Deep Structured Features for Semantic Segmentation

Michael Tschannen, Lukas Cavigelli, Fabian Mentzer, Thomas Wiatowski, and Luca Benini

Dept. IT & EE, ETH Zurich, Zurich, Switzerland

{michaelt, withomas}@nari.ee.ethz.ch, mentzerf@student.ethz.ch, {cavigelli, lbenini}@iis.ee.ethz.ch

Abstract—We propose a highly structured neural network architecture for semantic segmentation with an extremely small model size, suitable for low-power embedded and mobile platforms. Specifically, our architecture combines i) a Haar wavelet-based tree-like convolutional neural network (CNN), ii) a random layer realizing a radial basis function kernel approximation, and iii) a linear classifier. While stages i) and ii) are completely pre-specified, only the linear classifier is learned from data. We apply the proposed architecture to outdoor scene and aerial image semantic segmentation and show that the accuracy of our architecture is competitive with conventional pixel classification CNNs. Furthermore, we demonstrate that the proposed architecture is data efficient in the sense of matching the accuracy of pixel classification CNNs when trained on a much smaller data set.

I. INTRODUCTION

Semantic segmentation of images is an important step in many computer vision applications and refers to the task of identifying the semantic category of every pixel of an image. For example, in images depicting street scenes, the list of possible categories may include "car", "person", or "tree".

In recent years, convolutional neural networks (CNNs) have emerged as a popular method for semantic segmentation. To infer a dense labeling, most of the corresponding CNN architectures either rely on pixel classification [1]–[4] or on so-called deconvolution layers [4]–[10], which combine up-sampling, interpolation with a (possibly learned) kernel, and a non-linearity. Pixel classification approaches map each pixel to a feature vector by feeding the patch surrounding the pixel through a CNN and then applying a (respectively, the same) classifier to each feature vector. The so-obtained labeling is often refined using, e.g., superpixel and/or conditional random field (CRF)-based regularization. Deconvolution layer-based CNNs construct the labels by connecting deconvolution layers (with the output size being equal to the size of the original image) to one (in the case of encoder-decoder architectures [4], [8]) or multiple intermediate layers of a feed-forward CNN, the output of these deconvolution layers being finally combined. In both cases, the architectures are typically trained end-to-end. Deconvolution layer-based approaches are often more accurate and faster than pixel classification networks.

Both types of architectures have a rather complex structure resulting in large models, and may therefore not be suited for applications subject to strong resource constraints as present in embedded vision systems. Furthermore, the necessity of pre-training [5]–[7], [10], training on large labeled data sets, and parameter optimization requiring gradient back-propagation through the entire network, may hinder on-device learning [11]

and applications where only a small set of labeled images is available.

Contributions: We propose a novel, highly structured CNN architecture for semantic segmentation. Specifically, this architecture combines a tree-like CNN-based feature extractor [12], a random layer realizing a radial basis function (RBF) kernel approximation [13], and a linear classifier [14]. The feature extractor, typically the computational bottleneck in CNNs, allows for a very fast implementation—in particular when employed with separable wavelet filters as in our experiments. All trainable parameters of our architecture are in the last layer (i.e., the linear classifier), allowing for very simple stochastic gradient descent (SGD) weight updates, well suited for on-device learning. This is in contrast to conventional CNN architectures for which SGD weight updates typically require computationally more demanding gradient back-propagation. Using Haar wavelets as convolutional filters, we evaluate the architecture for semantic segmentation of two different image types, namely outdoor scene images (from the Stanford Background data set [15]) and aerial images (form the Vaihingen data set of the ISPRS 2D semantic labeling contest [16]), using identical values for almost all hyper-parameters of the architecture in both cases. For both image types, the performance of our architecture is comparable to that of end-to-end trained conventional pixel classification CNNs. However, our architecture has a model size of less than 350kBytes, which is 2-3 orders of magnitude smaller than the size of most conventional CNN models, and is therefore an ideal choice for inference on platforms with strong memory constraints. Further experiments show that our architecture is data efficient and matches the accuracy of the CNN in [1] using a much smaller training set.

II. NETWORK ARCHITECTURE

We set the stage by introducing our CNN architecture for semantic segmentation. The CNN we consider has a total depth of $D + 2$ where the first D layers correspond to a tree-like CNN-based feature extractor with pre-specified (i.e., hand-crafted) frame filters [12], followed by a non-linear classifier composed of a single fully connected RBF kernel approximation layer with pre-specified random filters [13], and a single fully connected linear classification layer based on learned filters [14].

