SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation

Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla, *Senior Member, IEEE,*

arXiv:1511.00561v3 [cs.CV] 10 Oct 2016

Abstract—We present a novel and practical deep fully convolutional neural network architecture for semantic pixel-wise segmentation termed SegNet. This core trainable segmentation engine consists of an encoder network, a corresponding decoder network followed by a pixel-wise classification layer. The architecture of the encoder network is topologically identical to the 13 convolutional layers in the VGG16 network [1]. The role of the decoder network is to map the low resolution encoder feature maps to full input resolution feature maps for pixel-wise classification. The novelty of SegNet lies is in the manner in which the decoder upsamples its lower resolution input feature map(s). Specifically, the decoder uses pooling indices computed in the max-pooling step of the corresponding encoder to perform non-linear upsampling. This eliminates the need for learning to upsample. The upsampled maps are sparse and are then convolved with trainable filters to produce dense feature maps. We compare our proposed architecture with the widely adopted FCN [2] and also with the well known DeepLab-LargeFOV [3], DeconvNet [4] architectures. This comparison reveals the memory versus accuracy trade-off involved in achieving good segmentation performance.

SegNet was primarily motivated by scene understanding applications. Hence, it is designed to be efficient both in terms of memory and computational time during inference. It is also significantly smaller in the number of trainable parameters than other competing architectures and can be trained end-to-end using stochastic gradient descent. We also performed a controlled benchmark of SegNet and other architectures on both road scenes and SUN RGB-D indoor scene segmentation tasks. These quantitative assessments show that SegNet provides good performance with competitive inference time and most efficient inference memory-wise as compared to other architectures. We also provide a Caffe implementation of SegNet and a web demo at http://mi.eng.cam.ac.uk/projects/segnet/.

Index Terms—Deep Convolutional Neural Networks, Semantic Pixel-Wise Segmentation, Indoor Scenes, Road Scenes, Encoder, Decoder, Pooling, Upsampling.

1 Introduction

Semantic segmentation has a wide array of applications ranging from scene understanding, inferring support-relationships among objects to autonomous driving. Early methods that relied on low-level vision cues have fast been superseded by popular machine learning algorithms. In particular, deep learning has seen huge success lately in handwritten digit recognition, speech, categorising whole images and detecting objects in images [5], [6]. Now there is an active interest for semantic pixel-wise labelling [7] [8], [9], [2], [4], [10], [11], [12], [13], [3], [14], [15], [16]. However, some of these recent approaches have tried to directly adopt deep architectures designed for category prediction to pixel-wise labelling [7]. The results, although very encouraging, appear coarse [3]. This is primarily because max pooling and sub-sampling reduce feature map resolution. Our motivation to design SegNet arises from this need to map low resolution features to input resolution for pixel-wise classification. This mapping must produce features which are useful for accurate boundary localization.

Our architecture, SegNet, is designed to be an efficient architecture for pixel-wise semantic segmentation. It is primarily motivated by road scene understanding applications which require the ability to model appearance (road, building), shape (cars, pedestrians) and understand the spatial-relationship (context) between different classes such as road and side-walk. In typical road scenes, the majority of the pixels belong to large classes such as road, building and hence the network must produce smooth segmentations. The engine must also have the ability to delineate objects based on their shape despite their small size. Hence it is important to retain boundary information in the extracted image representation. From a computational perspective, it is necessary for the network to be efficient in terms of both memory and computation time during inference. The ability to train end-to-end in order to jointly optimise all the weights in the network using an efficient weight update technique such as stochastic gradient descent (SGD) [17] is an additional benefit since it is more easily repeatable. The design of SegNet arose from a need to match these criteria.

The encoder network in SegNet is topologically identical to the convolutional layers in VGG16 [1]. We remove the fully connected layers of VGG16 which makes the SegNet encoder network significantly smaller and easier to train than many other recent architectures [2], [4], [11], [18]. The key component of SegNet is the decoder network which consists of a hierarchy of decoders one corresponding to each encoder. Of these, the appropriate decoders use the max-pooling indices received from the corresponding encoder to perform non-linear upsampling of their input feature maps. This idea was inspired from an architecture designed for unsupervised feature learning [19]. Reusing max-pooling indices in the decoding process has several practical

• *V. Badrinarayanan, A. Kendall, R. Cipolla are with the Machine Intelligence Lab, Department of Engineering, University of Cambridge, UK. E-mail: vb292,agk34,cipolla@eng.cam.ac.uk*

