# Scene Labeling with LSTM Recurrent Neural Networks

Wonmin Byeon[1 2] Thomas M. Breuel[1] Federico Raue[1 2] Marcus Liwicki[1]

[1] University of Kaiserslautern, Germany.
[2] German Research Center for Artificial Intelligence (DFKI), Germany.

{wonmin.byeon,federico.raue}@dfki.de {tmb,liwicki}@cs.uni-kl.de

## Abstract

*This paper addresses the problem of pixel-level segmentation and classification of scene images with an entirely learning-based approach using Long Short Term Memory (LSTM) recurrent neural networks, which are commonly used for sequence classification. We investigate two-dimensional (2D) LSTM networks for natural scene images taking into account the complex spatial dependencies of labels. Prior methods generally have required separate classification and image segmentation stages and/or pre- and post-processing. In our approach, classification, segmentation, and context integration are all carried out by 2D LSTM networks, allowing texture and spatial model parameters to be learned within a single model. The networks efficiently capture local and global contextual information over raw RGB values and adapt well for complex scene images. Our approach, which has a much lower computational complexity than prior methods, achieved state-of-the-art performance over the Stanford Background and the SIFT Flow datasets. In fact, if no pre- or post-processing is applied, LSTM networks outperform other state-of-the-art approaches. Hence, only with a single-core Central Processing Unit (CPU), the running time of our approach is equivalent or better than the compared state-of-the-art approaches which use a Graphics Processing Unit (GPU). Finally, our networks' ability to visualize feature maps from each layer supports the hypothesis that LSTM networks are overall suited for image processing tasks.*

## 1. Introduction

Accurate scene labeling is an important step towards image understanding. The scene labeling task consists of partitioning the meaningful regions of an image and labeling pixels with their regions. Pixel labels can (most likely) not only be decided with low-level features, such as color or texture, extracted from a small window around pixels. For instance, distinguishing "grass" from "tree" or "forest" would prove tedious under such a setting. As a matter of fact, humans perceptually distinguish regions via the spatial dependencies between them. For instance, visually similar regions could be predicted as "sky" or "ocean" depending on whether they are on the top or bottom part of a scene.

Consequently, a higher-level representation of scenes (their global context) is typically constructed based on the similarity of the low-level features of pixels and on their spatial dependencies using *a graphical model*. The graphical models construct the global dependencies based on the similarities of neighboring segments. The most popular graph-based approaches are Markov Random Fields (MRF) [4, 15, 16, 25] and Conditional Random Fields (CRF) [10, 20]. However, most such methods require pre-segmentation, superpixels, or candidate areas.

More recently, *deep learning* has become a very active area of research in scene understanding and vision in general. In [23], color and texture features from oversegmented regions are merged by Recursive Neural Networks. This work has been extended by Socher et al. [22] who combined it with convolutional neural networks. Among deep learning approaches, *Convolutional Neural Networks* (CNNs) [17] are one of the most successful methods for end-to-end supervised learning. This method has been widely used in image classification [14, 21], object recognition [12], face verification [24], and scene labeling [5, 3].

Farabet et al. [3] introduced multi-scale CNNs to learn scale-invariant features, but had problems with global contextual coherence and spatial consistency. These problems were addressed by combining CNNs with several post-processing algorithms, i.e., superpixels, CRF, and segmentation trees. Later, Kekeç et al. [13] improved CNNs by combining two CNN models which learn context information and visual features in separate networks. Both mentioned approaches improved accuracy through carefully designed pre-processing steps to help the learning, i.e., class frequency balancing by selecting the same amount of random patches per class, and specific color space for the input data.

In order to improve modeling of long range dependencies, Pinheiro et al. [19] first revealed the use of large input patches to consider larger contexts. However, that would reduce the resolution of the final label image, and a huge redundancy of overlapping regions would make the learning inefficient. Hence, they introduced Recurrent Convolutional Networks (RCNNs) for scene labeling. RCNNs train different input patch sizes (the instances) recurrently to learn increasingly large contexts for each pixel, whilst ensuring that the larger contexts are coherent with the smaller ones.

In this work, we investigate a two-dimensional (2D) Long Short Term Memory (LSTM) recurrent neural network architecture to take into account the local (pixel-by-pixel) and global (label-by-label) dependencies in a single process for scene labeling. We can skip any additional processing or conditions like multi-scale or different patch sizes to solve the scene labeling task with the least human or machine's effort. LSTM recurrent neural networks [11] were originally introduced for sequence learning. These networks include recurrently connected cells to learn the dependencies between two time frames, then transfer the probabilistic inference to the next frame. The LSTM memory block stores and retrieves this information over short or long periods of time.