A. CNN-based feature extractor

We briefly review the tree-like CNN-based feature extractor presented in [12], the basis of which is a convolutional
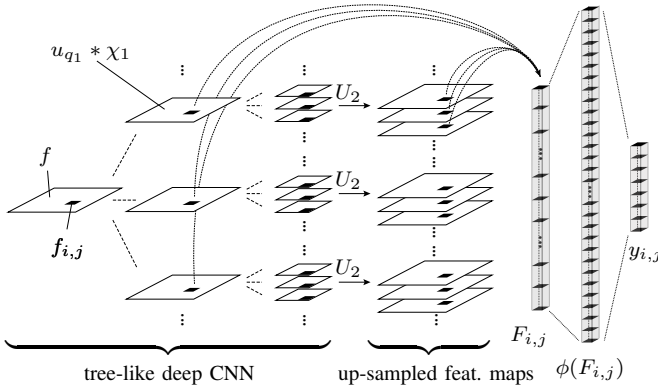
Fig. 1: Network architecture for semantic segmentation of total depth 4. The first $D = 2$ layers correspond to the tree-like CNN defined in Sec. II-A, which employs pooling by sub-sampling from the first to the second layer with pooling factor 2. The feature vector $F_{i,j} \in \mathbb{R}^m$ corresponding to the pixel $f_{i,j} \in \mathbb{R}$ is transformed to $\phi(F_{i,j})$ according to (4), and class scores $y_{i,j} \in \mathbb{R}^K$ are computed as in (5).

transform followed by a non-linearity and a pooling operation. Specifically, every network layer—specified by the layer index $1 \leq d \leq D$—is associated with

i) a collection of pre-specified filters $\{\chi_d\} \cup \{g_{\lambda_d}\}_{\lambda_d \in \Lambda_d} \subseteq \mathbb{R}^{N_d \times N_d}$, indexed by a countable set Λ_d and satisfying the Bessel condition

$$\|f * \chi_d\|_2^2 + \sum_{\lambda_d \in \Lambda_d} \|f * g_{\lambda_d}\|_2^2 \leq B_d \|f\|_2^2, \quad (1)$$

for all $f \in \mathbb{R}^{N_d \times N_d}$, for some $B_d > 0$, where $*$ denotes the circular convolution operator,

ii) a pointwise Lipschitz-continuous non-linearity $\rho_d : \mathbb{R} \to \mathbb{R}$,

iii) a Lipschitz-continuous pooling operator $P_d : \mathbb{R}^{N_d \times N_d} \to \mathbb{R}^{(N_d/S_d) \times (N_d/S_d)}$, where the integer $S_d \in \mathbb{N}$, with $N_d/S_d =: N_{d+1} \in \mathbb{N}$, is referred to as pooling factor, and determines the "size" of the neighborhood values are combined in.

Associated with these filters, non-linearities, and pooling operators, the feature maps are defined as

$$f_{q_d} := u_{q_d} * \chi_{d+1} \in \mathbb{R}^{N_{d+1} \times N_{d+1}}, \quad (2)$$

where $q_d = (\lambda_1, \lambda_2, \ldots, \lambda_d) \in \Lambda_1 \times \Lambda_2 \times \cdots \times \Lambda_d =: \Lambda_1^d$ is a path of indices of length d, and

$$u_{q_d} = P_d\big(\rho_d(u_{q_{d-1}} * g_{\lambda_d})\big),$$

for $d \in \{1, \ldots D-1\}$, are propagated signals with input signal $u_{q_0} := f \in \mathbb{R}^{N_1 \times N_1}$.

