How to transfer a semantic segmentation model from autonomous driving to other domains?

Ana B. Cambra, Adolfo Muñoz, and Ana C. Murillo

DIIS-I3A. University of Zaragoza, Spain,
acambra@unizar.es, adolfo@unizar.es, acm@unizar.es

Abstract. Semantic scene understanding is an important task for robots operating autonomously in real-world applications. Recent deep convolutional neural networks (CNNs) have demonstrated to be an effective approach for semantic image segmentation, especially for tasks where plenty of labeled data is available. However, many applications need to learn new specific classes but do not have a lot of labeled training data. This paper addresses the problem of transferring the knowledge from existing CNN models, e.g., from autonomous driving applications, to different classes and domains, e.g., different robotic platforms. Our work explores the two common transfer learning approaches for the particular problem of semantic segmentation: 1) fine-tuning existing models with the new training data, following a standard pipeline; 2) training a superpixel classifier using our proposed superpixel representation, which combines local and context information. We evaluate both approaches on three varied binary segmentation use cases from different domains. Our experiments demonstrate advantages and limitations from each alternative, showing that the proposed superpixel based strategies learn better models with limited amounts of labeled data.

Keywords: semantic segmentation, deep learning, superpixel segmentation

1 Introduction

Semantic scene understanding is crucial for autonomous systems to operate in real-world scenarios. Applications such as autonomous driving have already shown to benefit from an initial semantic segmentation of the scene, e.g., to locate drivable regions or identify obstacles [20, 12]. However, available semantic segmentation models are often targeted to a particular task and then the set of pre-defined classes is specific for such task. This paper studies alternatives to adapt existing models (e.g., trained for autonomous driving scenes) to segment new semantic classes not included in the original model (e.g, for aerial scenes or indoor service robotic scenarios) as Fig. 1 illustrates.

In the last years, CNNs have pushed the state-of-the-art on a broad variety of visual recognition problems, from object classification [13] to semantic segmentation [2]. However, robotic applications typically need to operate in very specific environments and often the amount of available training data is insufficient to train a CNN model from scratch. Recent research illustrates how to achieve this adaptation, denominated transfer learning. Two common strategies are: 1) to fine-tune a pre-trained model with

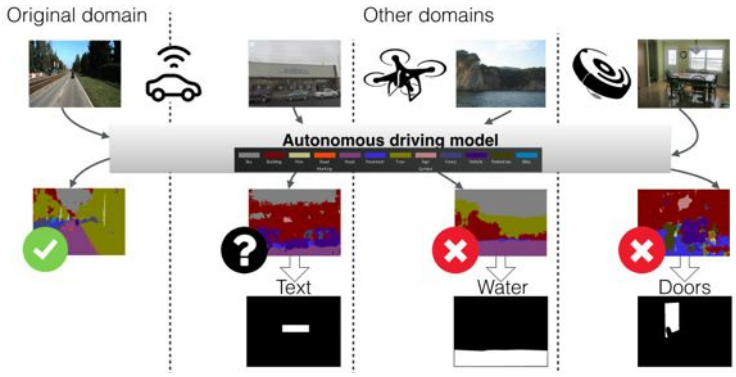


Fig. 1: This work explores how to transfer existing semantic segmentation models to other domains. *Given an existing model* (such as the example on the left, trained for autonomous driving related classes), *how can we use it to learn new classes for different domains and classes not included initially* (such as the three examples on the right)?

additional data from the new domain [17]; 2) to consider intermediate outputs of a pre-trained model as features and train a new classifier with those features extracted on the new domain data [3].

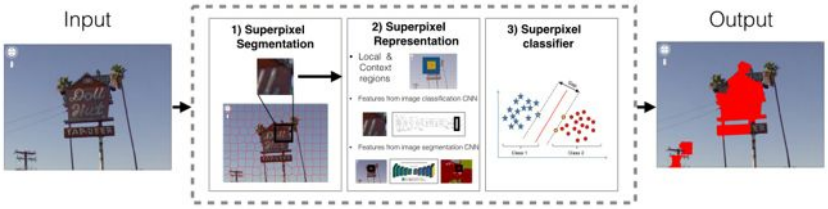
We experiment on how to apply these two common transfer learning strategies on situations where the amount of data and resources is limited. First, we propose how to integrate CNN based classification with a superpixel representation considering local and context information. Secondly, we build and evaluate three different transfer learning pipelines using three varied, complementary and realistic scenarios using different public data-sets for text, door and water region segmentation respectively.

