This addendum shows an extension to the original work in [Houthooft et al., 2016] and [Houthooft and De Turck, 2016], done between the years 2014 and 2016, which could not be released publicly at the time. We compare an end-to-end segmentation method with a convolutional implementation of the end-to-end SSVM method described in this chapter. We call this implementation a *deep SSVM*, and apply it to a newly constructed autonomous agricultural vehicle segmentation dataset.

Data Acquisition

As part of this use case, an autonomous agricultural vehicle segmentation dataset was created in collaboration with Case New Holland (CNH) Industrial. Camera video data was captured originating from ten different recording campaigns in the Styria region and Amstetten area in Austria, the Bologna province in Italy, West-Flanders in Belgium, Pinal County in Arizona, USA, the Indre and Lot-et-Garonne departments in France, the Thuringia region in Germany, and the North Brabant region in The Netherlands. To capture the data, different vehicles were equipped with GoPro Hero 3+ Black cameras, set to a resolution of 1920×1080 pixels and a frame rate of 30 Hz. One of the vehicles used in the campaigns is shown in Figure 1. A wide-angle field-of-view was chosen to capture additional surroundings using the following camera angle settings: 69.5° vertical, 118.2° horizontal, and 133.6° diagonal, with a focal length of 14 mm. The resulting fisheye effect was corrected post-hoc through an inverse transformation.

Figure 1: One of the vehicles used to capture the dataset images; cameras are mounted on the roof top center and mirrors.

The different captured videos were split into image sequences with a frequency of 1 Hz. From this set of images, for each measurement campaign, the most visibly informative and distinctive ones in terms of objects and perspective were selected. This resulted in a dataset of 595 training and 149 test images, which were labeled by an external party that was given labeling guidelines. This dataset is being continuously expanded. For this task, a labeling tool was built that allows

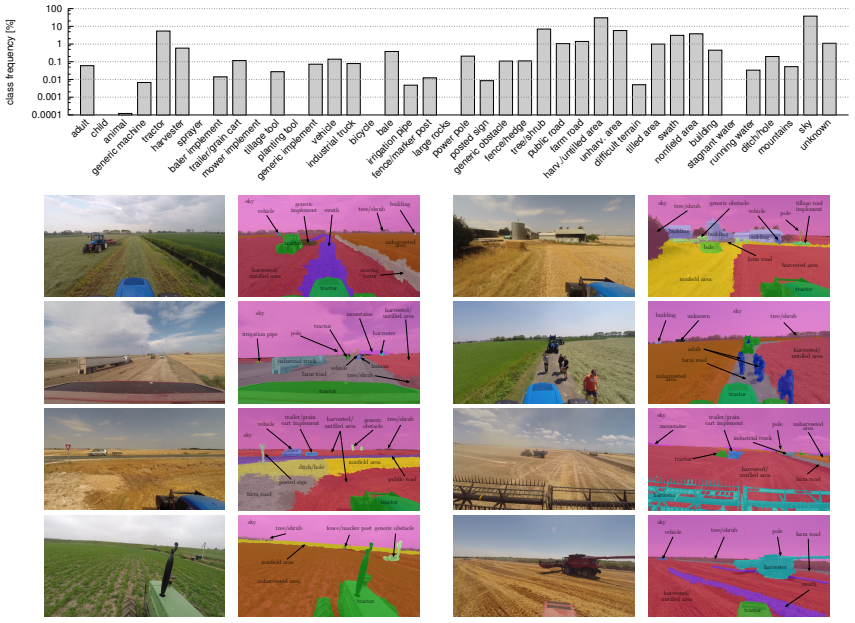

Figure 2: Illustation of dataset images (columns 1 and 3) and corresponding manual labelings (columns 2 and 4); class list and corresponding occurrence frequencies in the dataset (top histogram)

operators to load an image, segment it into over-segmentation regions, and assign classes to these regions. The dataset is illustrated in Figure 2, which shows the list of classes and their occurrence frequencies, as well as some representative images together with their corresponding ground truth labelings.

Convolutional Unary Classifier

A first extension proposes to use an alternative method for classifying individual superpixels. Instead of using a classifier that takes as input only region bag-of-word features, a convolutional neural network (CNN) is used to learn representations of each superpixel. The CNN is trained in a supervised fashion, taking $3 \times 84 \times 84$ square windows around the median center of each superpixel as input, and outputs a probability distribution over all possible classes. This CNN has the following architecture. First, the $3 \times 84 \times 84$ windows are downscaled 3-channel 64×64 patches. Next, a convolutional layer of 16 5×5 filters is used, after which a 2-dim max-pooling operation downsamples the 16 feature maps to 32×32 pixels. Hereafter, 2 additional convolutional layers with max-pooling operations are applied, with 32 and 64 3×3 filters respectively. Hereafter, 2 subsequent dense layers of 1024 units are used, which end in an 18-bin softmax output layer. Additionally, the (x, y) position of the superpixel median centers are added to the first dense layer, together with the previously mentioned bag-of-word features (450-dim). The CNN architecture is visually described in Figure 3.

