iterative or stochastic nature of current inference algorithms prevents the use of MRF models. In addition, there is no need to specify the feature filters since these are learned from data. Furthermore, as shown in the results, learning the filters from data allows for a smaller number of features to be utilized, leading to compact and more computationally efficient models without significantly reducing the performance.

Fast inference in the SCI algorithm comes at a price. Instead of learning only one MRF joint distribution model, one must learn several conditional distributions. This increases considerably the computational cost of modeling. Nevertheless, considering that in most practical applications modeling is a one-time process and can be done offline, this seems a beneficial trade-off in many cases. Still, in the future we plan to explore strategies to reduce the modeling complexity.

Acknowledgements

This work was supported in part by NIH RO1 EB005832.

References

[1] S. Z. Li, *Markov Random Field Modeling in Image Analysis*. Springer, 2001.

[2] J. Pearl, *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. San Francisco, CA: Morgan Kaufmann, 1988.

[3] S. Z. Li, H. Wang, and K. L. Chan, "Minimization of MRF energy with relaxation labeling," *Journal of Mathematical Imaging and Vision*, vol. 7, no. 2, pp. 149–161, Mar. 1997.

[4] J. Besag, "On the statistical analysis of dirty pictures," *Journal of the Royal Statistical Society – Series B*, vol. 48, no. 3, pp. 259–302, 1986.

[5] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–2139, Nov. 2001.

[6] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147–159, Feb. 2004.

[7] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: an empirical study," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, Stockholm, Sweden, 1999, pp. 467–475.

[8] Z. Kou and W. W. Cohen, "Stacked graphical models for efficient inference in Markov random fields," in *Proceedings of the SIAM International Conference on Data Mining*, Minneapolis, Minnesota, Apr. 2007.

[9] Z. Tu, "Auto-context and its application to high-level vision tasks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, USA, Jun. 2008.

[10] S. Haykin, *Neural networks - A comprehensive foundation*, 2nd ed. Prentice-Hall, 1999.

[11] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1992.

[12] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, Dec. 1989.

[13] E. Borenstein, E. Sharon, and S. Ullman, "Combining top-down and bottom-up segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Washington, D.C., USA, Jun. 2004.