ReSeg: A Recurrent Neural Network-based Model for Semantic Segmentation

Francesco Visin*†
francesco.visin@polimi.it
Marco Ciccone*
marco.ciccone@mail.polimi.it
Adriana Romero†
adriana.romero.soriano@umontreal.ca
Kyle Kastner†
kyle.kastner@umontreal.ca
Kyunghyun Cho‡
kyunghyun.cho@nyu.edu
Yoshua Bengio†§
yoshua.bengio@umontreal.ca
Matteo Matteucci*
matteo.matteucci@polimi.it
Aaron Courville†
aaron.courville@umontreal.ca

Abstract

We propose a structured prediction architecture, which exploits the local generic features extracted by Convolutional Neural Networks and the capacity of Recurrent Neural Networks (RNN) to retrieve distant dependencies. The proposed architecture, called ReSeg, is based on the recently introduced ReNet model for image classification. We modify and extend it to perform the more challenging task of semantic segmentation. Each ReNet layer is composed of four RNN that sweep the image horizontally and vertically in both directions, encoding patches or activations, and providing relevant global information. Moreover, ReNet layers are stacked on top of pre-trained convolutional layers, benefiting from generic local features. Upsampling layers follow ReNet layers to recover the original image resolution in the final predictions. The proposed ReSeg architecture is efficient, flexible and suitable for a variety of semantic segmentation tasks. We evaluate ReSeg on several widely-used semantic segmentation datasets: Weizmann Horse, Oxford Flower, and CamVid; achieving state-of-the-art performance. Results show that ReSeg can act as a suitable architecture for semantic segmentation tasks, and may have further applications in other structured prediction problems. The source code and model hyperparameters are available on https://github.com/fvisin/reseg.

*Dipartimento di Elettronica Informazione e Bioingegneria, Politecnico di Milano, Milan, 20133, Italy
†Montreal Institute for Learning Algorithms (MILA), University of Montreal, Montreal, QC, H3T 1J4, Canada
‡Courant Institute and Center for Data Science, New York University, New York, NY 10012, United States
§CIFAR Senior Fellow

1. Introduction

In recent years, Convolutional Neural Networks (CNN) have become the *de facto* standard in many computer vision tasks, such as image classification and object detection [23, 15]. Top performing image classification architectures usually involve *very* deep CNN trained in a supervised fashion on a large datasets [28, 39, 43] and have been shown to produce generic hierarchical visual representations that perform well on a wide variety of vision tasks. However, these deep CNNs heavily reduce the input resolution through successive applications of pooling or subsampling layers. While these layers seem to contribute significantly to the desirable invariance properties of deep CNNs, they also make it challenging to use these pre-trained CNNs for tasks such as semantic segmentation, where a per pixel prediction is required.

Recent advances in semantic segmentation tend to convert the standard deep CNN classifier into Fully Convolutional Networks (FCN) [30, 33, 2, 36] to obtain coarse image representations, which are subsequently upsampled to recover the lost resolution. However, these methods are not designed to take into account and preserve both *local* and *global* contextual dependencies, which has shown to be useful for semantic segmentation tasks [40, 17]. These models often employ Conditional Random Fields (CRFs) as a post-processing step to locally smooth the model predictions, however the long-range contextual dependencies remain relatively unexploited.

Recurrent Neural Networks (RNN) have been introduced in the literature to retrieve global spatial dependencies and further improve semantic segmentation [34, 17, 9, 8]. However, training spatially recurrent neural networks tends to be computationally intensive.

In this paper, we aim at the *efficient* application of Recurrent Neural Networks RNN to retrieve contextual information from images. We propose to extend the ReNet architecture [45], originally designed for image classification, to deal with the more ambitious task of semantic segmentation. ReNet layers can efficiently capture contextual dependencies from images by first sweeping the image horizontally, and then sweeping the output of hidden states vertically. The output of a ReNet layer is therefore implicitly encoding the local features at each pixel position with respect to the whole input image, providing relevant global information. Moreover, in order to *fully* exploit local and global pixel dependencies, we stack the ReNet layers on top of the output of a FCN, i.e. the intermediate convolutional output of VGG-16 [39], to benefit from generic local features. We validate our method on Weizmann Horse and Oxford Flower foreground/background segmentation datasets as a proof of concept for the proposed architecture. Then, we evaluate the performance in the standard benchmark of urban scenes CamVid; achieving state-of-the-art in all three datasets.