LSTM networks have been successfully applied to many tasks such as handwriting [8] and speech recognition [6]. They were extended to multi-dimensional learning which improved handwriting systems [9]. Image processing task were also considered with the multi-dimensional model [7, 2, 1]. This work demonstrated their utility for image labeling, binarization or texture analysis, but only on very simple data (simple digit images, graphical or texture data).

Our work builds on the foundations laid in [7]. However, we focus particularly on natural scene image labeling. RNN-based approaches are known to be difficult to train with high noise and large data. Particularly in large data, long-term dependencies are vanished while the information is accumulated by the recurrence. We now support large scene images which contain huge variations of instances for each label and which can have an enormous size. We show how LSTM networks can be generalized well to any vision-based task and efficiently learnt without any task-specific features. LSTM networks also automatically learn short and long-range contextual information by end-to-end entirely supervised training. The advantages of our work can be summarized as follows:

- Our model learns long range dependencies on their own in an efficient way without additional process or task-specific feature. In the network, the local prediction is influenced by its neighboring contexts which contain previous spatial dependencies. Therefore, each local prediction is implicitly affected by the global contextual information of the image. The benefits of the ability to learn local and global dependencies through the networks make the task completely end-to-end supervised learning.

- We avoid any hand-crafted feature extraction, multi-scale input pyramids, pre-, or post-processing, without performance penalty. The networks automatically learn relevant features and contexts of each label only from raw RGB values with only a small number of weights. This simpler architecture keeps the computing time per image low, and our system does not include any extra post-processing techniques — such additional steps would improve accuracy by a few extra percents, but often taken very long to process, e.g., Farabet et al. [3] reported an accuracy improvement of 1.4% at the expense of their CNN+CRF algorithm being 100 times slower.

- The choice of architecture or parameters for scene labeling (and vision-based tasks in general) are much simpler compared to other deep network approaches.

- The total number of weights in our networks is much smaller compared to other deep network approaches, but the performance is maintained. Our total training time was less than 10 days, and testing took around 1 second only per image on a single-core CPU, which is reasonable for practical applications. It can still be improved dramatically by implementing a GPU version.

- All above advantages indicate that our system needs a very little human effort and computational complexity to achieve near state-of-the-art performance.

- The internal feature activations were easily visualized and show what is learnt from images and how it is learnt. As remarked in [26], visualizing features is crucial for neural-network-based approaches to intuitively understand the learning process, but challenging after the first layer since activations of networks does not correspond to the pixel space. Visualization also suggests that LSTM networks can be applied well to vision-based tasks in general.

## 2. 2D LSTM Recurrent Neural Networks for Scene Labeling

In this paper, the networks are divided into the three main layers: input layer, hidden layer, and output layer. The hidden layer consists of 2D LSTM layer and feedforward layer, and is stacked as deep networks. The architecture of 2D LSTM networks is illustrated in Figure 1.