Our results demonstrate how we can efficiently adapt existing CNN models to obtain a binary semantic segmentation of a new class, in spite of having limited amount of training data. We emphasize the relevance of modeling the context and validate the benefits of using a superpixel segmentation. Although fine-tuning an existing CNN model is the most straightforward way for transfer learning; our experiments show that a superpixel classifier can provide interesting advantages. Using CNNs as superpixel feature extractor with a simple classifier requires less training resources while keeps a similar performance, or even higher, especially when the available new dataset is small and different in content type compared to the original dataset.

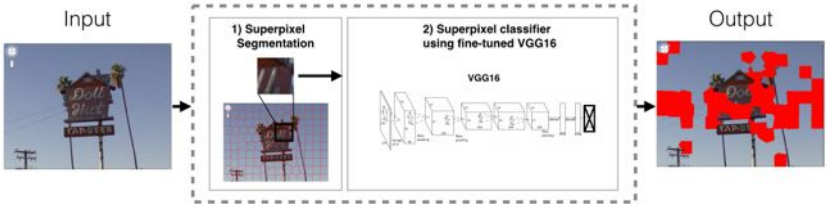
2 Related work

CNNs have become a popular choice for many applications, because of their effective feature generation and end-to-end training. In order to train a new CNN model from scratch, or to adapt an existing CNN to new applications or inputs, it is required to have large amounts of annotated training data. In many cases, obtaining this data may be not feasible, so we find many recent works focused on how to deal with this lack of ground truth data, for example by generating photo-realistic imagery as ground truth [8].

(a) Superpixel classification using SVM and CNN based features



(b) Superpixel classification using fine-tuned CNN for image classification



(c) Fine-tuned CNN for image segmentation

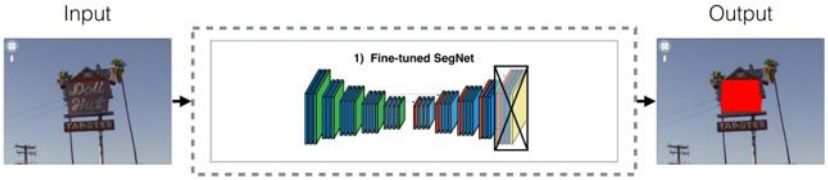


Fig. 2: Transfer learning strategies for semantic segmentation CNN model: (a)(b) superpixel segmentation and classification variations; (c) end-to-end pipeline for per pixel semantic labeling.

A common approach to adapt pre-trained models is fine-tuning. It consists of training a neural network with the new data and target classes, but initializing part of the network with parameters previously learned for other tasks. For example we find works to learn to transfer image style [9] or to adapt to medical imagery tasks [21].

Another common approach is to use an existing model as feature extractor, since the intermediate layers have been observed to correspond to representative features [7]. Recent works demonstrate the features learned when training on ImageNet data for object recognition [13] have been applied successfully on other problems such as activity recognition [5]. These approaches based on [10], generate region proposals and calculate CNN features to classify separately each proposal. More recent works, as [18], propose to generate and classify region proposals simultaneously, in order to eliminate the bottleneck of region proposal computation and achieve real-time rates. We focus on the problem of semantic segmentation, and how to use deep features as superpixel descriptors. Superpixels are groups of contiguous pixels that share appearance, location or other image properties, depending on the algorithm used to compute them (in our work we use SLIC [1]). Similar to prior work [24, 14], we use superpixels as CNN region proposals, but we build a rich superpixel representation calculating features in different scopes, in a simpler way than existing complex approaches [15].