Adam [Kingma and Ba, 2015] is used as a learning scheme to minimize the

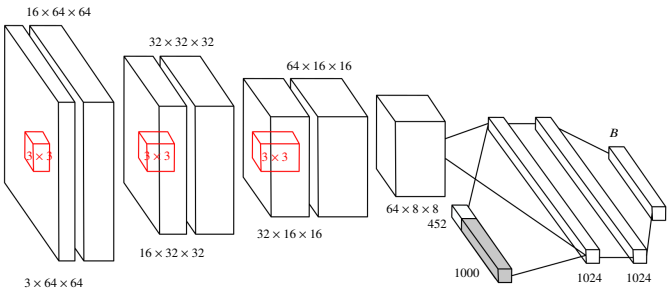

Figure 3: The convolutional neural network architecture used for the unary predictor which takes as input windows centered around each superpixel

cross-entropy, using minibatches of size 32. The cross-entropy term is weighted by the inverse of the square root of the class frequencies to correct for class imbalance. Batch normalization [Ioffe and Szegedy, 2015] is applied to every non-output layer; the layer weights are initialized according to [Glorot and Bengio, 2010]; a dropout [Srivastava et al., 2014] rate of 50% is applied to both dense layers; all nonlinearities are exponential linear units (ELUs) [Clevert et al., 2015]; label smoothing of 0.0003 is applied to the softmax output. Furthermore, the dataset is augmented through random mirroring over the vertical axis. After the unary convolutional classifier has been trained, the 2-table SSVM is trained as described previously, using the CNN softmax ouput probabilities as input features. Recently, related methods have been proposed by [Mostajabi et al., 2015].

Transfer learning To enhance the accuracy of the proposed unary classification model, transfer learning is used through leverage of a CNN that was pretrained for classification on the ImageNet dataset [Russakovsky et al., 2015]. In particular, we use the OverFeat CNN model [Sermanet et al., 2013], which classifies 3×232×232 images into 1000 distinct ImageNet classes. Although the majority of these classes do not correspond the classes in our dataset, the goal is to reuse certain learned features in the segmentation model. This can aid in lowering the overfitting behavior caused by the bias in the proposed agricultural dataset to particular types of images. The pretrained CNN model is applied to the same windows as used by the classifier described in the previous paragraph. The $3 \times 84 \times 84$ windows are scaled up to $3 \times 232 \times 232$ to match the pretrained CNN input dimensions. The class probabilities that are outputted for each superpixel act as an additional 1000-dim feature vector, which is fed into the first dense layer of the CNN model described in the previous paragraph. This is shown in Figure 3 as the gray-colored block.

End-to-end Segmentation

A second extension avoids the use of an SSVM model completely. Here, a CNN is modeled to take as input a complete $3 \times 640 \times 320$ image in order to output a full segmentation. This avoids the need for an over-segmentation preprocessing method. Its advantage is that the segmentations are no longer limited by errors

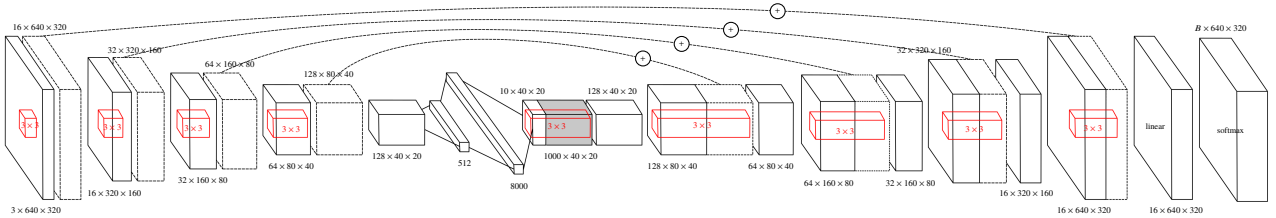

Figure 4: End-to-end convolutional segmentation architecture; the circled plus connections represent the skip-layer connections; the gray box represents the OverFeat CNN class probabilities according to the window extraction process.

made by the SLIC procedure. A possible disadvantage is that less prior structure is imposed through both the superpixel method and the SSVM connectivity graph. The proposed model is a type of convolutional autoencoder with the following architecture, which is also visually described in Figure 4. First a stack of convolutional layers is applied to the 3-channel input images, composed of respectively 16, 32, 64, and 128 filters of size 3×3. After each convolutional layer, a 2-dim max-pooling operation is applied, halving the size of each set of feature maps.

After the convolutional stack, 2 dense layers of respectively 512 and 8000 units are used, which feed into the stack of transposed convolutional layers. This stack first reshapes the set of 8000 units into a tensor of size $10 \times 20 \times 40$, after which layers of respectively 128, 64, 32, and 16 filters of size 3×3 are used. After each of these layers, a 2-dim upscaling layer is applied, doubling the size of each feature map. The final feature map is fed into a convolutional layer of 16 3×3 filters, after which a linear transformation is applied, which connects to a final 18-bin softmax layer.