Remark 1. *It is shown in [12, Thm. 2] that the feature maps (2) are vertically translation-invariant in the sense of the layer index d determining the extent to which the feature maps are translation-invariant, and deformation insensitive w.r.t. to small non-linear deformations. We refer the reader to [17]–[19] for similar translation-invariant and deformation-insensitive tree-like CNNs.*

To leverage the network-based feature maps (2) for semantic segmentation, we now bi-linearly interpolate the feature maps (2) to the size of the input image f, i.e., $N_1 \times N_1$, according to $U_d f_{q_d} \in \mathbb{R}^{N_1 \times N_1}$, where $U_d : \mathbb{R}^{N_{d+1} \times N_{d+1}} \to \mathbb{R}^{N_1 \times N_1}$ is the corresponding bi-linear interpolation operator with $U_0 f = f$. The feature vector $F_{i,j}$ of the pixel $f_{i,j} \in \mathbb{R}$ (of the input image $f \in \mathbb{R}^{N_1 \times N_1}$) is defined as

$$F_{i,j} := \left\{ (U_d f_{q_d})_{i,j} \right\}_{q_d \in \cup_{k=0}^{D-1} \Lambda_1^k}, \quad (3)$$

i.e., $F_{i,j}$ is obtained by collecting the entry at location (i, j) from each (up-sampled) feature map in the network (see Fig. 1).

For the experiments in the present paper, we particularize the feature extractor as follows: In every network layer, we employ tensorized (i.e., separable) Haar wavelets $\{g_\lambda\}_{\lambda \in \Lambda}$, sensitive to $R = 3$ directions (horizontal, vertical, and diagonal) and—for each direction—sensitive to $J = 4$ scales, with corresponding wavelet low-pass filter χ, together satisfying the Bessel condition (1) with $B_d = 1$, see [20].[1] We set $D = 2$ and employ the modulus non-linearity $\rho = |\cdot|$ as well as pooling by sub-sampling where we retain every second pixel, i.e., $P : \mathbb{R}^{N \times N} \to \mathbb{R}^{N/2 \times N/2}$ with $(Pf)_{i,j} = f_{2i,2j}$. Note that pooling increases the robustness of the feature vector w.r.t. non-linear deformations [12] and hence allows our architecture to deal with variation in appearance of the semantic categories. Finally, similarly to [1], we also employ a variant of our network that extracts features at multiple image scales $\{s_\ell\}_{\ell=1}^L \subset \mathbb{N}$ by applying the feature extractor to multiple sub-sampled versions $\{f^\ell\}_{\ell=1}^L$ of the input image f, i.e., $(f^\ell)_{i,j} = f_{s_\ell i, s_\ell j}$. We then bi-linearly interpolate the resulting feature maps to the size $N_1 \times N_1$ of the input image f and, for every pixel, concatenate the feature vectors from all scales into a single feature vector accordingly.

B. RBF kernel approximation layer

In the $(D+1)$-st layer of our CNN, we map the feature vectors $F_{i,j} \in \mathbb{R}^m$ to randomized feature vectors $\phi(F_{i,j}) \in \mathbb{R}^{\widetilde{m}}$ with $\widetilde{m} > m$, such that $\langle \phi(F_{i,j}), \phi(F_{i',j'}) \rangle \approx k(F_{i,j}, F_{i',j'})$, where $k(F_{i,j}, F_{i',j'}) := \exp(-\gamma \|F_{i,j} - F_{i',j'}\|_2^2)$ is a RBF kernel with parameter γ. In large-scale classification tasks as the one considered here, such randomized feature vectors combined with a linear classifier (see Sec. II-C) typically allow for much faster training and inference than non-linear kernel-based classifiers [13]. We follow the construction given in [13]: Let $G \in \mathbb{R}^{\widetilde{m} \times m}$ be a pre-specified matrix with i.i.d. standard normal entries, and let $b \in \mathbb{R}^{\widetilde{m}}$ be a pre-specified vector with entries i.i.d. uniform on $[0, 2\pi]$. We define the transformed feature vectors as

$$\phi(F_{i,j}) := \sqrt{\frac{2}{\widetilde{m}}} \cos\big(\gamma G F_{i,j} + b\big) \in \mathbb{R}^{\widetilde{m}}, \quad (4)$$

where $\cos(v)$, for $v \in \mathbb{R}^{\widetilde{m}}$, refers to element-wise application of cos. We note that the RBF kernel approximation can be

[1]The choice of this wavelet type is motivated by computational efficiency (see Sec. II-E). We did not explore other wavelet types, in particular not directional wavelets.

interpreted as a single fully connected layer with random filters and cos non-linearity (see Fig. 1).