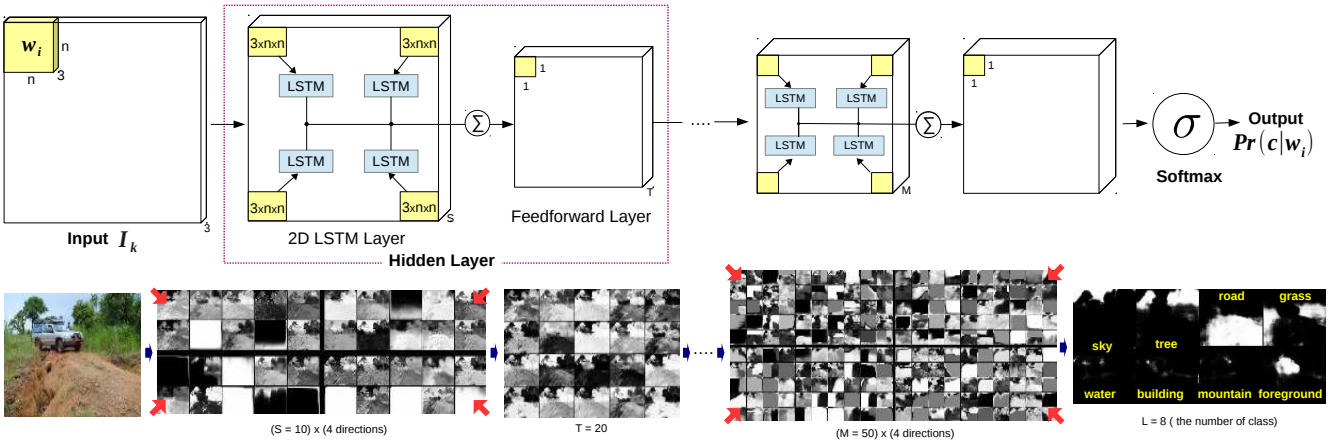


Figure 1: 2D LSTM network architecture. A input image $I_k$ is divided into non-overlapping windows $w_i$ (a grid) sized $n \times n$. Each window with RGB channels ($3 \times n \times n$) are fed into four separate LSTM memory blocks. The current window of LSTM block is connected to its surrounding directions x and y, i.e., left-top, left-bottom, right-top, and right-bottom; it propagates surrounding contexts. The output of each LSTM block is then passed to the Feedforward layer, where sums all directions and squash it by the Hyperbolic tangent ($tanh$). At the last layer, the outputs of the final LSTM blocks are summed up and sent to the *softmax* layer. Finally, the networks output the class probabilities ($Pr(c|w_i)$) for each input window. The bottom images are corresponding outputs for each layer.

## 2.1. Input Layer

The input can be one RGB pixel ($3 \times 1 \times 1$) or window ($3 \times n \times n$). With pixel-wise input, the network is able to give each pixel a precise label, but the sequence of pixels can become very long if the size of the image is big. Besides, in large images adjacent pixels tend to have similar values. It means the dependencies within the pixels is redundant. Therefore, non-overlapping windows were used as input — the input images were split into grids. The window-wise input combines the local correlation of the pixels but maintains global coherence of the image. Unlike previous tasks with 2D LSTM networks, scene images are big and high quality. Thus, the window integrates the local context and reduces the total sequence size. It will help both the localization, and the keeping of longer range dependencies without losing global context. Moreover, it will speed-up the inference process. The size of the final prediction is reduced, but can be easily upscaled by some interpolation method (e.g., cubic interpolation) and does not influence the final results much since the same label may occur several times within a small window. In Section 2.3, we will introduce a probabilistic target coding scheme to handle multiple labels in one window.

## 2.2. Hidden Layer

The hidden layer includes 2D LSTM layer (four LSTM memory blocks) and feedforward layer. The activations from the 2D LSTM layer represent the surrounding context in all directions and are combined in the feedforward layer.

**2D LSTM Layer:** LSTM is a subnet that allows to easily memorize the context information for long periods of time in sequence data. In images, this temporal dependency learning is converted to the spatial domain. The subnet includes three gates: the input gate ($i$), the forget gate ($f$), and the output gate ($o$) which overwrite, keep, or retrieve the memory cell $c$ respectively at the position $t$. $W$, $H$, and $C$ are weight matrices for input to gates, recurrent connections, and cell to gates. $h^x_{t-1}$ and $h^y_{t-1}$ are the output activation from previous direction x and y. $d$ and $d'$ indicate x or y direction, and $f_1(\cdot)$ and $f_2(\cdot)$ are logistic sigmoid ($sigm$) and hyperbolic tangent ($tanh$), respectively.

First, input gate ($i_t$) and forget gates for the x and y directions ($f^x_t$ and $f^y_t$) are computed by

$$i_t = f_1(W_i \cdot a_t + \textstyle\sum_d (H^d_i \cdot h^d_{t-1} + C^d_i \cdot c^d_{t-1}) + b_i)$$

$$f^{d'}_t = f_1(W_f \cdot a_t + \textstyle\sum_d (H^d_f \cdot h^d_{t-1}) + C^{d'}_f \cdot c^{d'}_{t-1} + b^{d'}_f)$$

After, the current memory cell ($c_t$) is updated by forgetting the previous contents ($c^x_{t-1}$ for x and $c^y_{t-1}$ for y direction) and including the new memory ($\widetilde{c_t}$)

$$\widetilde{c_t} = f_2(W_{\widetilde{c_t}} \cdot a_t + \textstyle\sum_d (H^d_{\widetilde{c_t}} \cdot h^d_{t-1}) + b_{\widetilde{c_t}})$$

$$c_t = \textstyle\sum_d (f^d_t \circledcirc c^d_{t-1}) + i_t \circledcirc \widetilde{c_t}$$

At the end, the final activation at the current position ($h_t$) is calculated with the output gate ($o_t$) which regulates the amount of information to be output.