To ensure sufficiently detailed segmentations, skip-layer connections are added that connect the convolutional layers, before their max-pooling operation, to their transposed convolutional counterparts. As such, the input of each transposed convolutional layer is expanded through concatenation. This is made clear in Figure 4 by the horizontal dashed connections with the circled plus symbol. All nonlinear transformations are composed of ELUs [Clevert et al., 2015]; the learning scheme Adam [Kingma and Ba, 2015] optimizes the cross-entropy augmented with L_2-regularization using minibatches of size 4. The cross-entropy is weighted by the inverse of the square root of the class frequencies. The training set is augmented through random image mirroring over the vertical axis. Recently a related method has been proposed by [Long et al., 2015] and [Ronneberger et al., 2015].

Transfer learning Transfer learning is used to leverage of information learned on the ImageNet [Russakovsky et al., 2015] dataset. Again, we make use of the OverFeat pretrained CNN model [Sermanet et al., 2013]. However, this time the model is slided with a fixed stride over 2x upscaled input images and applied to the resulting $3 \times 232 \times 232$ windows. This transformation results in a 1000-dim class probability vector extracted along a grid of 20×40 points. As such, a tensor of size $1000 \times 20 \times 40$ is formed, which is concatenated to the reshaped tensor of the 8000 units of the second dense layer, resulting in a feature map of size $1010 \times 20 \times 40$, shown in Figure 4 by the gray-colored block. The transposed convolutional stack of layers can now make use of transferred external knowledge in order to make more informed decisions about the segmentation outputs.

Results and Discussion

Quantitative results for both models on the test dataset are shown in Figures 6 and 5 through confusion matrices. The total set of classes in the dataset was translated into a set of 18 most interesting class groups. In these matrices, each row i represents pixels that have a ground truth label of class i, while each column j represents predictions made by the model as class j. Therefore, in a confusion matrix, at position (i, j), the number of pixels classified as j but actually belonging to class i is shown. In this work, rather than showing the actual pixel counts at each

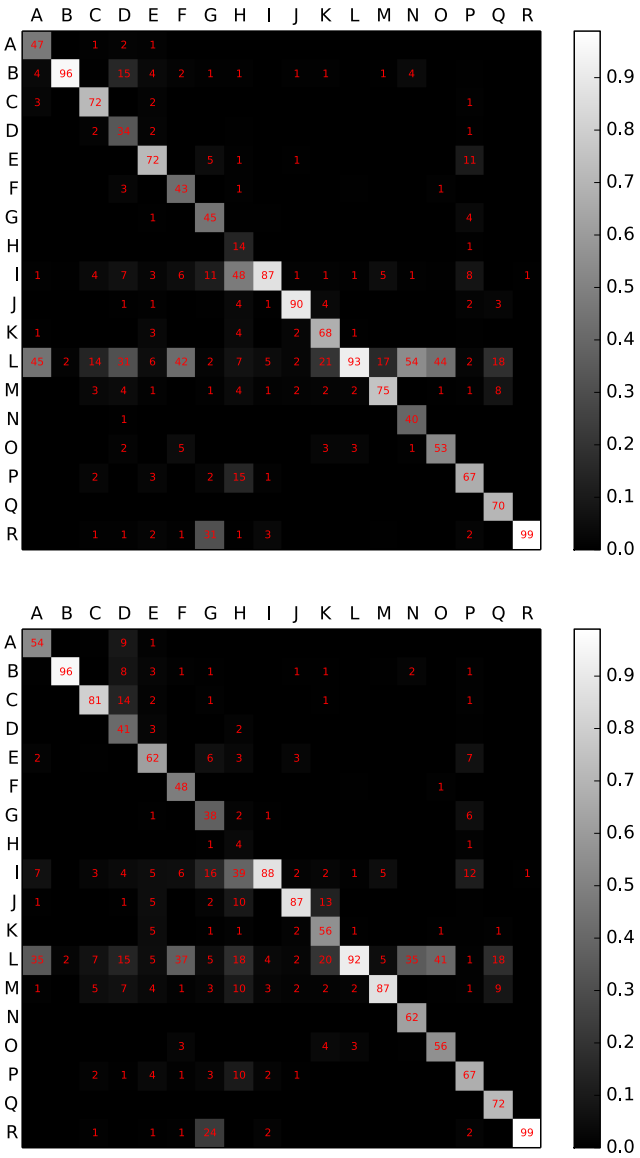

Figure 5: SSVM with a CNN unary classifier (top) and end-to-end segmentation model (bottom): pixel-wise precision results on the autonomous agricultural vehicle test dataset, described as a confusion matrix. Class legend: person (A), tractor (B), harvester (C), implement (D), moving object (E), nonmoving object (F), power pole (G), fence/hedge (H), tree/shrubbery (I), public road (J), farm road (K), harvested untilled area (L), unharvested area (M), tilled area (N), swath (O), building (P), water (Q), sky (R).