Fig. 3: Text region classification considering regular grid or superpixel segmentation. Superpixels adapt their shape to object boundaries and then produce better text segmentation.

3 Semantic Segmentation pipelines

This section details the two approaches considered for transfer learning: CNN based feature extraction and CNN fine-tuning.

3.1 CNN-based features for superpixel classification

A typical alternative to obtain semantic segmentation of an image is to segment the image in smaller regions and classify each of them. We have built a superpixel classifier, based on a CNN based representation of the superpixel neighbourhood (Fig. 2 a).

Superpixel segmentation. Superpixels adapt their shape to image content and contours, while typical sliding windows or patches from a regular grid do not (Fig. 3) In our work we obtain superpixels using the SLIC algorithm [1], but other superpixel approaches could be similarly applied.

Superpixel description. Our proposed superpixel representation considers superpixel appearance as well as contextual information. All the descriptors are computed in two superpixel scopes, illustrated in Fig. 6 a). While *Local* scope corresponds to information of pixels within each superpixel; *Context* scope corresponds to information from the pixels of neighboring. superpixels. Section 5.1 includes a more detailed analysis of the effects of representing the contextual information around a superpixel. Using CNN based features is a common transfer learning strategy which consists of using the output of intermediate layers from existing networks as image descriptor. We propose two different types of CNN based superpixel descriptors:

First, obtaining *descriptors from Classification CNN models*. We select the weights from last fully connected layer (fc_7), resulting in 4096 descriptor dimensions, from each of the models studied. We consider three well known image classification models: *Alexnet* [13] and *Hybrid-CNN* [25], which share the architecture but use different set of training data and classes, and *VGG* model[22], which is a larger architecture.

Second, obtaining *descriptors from Segmentation CNN models*. We additionally propose how to get a semantic descriptor based on the output of a CNN trained to directly perform image segmentation, i.e. classify each pixel with a semantic label. We propose to build a histogram of the distribution of the semantic labels, obtained with the public SegNet model [2] within the corresponding scope region:

- Local: we build 12-bin histogram (a bin for every label) of the distribution of the semantic labels inside a superpixel.
- Context: neighbouring superpixels areas are grouped in four sets (attending to their position with respect to the superpixel centroid: top, bottom, left and right). We compute a 12-bin histogram in each set.

Superpixel classification. Lastly, superpixels are classified using a standard Support Vector Machine (SVM).

3.2 Fine-tuned CNN models

Another common solution to adapt deep learning models to a new domain is to fine-tune an existing model. We follow a common procedure to fine-tune a CNN model [9, 21]. First, we replace the last CNN classification layer with a new classification layer. The output of this new layer corresponds with the desired number of outputs for our binary segmentation task. Next, we train the adapted network structure with labeled data from the new target class, initializing this process with the original network parameters. The back-propagation algorithm run during training optimizes the network parameters using Stochastic Gradient Descent algorithm (SGD). The two different pipelines we have built based on fine-tuned existing CNN models are detailed next.

Fine-tuning CNN for image classification. This pipeline is illustrated in Fig. 2 b). We fine-tune the publicly available *VGG-16* model [22], which was originally trained to classify images into the 1000 ImageNet object categories [6]. Following a similar idea than R-CNN framework [10], we classify each superpixel separately and build the final image semantic segmentation concatenating the fine-tuned network outputs.

Fine-tuning CNN for image segmentation. We also consider fine-tuning a recent CNN-based semantic segmentation model SegNet [2]. As illustrated in Fig. 2 c), we fine-tune this model to perform a binary segmentation of a new target class. Following standard fine-tuning ideas, we slightly modify the network design to achieve the desired output.

4 Experimental setup.

Our experiments consist of training and evaluating a new binary classifier for a new target semantic class when the new target dataset is limited.

Datasets. We have run three different experiments to learn a new semantic class. These three new target classes were chosen among classes where we could find enough labeled data and were not already included on the reference semantic segmentation models. Besides, they cover a variety of use cases, since they span different levels of man-made characteristics and different levels of difficulty due to the variety of samples.