C. Linear classification layer

In the last, i.e., $(D + 2)$-nd layer of our CNN, we employ a linear classifier, shared across all pixels, meaning that we apply the same classifier to all $\phi(F_{i,j})$. Formally, we apply a matrix $W \in \mathbb{R}^{K \times \tilde{m}}$, add a bias vector $v \in \mathbb{R}^K$ according to

$$y_{i,j} := W\phi(F_{i,j}) + v \in \mathbb{R}^K, \qquad (5)$$

and determine the class label via a one-versus-the-rest scheme as $\arg\max_{k \in \{1,...,K\}}(y_{i,j})_k$. The matrix W and the vector b are learned by minimizing the hinge loss with an ℓ_2-regularization term using SGD [14], i.e., we learn the last layer (see Fig. 1) using a support vector machine (SVM).

D. Relation to prior work

Although the tree-like feature extractor from [12] and those described in [17]–[19] were employed previously for different computer vision applications, to the best of our knowledge we are the first to use this type of feature extractor for semantic segmentation. A key difference to previous applications, which use the union of all feature maps to to characterize the entire image, is the aggregation of features across feature maps to characterize individual pixels. This is also in contrast to conventional pixel classification networks [1]–[4] which (unlike tree-like architectures) recombine feature channels in each layer and generate pixel-wise features only in the last layer. Our architecture is hence not a special case of pixel classification networks. Finally, a similar concept of aggregating pixel-wise features across feature maps was proposed in [21] for simultaneous detection and segmentation (a computer vision task that differs significantly from the semantic segmentation task considered here). However, [21] requires a pre-trained detection network and end-to-end training, while here we rely on pre-specified wavelet filters as well as pre-specified random filters and train the last classification layer only.

E. Efficient implementation and storage requirements

Even though we leave a highly optimized implementation of the proposed architecture for future work, we discuss possible optimizations and resulting gains.

CNN-based semantic segmentation implementations typically spend around 90% of the overall execution time computing the feature maps before applying pixel-wise classification. During feature extraction the vast majority of of computation effort is incurred by the convolution layers [9], with most CNNs using in the order of 10 GOp/image [5], [9], where one operation (Op) is an addition or a multiplication.

Our feature extractor only performs convolutions with tensorized (separable) Haar wavelets, which can be implemented efficiently with the *algorithme à trous* [20, Sec. 5.2.2]. For a 1D Haar wavelet transform, $2NJ$ add/subtract operations are required for a length-N signal, for J scales. For the separable 2D case, the filtering is first applied in horizontal direction before applying it to the two resulting signals in vertical

direction [22, Fig. "Decomposition Step"], and then repeating for the remaining scales using a strided convolution pattern, resulting in a total of $2(W + 2H)J$ operations. Performing the χ filter operations before interpolating and aggregating the last layer's feature maps (which are not included in the 2D stationary wavelet transform (SWT)) requires $\frac{1}{2}WHJ(R^2J^2)$ more additions. For all 2D SWT convolution operations in the feature extractor the number of additions amounts to $6WHJ + \frac{3}{2}WHRJ^2$ (recall that we employ pooling by sub-sampling). Additionally, $WHRJ(1 + RJ/4)$ absolute-value operations are required and $WH(RJ + 1)RJ$ pixels need to be interpolated with 4 multiply and 4 add operations each. For the configuration specified in Sec. II-A applied to each of the three channels of the input image, the evaluation of the feature extraction network is dominated by the interpolation effort and results in $387\,\mathrm{MOp}$ for a 320×240 image—about $20\times$ less computation effort than competing CNNs and, just as important for embedded devices, without any trained parameters to be stored.

Furthermore, note that the evaluation of the RBF approximation layer can be accelerated from $\mathcal{O}(\tilde{m}m)$ operations to $\mathcal{O}(\tilde{m}\log m)$ operations by using a fast RBF kernel approximation such as [23], [24]. These methods (just like the one used for our evaluations) only require randomly generated values, which can be regenerated on-the-fly using an identically seeded pseudo-random number generator.

The only parameters to be stored are the matrix W in the final linear layer and the corresponding bias vector v. Our architecture thus has an extremely small memory footprint of only $(\tilde{m}+1)K$ values, which typically amounts to a few tens of thousands of parameters—a strong requirement for performing inference on tightly resource-bound hardware platforms. In contrast, many popular (pre-trained) CNN architectures such as AlexNet, VGG, or GoogLeNet, which the semantic segmentation CNNs in [5]–[7], [10], [25] build upon, have millions of parameters, even when truncated [26].