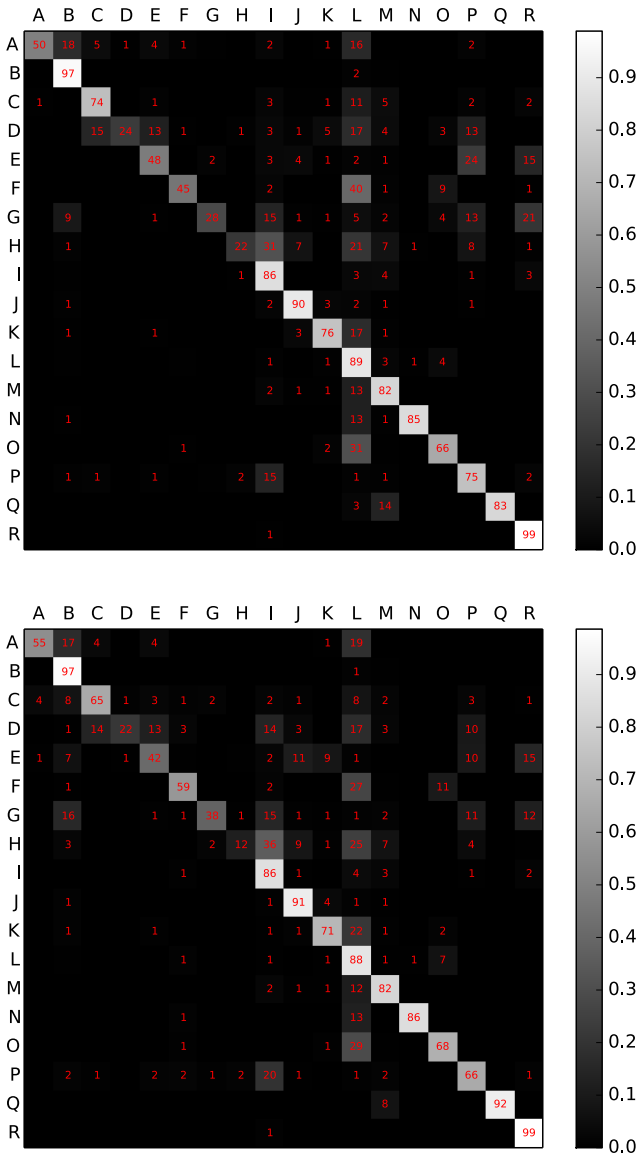

Figure 6: SSVM with a CNN unary classifier (top) and end-to-end segmentation model (bottom): pixel-wise recall results on the autonomous agricultural vehicle test dataset, described as a confusion matrix. Class legend: person (A), tractor (B), harvester (C), implement (D), moving object (E), nonmoving object (F), power pole (G), fence/hedge (H), tree/shrubbery (I), public road (J), farm road (K), harvested untilled area (L), unharvested area (M), tilled area (N), swath (O), building (P), water (Q), sky (R).

position (i, j), we show this count divided by the row sum (in %), which forms the recall confusion matrix. Hence, the values in each row sum to 100%[1]. The diagonal values of the matrix can be interpreted as the per-class accuracy values, while the off-diagonal values represent the percentage of pixels (belonging to a particular class) mispredicted as another class. If we would interpret the ground truth labels as model predictions, a matrix would be obtained with only values of 100% on its diagonal, and 0% everywhere else. Moreover, we show the precision confusion matrix, which is composed of the actual pixel counts divided by the column sum (in %).

These results reveal that the mistakes made by both methods are logically interpretable, e.g., mistaking types of land or types of vehicle, while their accuracy results are very similar. When looking at the average accuracy values, the unary classification model achieves an average precision of 64.7% and an average recall of 67.7%. The end-to-end model achieves an average precision of 62.6% and an average recall of 67.7%. The global pixel-wise accuracy, which is the fraction of correctly classified pixels, is 91.2% for the SSVM method and 90.9% for the end-to-end method.

Since the quantitative results of both methods are very similar, illustrative qualitative results on the test dataset are shown in Figure 7. On the one hand, these results reveal that the SSVM method tends to overshoot when the over-segmentations are inaccurate, while the end-to-end method suffers from no such limitation. On the other hand, when the superpixels do align, they nicely delineate objects, leading to highly accurate segmentations. This is true even when the actual ground truth labels do not correctly delineate the objects. Although the quantitative results of both models are very similar, it can be noticed that the end-to-end segmentation method is capable of correctly predicting objects that are very far away, which is its main advantage over the over-segmentation method.

The results of both models highlight their capability to operate in highly cluttered environments by achieving accurate image segmentations. Such predictions could form a basis for steering autonomous vehicles towards particular types of land, e.g., unharvested fields, or to avoid dangerous areas, e.g., public roads. The accurate segmentation of trees and shrubbery can aid in marking field boundaries or in avoiding collisions, as does the segmentation of objects in general, e.g., power poles. Segmenting sky regions can prove to be helpful in horizon estimation, or to avoid running other expensive algorithms on image regions of which we know a priori that they do not contain important information, e.g., when trying to localize particular objects.