$$o_t = f_1(W_o \cdot a_t + \sum_d (H_o^d \cdot h_{t-1}^d) + C_o \cdot c_t + b_o)$$

$$h_t = o_t \circledcirc f_2(c_t)$$

**Feedforward Layer:** Feedforward layer includes fully connected layer with a nonlinear activation function and the summation of the LSTM layer activations from all directions. From the last layer, each LSTM memory block scans on all directions (left-top, left-bottom, right-top, and right-bottom, see Figure 1), then the outputs are combined together in this layer. For mid-level hidden layers, outputs from each LSTM memory block are convoluted, and squashed by the hyperbolic tangent (tanh). These results are used as an input in the next level. This layer acts as a feature mapping from 2D LSTM layer, so the amount of features from all contextual information on the image is generated and combined together in this layer. The size of the layer corresponds to the number of feature maps — the bigger size of feedforward layer create more features. As can be seen in Figure 2, more detailed features are created in lower levels and more abstract and complex features at higher levels with global contexts. At the end, at the final level, all features from all directions are accumulated and sent to the output layer.

### 2.3. Output Layer

The outputs from the last hidden layer is normalized with the *softmax* function:

$$Pr(c|w_i) = \frac{e^{a_c(w_i)}}{\sum_{l \in \{1,\ldots,L\}} e^{a_l(w_i)}}$$

where $a_c(w_i)$ is the final activation of the input window $w_i$ obtained from the last hidden layer for corresponding target $c$, and $l$ is the one of target classes.

Our goal is to find the maximum likelihood of all training samples. As an objective function, we apply the negative log probability, i.e., cross entropy error function:

$$E = -\sum_{c=1}^{C} z_c \; ln \; Pr(c|w_i) \qquad (1)$$

where $z_c \in \{0,\ 1\}$. $Pr(c|w_i)$ is the predicted probability of the class $c$.

**Probabilistic Target Coding:** To compare the multi-class probability ($Pr(c|w_i)$) with the target $c$ for errors, a *1-of-K coding scheme* is typically used, i.e., a binary vector with all elements set to zero except one which corresponds to the correct class. It encodes the desired output — In Eq. ( 1), $z_c$ is decided by the encoding vector. However, in our scenario, the size of the target vector is the same as the size of input window, $n \times n$. Thus, the target is not a single class, but a vector of size $n \times n$ which can contain more than one class. We quantify errors within each window containing

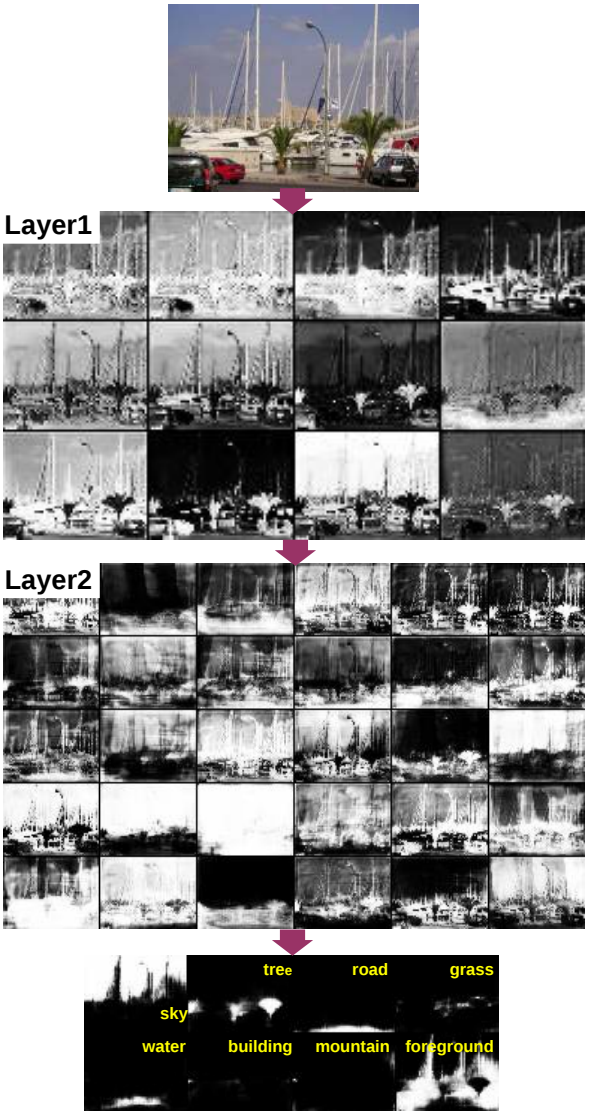


Figure 2: Visualization of feature maps in each layer. The activations are sampled after the convolutional summing of four LSTM blocks. Each activation from each input window is projected down to the image space. Each image represents the features from each hidden node of the corresponding layer. Note that a lighter color represents higher activations.