III. EXPERIMENTS

We evaluate the performance of the proposed CNN architecture in semantic segmentation of outdoor scene images and aerial images. For both segmentation tasks we set the output dimension of the RBF approximation layer to $\tilde{m} = 5000$, resulting in a model size of less than 350kByte in all cases (for model parameters with 32bit float precision, stored using Python built-in serialization via `pickle`). For training, we first randomly draw a subset containing 2% of all pixels in the training set, collect the corresponding feature vectors, and then perform SGD passes over randomly shuffled versions of the so-obtained feature vector set. Furthermore, we tune the RBF kernel parameter γ as well as the parameter λ balancing the hinge loss and the regularization term in the SVM objective (see [14, Tab. 1]). We evaluate the segmentation performance for feature extraction at a single scale (i.e., the original image size) and at scales $s_\ell \in \{1, 2, 4\}$. Preliminary experiments showed that using the full training set for the SGD passes or

increasing the feature extraction network depth D does not significantly increase the accuracy. We implemented the proposed architecture in Python and Matlab (for the feature extractor) on CPU. Note that we did not optimize the implementation for speed. The runtimes we report were obtained on a desktop computer with 2.5 GHz Intel Core i7 (I7-4870HQ) and 16 GB RAM, and can be drastically reduced by leveraging the parallel processing capabilities of GPUs for the the evaluation of the feature extractor, the random layer, and the classification layer. To ensure a fair comparison with accuracies reported in the literature for other architectures, we always refer to the accuracies obtained without post-processing (using, e.g., a CRF or superpixels).

A. Outdoor scene semantic segmentation

We use the Stanford Background data set [15] containing 715 RGB images of outdoor scenes of size approximately 320×240 pixels. Each pixel is labeled with one of eight semantic categories ("sky", "tree", "road", "grass", "water", "building", "mountain", and "foreground object"). An image is processed by first transforming it to YUV color space, applying the feature extraction network (possibly at multiple scales) to each color channel, and subsequently concatenating the extracted feature vectors. This results in feature vectors $F_{i,j}$ of dimension $m = 309$ for 1 scale and $m = 927$ for 3 scales. The runtime per image for segmentation was 9.6s and 23.4s for 1 scale and 3 scales, respectively.

Fig. 2 (b, c) shows example outputs of our CNN for images from the Stanford Background data set. Following [1], we estimate the accuracy of our method using 5-fold cross-validation (CV). Table I (top) shows the pixel accuracy (averaged over all pixels) and the class accuracy (i.e., the average class precision) of our CNN architecture, along with the end-to-end trained pixel classification CNN from [1] (we refer to [1, Tab. 1] for comparison with non-CNN-based methods). For 1 scale, our network outperforms the CNN from [1]. Using 3 scales instead of 1 increases the pixel accuracy of our CNN by 3.4%, i.e., the gain from using multiple scales is smaller than for the CNN from [1]. A possible reason for this could be that the wavelet filters used in our network already capture the multi-scale nature of the images quite well. We note that other CNN architectures [3], [5]–[9] achieve higher accuracies in semantic segmentation of outdoor scenes (mostly on other data sets). However, these architectures are all trained end-to-end and rely either on deconvolution layers or recurrent structure, resulting in orders of magnitude larger models and requiring considerably larger computational effort for training. In addition, the architectures from [5]–[8] are based on pre-trained classification CNNs.

Effect of training set size on pixel accuracy: We investigate the effect of the number of training images n_{train} on the pixel accuracy. Specifically, we fix γ and λ to the values obtained for 5-fold CV, randomly split the data set into 500 training and 215 testing images, and draw n_{train} images from the training set (keeping the same training/testing split for all n_{train}). Fig. 2 (a) shows the pixel accuracy of our architecture
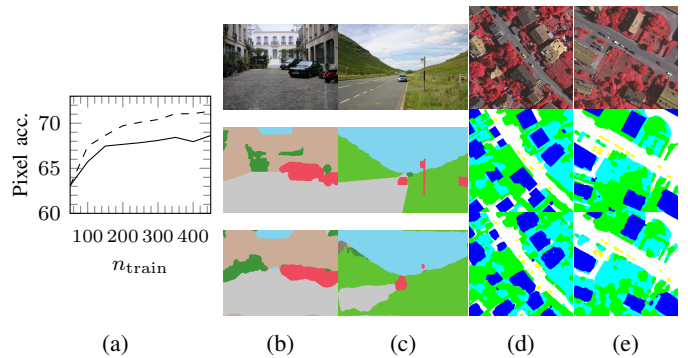