Deep Structural Support Vector Machine

We propose the use of an SSVM with an underlying grid-structured graphical model. This model connects each pixel to its four adjacent neighbors in the left, top, right, and bottom directions. It uses a unary energy function $f(x, y; \theta)$ modeled by a convolutional neural network (CNN) which outputs an energy value for each distinct class. The interaction energy function $h(x, y; \gamma)$ is modeled by the same CNN, except that it uses a different output branch that ends in two distinct output blocks of 18^2 values. Once 18^2 outputs for the horizontal connections (left and

[1]Note that due to rounding errors in the figures, the sum may not be exactly 100%.

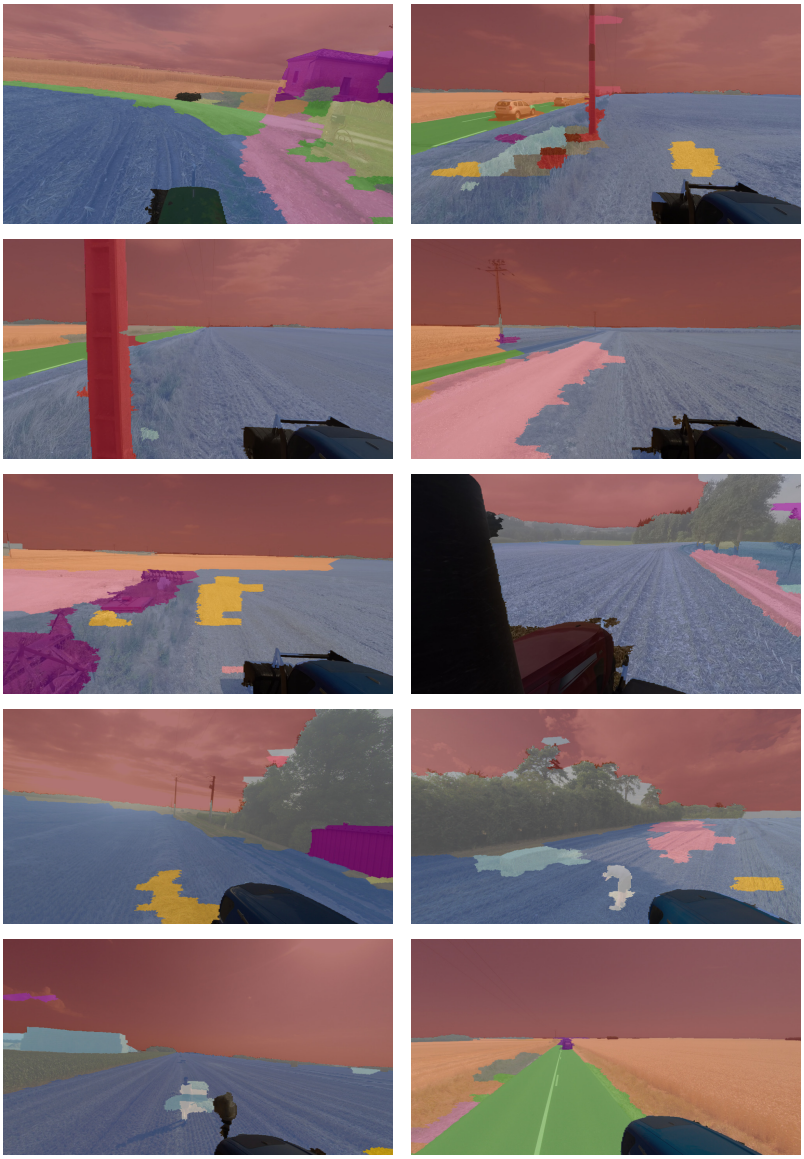

Figure 7: SSVM combined with the unary CNN classifier: illustration of segmented test images from the autonomous agricultural vehicle dataset; visible class legend: public road (green), sky (red), private road (pink), unharvested area (orange), human (white), tractor (black), building or implement (purple), swath (yellow), harvester (bright yellow), harvested area (blue), trees/shrubbery (gray), generic nonmoving object (cyan), power pole (dark red), car (dark orange), fence/hedge (khaki)