multiple classes in their target vector by applying a *probabilistic target coding scheme*. Finally, the error function Eq. ( 1) is modified using the probability of an occurrence at class $c$.

$$E = -\sum_{c=1}^{C} \frac{f_c}{n \times n} \; ln \; Pr(c|w_i) \qquad (2)$$

where $f_c$ is the frequency of occurrence for class $c$ in $w_i$,

Table 1: Pixel and averaged per class accuracy comparison on the Stanford Background dataset (in %). Approaches which include pre- or post-processing, and/or multi-scale pyramids are not reported here as they cannot be directly compared. LSTM networks lead to high performance with a very fast inference time on CPU. Balancing the class frequencies of input images would improve the class-average accuracy, but is not realistic for scene labeling in general. The performance of recurrent CNNs (RCNNs) reported here is from two instances. For more details, see Section 3.1. CT indicates the averaged computing time per image

| Method | Pixel Acc. | Class Acc. | class frequency | CT (sec.) | # parameters |
|---|---|---|---|---|---|
| Superparsing 2010 [25] | 77.5 | - | - | 10 to 300 | - |
| Singlescale ConvNet 2013 [3] | 66 | 56.5 | balanced | 0.35 (GPU) | - |
| Augmented CNNs 2014 [13] | 71.97 | 66.16 | balanced | - | 701K |
| Recurrent CNNs 2014 [19] | 76.2 | 67.2 | unbalanced | 1.1 (GPU) | - |
| LSTM networks (window 5×5) | 77.73 | **68.26** | unbalanced | **1.3** (CPU) | 173K |
| LSTM networks (window 3×3) | **78.56** | 68.79 | unbalanced | 3.7 (CPU) | 155K |

Table 2: Pixel and averaged per class accuracy comparison on the SIFT Flow dataset (in %). For performance comparison, approaches which include pre- or post-processing, and/or multi-scale pyramids are not reported here (For the SIFT Flow dataset, we compared multi-scale ConvNet as no single-scale version is reported in [3]). LSTM networks lead to high performance with a very fast inference time on CPU. Balancing the class frequencies of input images would improve the class-average, but is not realistic for scene labeling in general. The performance of RCNNs reported here is with two instances. For more details, see Section 3.1. CT indicates the averaged computing time per image.

| Method | Pixel Acc. | Class Acc. | class frequency | CT (sec.) | # parameters |
|---|---|---|---|---|---|
| Multi-scale ConvNet 2013 [3] | 67.9 | 45.9 | balanced | - | - |
| Augmented CNNs 2014 [13] | 49.39 | 44.54 | balanced | - | 1225K |
| Recurrent CNNs 2014 [19] | 65.5 | 20.8 | unbalanced | - | - |
| LSTM networks (window 5×5) | 68.74 | **22.59** | unbalanced | **1.2** (CPU) | 178K |
| LSTM networks (window 3×3) | **70.11** | 20.90 | unbalanced | 3.1 (CPU) | 168K |

and $n \times n$ is the size of $w_i$.

## 3. Experiments

**Datasets:** Our approach has been tested on two fully labeled outdoor scene datasets: the Stanford Background dataset [4] and the SIFT Flow dataset [18]. The Stanford Background dataset contains 715 images composed of 8 common labels chosen from existing public datasets (572 images used for training, the rest for testing). Note that one of the labels, "foreground" contains unknown objects. Each image has a different resolution (on average $320 \times 240$). The SIFT Flow dataset is more challenging with 2688 images of $256 \times 256$ pixels each. The dataset is split into 2488 images used for training and the rest for testing. The images include 33 semantic labels labeled by LabelMe users.

We report both pixel accuracy and class-average accuracy to compare our performance with other approaches. The pixel accuracy measures the ratio of true positive over all pixels, and class-average accuracy averages all class accuracies using the equal weight for all classes. The measure of class-average accuracy has more impact when the classes are imbalanced in the test images (which is common on outdoor scene images as most scenes contain "sky" but not e.g., "tree").