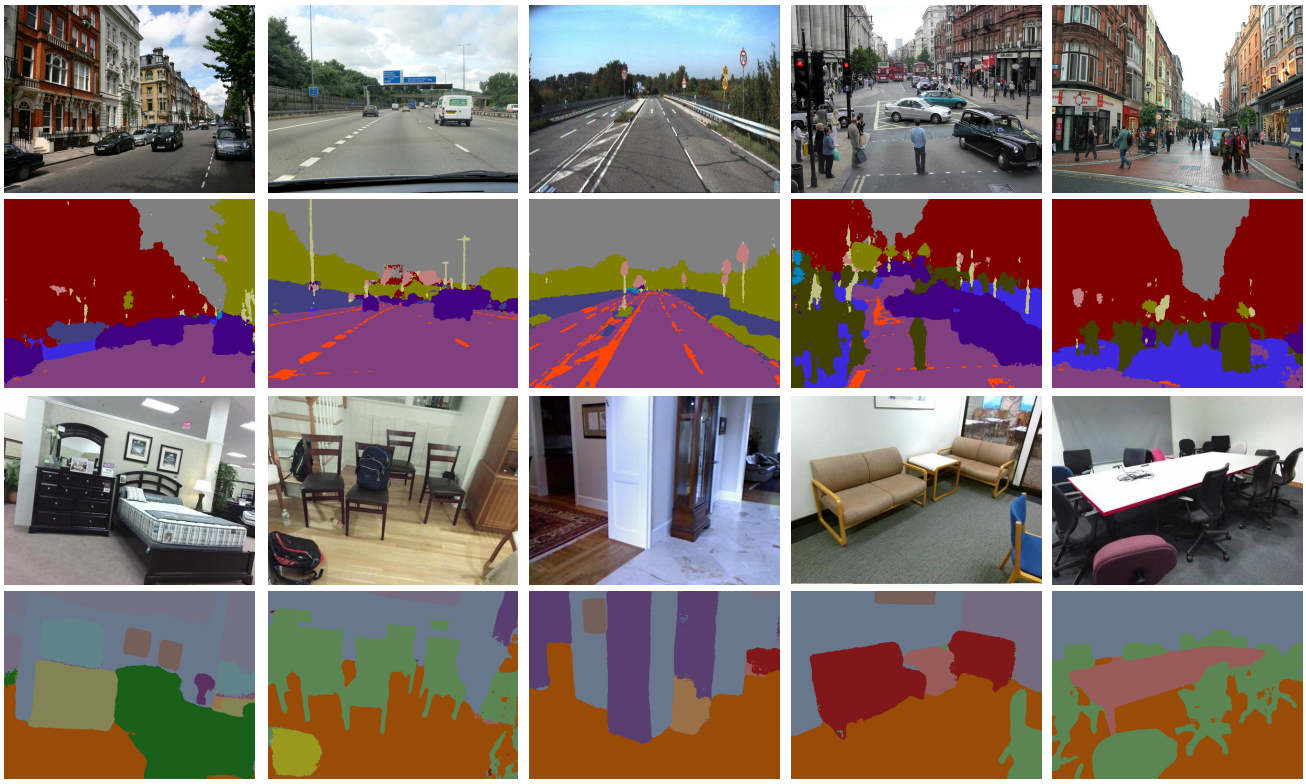

Fig. 1. SegNet predictions on road scenes and indoor scenes. To try our system yourself, please see our online web demo at http://mi.eng.cam.ac.uk/projects/segnet/.

advantages; (i) it improves boundary delineation , (ii) it reduces the number of parameters enabling end-to-end training, and (iii) this form of upsampling can be incorporated into any encoder-decoder architecture such as [2], [10] with only a little modification.

One of the main contributions of this paper is our analysis of the SegNet decoding technique and the widely used Fully Convolutional Network (FCN) [2]. This is in order to convey the practical trade-offs involved in designing segmentation architectures. Most recent deep architectures for segmentation have identical encoder networks, i.e VGG16, but differ in the form of the decoder network, training and inference. Another common feature is they have trainable parameters in the order of hundreds of millions and thus encounter difficulties in performing end-to-end training [4]. The difficulty of training these networks has led to multi-stage training [2], appending networks to a pre-trained architecture such as FCN [10], use of supporting aids such as region proposals for inference [4], disjoint training of classification and segmentation networks [18] and use of additional training data for pre-training [11] [20] or for full training [10]. In addition, performance boosting post-processing techniques [3] have also been popular. Although all these factors improve performance on challenging benchmarks [21], it is unfortunately difficult from their quantitative results to disentangle the key design factors necessary to achieve good performance. We therefore analysed the decoding process used in some of these approaches [2], [4] and reveal their pros and cons.

We evaluate the performance of SegNet on two scene segmentation tasks, CamVid road scene segmentation [22] and SUN RGB-D indoor scene segmentation [23]. Pascal VOC12 [21] has been the benchmark challenge for segmentation over the years. However, the majority of this task has one or two foreground classes surrounded by a highly varied background. This implicitly favours techniques used for detection as shown by the recent work on a decoupled classification-segmentation network [18] where the classification network can be trained with a large set of weakly labelled data and the independent segmentation network performance is improved. The method of [3] also use the feature maps of the classification network with an independent CRF post-processing technique to perform segmentation. The performance can also be boosted by the use additional inference aids such as region proposals [4], [24]. Therefore, it is different from scene understanding where the idea is to exploit co-occurrences of objects and other spatial-context to perform robust segmentation. To demonstrate the efficacy of SegNet, we present a real-time online demo of road scene segmentation into 11 classes of interest for autonomous driving (see link in Fig. 1). Some example test results produced on randomly sampled road scene images from Google and indoor test scenes from the SUN RGB-D dataset [23] are shown in Fig. 1.

The remainder of the paper is organized as follows. In Sec. 2 we review related recent literature. We describe the SegNet architecture and its analysis in Sec. 3. In Sec. 4 we evaluate the performance of SegNet on outdoor and indoor scene datasets. This is followed by a general discussion regarding our approach with pointers to future work in Sec. 5. We conclude in Sec. 6.