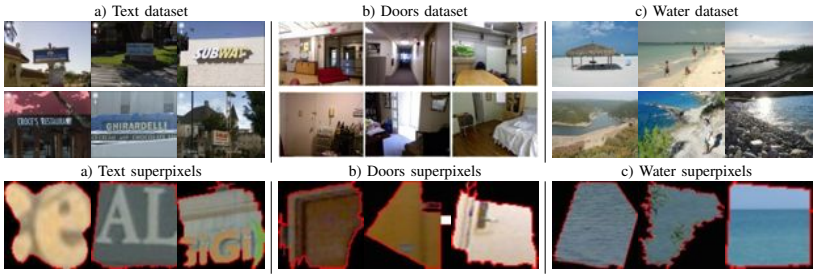


Fig. 4: Sample images and superpixels from the datasets used in each of use cases studied.

Text segmentation: text regions in natural images present a huge variability in appearance and layout, but they possess distinguishable and discriminative properties. The data used in this experiment is from the *Street View Text* dataset [23] (Figure 4 a), where most of the images come from business signage and exhibit a high degree of variability in appearance and resolution. The provided training set contains 100 images, while the testing set contains 249 images. In average 3.38% pixels per image are labeled as Text.

Doors segmentation: doors are important landmarks indoors, and automatic visual door recognition has major difficulties due to the variety and diversity of appearance and shapes of door regions. We use RGB data from the *NYU v2* dataset [16] (Figure 4 b). In this experiment, we only consider the images which have pixels labeled as door. The resultant training and testing sets contain 239 and 165 images respectively. In average 7.63% pixels per image are labeled as Door.

Water segmentation: detection of image regions containing water is a challenging problem, since they do not have a defined shape and the appearance suffers strong changes do to lighting, reflection, etc. We use the data from the *LabelME* dataset [19] (Figure 4 c). As positive Water class examples, we use regions from all labels related to it: sea, water, lake, pond and ocean; and the rest of labels are considered as no-water. Our training and testing sets contain 100 and 194 images randomly picked from these collections: *static waterfront boston outdoor july 2005, web static park garden outdoor, static web outdoor spain, static coast palafrugel outdoor spain*. In average 28.16% of the image pixels from the images used from these collections are labeled as Water.

Training details. For the *superpixel classifier*, we use the standard SVM implementation in OpenCV which is based on LibSVM library [4]. In all the experiments, we consider positive training samples the superpixels which contain more than 50% of their pixels labeled as the target class. Note that superpixel segmentation solution can often provide superpixels where not all pixels belong to the same class. As negative examples, we randomly choose superpixels from images in the training set with zero pixels labeled as the target class. For both the extraction of CNN features and the *fine-tuning* of CNN models, we have used the open source Deep Learning library Caffe [11]. We have run the fine-tuning of the modified version of the networks for around 60,000 iterations, using a NVIDIA GeForce GTX TITAN X GPU with 12GB RAM.

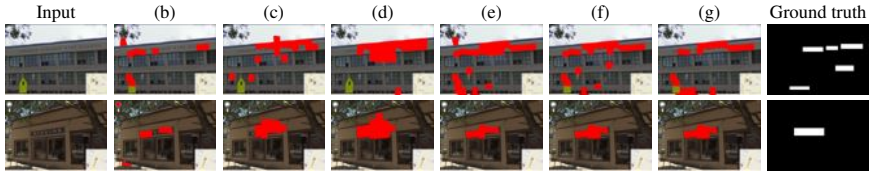
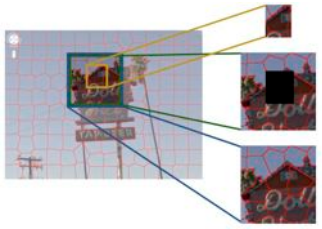


Fig. 5: Different Superpixel Context Representation (Text classification task): (b) *local*, (c) $context_1$, (d) $context_2$, (e) $local + context_1$, (f) $local + context_2$, (g) $local + context_1 + context_2$.