All of LSTM network models in our experiment have three layers. The second and third layer have 20 LSTM with 30 units in the feedforward layer and 50 LSTM (no feedforward layer at the final level) respectively. In the first layer, the hidden size was decided based on the size of window, i.e., 4 LSTM with 10 units in the feedforward layer for $3 \times 3$ input windows and 10 LSTM with 12 units in the feedforward layer for $5 \times 5$ input windows. At the end, the output of the networks was reduced by a factor of window-size which is up-scaled with cubic interpolation for final evaluation. Online gradient descent was used for training with a learning rate of $10^{-6}$ and a momentum of $0.9$.

### 3.1. Results and Comparisons

Table 1 compares the performance of LSTM networks with current state-of-the-art methods on the Stanford Background dataset, and Table 2 compares the performance on the SIFT Flow dataset. Note that we did not consider other methods which include pre- or post-processing, and multi-scale version in the tables to make the comparison as fair as possible. However, our single-scale networks are still comparable to the multi-scale versions of state-of-art-methods, e.g., pixel accuracy of 78.8% for multi-scale CNNs [3],

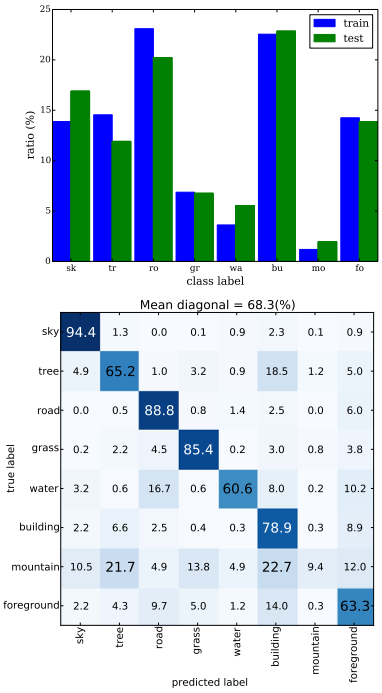


Figure 3: Class frequency distribution (top) and confusion table on the Stanford Background dataset (bottom). As can be observed, the percentage of misclassification labels is correlated with the frequency of the class in the dataset, e.g., The training and testing sets contain only 2% of "mountain" labels and less than 5% of "water" and "grass" labels. As a consequence, "mountain" was confused with "tree" and "building", "water" with "road", and 'tree" with "building". From these mislabeling results, we can observe that the wrong predictions were often caused in the label prediction level but not the segmentation level (some examples are shown in Figure 5).

76.36% for multi-scale augmented CNNs [13], and 78.56% for single-scale LSTM networks. Multi-scale CNNs with further post-processing, i.e., superpixel, CRF, and segmentation tree, slightly improve the accuracy by a few more percents, but are around 15-100 times slower than without post-processing.

With RCNNs, two instances were considered for the accuracy comparison. Note that higher network instances increase the context patch size to correct a final prediction. An increase in the number of instances will maintain the capacity of the system constant (since the network weights are shared), but it will cause a dramatic growth in training time. The testing time was reported as increasing tenfold when one more instance was added on the network. With pixel accuracy, LSTM networks performed about 2% better compared to RCNNs with two instances, but around 2% worse than RCNNs with three instances on the Stanford Background dataset. On the SIFT Flow dataset, the accuracy of LSTM networks was around 4% higher than RCNNs with two instances, but around 7% lower than with three instances. However, the differences for average per class accuracy were less than 1% with both two and three instances on all datasets. Overall, our network model is efficient in training and testing — LSTM networks do not need time-consuming computations to combine long-range context information. Our approach achieved results higher than compared methods without extra effort in an end-to-end manner.

### 3.2. Result Analysis

The confusion matrix on the Stanford Background dataset is reported in Figure 3. "mountain" is the hardest class (only 9.5% class accuracy), but this is explained by the extremely small size of the training and testing sets compared to other classes (see the class frequency distribution in Figure 3). Note that other methods solved this issue by balancing class frequencies, yet doing so is not realistic. The most often confused labels are "mountain" with "tree" or "building", and "water" with "road'. It is clear from class frequency distribution in Figure 3 that "mountain" and "water" have the least frequent samples in both the training and testing sets. The visual labeling results of confused classes will be shown in Figure 5. These are mostly well-segmented but mislabeled.

Selected examples of labeling results from the Stanford dataset are shown in Figure 4. Our approach yields very precise pixel-wise labeling. Figure 5 shows an example of misclassification compared to the ground-truth. The results in the first and second row were mislabeled in the ground-truth image (human mistakes), but correctly classified by LSTM networks. The misclassification regions from the third row mostly include very foggy mountains, so is not so visible even to the human eyes. The results from the fourth to sixth rows show a precise segmentation but an incorrect labeling (because of the ambiguity of the label's characteristics). All of these examples reflect difficulties of the datasets and task.