2 Literature Review

Semantic pixel-wise segmentation is an active topic of research, fuelled by challenging datasets [21], [22], [23], [25], [26]. Before the arrival of deep networks, the best performing methods mostly relied on hand engineered features classifying pixels independently. Typically, a patch is fed into a classifier *e.g.* Random

Forest [27], [28] or Boosting [29], [30] to predict the class probabilities of the center pixel. Features based on appearance [27] or SfM and appearance [28], [29], [30] have been explored for the CamVid road scene understanding test [22]. These per-pixel noisy predictions (often called *unary* terms) from the classifiers are then smoothed by using a pair-wise or higher order CRF [29], [30] to improve the accuracy. More recent approaches have aimed to produce high quality unaries by trying to predict the labels for all the pixels in a patch as opposed to only the center pixel. This improves the results of Random Forest based unaries [31] but thin structured classes are classified poorly. Dense depth maps computed from the CamVid video have also been used as input for classification using Random Forests [32]. Another approach argues for the use of a combination of popular hand designed features and spatio-temporal super-pixelization to obtain higher accuracy [33]. The best performing technique on the CamVid test [30] addresses the imbalance among label frequencies by combining object detection outputs with classifier predictions in a CRF framework. The result of all these techniques indicate the need for improved features for classification.

Indoor RGBD pixel-wise semantic segmentation has also gained popularity since the release of the NYU dataset [25]. This dataset showed the usefulness of the depth channel to improve segmentation. Their approach used features such as RGB-SIFT, depth-SIFT and pixel location as input to a neural network classifier to predict pixel unaries. The noisy unaries are then smoothed using a CRF. Improvements were made using a richer feature set including LBP and region segmentation to obtain higher accuracy [34] followed by a CRF. In more recent work [25], both class segmentation and support relationships are inferred together using a combination of RGB and depth based cues. Another approach focuses on real-time joint reconstruction and semantic segmentation, where Random Forests are used as the classifier [35]. Gupta *et al.* [36] use boundary detection and hierarchical grouping before performing category segmentation. The common attribute in all these approaches is the use of hand engineered features for classification of either RGB or RGBD images.

The success of deep convolutional neural networks for object classification has more recently led researchers to exploit their feature learning capabilities for structured prediction problems such as segmentation. There have also been attempts to apply networks designed for object categorization to segmentation, particularly by replicating the deepest layer features in blocks to match image dimensions [7], [37], [38], [39]. However, the resulting classification is blocky [38]. Another approach using recurrent neural networks [40] merges several low resolution predictions to create input image resolution predictions. These techniques are already an improvement over hand engineered features [7] but their ability to delineate boundaries is poor.

Newer deep architectures [2], [4], [10], [13], [18] particularly designed for segmentation have advanced the state-of-the-art by learning to decode or map low resolution image representations to pixel-wise predictions. The encoder network which produces these low resolution representations in all of these architectures is the VGG16 classification network [1] which has 13 convolutional layers and 3 fully connected layers. This encoder network weights are typically pre-trained on the large ImageNet object classification dataset [41]. The decoder network varies between these architectures and is the part which is responsible for producing multi-dimensional features for each pixel for classification.

Each decoder in the Fully Convolutional Network (FCN) architecture [2] learns to upsample its input feature map(s) and combines them with the corresponding encoder feature map to produce the input to the next decoder. It is an architecture which has a large number of trainable parameters in the encoder network (134M) but a very small decoder network (0.5M). The overall large size of this network makes it hard to train end-to-end on a relevant task. Therefore, the authors use a stage-wise training process. Here each decoder in the decoder network is progressively added to an existing trained network. The network is grown until no further increase in performance is observed. This growth is stopped after three decoders thus ignoring high resolution feature maps can certainly lead to loss of edge information [4]. Apart from training related issues, the need to reuse the encoder feature maps in the decoder makes it memory intensive in test time. We study this network in more detail as it the core of other recent architectures [10], [11].

The predictive performance of FCN has been improved further by appending the FCN with a recurrent neural network (RNN) [10] and fine-tuning them on large datasets [21], [42]. The RNN layers mimic the sharp boundary delineation capabilities of CRFs while exploiting the feature representation power of FCN's. They show a significant improvement over FCN-8 but also show that this difference is reduced when more training data is used to train FCN-8. The main advantage of the CRF-RNN is revealed when it is jointly trained with an architecture such as the FCN-8. The fact that joint training helps is also shown in other recent results [43], [44]. Interestingly, the deconvolutional network [4] performs significantly better than FCN although at the cost of a more complex training and inference. This however raises the question as to whether the perceived advantage of the CRF-RNN would be reduced as the core feed-forward segmentation engine is made better. In any case, the CRF-RNN network can be appended to any deep segmentation architecture including SegNet.

Multi-scale deep architectures are also being pursued [13], [44]. They come in two flavours, (i) those which use input images at a few scales and corresponding deep feature extraction networks, and (ii) those which combine feature maps from different layers of a single deep architecture [45] [11]. The common idea is to use features extracted at multiple scales to provide both local and global context [46] and the using feature maps of the early encoding layers retain more high frequency detail leading to sharper class boundaries. Some of these architectures are difficult to train due to their parameter size [13]. Thus a multi-stage training process is employed along with data augmentation. The inference is also expensive with multiple convolutional pathways for feature extraction. Others [44] append a CRF to their multi-scale network and jointly train them. However, these are not feed-forward at test time and require optimization to determine the MAP labels.