2. Related Work

Methods based on FCN tackle the information recovery (upsampling) problem in a large variety of ways. For instance, Eigen et al. [14] introduce a multi-scale architecture, which extracts coarse predictions, which are then refined using finer scales. Farabet et al. [16] introduce a multi-scale CNN architecture; Hariharan et al. [19] combine the information distributed over all layers to make accurate predictions. Other methods such as [30, 2] use simple bilinear interpolation to upsample the feature maps of increasingly abstract layers. More sophisticated upsampling methods, such as unpooling [2, 33] or deconvolution [30], are introduced in the literature. Finally, [36] concatenate the feature maps of the downsampling layers with the feature maps of the upsampling layers to help recover finer information.

RNN and RNN-like models have become increasingly popular in the semantic segmentation literature to capture long distance pixel dependencies [34, 17, 8, 41]. For instance, in [34, 17], CNN are unrolled through different time steps to include semantic feedback connections. In [8], 2-dimensional Long Short Term Memory (LSTM), which consist of 4 LSTM blocks scanning all directions of an image (left-bottom, left-top, right-top, right-bottom), are introduced to learn long range spatial dependencies. Following a similar direction, in [41], multi-dimensional LSTM are swept along different image directions; however, in this case, computations are re-arranged in a pyramidal fashion for efficiency reasons. Finally, in [45], ReNet is proposed to model pixel dependencies in the context of image classification. It is worth noting that one important consequence of the adoption of the ReNet spatial sequences is that they are even more easily parallelizable, as each RNN is dependent only along a horizontal or vertical sequence of pixels; i.e., all rows/columns of pixels can be processed at the same time.

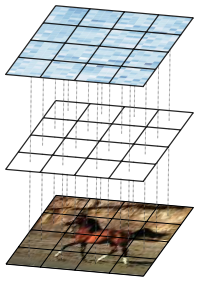

Figure 1. A ReNet layer. The blue and green dots on the input image/feature map represent the steps of $f^{\downarrow}$ and $f^{\uparrow}$ respectively. On the concatenation of the resulting feature maps, $f^{\rightarrow}$ (yellow dots) and $f^{\leftarrow}$ (red dots) are subsequently swept. Their feature maps are finally concatenated to form the output of the ReNet layer, depicted as a blue heatmap in the figure.

3. Model Description

The proposed ReSeg model builds on top of ReNet [45] and extends it to address the task of semantic segmentation. The model pipeline involves multiple stages.

First, the input image is processed with the first layers of VGG-16 [39] network, pre-trained on ImageNet [11] and not fine-tuned, and is set such that the image resolution does not become too small. The resulting feature maps are then fed into one or more *ReNet layers* that sweep over the image. Finally, one or more *upsampling layers* are employed to resize the last feature maps to the same resolution as the input and a softmax non-linearity is applied to predict the probability distribution over the classes for each pixel.

The recurrent layer is the core of our architecture and is composed by multiple RNN that can be implemented as a vanilla $\tanh$ RNN layer, a Gated Recurrent Unit (GRU) layer [10] or a LSTM layer [20]. Previous work has shown that the ReNet model can perform well with little concern for the specific recurrent unit used, therefore, we have chosen to use GRU units as they strike a good balance between memory usage and computational power.

In the following section we will define the recurrent and the upsampling layers in more detail.

3.1. Recurrent layer

As depicted in Figure 1, each recurrent layer is composed by 4 RNNs coupled together in such a way to capture the local and global spatial structure of the input data.

Specifically, we take as an input an image (or the feature map of the previous layer) $\mathbf{X}$ of elements $x \in \mathbb{R}^{H \times W \times C}$, where H, W and C are respectively the height, width and number of channels (or features) and we split it into $I \times J$

patches $p_{i,j} \in \mathbb{R}^{H_p \times W_p \times C}$. We then sweep vertically a first time with two RNNs $f^{\downarrow}$ and $f^{\uparrow}$, with U recurrent units each, that move top-down and bottom-up respectively. Note that the processing of each column is independent and can be done in parallel.