(a)

Scopes	Precision	Recall
local	0.34	0.70
$context_1$	0.23	0.59
$context_2$	0.27	0.68
$local + context_1$	**0.35**	**0.76**
$local + context_2$	**0.34**	**0.77**
$local + context_1 + context_2$	0.37	0.74

(b)

Fig. 6: (a) Superpixel scopes. (b) Superpixel Context Representation in Text classification.

Evaluation metrics. We compute the average *Precision and Recall* in the pixel classification for all images, using the sets provided in the corresponding public datasets.

5 Experimental Results

This section evaluates and discusses the results achieved with the presented transfer learning approaches on the different use cases described in previous section.

5.1 Context representation

We proposed a feature representation considering two different scopes: from the superpixel itself to a region around it. To analyze each representation, we have trained various superpixel classifiers using the same CNN based features but varying the region which these features have been calculated. In particular, we focused on different ways to represent the contextual information around a superpixel. Given a superpixel, we define its context region as the region encloses all pixels from its adjacent superpixels. We have computed the CNN features into the context region excluding or not the superpixel itself, i.e. $context_1$ and $context_2$ regions respectively (Fig. 6 a).

We have tested each superpixel classifier in the Text classification problem. The results are summarized in Fig. 6 (b) and Fig. 5. Based on these results, we observe that including context features ($context_1$ or $context_2$) help to detect more Text regions (*Recall*) and with less false-positives regions (*Precision*), independently that context region considered. Both context descriptors obtain similar results, then, we propose using $context_1$ to represent the superpixel context.

Table 1: Precision and Recall of semantic pixel segmentation for each application (Text, Door, Water) using different transfer learning pipelines. Figure 7 shows examples of the best ones.

Pipelines			Text		Door		Water	
			Precision	Recall	Precision	Recall	Precision	Recall
CNN based SUPERPIXEL FEATURES + SVM superpixel classifier								
	Feature	size						
	AlexNet_fc7 (*A*)	8192	0.33	0.65	0.13	0.44	0.90	0.85
	HybridCNN_fc7 (*H*)	8192	0.34	0.70	0.12	0.40	0.87	0.86
Fig 7(b)	VGG_fc7 (*V*)	8192	**0.35**	**0.76**	0.12	0.44	0.88	0.85
Fig 7(c)	SegNet_hist (*S*)	60	0.19	0.43	0.11	0.30	0.66	0.81
	Combined							
	$V+S$	8252	0.32	0.67	0.11	0.44	0.90	0.69
Superpixel segmentation + FINE-TUNING of CNN for image classification								
Fig 7(d)	fine-tuned_VGG		**0.20**	**0.77**	**0.20**	**0.44**	**0.88**	**0.87**
FINE-TUNING of CNN for image segmentation								
Fig 7(e)	fine-tuned_SegNet		**0.53**	**0.58**	**0.28**	**0.10**	**0.58**	**0.27**

5.2 Performance of transfer learning pipelines

This section evaluates the studied transfer learning strategies to achieve the new semantic segmentation class.

As we can observe in the examples from Fig. 7, the three applications achieve different performance. As it could be expected, Door classification is a more challenging tasks, due to large variations on appearance and shape of door regions, as well as being hard to discriminate from other common indoor surfaces, such as furniture. As we can observe in Fig. 4 the set of training data available is not enough to cover all the variability of this class, while the other experiments intra-class variation (water and text) seems to be better captured by the training set, even though we have similar amount of images for each of them. Note that the ground truth images contain some errors as shown in Fig 7, e.g., the second row shows ground truth with actual text regions unlabeled.

Table 1 shows the average Precision and Recall obtained for each of the three experiments (Text, Doors, Water). Each row corresponds to a different configuration of the semantic segmentation approach (Sec. 3). All descriptors are calculated on the two scopes: *Local* and *Context* (in particular we use $context_1$, since we already demonstrated in previous section 5.1 that using both scopes significantly improves performance. While recall is always higher on the superpixel based pipelines (i.e., we were able to correctly label more image pixels from the target semantic class), the accuracy results point to two different conclusions. On one hand, in *Text* classification, the *fine-tuned_SegNet* end-to-end classifier, achieved the best precision. This can be explained because the *Text* classification problem is the closest domain to the type of images that were used to train SegNet model (urban scenes, including a class for signs, which often contained text-like areas). On the other hand, in *Water* and *Text* classification, we observe that the superpixel-based classifiers obtain significantly higher recall (and comparable precision) than the fine-tuned model for image segmentation.