Several of the recently proposed deep architectures for segmentation are not feed-forward in inference time [4], [3], [18]. They require either MAP inference over a CRF [44], [43] or aids such as region proposals [4] for inference. We believe the perceived performance increase obtained by using a CRF is due to the lack of good decoding techniques in their core feed-forward segmentation engine. SegNet on the other hand uses decoders to obtain features for accurate pixel-wise classification.

The recently proposed Deconvolutional Network [4] and its semi-supervised variant the Decoupled network [18] use the max locations of the encoder feature maps (pooling indices) to perform non-linear upsampling in the decoder network. The authors of these architectures, independently of SegNet (first submitted to

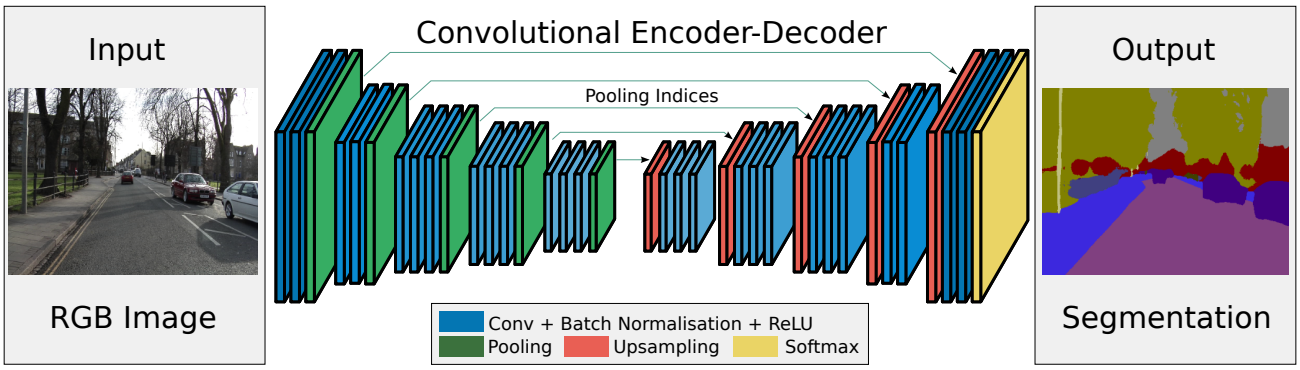

Fig. 2. An illustration of the SegNet architecture. There are no fully connected layers and hence it is only convolutional. A decoder upsamples its input using the transferred pool indices from its encoder to produce a sparse feature map(s). It then performs convolution with a trainable filter bank to densify the feature map. The final decoder output feature maps are fed to a soft-max classifier for pixel-wise classification.

CVPR 2015 [12]), proposed this idea of decoding in the decoder network. However, their encoder network consists of the fully connected layers from the VGG-16 network which consists of about 90% of the parameters of their entire network. This makes training of their network very difficult and thus require additional aids such as the use of region proposals to enable training. Moreover, during inference these proposals are used and this increases inference time significantly. From a benchmarking point of view, this also makes it difficult to evaluate the performance of their architecture (encoder-decoder network) without other aids. In this work we discard the fully connected layers of the VGG16 encoder network which enables us to train the network using the relevant training set using SGD optimization. Another recent method [3] shows the benefit of reducing the number of parameters significantly without sacrificing performance, reducing memory consumption and improving inference time.

Our work was inspired by the unsupervised feature learning architecture proposed by Ranzato *et al.* [19]. The key learning module is an encoder-decoder network. An encoder consists of convolution with a filter bank, element-wise tanh non-linearity, max-pooling and sub-sampling to obtain the feature maps. For each sample, the indices of the max locations computed during pooling are stored and passed to the decoder. The decoder upsamples the feature maps by using the stored pooled indices. It convolves this upsampled map using a trainable decoder filter bank to reconstruct the input image. This architecture was used for unsupervised pre-training for classification. A somewhat similar decoding technique is used for visualizing trained convolutional networks [47] for classification. The architecture of Ranzato *et al.* mainly focused on layer-wise feature learning using small input patches. This was extended by Kavukcuoglu et. al. [48] to accept full image sizes as input to learn hierarchical encoders. Both these approaches however did not attempt to use *deep encoder-decoder* networks for unsupervised feature training as they discarded the decoders after each encoder training. Here, SegNet differs from these architectures as the deep encoder-decoder network is trained jointly for a supervised learning task and hence the decoders are an integral part of the network in test time.

Other applications where pixel wise predictions are made using deep networks are image super-resolution [49] and depth map prediction from a single image [50]. The authors in [50] discuss the need for learning to upsample from low resolution feature maps which is the central topic of this paper.

3 Architecture

SegNet has an encoder network and a corresponding decoder network, followed by a final pixelwise classification layer. This architecture is illustrated in Fig. 3. The encoder network consists of 13 convolutional layers which correspond to the first 13 convolutional layers in the VGG16 network [1] designed for object classification. We can therefore initialize the training process from weights trained for classification on large datasets [41]. We can also discard the fully connected layers in favour of retaining higher resolution feature maps at the deepest encoder output. This also reduces the number of parameters in the SegNet encoder network significantly (from 134M to 14.7M) as compared to other recent architectures [2], [4] (see. Table 6). Each encoder layer has a corresponding decoder layer and hence the decoder network has 13 layers. The final decoder output is fed to a multi-class soft-max classifier to produce class probabilities for each pixel independently.