At every time step each RNN reads the next non-overlapping patch $p_{i,j}$ and, based on its previous state, emits a projection $o^{\star}_{i,j}$ and updates its state $z^{\star}_{i,j}$:

$$o^{\downarrow}_{i,j} = f^{\downarrow}(z^{\downarrow}_{i-1,j}, p_{i,j}), \text{ for } i = 1, \cdots, I \quad (1)$$

$$o^{\uparrow}_{i,j} = f^{\uparrow}(z^{\uparrow}_{i+1,j}, p_{i,j}), \text{ for } i = I, \cdots, 1 \quad (2)$$

We stress that the decision to read non-overlapping patches is a modeling choice to increase the image scan speed and lower the memory usage, but is not a limitation of the architecture.

Once the first two vertical RNNs have processed the whole input X, we concatenate their projections $o^{\downarrow}_{i,j}$ and $o^{\uparrow}_{i,j}$ to obtain a composite feature map $\mathbf{O}^{\updownarrow}$ whose elements $o^{\updownarrow}_{i,j} \in \mathbb{R}^{2U}$ can be seen as the activation of a feature detector at the location (i, j) with respect to all the patches in the j-th column of the input. We denote what we described so far as the *vertical recurrent sublayer*.

After obtaining the concatenated feature map $\mathbf{O}^{\updownarrow}$, we sweep over each of its rows with a pair of new RNNs, $f^{\rightarrow}$ and $f^{\leftarrow}$. We chose not to split $\mathbf{O}^{\updownarrow}$ into patches so that the second recurrent sublayer has the same granularity as the first one, but this is not a constraint of the model and different architectures can be explored. With a similar but specular procedure as the one described before, we proceed reading one element $o^{\updownarrow}_{i,j}$ at each step, to obtain a concatenated feature map $\mathbf{O}^{\leftrightarrow} = \left\{h^{\leftrightarrow}_{i,j}\right\}^{j=1\ldots J}_{i=1\ldots I}$, once again with $o^{\leftrightarrow}_{i,j} \in \mathbb{R}^{2U}$. Each element $o^{\leftrightarrow}_{i,j}$ of this *horizontal recurrent sublayer* represents the features of one of the input image patches $p_{i,j}$ *with contextual information from the whole image*.

It is trivial to note that it is possible to concatenate many recurrent layers $\mathbf{O}^{(\mathbf{1}\cdots\mathbf{L})}$ one after the other and train them with any optimization algorithm that performs gradient descent, as the composite model is a smooth, continuous function.

3.2. Upsampling layer

Since by design each recurrent layer processes non-overlapping patches, the size of the last composite feature map will be smaller than the size of the initial input $\mathbf{X}$, whenever the patch size is greater than one. To be able to compute a segmentation mask at the same resolution as the ground truth, the prediction should be expanded back before applying the softmax non-linearity.

Several different methods can be used to this end, e.g., fully connected layers, full convolutions and transposed convolutions. The first is not a good candidate in this domain as it does not take into account the topology of the input, which is essential for this task; the second is not optimal either, as it would require large kernels and stride sizes to upsample by the required factor. Transposed convolutions are both memory and computation efficient, and are the ideal method to tackle this problem.

Transposed convolutions – also known as *fractionally strided convolutions* – have been employed in many works in recent literature [48, 50, 31, 35, 21]. This method is based on the observation that direct convolutions can be expressed as a dot product between the flattened input and a sparse matrix, whose non-zero elements are elements of the convolutional kernel. The equivalence with the convolution is granted by the connectivity pattern defined by the matrix.

Transposed convolutions apply the transpose of this transformation matrix to the input, resulting in an operation whose input and output shapes are inverted with respect to the original direct convolution. A very efficient implementation of this operation can be obtained exploiting the gradient operation of the convolution – whose optimized implementation can be found in many of the most popular libraries for neural networks. For an in-depth and comprehensive analysis of each alternative, we refer the interested reader to [13].