Fig. 2: (a): Pixel accuracy (in %) of our architecture as a function of the number of training images n_{train} using 1 scale (solid) and 3 scales (dashed), for outdoor scene semantic segmentation. (b–e): Input image (top), ground truth labeling (middle), and prediction by our architecture (3 scales; bottom) for example outdoor scenes (b, c) and aerial images (d, e).

as a function of n_{train} (averaged over 3 random training/testing splits). For $n_{\text{train}} \geq 100$ the pixel accuracy exceeds 93% of the accuracy obtained for the full training set containing 575 images (for 5-fold CV), for both 1 scale and 3 scales. Our architecture is thus quite data efficient. In particular, in the single scale case, it matches the pixel accuracy of the CNN from [1] using 5 times less training images.

B. Aerial image semantic segmentation

We evaluate our architecture on the Vaihingen data set[2] of the ISPRS 2D semantic labeling contest [16], which contains 33 aerial images of varying size (average size 2494×2064 pixels [4]). Pixel-level semantic labels (categories: "impervious surface", "building", "low vegetation", "tree", "car", and "background") are available for 16 images, the labels of the remaining images serve as private testing set for the contest. The images have three channels (near infrared, red, and green) and come with a (coregistered) digital surface model (DSM). Following [4], [25], we retain images 11, 15, 28, 30, and 34 for testing and use the remaining labeled images in the data set for training. Similarly as for outdoor scene semantic labeling, we apply the feature extraction network to each channel and to the normalized version of the DSM provided by [28]. We concatenate the feature vectors extracted for each channel to obtain feature vectors $F_{i,j}$ of dimension $m = 412$ for 1 scale and $m = 1236$ for 3 scales. The runtime per megapixel for segmentation was $2.6\,\text{min}$ and $6.6\,\text{min}$ for 1 and 3 scales, respectively.

Fig. 2 (d, e) show an example of an aerial image patch along with the ground truth labeling and the predictions of our architecture. In Table I (bottom) we report the pixel accuracy and the F1 score (averaged over classes) of our CNN architecture as well as that of two end-to-end trained pixel classification CNNs [2], [4]. As in [2], [4] (and following the rules of the ISPRS 2D semantic labeling contest) labeling

[2]The Vaihingen data set was provided by the German Society for Photogrammetry, Remote Sensing and Geoinformation (DGPF) [27]: http://www.ifp.uni-stuttgart.de/dgpf/DKEP-Allg.html.

	Number of scales	Pixel acc.		Class acc.	
		1	3	1	3
out.	Pixel class. CNN [1]	66.0	78.8	56.5	72.4
	Our CNN	68.3	71.7	62.1	63.3

		Pixel acc.		F1 score	
aerial	Pixel class. CNN [2]	83.46	85.56	77.84	81.74
	CNN-PC [4]	86.67	-	68.01	-
	Our CNN	85.16	85.45	66.28	75.73

TABLE I: Pixel accuracy (in %) and average Class accuracy/F1 score (in %) for outdoor scene (top, "out.") and aerial image (bottom, "aerial") semantic segmentation on the test set.

errors within a 3 pixel radius of the (true) category boundaries are excluded for the computation of the accuracy and the F1 score. It can be seen that the accuracy and the average F1 score of our architecture is comparable to the pixel classification CNNs proposed in [2], [4]. On the private ISPRS testing set our architecture with 3 scales obtained a pixel accuracy of 85.9 %, thus outperforming the algorithms from [29], each of which combines a pre-trained classification CNN with a SVM. We note that other CNN architectures [2], [4], [10], [25] achieve higher accuracies and F1 scores on the Vaihingen data set. Again, these architectures are all trained end-to-end and rely on deconvolution layers [4], [10], pre-trained classification CNNs [10], [25], CRF-based refinement [2], [25], and/or additional hand-crafted features [2].

IV. CONCLUSION

We proposed a simple highly structured Haar wavelet-based CNN architecture for semantic segmentation with an extremely small model size, potentially allowing for a very fast implementation thanks to its structure. We demonstrated that our architecture is very data efficient and that its accuracy is comparable to that of conventional pixel classification CNNs, even though we used pre-defined features in place of computationally demanding end-to-end feature learning.