Each *encoder* in the encoder network performs convolution with a filter bank to produce a set of feature maps. These are then batch normalized [51], [52]). Then an element-wise rectified-linear non-linearity (ReLU) $max(0, x)$ is applied. Following that, max-pooling with a 2×2 window and stride 2 (non-overlapping window) is performed and the resulting output is sub-sampled by a factor of 2. Max-pooling is used to achieve translation invariance over small spatial shifts in the input image. Sub-sampling results in a large input image context (spatial window) for each pixel in the feature map. While several layers of max-pooling and sub-sampling can achieve more translation invariance for robust classification correspondingly there is a loss of spatial resolution of the feature maps. The increasingly lossy (boundary detail) image representation is not beneficial for segmentation where boundary delineation is vital. Therefore, it is necessary to *capture and store* boundary information in the encoder feature maps before sub-sampling is performed. If memory during inference is not constrained, then all the encoder feature maps (after sub-sampling) can be stored. This is usually not the case in practical applications and hence we propose a more efficient way to store this information. It involves storing only the max-pooling *indices*, i.e, the locations of the maximum feature value in each pooling window is memorized for each encoder feature map. In principle, this can be done using 2 bits for each 2×2 pooling window and is thus much more efficient to store as compared to memorizing feature map(s) in float precision. As we show later in this work,

this lower memory storage results in a slight loss of accuracy but is still suitable for practical applications.

The appropriate *decoder* in the decoder network upsamples its input feature map(s) using the memorized max-pooling indices from the corresponding encoder feature map(s). This step produces sparse feature map(s). This SegNet decoding technique is illustrated in Fig. 3. These feature maps are then convolved with a trainable decoder filter bank to produce dense feature maps. A batch normalization step is then applied to each of these maps. Note that the decoder corresponding to the first encoder (closest to the input image) produces a multi-channel feature map, although its encoder input has 3 channels (RGB). This is unlike the other decoders in the network which produce feature maps with the same number of size and channels as their encoder inputs. The high dimensional feature representation at the output of the final decoder is fed to a trainable soft-max classifier. This soft-max classifies each pixel independently. The output of the soft-max classifier is a K channel image of probabilities where K is the number of classes. The predicted segmentation corresponds to the class with maximum probability at each pixel.

We add here that two other architectures, DeconvNet [53] and U-Net [16] share a similar architecture to SegNet but with some differences. DeconvNet has a much larger parameterization, needs more computational resources and is harder to train end-to-end (Table 6), primarily due to the use of fully connected layers (albeit in a convolutional manner) We report several comparisons with DeconvNet later in the paper Sec. 4.

As compared to SegNet, U-Net [16] (proposed for the medical imaging community) does not reuse pooling indices but instead transfers the entire feature map (at the cost of more memory) to the corresponding decoders and concatenates them to upsampled (via deconvolution) decoder feature maps. There is no conv5 and max-pool 5 block in U-Net as in the VGG net architecture. SegNet, on the other hand, uses all of the pre-trained convolutional layer weights from VGG net as pre-trained weights.

3.1 Decoder Variants

Many segmentation architectures [2], [3], [4] share the same encoder network and they only vary in the form of their decoder network. Of these we choose to compare the SegNet decoding technique with the widely used Fully Convolutional Network (FCN) decoding technique [2], [10].

In order to analyse SegNet and compare its performance with FCN (decoder variants) we use a smaller version of SegNet, termed **SegNet-Basic** [1], which has 4 encoders and 4 decoders. All the encoders in SegNet-Basic perform max-pooling and subsampling and the corresponding decoders upsample its input using the received max-pooling indices. Batch normalization is used after each convolutional layer in both the encoder and decoder network. No biases are used after convolutions and no ReLU nonlinearity is present in the decoder network. Further, a constant kernel size of 7×7 over all the encoder and decoder layers is chosen to provide a wide context for smooth labelling *i.e.* a pixel in the deepest layer feature map (layer 4) can be traced back to a context window in the input image of 106×106 pixels. This small size of SegNet-Basic allows us to explore many different variants (decoders) and train them in reasonable time. Similarly we create **FCN-Basic**, a comparable version of FCN for our analysis which shares the same encoder network as SegNet-Basic but with the FCN decoding technique (see Fig. 3) used in all its decoders.

On the left in Fig. 3 is the decoding technique used by SegNet (also SegNet-Basic), where there is no learning involved in the upsampling step. However, the upsampled maps are convolved with trainable multi-channel decoder filters to densify its sparse inputs. Each decoder filter has the same number of channels as the number of upsampled feature maps. A smaller variant is one where the decoder filters are single channel, i.e they only convolve their corresponding upsampled feature map. This variant (**SegNet-Basic-SingleChannelDecoder**) reduces the number of trainable parameters and inference time significantly.