4. Experiments

4.1. Datasets

We evaluated the proposed ReSeg architecture on several benchmark datasets. We proceeded by first assessing the performances of the model on the Weizmann Horse and the Oxford Flowers datasets and then focused on the more challenging Camvid dataset. We will describe each dataset in detail in this section.

4.1.1 Weizmann Horse

The Weizmann Horse dataset, introduced in [6], is an image segmentation dataset consisting of 329 variable size images in both RGB and gray scale format, matched with an equal number of groundtruth segmentation images, of the same size as the corresponding image. The groundtruth segmentations contain a foreground/background mask of the focused horse, encoded as a real-value between 0 and 255. To convert this into a boolean mask, we threshold in the center of the range setting all smaller values to 0, and all greater values to 1.

4.1.2 Oxford Flowers 17

The Oxford Flowers 17 class dataset from [32] contains 1363 variable size RGB images, with 848 image segmentations maps associated with a subset of the RGB images.

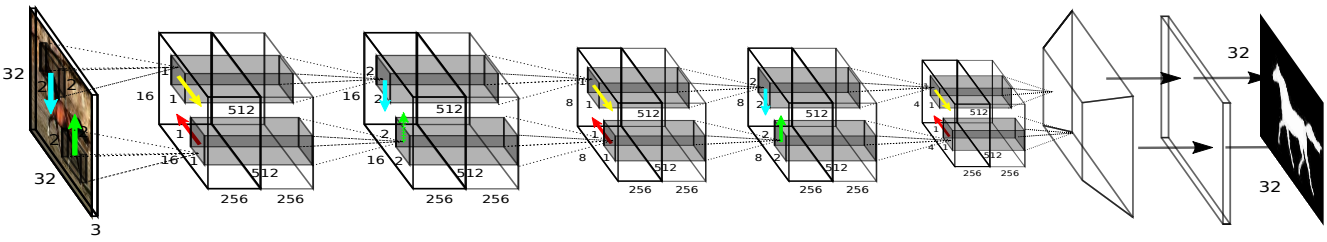

Figure 2. The ReSeg network. For space reasons we do not represent the pretrained VGG-16 convolutional layers that we use to preprocess the input to ReSeg. The first 2 RNNs (blue and green) are applied on 2x2x3 patches of the image, their 16x16x256 feature maps are concatenated and fed as input to the next two RNNs (red and yellow) which read 1x1x512 patches and emit the output of the first ReNet layer. Two similar ReNet layers are stacked, followed by an upsampling layer and a softmax nonlinearity.

There are 8 unique segmentation classes defined over all maps, including flower, sky, and grass. To build a foreground/background mask, we take the original segmentation maps, and set any pixel not belonging to class 38 (flower class) to 0, and setting the flower class pixels to 1. This binary segmentation task for Oxford Flowers 17 is further described in [46].

4.1.3 CamVid Dataset

The Cambridge-driving Labeled Video Database (CamVid) [7] is a real-world dataset which consists of images recorded from a car with an internally mounted camera, capturing frames of 960×720 RGB pixels per frame, with a recording frame rate of 30 frames per second. A total of ten minutes of video was recorded, and approximately one frame per second has been manually annotated with per pixel class labels, from one of 32 possible classes. A small number of pixels were labelled as void in the original dataset. These do not belong to any of the 32 classes prescribed in the original data, and are ignored during evaluation. We used the same subset of 11 class categories as [2] for experimental analysis. The CamVid dataset itself is split into 367 training, 101 validation and 233 test images, and in order to make our experimental setup fully comparable to [2], we downsampled all the images by a factor of 2 resulting in a final 480×360 resolution.

4.2. Experimental settings

To gain confidence with the sensitivity of the model to the different hyperparameters, we decided to evaluate it first on the Weissman Horse and Oxford Flowers datasets on a binary segmentation task; we then focused the most of our efforts on the more challenging semantic segmentation task on the CamVid dataset.

The number of hyperparameters of this model is potentially very high, as for each ReNet layer different implementations are possible (namely vanilla RNN, GRU or LSTM), each one with its specific parameters. Furthermore, the number of features, the size of the patches and the initialization scheme have to be defined for each ReNet layer as well as for each transposed convolutional layer. To make it feasible to explore the hyperparameter space, some of the hyperparameters have been fixed by design and the remaining have been finetuned. In the rest of this section, the architectural choices for both sets of parameters will be detailed.