5.3 Discussion

Different superpixel representations. The first five rows in Table 1 show Precision and Recall of the semantic classification of pixels using different CNN superpixel represen-

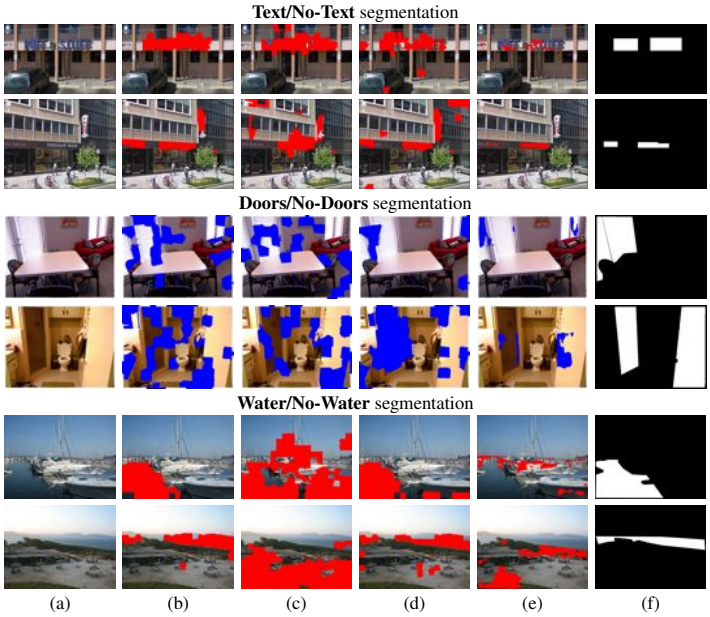


Fig. 7: Representative examples of pixel segmentation (in red or blue) using different transfer learning pipelines. (a) Original image. (b) CNN-based features. (c) Semantic features. (d) Fine-tuned CNN model for image classification and (e) for image segmentation. (f) Ground truth.

tations and SVM classifier. Given a superpixel patch, rows A, H and V show results using features extracted from layers of different CNN models for image classification, while S descriptor is obtained as the histogram of the output within the superpixel patch from a CNN model for image segmentation. There are several common points for all these experiments: combining the two types of CNN based descriptors ($V+S$) does not improve the Recall of the best of them separately. This confirms the amount of information enclosed on S is definitely more compressed but less detailed, something that could be expected from the very different size of descriptors (column *size* in the table).

If we compare both types of superpixel based classification pipelines (the best SVM based superpixel classifier, row V, and the fine-tuned model for image classification, *fine-tuned_VGG*), is not surprising that they get to similar quality on the results. They both take into account superpixel restrictions and use features learned in previous CNN trained for image classification. The difference is how they train given this information. The first case trains an SVM with this information, while the second fine-tunes the existing CNN, i.e., trains the last layers to adapt them to the new domain.

Therefore, using CNN features extracted by last CNN layer is equivalent to fine-tune the CNN model in terms of accuracy of the results (only in *Text* experiment, the fine-tuned model *fine-tuned_VGG* obtains worse results). The main difference between these pipelines is during the training stage. While a simple SVM can be trained on the CPU of a common desktop computer and it just takes the order of minutes, fine-tuning a CNN model in the same conditions can take several days. At the end of this section,

we present a more detailed analysis of which pipelines are more suitable when we have time or resources restrictions.