On the right in Fig. 3 is the FCN (also FCN-Basic) decoding technique. The important design element of the FCN model is dimensionality reduction step of the encoder feature maps. This *compresses* the encoder feature maps which are then used in the corresponding decoders. Dimensionality reduction of the encoder feature maps, say of 64 channels, is performed by convolving them with $1 \times 1 \times 64 \times K$ trainable filters, where K is the number of classes. The compressed K channel final encoder layer feature maps are the input to the decoder network. In a decoder of this network, upsampling is performed by *inverse convolution* using a fixed or trainable *multi-channel upsampling kernel*. We set the kernel size to 8×8. This manner of upsampling is also termed as *deconvolution*. Note that, in comparison, SegNet the multi-channel convolution using trainable decoder filters is performed after upsampling to densifying feature maps. The upsampled feature map in FCN has K channels. It is then added element-wise to the corresponding resolution encoder feature map to produce the output decoder feature map. The upsampling kernels are initialized using bilinear interpolation weights [2].

The FCN decoder model requires storing encoder feature maps during inference. This can be memory intensive for embedded applications; for e.g. storing 64 feature maps of the first layer of FCN-Basic at 180×240 resolution in 32 bit floating point precision takes 11MB. This can be made smaller using dimensionality reduction to the 11 feature maps which requires $\approx$ 1.9MB storage. SegNet on the other hand requires almost negligible storage cost for the pooling indices (.17MB if stored using 2 bits per 2×2 pooling window). We can also create a variant of the FCN-Basic model which discards the encoder feature map addition step and only learns the upsampling kernels (**FCN-Basic-NoAddition**).

In addition to the above variants, we study upsampling using fixed bilinear interpolation weights which therefore requires no learning for upsampling (**Bilinear-Interpolation**). At the other extreme, we can add 64 encoder feature maps at each layer to the corresponding output feature maps from the SegNet decoder to create a more memory intensive variant of SegNet (**SegNet-Basic-EncoderAddition**). Here both the pooling indices for upsampling are used, followed by a convolution step to densify its sparse input. This is then added element-wise to the corresponding encoder feature maps to produce a decoders output.

Another and more memory intensive FCN-Basic variant (**FCN-Basic-NoDimReduction**) is where there is no dimensionality reduction performed for the encoder feature maps. This implies that unlike FCN-Basic the final encoder feature map is not compressed to K channels before passing it to the decoder network. Therefore, the number of channels at the end of each decoder is the same as the corresponding encoder (i.e 64).

We also tried other generic variants where feature maps are simply upsampled by *replication* [7], or by using a fixed (and

1. SegNet-Basic was earlier termed SegNet in a archival version of this paper [12]

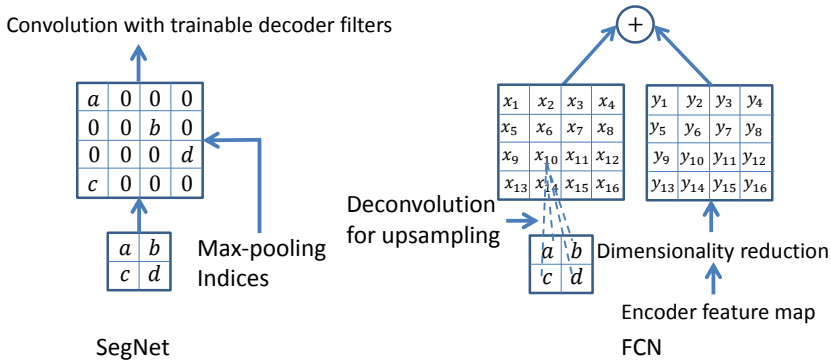

Fig. 3. An illustration of SegNet and FCN [2] decoders. a, b, c, d correspond to values in a feature map. SegNet uses the max pooling indices to upsample (without learning) the feature map(s) and convolves with a trainable decoder filter bank. FCN upsamples by learning to deconvolve the input feature map and adds the corresponding encoder feature map to produce the decoder output. This feature map is the output of the max-pooling layer (includes sub-sampling) in the corresponding encoder. Note that there are no trainable decoder filters in FCN.

				Median frequency balancing							Natural frequency balancing						
				Test				Train			Test				Train		
Variant	Params (M)	Storage multiplier	Infer time (ms)	G	C	mIoU	BF	G	C	mIoU	G	C	mIoU	BF	G	C	mIoU
Fixed upsampling																	
Bilinear-Interpolation	0.625	0	24.2	77.9	61.1	43.3	20.83	89.1	90.2	82.7	82.7	52.5	43.8	23.08	93.5	74.1	59.9
Upsampling using max-pooling indices																	
SegNet-Basic	1.425	1	52.6	82.7	62.0	47.7	35.78	94.7	96. 2	92.7	84.0	54.6	46.3	36.67	96.1	83.9	73.3
SegNet-Basic-EncoderAddition	1.425	64	53.0	83.4	**63.6**	48.5	35.92	94.3	95.8	92.0	**84.2**	56.5	**47.7**	36.27	95.3	80.9	68.9
SegNet-Basic-SingleChannelDecoder	0.625	1	33.1	81.2	60.7	46.1	31.62	93.2	94.8	90.3	83.5	53.9	45.2	32.45	92.6	68.4	52.8
Learning to upsample (bilinear initialisation)																	
FCN-Basic	0.65	11	24.2	81.7	62.4	47.3	**38.11**	92.8	93.6	88.1	83.9	55.6	45.0	**37.33**	92.0	66.8	50.7
FCN-Basic-NoAddition	0.65	n/a	23.8	80.5	58.6	44.1	31.96	92.5	93.0	87.2	82.3	53.9	44.2	29.43	93.1	72.8	57.6
FCN-Basic-NoDimReduction	1.625	64	44.8	**84.1**	63.4	**50.1**	37.37	**95.1**	**96.5**	**93.2**	83.5	**57.3**	47.0	37.13	**97.2**	**91.7**	**84.8**
FCN-Basic-NoAddition-NoDimReduction	1.625	0	43.9	80.5	61.6	45.9	30.47	92.5	94.6	89.9	83.7	54.8	45.5	33.17	95.0	80.2	67.8