All the transposed convolution upsampling layers were followed by a ReLU [24] non-linearity and initialized with the fan-in plus fan-out initialization scheme described in [18]. The recurrent weight matrices were instead initialized to be orthonormal, following the procedure defined in [38]. We also constrained the stride of the upsampling transposed convolutional layers to be tied to their filter size.

In the segmentation task, each training image carries classification information for all of its pixels. Differently from the image classification task, small batch sizes provide the model with a good amount of information with sufficient variance to learn and generalize well. We experimented with various batch sizes going as low as processing a single image at the time, obtaining comparable results in terms of performance. In our experiments we kept a fixed batch size of 5, as a compromise between train speed and memory usage. In all our experiments, we used L2 regularization [25], also known as weight decay, set to 0.001 to avoid instability at the end of training. We trained all our models with the Adadelta [49] optimization algorithm, for its desired property of not requiring a specific hyperparameter tuning. The effect of Batch Normalization in RNNs has been a focus of attention [27], but it does not seem to provide a reliable improvement in performance, so we decided not to adopt it.

In the experiments, we varied the number of ReNet layers and the number of upsampling transposed convolutional layers, each of them defined respectively by the number of features $d_{\text{RE}}(l)$ and $d_{\text{UP}}(l)$, the size of the input patches (or equivalently of the filters) $ps_{\text{RE}}(l)$ and $fs_{\text{UP}}(l)$.

Method	Global acc	**Avg IoU**
All foreground baseline	25.4	79.9
All background baseline	74.7	0.0
Kernelized structural SVM [5]	94.6	80.1
ReSeg (no VGG)	94.9	79.9
CRF learning [29]	95.7	84.0
PatchCut [47]	95.8	84.0
ReSeg	96.8	**91.6**

Table 1. Weizmann Horses. Per pixel accuracy and IoU are reported.

Method	Global acc	**Avg IoU**
All background baseline	71.0	0.0
All foreground baseline	29.0	29.2
GrabCut [37]	95.9	89.3
Tri-map [46]	96.7	91.7
ReSeg	98	**93.7**

Table 2. Oxford Flowers. Per pixel accuracy and IoU are reported.

Method	Building	Tree	Sky	Car	Sign-Symbol	Road	Pedestrian	Fence	Column-Pole	Side-walk	Bicyclist	Avg class acc	Global acc	**Avg IoU**
Segmentation models														
Super Parsing [44]	**87.0**	67.1	**96.9**	62.7	30.1	95.9	14.7	17.9	1.7	70.0	19.4	51.2	83.3	n/a
Boosting+Higher order [42]	84.5	72.6	**97.5**	72.7	34.1	95.3	34.2	45.7	8.1	77.6	28.5	59.2	83.8	n/a
Boosting+Detectors+CRF [26]	81.5	76.6	96.2	78.7	40.2	93.9	43.0	47.6	14.3	81.5	33.9	62.5	83.8	n/a
Neural Network based segmentation models														
SegNet-Basic (layer-wise training [1])	75.0	84.6	91.2	82.7	36.9	93.3	55.0	37.5	44.8	74.1	16.0	62.9	84.3	n/a
SegNet-Basic [2]	80.6	72.0	93.0	78.5	21.0	94.0	62.5	31.4	36.6	74.0	42.5	62.3	82.8	46.3
SegNet [2]	**88.0**	**87.3**	92.3	80.0	29.5	**97.6**	57.2	**49.4**	27.8	84.8	30.7	65.9	88.6	50.2
ReSeg + Class Balance	70.6	84.6	89.6	81.1	**61.0**	95.1	**80.4**	35.6	**60.6**	**86.3**	**60.0**	73.2	83.5	53.7
ReSeg	86.8	84.7	93.0	**87.3**	48.6	**98.0**	63.3	20.9	35.6	**87.3**	43.5	68.1	88.7	**58.8**
Sub-model averaging														
Bayesian SegNet-Basic [22]	75.1	68.8	91.4	77.7	52.0	92.5	71.5	44.9	52.9	79.1	69.6	70.5	81.6	55.8
Bayesian SegNet [22]	80.4	85.5	90.1	86.4	67.9	93.8	73.8	64.5	50.8	91.7	54.6	76.3	86.9	63.1

Table 3. CamVid. The table reports the per-class accuracy, the average per-class accuracy, the global accuracy and the average intersection over union. The best values and the values within 1 point from the best are highlighted in bold for each column. For completeness we report the Bayesian Segnet models even if they are not directly comparable to the others as they perform a form of model averaging.