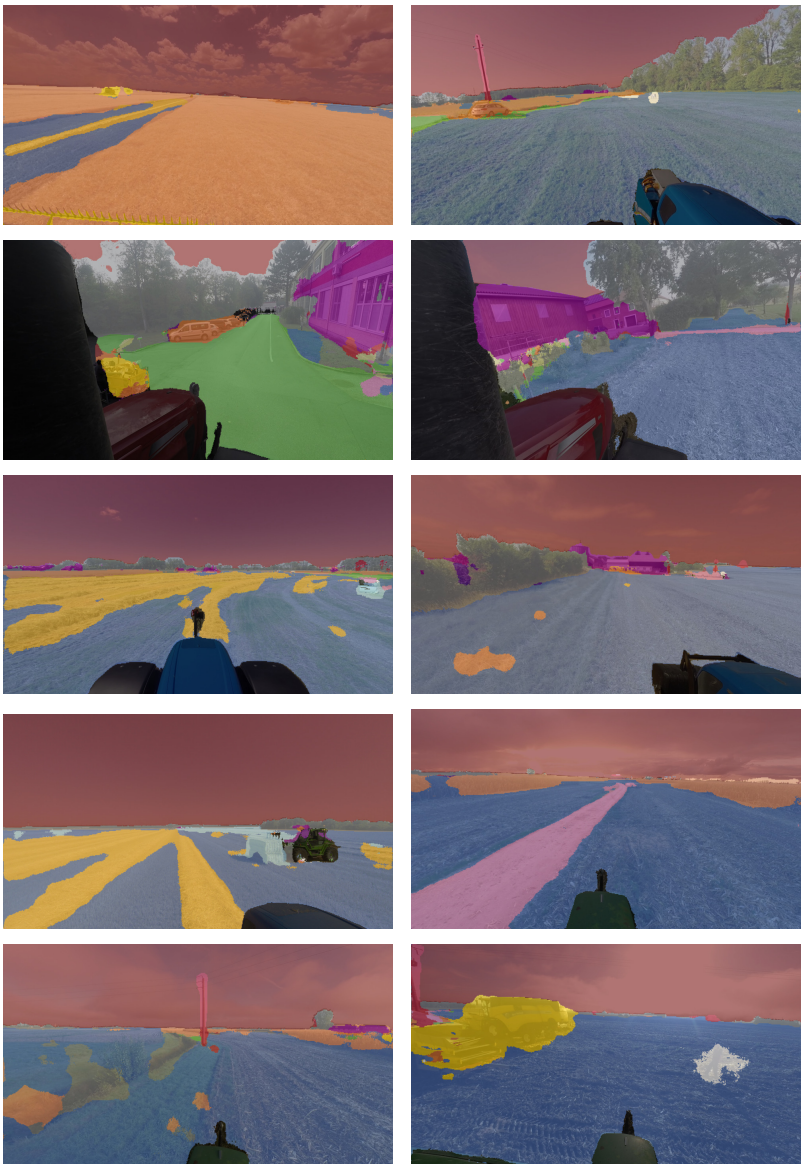

Figure 8: End-to-end segmentation: illustration of segmented test images from the autonomous agricultural vehicle dataset; visible class legend: public road (green), sky (red), private road (pink), unharvested area (orange), human (white), tractor (black), building or implement (purple), swath (yellow), harvester (bright yellow), harvested area (blue), trees/shrubbery (gray), generic nonmoving object (cyan), power pole (dark red), car (dark orange), fence/hedge (khaki)

right), once 18^2 outputs for the vertical connections (top and bottom). The right column and bottom row are stripped away from both branch outputs, to match the actual grid layout. The CNN factors are composed of stack of convolutional layers, feeding into a set of dense layers, feeding into a stack of transposed convolutional layers, which output the SSVM energy values. The whole system is trained end-to-end using back-propagation and α-expansion for loss-augmented inference. This means, rather than applying the unary and pairwise neural networks as presented in this chapter separately for each individual unary or pairwise connection, that the CNN outputs all SSVM energy values simultaneously.

The model has the following architecture, which is also shown visually in Figure 9. The SSVM graphical model is a grid that connects adjacent pixels, as described previously, consisting of $V_n = 160 \times 80$ nodes and $E_n = (160 \times 79) + (159 \times 160)$ undirected edges. It takes as input $3 \times 640 \times 320$ images downscaled to $3 \times 160 \times 80$, and outputs a 160×80 segmentation. This means that both the unary features x_i^U as well as the interaction features x_i^I are the complete 3-channel input images. We specifically use downscaled images of 160×80 pixels to reduce the processing time, since our α-expansion algorithm implementation cannot be run parallely on a GPU. A possible solution to this is described in [Vineet and Narayanan, 2009].

The CNN consists of consecutive convolutional layers with 16, 32, and 64 filters of size 3×3. After each layer, a 2-dim max-pooling operation is applied, halving the size of each set of feature maps. Next, 2 dense layers of respectively 512 and 2000 units are used, which feed into the stack of transposed convolutional layers. The set of 2000 dense outputs is first reshaped into a tensor of size $10 \times 10 \times 20$, after which layers of respectively 128, 32, and 32 filters of size 3×3 are used. Each of these layers is followed by a 2-dim upscaling layer, doubling the size of each feature map.

The final feature map is fed into two parallel transposed convolutional layers, one with 18 3×3 filters and one with 2×18^2 filters. The first output branch feeds into the affine unary energy layer, with output size $18 \times 160 \times 80$, while the second branch feeds into an affine interaction energy layer, with output size $324 \times 160 \times 80$ (which is cropped to match the actual interaction grid). Skip-layer connections are added that connect the convolutional layers, before their max-pooling operation, to their transposed convolutional counterparts. As such, the input of each transposed convolutional layer is expanded through concatenation. This is made clear in Figure 9 by the horizontal dashed connections with the circled plus symbol. All nonlinear transformations are composed of ELUs [Clevert et al., 2015]; the learning scheme Adam [Kingma and Ba, 2015] optimizes the proposed SSVM objective function, using minibatches of size 6. The loss function $\Delta(\cdot, \cdot)$ is weighted by the inverse of the class frequencies. Transfer learning is used through concatenation of the $1000 \times 20 \times 40$ feature tensor with tensor of feature maps after the first upscaling layer.