TABLE 1
Comparison of decoder variants. We quantify the performance using global (G), class average (C), mean of intersection over union (mIoU) and a semantic contour measure (BF). The testing and training accuracies are shown as percentages for both natural frequency and median frequency balanced training loss function. SegNet-Basic performs at the same level as FCN-Basic but requires only storing max-pooling indices and is therefore more memory efficient during inference. Note that the theoretical memory requirement reported is based only on the size of the first layer encoder feature map. FCN-Basic, SegNet-Basic, SegNet-Basic-EncoderAddition all have high BF scores indicating the need to use information in encoder feature maps for better class contour delineation. Networks with larger decoders and those using the encoder feature maps in full perform best, although they are least efficient in terms of inference time and memory.

sparse) array of indices for upsampling. These performed quite poorly in comparison to the above variants. A variant without max-pooling and sub-sampling in the encoder network (decoders are redundant) consumes more memory, takes longer to converge and performs poorly. Finally, please note that to encourage reproduction of our results we release the Caffe implementation of all the variants [2].

3.2 Training

We use the CamVid road scenes dataset to benchmark the performance of the decoder variants. This dataset is small, consisting of 367 training and 233 testing RGB images (day and dusk scenes) at 360×480 resolution. The challenge is to segment 11 classes such as road, building, cars, pedestrians, signs, poles, side-walk etc. We perform local contrast normalization [54] to the RGB input.

The encoder and decoder weights were all initialized using the technique described in He *et al.* [55]. To train all the variants we use stochastic gradient descent (SGD) with a fixed learning rate of 0.1 and momentum of 0.9 [17] using our Caffe implementation of SegNet-Basic [56]. We train the variants until the training loss converges. Before each epoch, the training set is shuffled and each mini-batch (12 images) is then picked in order thus ensuring that each image is used only once in an epoch. We select the model which performs highest on a validation dataset.

We use the cross-entropy loss [2] as the objective function for training the network. The loss is summed up over all the pixels in a mini-batch. When there is large variation in the number of pixels in each class in the training set (e.g road, sky and building pixels dominate the CamVid dataset) then there is a need to weight the loss differently based on the true class. This is termed *class balancing*. We use *median frequency balancing* [13] where the weight assigned to a class in the loss function is the ratio of the median of class frequencies computed on the entire training set divided by the class frequency. This implies that larger classes in the training set have a weight smaller than 1 and the weights of the smallest classes are the highest. We also experimented with training the different variants without class balancing or equivalently using *natural frequency balancing*.

3.3 Analysis

To compare the quantitative performance of the different decoder variants, we use three commonly used performance measures: global accuracy (G) which measures the percentage of pixels

2. See http://mi.eng.cam.ac.uk/projects/segnet/ for our SegNet code and web demo.

correctly classified in the dataset, class average accuracy (C) is the mean of the predictive accuracy over all classes and mean intersection over union (mIoU) over all classes as used in the Pascal VOC12 challenge [21]. The mIoU metric is a more stringent metric than class average accuracy since it penalizes false positive predictions. However, mIoU metric is not optimized for directly through the class balanced cross-entropy loss.

The mIoU metric otherwise known as the Jacard Index is most commonly used in benchmarking. However, Csurka et al. [57] note that this metric does not always correspond to human qualitative judgements (ranks) of good quality segmentation. They show with examples that mIoU favours region smoothness and does not evaluate boundary accuracy, a point also alluded to recently by the authors of FCN [58]. Hence they propose to complement the mIoU metric with a boundary measure based on the Berkeley contour matching score commonly used to evaluate unsupervised image segmentation quality [59]. Csurka et al. [57] simply extend this to semantic segmentation and show that the measure of semantic contour accuracy used in conjunction with the mIoU metric agrees more with human ranking of segmentation outputs.

The key idea in computing a semantic contour score is to evaluate the F1-measure [59] which involves computing the precision and recall values between the predicted and ground truth class boundary given a pixel tolerance distance. We used a value of 0.75% of the image diagonal as the tolerance distance. The F1-measure for each class that is present in the ground truth test image is averaged to produce an image F1-measure. Then we compute the whole test set average, denoted the boundary F1-measure (BF) by average the image F1 measures.

We test each architectural variant after each 1000 iterations of optimization on the CamVid validation set until the training loss converges. With a training mini-batch size of 12 this corresponds to testing approximately every 33 epochs (passes) through the training set. We select the iteration wherein the global accuracy is highest amongst the evaluations on the validation set. We report all the three measures of performance at this point on the held-out CamVid test set. Although we use class balancing while training the variants, it is still important to achieve high global accuracy to result in an overall smooth segmentation. Another reason is that the contribution of segmentation towards autonomous driving is mainly for delineating classes such as roads, buildings, side-walk, sky. These classes dominate the majority of the pixels in an image and a high global accuracy corresponds to good segmentation of these important classes. We also observed that reporting the numerical performance when class average is highest can often correspond to low global accuracy indicating a perceptually noisy segmentation output.