Superpixel-based classifiers vs end-to-end segmentation. As previously mentioned, if we compare the results obtained by superpixel-based classifiers (*V* and *fine-tuned_VGG*) and the end-to-end pipeline obtained with the fine-tuned model for image segmentation (*fine-tuned_SegNet*), we can observe more significant differences in the performance obtained, which highlight the influence of the following aspects, which should be taken into account when targeting a new semantic class segmentation:

Different domains. As previously mentioned, the different behaviour of the *Text* experiment (achieving the best precision using the fine-tuned SegNet to directly obtain the binary image segmentation) is due to the fact that this target class is closer to the domain considered in the original label set. So in general, we could suggest to use a superpixel-based classifier for the transfer learning pipeline when the new target class is from a very different domain than the data from the existing CNN model.

Amount of training data. Our use cases are motivated by not having enough labeled data to train a model from scratch. A benefit we have observed in the superpixel based approaches is that with the same amount of labeled training images, we actually have a larger set of training samples for superpixel based approaches. This provides better models and results with limited amount of training data.

Limited resources. Another important aspect to study when choosing the most convenient way to implement a new classifier based on existing models is the type of resources we have available (time restriction or availability of GPU), and how they affect our target application or platform, during training or prediction stages.

Regarding the **training** stage, we should note that the *SVM based superpixel classifier* has much lower computational requirement to train a new binary segmentation model, because we only run CNNs forward steps to obtain the features, which is relatively fast, even without using GPU (around 5s to get features for a 50 image batch on a CPU). Then, the SVM training is a fast step, e.g., training a superpixel classifier with 15766 examples using a 4096-dimensional descriptor for each sample, takes around 4 minutes in a common desktop computer. However, we estimate that fine-tuning a CNN model could take around 30 hours executing 60.000 iterations in similar conditions (CPU). For this option to become more affordable (as we have done in our experiments) we should fine-tune the models using a GPU. Therefore, the SVM based pipeline is the best option without a GPU available for the training stage, since using a CNN only to extract features is not very demanding operation that can be run without a GPU.

Regarding the **prediction** stage, i.e., to achieve the semantic segmentation of a new image, we should note that there are smaller execution time differences among the different pipelines. This is due mainly because we always use a CNN model, either to extract the features or to directly predict the final segmentation. The execution time in the different pipelines is summarized in (1), where T_a is the time for a superpixel based SVM classifier, T_b is a superpixel based CNN image classifier, and T_c is an end-to-end CNN image segmentation pipeline:

$$T_a = T_{sp} + |sp| \cdot (T_{cnn} + T_{svm} + T_{out}) \qquad T_b = T_{sp} + |sp| \cdot (T_{cnn} + T_{out}) \qquad T_c = T_{cnn} \quad (1)$$

T_{sp} is the time to segment an image into $|sp|$ superpixels, T_{cnn} is the time to run the a forward pass of an image through a CNN, T_{svm} is the time to classify an superpixel with an SVM and T_{out} is the time to build the output image combining individual superpixel classification. Note that, T_{svm} and T_{out} can be ignored, since their times are despicable compared to the other steps. Besides, $|sp| \cdot T_{cnn}$ is almost equivalent to T_{cnn} if we process the superpixel patches in batches (note that CNN evaluation can process batches of images in a parallel manner), so the main difference between T_a, T_b and T_c is the superpixel extraction time (6s per image in the algorithm we use). An important aspect to consider about the requirements during prediction stage is that often the deployment of trained models for the final applications (e.g., deploy the learned model on a robot on-board computer), has stronger time and resources limitations. Therefore, the end-to-end pipeline is the best option for a resource limited platform for the *prediction* stage, since as we have seen, the training is performed beforehand, and the only step needed for prediction is a task that can be performed reasonably fast without a GPU.

6 Conclusions

Many applications require semantic segmentation of specific classes for which there is not many labeled training data. Recent CNN approaches have proven useful for transfer knowledge from existing models to new tasks. This work evaluates the two major CNN-based transfer learning strategies: fine-tuning existing models and CNN based features for classification. We have proposed how to integrate a superpixel representation with these strategies and combining effectively local and context superpixel description.