Model	ps_{RE}	d_{RE}	fs_{UP}	d_{UP}	Building	Tree	Sky	Car	Sign-Symbol	Road	Pedestrian	Fence	Column-Pole	Side-walk	Bicyclist	Avg class acc	Global acc	**Avg IoU**
ReSeg + LCN	$(2\times2),(1\times1)$	(100, 100)	(2×2)	(50, 50)	81.5	80.3	**94.7**	78.1	42.8	**97.4**	53.5	34.3	36.8	68.9	47.9	65.1	84.8	52.6
ReSeg + Class Balance	$(2\times2),(1\times1)$	(100, 100)	(2×2)	(50, 50)	70.6	**84.6**	89.6	81.1	**61.0**	95.1	**80.4**	**35.6**	**60.6**	**86.3**	**60.0**	73.2	83.5	53.7
ReSeg	$(2\times2),(1\times1)$	(100, 100)	(2×2)	(50, 50)	**86.8**	**84.7**	93.0	**87.3**	48.6	**98.0**	63.3	20.9	35.6	**87.3**	43.5	68.1	88.7	**58.8**

Table 4. Comparison of the performance of different hyperparameter on CamVid.

4.3. Results

In Table 1, we report the results on the Weizmann Horse dataset. On this dataset, we verified the assumption that processing the input image with some pre-trained convolutional layers from VGG-16 could ease the learning. Specifically, we restricted ourselves to only using the first 7 convolutional layers from VGG, as we only intended to extract some low-level generic features and learn the task-specific high-level features with the ReNet layers. The results indeed show an increase in terms of average Intersection over Union (*IoU*) when these layers are being used, confirming our hypothesis.

Table 2 shows the results for Oxford Flowers dataset, when using the full ReSeg architecture (i.e., including VGG convolutional layers). As shown in the table, our method clearly outperforms the state-of-the-art both in terms of global accuracy and average IoU.

Table 3 presents the results on CamVid dataset using the full ReSeg architecture. Our model exhibits state-of-the-art performance in terms of IoU when compared to both standard segmentation methods and neural network based methods, showing an increase of 17% w.r.t. to the recent SegNet model. It is worth highlighting that incorporating sub-model averaging to SegNet model, as in [22], boosts the original model performance, as expected. Therefore, introducing sub-model averaging to ReSeg would also presumably result in significant performance increase. However, this remains to be tested.

5. Discussion

As reported in the previous section, our experiments on the Weizmann Horse dataset show that processing the input images with some layers of VGG-16 pre-trained network improves the results. In this setting, pre-processing the input with Local Contrast Normalization (LCN) does not seem to give any advantage (see Table 4). We did not use any other kind of pre-processing.

While on both the Weizmann Horse and the Oxford Flowers datasets we trained on a binary background/foreground segmentation task, on CamVid we addressed the full semantic segmentation task. In this setting, when the dataset is highly imbalanced, the segmentation performance of some classes can drop significantly as the network tries to maximize the score on the high-occurrence classes, *de facto* ignoring the low-occurrence ones. To overcome this behaviour, we added a term to the cross-entropy loss to bias the prediction towards the low-occurrence classes. We use *median frequency balancing* [14], which re-weights the class predictions by the ratio between the median of the frequencies of the classes (computed on the training set) and the frequency of each class. This increases the score of the low frequency classes (see Table 4) at the price of a more noisy segmentation mask, as the probability of the underrepresented classes is overestimated and can lead to an increase in misclassified pixels in the output segmentation mask, as shown in Figure 3.