Results and Discussion

This section demonstrates that the presented end-to-end SSVM training method can be used in conjunction with highly complex underlying neural factor models. The prediction accuracy between the deep SSVM and an end-to-end segmentation

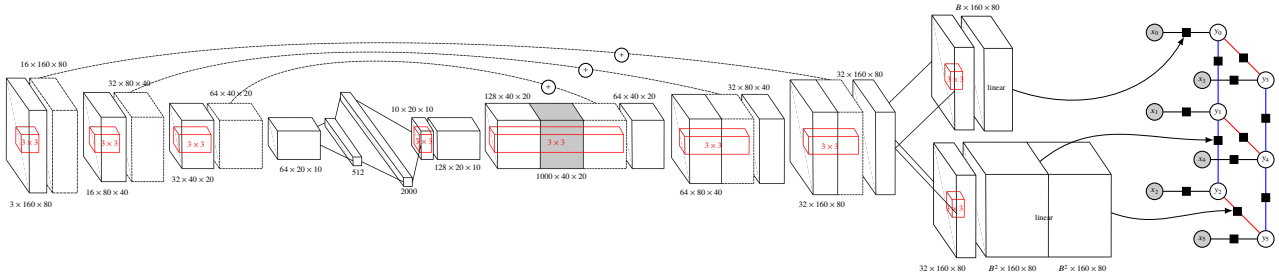

Figure 9: Deep SSVM using a convolutional architecture; the circled plus connections represent the skip-layer connections; the gray box represents the OverFeat CNN class probabilities according to the window extraction process.

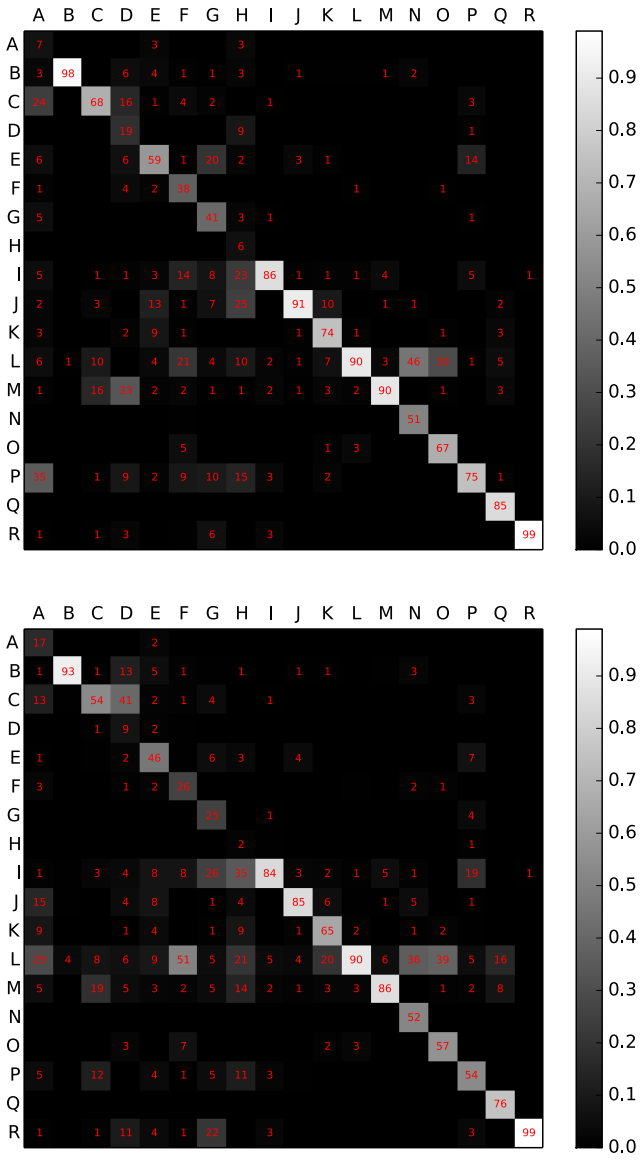

Figure 10: Deep SSVM (top) and end-to-end segmentation model (bottom): pixel-wise precision results on the autonomous agricultural vehicle test dataset, described as a confusion matrix. Class legend: person (A), tractor (B), harvester (C), implement (D), moving object (E), nonmoving object (F), power pole (G), fence/hedge (H), tree/shrubbery (I), public road (J), farm road (K), harvested untilled area (L), unharvested area (M), tilled area (N), swath (O), building (P), water (Q), sky (R).