Replacing the pixel classification network by deconvolution layers might improve the segmentation accuracy and is an interesting direction to be explored in the future.

ACKNOWLEDGMENTS

REFERENCES

[1] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 35, no. 8, pp. 1915–1929, 2013.

[2] S. Paisitkriangkrai, J. Sherrah, P. Janney, and A. Van-Den Hengel, "Effective semantic pixel labelling with convolutional networks and conditional random fields," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 36–43.

[3] P. H. Pinheiro and R. Collobert, "Recurrent convolutional neural networks for scene labeling.," in *Proc. of Int. Conf. on Machine Learning*, 2014, pp. 82–90.

[4] M. Volpi and D. Tuia, "Dense semantic labeling of sub-decimeter resolution images with convolutional neural networks," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 2, pp. 881–893, 2017.

[5] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[6] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proc. of IEEE Int. Conf. on Computer Vision*, 2015, pp. 2650–2658.

[7] C. Liang-Chieh, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," in *Int. Conf. on Learning Representations*, 2015.

[8] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *arXiv:1511.00561*, 2015.

[9] L. Cavigelli, M. Magno, and L. Benini, "Accelerating real-time embedded scene labeling with convolutional networks," in *Proc. of Annual Design Automation Conference*. ACM, 2015, p. 108.

[10] D. Marmanis, J. D. Wegner, S. Galliani, K. Schindler, M. Datcu, and U. Stilla, "Semantic segmentation of aerial images with an ensemble of cnns," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pp. 473–480, 2016.

[11] N. Shanbhag, "Energy-efficient machine learning in silicon: A communications-inspired approach," in *ICML 2016 Workshop on On-Device Intelligence*, 2016.

[12] T. Wiatowski, M. Tschannen, A. Stanić, P. Grohs, and H. Bölcskei, "Discrete deep feature extraction: A theory and new architectures," in *Proc. of Int. Conf. on Machine Learning*, June 2016, pp. 2149–2158.

[13] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems*, 2007, pp. 1177–1184.

[14] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

[15] S. Gould, R. Fulton, and D. Koller, "Decomposing a scene into geometric and semantically consistent regions," in *Proc. of IEEE Int. Conf. on Computer Vision*, 2009.

[16] "ISPRS 2D semantic labeling contest," http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html.

[17] T. Wiatowski and H. Bölcskei, "A mathematical theory of deep convolutional neural networks for feature extraction," *arXiv:1512.06293*, 2015.

[18] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1872–1886, 2013.

[19] S. Mallat, "Group invariant scattering," *Comm. Pure Appl. Math.*, vol. 65, no. 10, pp. 1331–1398, 2012.

[20] S. Mallat, *A wavelet tour of signal processing: The sparse way*, Academic Press, 3rd edition, 2009.

[21] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *Proc. IEEE CVPR*, 2015, pp. 447–456.

[22] MathWorks, "Matlab R2016b Documentation: Discrete stationary wavelet transform 2-D (swt2)," https://ch.mathworks.com/help/wavelet/ref/swt2.htm, 2016.

[23] Q. Le, T. Sarlós, and A. Smola, "Fastfood-approximating kernel expansions in loglinear time," in *Proc. of Int. Conf. on Machine Learning*, 2013, pp. 244–252.

[24] K. Choromanski and V. Sindhwani, "Recycling randomness with structure for sublinear time kernel expansions," in *Proc. of Int. Conf. on Machine Learning*, 2016, pp. 2502–2510.

[25] J. Sherrah, "Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery," *arXiv:1606.02585*, 2016.

[26] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, 2015, pp. 1135–1143.

[27] M. Cramer, "The DGPF-test on digital airborne camera evaluation–overview and test design," *Photogrammetrie-Fernerkundung-Geoinformation*, vol. 2010, no. 2, pp. 73–82, 2010.

[28] M. Gerke, "Use of the stair vision library within the ISPRS 2D semantic labeling benchmark (Vaihingen)," Tech. Rep., University of Twente, 2015.

[29] A. Lagrange and B. Le Saux, "Convolutional neural networks for semantic labeling," Tech. Rep., Onera – The French Aerospace Lab, 2015.