On all datasets we report the per-pixel accuracy (*Global acc*), computed as the percentage of true positives w.r.t. the total number of pixels in the image, and the average per-class Intersection over Union (*Avg IoU*), computed on each class as true positive divided by the sum of true positives, false positives and false negatives and then averaged. In the full semantic segmentation setting we also report the per-class accuracy and the average per-class accuracy (*Avg class acc*).

6. Conclusion

We introduced the ReSeg model, an extension of the ReNet model for image semantic segmentation. The proposed architecture shows state-of-the-art performances on CamVid, a widely used dataset for urban scene semantic segmentation, as well as on the much smaller Oxford Flowers dataset. We also report state-of-the-art performances on the Weizmann Horses.

In our analysis, we discuss the effects of applying some layers of VGG-16 to process the input data, as well as those of introducing a class balancing term in the cross-entropy loss function to help the learning of under-represented classes. Notably, it is sufficient to process the input images with just a few layers of VGG-16 for the ReSeg model to gracefully handle the semantic segmentation task, confirming its ability to encode contextual information and long term dependencies.

Acknowledgments

We would like to thank all the developers of Theano [4, 3] and in particular Pascal Lamblin, Arnaud Bergeron and Frédéric Bastien for their dedication. We are also thankful to César Laurent for the moral support and to Vincent Dumoulin for the insightful discussion on transposed convolutions. We are also very grateful to the developers of Lasagne [12] for providing a light yet powerful framework and to the reviewers for their valuable feedback. We finally acknowledge the support of the following organizations for research funding and computing support: NSERC, IBM Watson Group, IBM Research, NVIDIA, Samsung, Calcul Québec, Compute Canada, the Canada Research Chairs and CIFAR. F.V. was funded by the AI*IA Young Researchers Mobility Grant and the Politecnico di Milano PHD School International Mobility Grant.

References

[1] V. Badrinarayanan, A. Handa, and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling.

[2] V. Badrinarayanan, A. Handa, and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. page 5, 2015.

[3] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio. Theano: new features and speed improvements. Submited to the Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[4] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, 2010.

[5] L. Bertelli, T. Yu, D. Vu, and B. Gokturk. Kernelized structural svm learning for supervised object segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2153–2160. IEEE, 2011.

[6] E. Borenstein. Combining top-down and bottom-up segmentation. In *In Proceedings IEEE workshop on Perceptual Organization in Computer Vision, CVPR*, page 46, 2004.

[7] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009.

[8] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki. Scene labeling with lstm recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3547–3555, 2015.

[9] L.-C. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille. Semantic image segmentation with task-specific

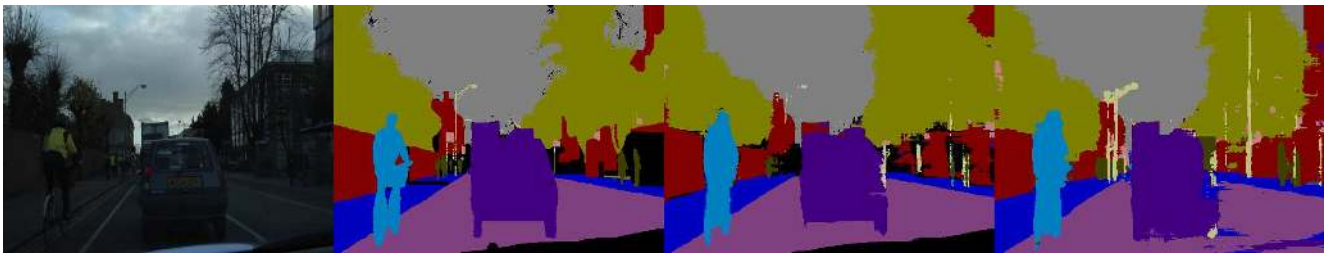

Figure 3. Camvid segmentation example with and without class balancing. From the left: input image, ground truth segmentation, ReSeg segmentation, ReSeg segmentation with class balancing. Class balancing improves the low frequency classes as e.g., the street lights, at the price of a worse overall segmentation.

edge detection using cnns and a discriminatively trained domain transform. *arXiv preprint arXiv:1511.03328*, 2015.