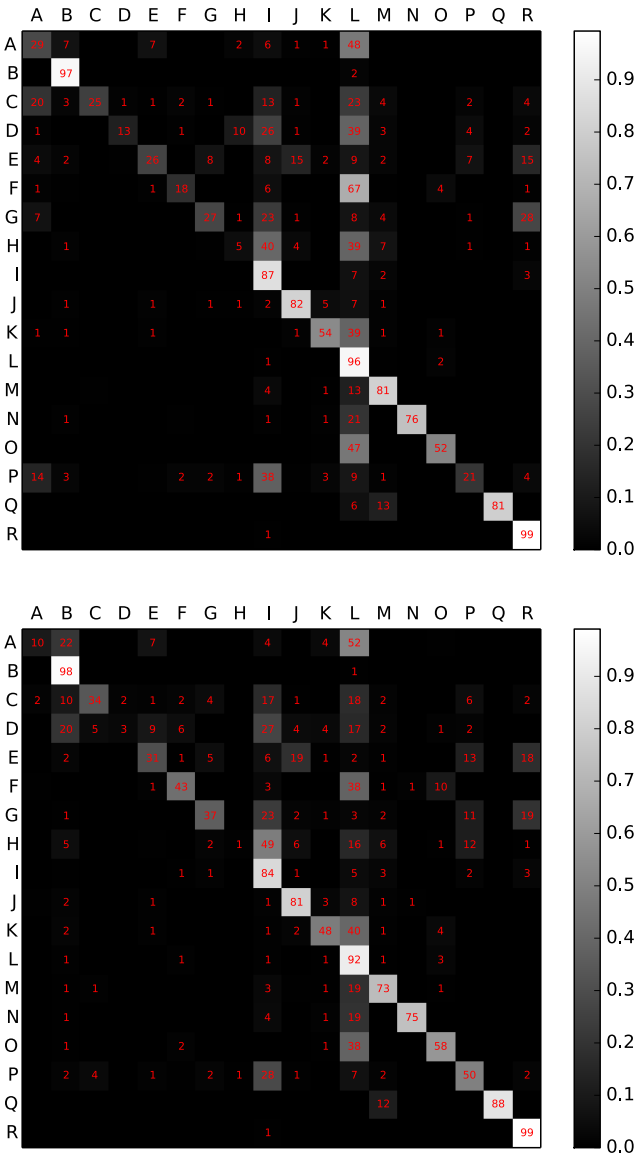

Figure 11: Deep SSVM (top) and end-to-end segmentation model (bottom): pixel-wise recall results on the autonomous agricultural vehicle test dataset, described as a confusion matrix. Class legend: person (A), tractor (B), harvester (C), implement (D), moving object (E), nonmoving object (F), power pole (G), fence/hedge (H), tree/shrubbery (I), public road (J), farm road (K), harvested untilled area (L), unharvested area (M), tilled area (N), swath (O), building (P), water (Q), sky (R).

model that uses the exact same convolutional base (but without the lower branch, the graphical model (SSVM), and with a softmax output of 18 classes rather than an affine transformation (linear) into 18 unary energy values) is compared through precision and recall on the autonomous agricultural test dataset. The deep SSVM is compared to an end-to-end segmentation model with the same architecture as the deep SSVM.

The results are shown in Figures 10 and 11 through confusion matrices. The mistakes made by both models are highly interpretable, for example incorrectly classifying 'fence/hedge' as 'tree/shrubbery'. On average, it can be noticed that the deep SSVM method attains a slightly lower recall (53.3% compared to 55.8%), but a higher precision (63.6% compared to 56.7%) than the end-to-end segmentation model. This means that if the deep SSVM model makes a prediction, on average, it is more likely to be correct than its competitor. However, this leads to a slightly lower detection rate for particular classes. When looking at the global accuracy, which is the total number of correctly classified pixels, the end-to-end segmentation model attains an accuracy of 90.5%, while the deep SSVM scores slightly higher with 92.3%. These results indicate that the deep SSVM model is capable of adding value to the segmentation process. However, the most important point to take away from these results is that they reinforce the conclusion made previously, namely that the end-to-end SSVM training method can serve as a foundation for highly complex underlying models.

References

[Clevert et al., 2015] Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289*.

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference Artificial Intelligence and Statistics (AISTATS)*, pages 249–256.

[Houthooft et al., 2016] Houthooft, R., De Boom, C., Verstichel, S., Ongenae, F., and De Turck, F. (2016). Structured output prediction for semantic perception in autonomous vehicles. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*.

[Houthooft and De Turck, 2016] Houthooft, R. and De Turck, F. (2016). Integrated inference and learning of neural factors in structural support vector machines. *Pattern Recognition*, 59:292–301.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 448–456.

[Kingma and Ba, 2015] Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440.

[Mostajabi et al., 2015] Mostajabi, M., Yadollahpour, P., and Shakhnarovich, G. (2015). Feedforward semantic segmentation with zoom-out features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3376–3385.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the 18th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

[Sermanet et al., 2013] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). OverFeat: Integrated recognition, localization and detection using convolutional networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[Srivastava et al., 2014] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

[Vineet and Narayanan, 2009] Vineet, V. and Narayanan, P. (2009). Solving multilabel MRFs using incremental α-expansion on the GPUs. In *Proceedings of the 9th Asian Conference on Computer Vision (ACCV)*, pages 633–643.