## 4. Conclusion

We have presented a completely learning-based approach for scene labeling. LSTM networks learn the neighboring context information of every pixel and internally model the global dependencies between labels using recurrent connections. This architecture is simple and well-adapted for natural scene images. Our experiments show performance gains without any hand-crafted features, pre- or post-processing techniques, and multi-scale pyramids from input images. Moreover, our architecture has a (comparatively) fast training and testing time on a single-core CPU, as it uses a smaller number of parameters than other deep-learning approaches.

Figure 4: The results of scene labeling on the Stanford Background dataset. First row: input image; Second row: ground-truth; Third row: predicted image. Colors on images indicate labels — identical colors on ground-truth and predicted images indicate a correct labeling

As future work, we will investigate solutions for the failure cases shown in Figure 5. A possible direction will be the use of a chromatic opponent to avoid the confusion of ambiguous labels by increasing the contrast among pixels (the limitation of RGB input). These mistakes are caused by the behavior of LSTM networks, i.e., the context information is propagated on a large image (a long sequence) and the contexts from small regions are collapsed by the big regions. By solving this issue, our labeling results will give more precise labeling and reduce noises. Furthermore, the computational time of our approach can even be improved by the GPU version of LSTM networks. Other directions include the analysis of feature maps and internal representation in deeper RNN models to adapt or generalize for other vision-based tasks in general.

## References


[1] W. Byeon and T. M. Breuel. Supervised texture segmentation using 2d lstm networks. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 4373–4377, Oct 2014. 2

[2] W. Byeon, M. Liwicki, and T. Breuel. Texture classification using 2d lstm networks. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 1144–1149, Aug 2014. 2

[3] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1915–1929, Aug 2013. 1, 2, 5

[4] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1–8, Sept 2009. 1, 4

[5] D. Grangier, L. Bottou, and R. Collobert. Deep convolutional networks for scene parsing. 1

[6] A. Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. 2

[7] A. Graves, S. Fernndez, and J. Schmidhuber. Multi-dimensional recurrent neural networks. In J. de S, L. Alexandre, W. Duch, and D. Mandic, editors, *Artificial Neural Networks ICANN 2007*, volume 4668 of *Lecture Notes in Computer Science*, pages 549–558. Springer Berlin Heidelberg, 2007. 2

[8] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(5):855–868, May 2009. 2

[9] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. 2

[10] X. He, R. Zemel, and M. Carreira-Perpindn. Multiscale conditional random fields for image labeling. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–695–II–702 Vol.2, June 2004. 1

[11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, Nov 1997. 2

[12] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153, Sept 2009. 1

[13] T. Kekeç, R. Emonet, E. Fromont, A. Trémeau, C. Wolf, and F. Saint-Etienne. Contextually constrained deep networks for scene labeling. In *Proceedings of the British Machine Vision Conference, 2014*, 2014. 1, 5