[10] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, Oct. 2014. to appear.

[11] 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.

[12] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri, D. Maturana, M. Thoma, E. Battenberg, J. Kelly, J. D. Fauw, M. Heilman, diogo149, B. McFee, H. Weideman, takacsg84, peterderivaz, Jon, instagibbs, D. K. Rasul, CongLiu, Britefury, and J. Degrave. Lasagne: First release., Aug. 2015.

[13] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning, 2016. cite arxiv:1603.07285.

[14] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *CoRR*, abs/1411.4734, 2014.

[15] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 2155–2162, Washington, DC, USA, 2014. IEEE Computer Society.

[16] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE TPAMI*, 35(8):1915–1929, 2013.

[17] C. Gatta, A. Romero, and J. van de Weijer. Unrolling loopy top-down semantic feedback in convolutional deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2014, Columbus, OH, USA, June 23-28, 2014*, pages 504–511, 2014.

[18] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256, 2010.

[19] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

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

[21] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic. Generating images with recurrent adversarial networks. *arXiv preprint arXiv:1602.05110*, 2016.

[22] A. Kendall, V. Badrinarayanan, and R. Cipolla. Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding. 2015.

[23] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25 (NIPS'2012)*. 2012.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[25] A. Krogh and J. A. Hertz. A simple weight decay can improve generalization. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 4*, pages 950–957. Morgan Kaufmann, 1992.

[26] L. Ladický, P. Sturgess, K. Alahari, C. Russell, and P. H. S. Torr. What, where and how many? Combining object detectors and CRFs. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6314 LNCS(PART 4):424–437, 2010.

[27] C. Laurent, G. Pereyra, P. Brakel, Y. Zhang, and Y. Bengio. Batch normalized recurrent neural networks. *CoRR*, abs/1510.01378, 2015.

[28] M. Lin, Q. Chen, and S. Yan. Network in network. In *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)*, Apr. 2014.

[29] F. Liu, G. Lin, and C. Shen. Crf learning with cnn features for image segmentation. *Pattern Recognition*, 2015.

[30] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CVPR (to appear)*, Nov. 2015.

[31] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[32] M.-E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1447–1454, 2006.

[33] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. *arXiv preprint arXiv:1505.04366*, 2015.

[34] P. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene labeling. *JMLR*, 1(32):82–90, 2014.

[35] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[36] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]).

[37] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG)*, 23(3):309–314, 2004.

[38] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)*, Apr. 2014.

[39] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[40] G. Singh and J. Kosecka. Nonparametric scene parsing with adaptive feature relevance and semantic context. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 3151–3157, 2013.

[41] M. F. Stollenga, W. Byeon, M. Liwicki, and J. Schmidhuber. Parallel multi-dimensional lstm, with application to fast biomedical volumetric image segmentation. In *Advances in Neural Information Processing Systems*, pages 2980–2988, 2015.

[42] P. Sturgess, K. Alahari, L. Ladicky, and P. H. S. Torr. Combining Appearance and Structure from Motion Features for Road Scene Understanding. *Procedings of the British Machine Vision Conference 2009*, pages 62.1–62.11, 2009.

[43] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[44] J. Tighe and S. Lazebnik. Superparsing: Scalable nonparametric image parsing with superpixels. *International Journal of Computer Vision*, 101(2):329–349, 2013.

[45] F. Visin, K. Kastner, K. Cho, M. Matteucci, A. Courville, and Y. Bengio. Renet: A recurrent neural network based alternative to convolutional networks. *arXiv preprint arXiv:1505.00393*, 2015.

[46] X. Wu and K. Kashino. Tri-map self-validation based on least gibbs energy for foreground segmentation. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014.

[47] J. Yang, B. Price, S. Cohen, Z. Lin, and M.-H. Yang. Patchcut: Data-driven object segmentation via local shape transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1770–1778, 2015.

[48] M. Zeiler, G. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *Proc. International Conference on Computer Vision (ICCV'11)*, pages 2146–2153. IEEE, 2011.

[49] M. D. Zeiler. ADADELTA: an adaptive learning rate method. Technical report, arXiv 1212.5701, 2012.

[50] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV'14*, 2014.