Our three experiments have covered a wide range of semantic segmentation problems. Our results with different real scenarios have demonstrated the benefits of our pipelines to learn how to segment new classes with limited data and resources. Although fine-tuning existing model seems the most direct way to transfer information learned on those models, our experiments have shown that using the proposed CNN-based representation and superpixels provides a more effective transfer learning pipeline than fine-tuning a model. Future works on optimizing and enabling this kind of visual recognition systems on embedded platforms are a challenging problem that would enable autonomous systems to have richer perception modules.

Acknowledgments. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research. This research has been partially funded by the European Research Council (ERC) under the EU Horizon 2020 program (CHAMELEON project, grant agreement No 682080), the Spanish Government (projects DPI2015-65962-R, DPI2015-69376-R) and Aragon regional government (Grupo DGA T04-FSE).

References

1. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Susstrunk, S.: SLIC superpixels compared to state-of-the-art superpixel methods. IEEE Trans. on PAMI 34(11), 2274–2282 (2012)

2. Badrinarayanan, V., Kendall, A., Cipolla, R.: SegNet: A deep convolutional encoder-decoder architecture for image segmentation. arXiv preprint arXiv:1511.00561 (2015)
3. Bell, S., Lawrence Zitnick, C., Bala, K., Girshick, R.: Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In: CVPR. pp. 2874–2883 (2016)
4. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST) 2(3), 27 (2011)
5. Chéron, G., Laptev, I., Schmid, C.: P-cnn: Pose-based cnn features for action recognition. In: Proc. of the IEEE International Conf. on Computer Vision. pp. 3218–3226 (2015)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. pp. 248–255. IEEE (2009)
7. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: DeCAF: A deep convolutional activation feature for generic visual recognition. In: Proc. of the 31st Int. Conf. on Machine Learning. pp. 647–655 (2014)
8. Gaidon, A., Wang, Q., Cabon, Y., Vig, E.: Virtual worlds as proxy for multi-object tracking analysis. In: CVPR. pp. 4340–4349 (2016)
9. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: CVPR. pp. 2414–2423 (2016)
10. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: IEEE Conf. on CVPR. pp. 580–587 (2014)
11. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. In: Proc. of the 22Nd ACM International Conference on Multimedia. pp. 675–678. MM '14, ACM (2014)
12. Kochanov, D., Ošep, A., Stückler, J., Leibe, B.: Scene flow propagation for semantic mapping and object discovery in dynamic street scenes. In: IROS. pp. 1785–1792. IEEE (2016)
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. pp. 1097–1105 (2012)
14. Liu, F., Shen, C., Lin, G.: Deep convolutional neural fields for depth estimation from a single image. In: CVPR. pp. 5162–5170 (2015)
15. Mostajabi, M., Yadollahpour, P., Shakhnarovich, G.: Feedforward semantic segmentation with zoom-out features. In: Proc. of the IEEE Conf. on CVPR. pp. 3376–3385 (2015)
16. Nathan Silberman, Derek Hoiem, P.K., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: ECCV (2012)
17. Pinheiro, P.O., Collobert, R., Dollar, P.: Learning to segment object candidates. In: NIPS. pp. 1990–1998 (2015)
18. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: NIPS (2015)
19. Russell, B.C., Torralba, A., Murphy, K.P., Freeman, W.T.: Labelme: a database and web-based tool for image annotation. IJCV 77(1-3), 157–173 (2008)
20. Schneider, L., Cordts, M., Rehfeld, T., Pfeiffer, D., Enzweiler, M., Franke, U., Pollefeys, M., Roth, S.: Semantic stixels: Depth is not enough. In: Intelligent Vehicles Symposium. pp. 110–117. IEEE (2016)
21. Shin, H.C., Roth, H.R., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D., Summers, R.M.: Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. IEEE T. on Medical Imaging PP(99), 1–1 (2016)
22. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556 (2014)
23. Wang, K., Belongie, S.: Word spotting in the wild. In: ECCV. pp. 591–604 (2010)
24. Yan, J., Yu, Y., Zhu, X., Lei, Z., Li, S.Z.: Object detection by labeling superpixels. In: CVPR (2015)
25. Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., Oliva, A.: Learning deep features for scene recognition using places database. In: NIPS. pp. 487–495 (2014)