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In

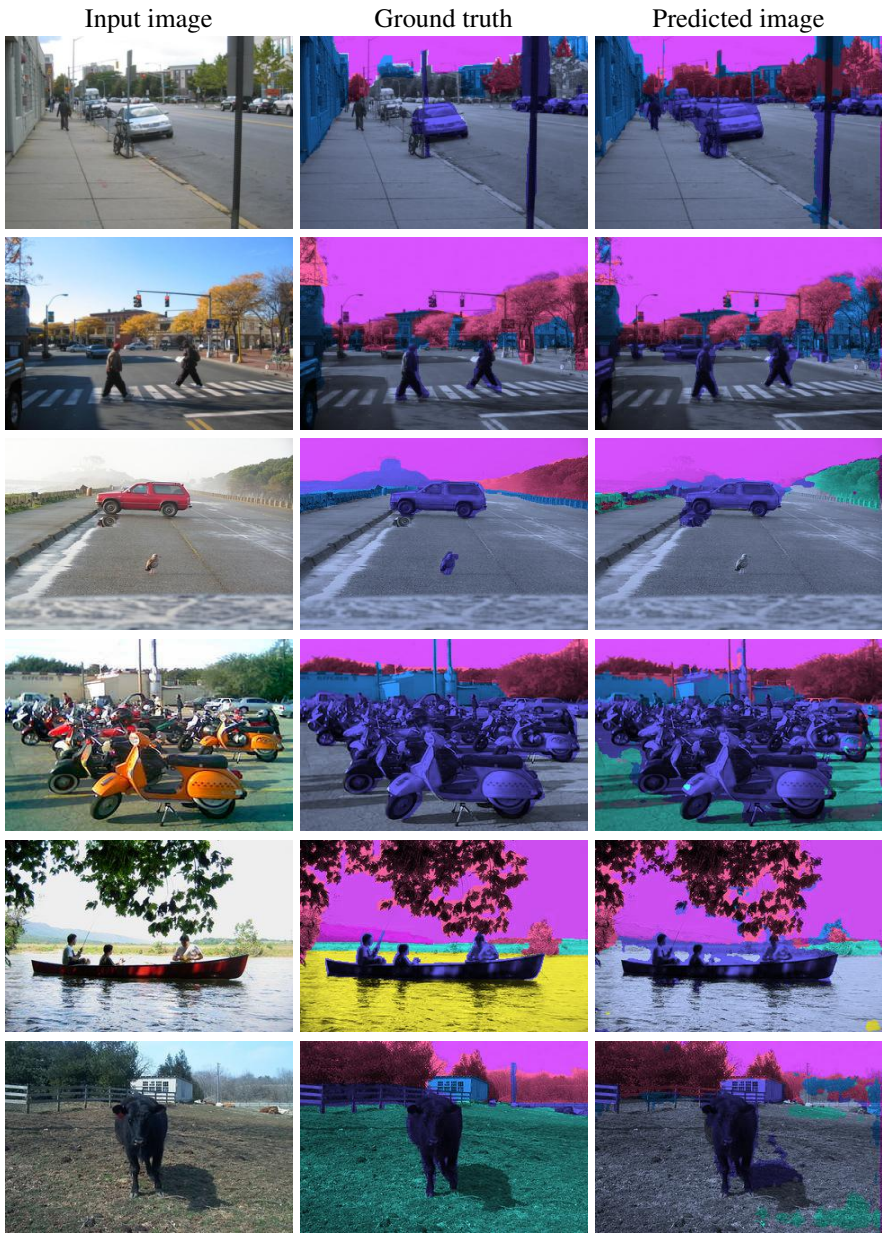


Figure 5: Selected mislabeled examples from LSTM networks. The first and second errors are mainly from mislabeling in the ground-truth (gt): human mistake — the car and human parts are labeled as "road" instead of its label "foreground" in the ground-truth. The third misclassified regions are from very foggy forest. Furthermore, the reflection of a car wheel on the water is misclassified as a wheel (label "foreground"). These examples are understandable mistakes. The results show the difficulties of our dataset and the ambiguity of actual labels. The fourth to sixth examples show the common mistakes of LSTM networks — the well-segmented regions were mislabeled. First row: road (gt) to grass (predicted); Second row: water (gt) to road (predicted); Third row: grass (gt) to road (predicted)

F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 1

[15] M. Kumar and D. Koller. Efficiently selecting regions for scene understanding. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3217–3224, June 2010. 1

[16] D. Larlus and F. Jurie. Combining appearance models and markov random fields for category level object segmentation. In *Computer Vision and Pattern Recognition, 2008. CVPR*

*2008. IEEE Conference on*, pages 1–7, June 2008. 1

[17] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551, Dec. 1989. 1

[18] C. 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. 4

[19] P. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene labeling. In T. Jebara and E. P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 82–90. JMLR Workshop and Conference Proceedings, 2014. 1, 5

[20] C. Russell, P. H. S. Torr, and P. Kohli. Associative hierarchical crfs for object class image segmentation. In *in Proc. ICCV*, 2009. 1

[21] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013. 1

[22] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng. Convolutional-recursive deep learning for 3d object classification. In *Advances in Neural Information Processing Systems*, pages 665–673, 2012. 1

[23] R. Socher, C. C. yu Lin, A. Y. Ng, and C. D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the International Conference on Machine Learning (ICML-11)*, 2011. 1

[24] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1701–1708, June 2014. 1

[25] J. Tighe and S. Lazebnik. Superparsing: Scalable nonparametric image parsing with superpixels. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision ECCV 2010*, volume 6315 of *Lecture Notes in Computer Science*, pages 352–365. Springer Berlin Heidelberg, 2010. 1, 5

[26] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision ECCV 2014*, volume 8689 of *Lecture Notes in Computer Science*, pages 818–833. Springer International Publishing, 2014. 2