In Table 1 we report the numerical results of our analysis. We also show the size of the trainable parameters and the highest resolution feature map or pooling indices storage memory, i.e, of the first layer feature maps after max-pooling and sub-sampling. We show the average time for one forward pass with our Caffe implementation, averaged over 50 measurements using a 360×480 input on an NVIDIA Titan GPU with cuDNN v3 acceleration. We note that the upsampling layers in the SegNet variants are not optimised using cuDNN acceleration. We show the results for both testing and training for all the variants at the selected iteration. The results are also tabulated without class balancing (natural frequency) for training and testing accuracies. Below we analyse the results with class balancing.

From the Table 1, we see that bilinear interpolation based upsampling without any learning performs the worst based on all the measures of accuracy. All the other methods which either use learning for upsampling (FCN-Basic and variants) or learning decoder filters after upsampling (SegNet-Basic and its variants) perform significantly better. This emphasizes the need to learn decoders for segmentation. This is also supported by experimental evidence gathered by other authors when comparing FCN with SegNet-type decoding techniques [4].

When we compare SegNet-Basic and FCN-Basic we see that both perform equally well on this test over all the measures of accuracy. The difference is that SegNet uses less **memory** during inference since it only stores max-pooling indices. On the other hand FCN-Basic stores **encoder feature maps** in full which consumes much more memory (11 times more). SegNet-Basic has a decoder with 64 feature maps in each decoder layer. In comparison FCN-Basic, which uses dimensionality reduction, has fewer (11) feature maps in each decoder layer. This reduces the number of convolutions in the decoder network and hence FCN-Basic is faster during inference (forward pass). From another perspective, the decoder network in SegNet-Basic makes it overall a larger network than FCN-Basic. This endows it with more flexibility and hence achieves higher training accuracy than FCN-Basic for the same number of iterations. Overall we see that SegNet-Basic has an advantage over FCN-Basic when inference time memory is constrained but where inference time can be compromised to some extent.

SegNet-Basic is most similar to FCN-Basic-NoAddition in terms of their decoders, although the decoder of SegNet is larger. Both learn to produce dense feature maps, either directly by learning to perform deconvolution as in FCN-Basic-NoAddition or by first upsampling and then convolving with trained decoder filters. The performance of SegNet-Basic is superior, in part due to its larger decoder size. The accuracy of FCN-Basic-NoAddition is also lower as compared to FCN-Basic. This shows that it is vital to capture the information present in the encoder feature maps for better performance. In particular, note the large drop in the BF measure between these two variants. This can also explain the part of the reason why SegNet-Basic outperforms FCN-Basic-NoAddition.

The size of the FCN-Basic-NoAddition-NoDimReduction model is slightly larger than SegNet-Basic since the final encoder feature maps are not compressed to match the number of classes K. This makes it a fair comparison in terms of the size of the model. The performance of this FCN variant is poorer than SegNet-Basic in test but also its training accuracy is lower for the same number of training epochs. This shows that using a larger decoder is not enough but it is also important to capture encoder feature map information to learn better, particular the fine grained contour information (notice the drop in the BF measure). Here it is also interesting to see that SegNet-Basic has a competitive training accuracy when compared to larger models such FCN-Basic-NoDimReduction.

Another interesting comparison between FCN-Basic-NoAddition and SegNet-Basic-SingleChannelDecoder shows that using max-pooling indices for upsampling and an overall larger decoder leads to better performance. This also lends evidence to SegNet being a good architecture for segmentation, particularly when there is a need to find a compromise between storage cost, accuracy versus inference time. In the best case, when both memory and inference time is not constrained, larger models such as FCN-Basic-NoDimReduction and SegNet-EncoderAddition are

both more accurate than the other variants. Particularly, discarding dimensionality reduction in the FCN-Basic model leads to the best performance amongst the FCN-Basic variants with a high BF score. This once again emphasizes the trade-off involved between memory and accuracy in segmentation architectures.

The last two columns of Table 1 show the result when no class balancing is used (natural frequency). Here, we can observe that without weighting the results are poorer for all the variants, particularly for class average accuracy and mIoU metric. The global accuracy is the highest without weighting since the majority of the scene is dominated by sky, road and building pixels. Apart from this all the inference from the comparative analysis of variants holds true for natural frequency balancing too, including the trends for the BF measure. SegNet-Basic performs as well as FCN-Basic and is better than the larger FCN-Basic-NoAddition-NoDimReduction. The bigger but less efficient models FCN-Basic-NoDimReduction and SegNet-EncoderAddition perform better than the other variants.

We can now summarize the above analysis with the following general points.

1) The best performance is achieved when encoder feature maps are stored in full. This is reflected in the semantic contour delineation metric (BF) most clearly.
2) When memory during inference is constrained, then compressed forms of encoder feature maps (dimensionality reduction, max-pooling indices) can be stored and used with an appropriate decoder (e.g. SegNet type) to improve performance.
3) Larger decoders increase performance for a given encoder network.

4 BENCHMARKING

We quantify the performance of SegNet on two scene segmentation benchmarks using our Caffe implementation [3]. The first task is road scene segmentation which is of current practical interest for various autonomous driving related problems. The second task is indoor scene segmentation which is of immediate interest to several augmented reality (AR) applications. The input RGB images for both tasks were 360×480.

We benchmarked SegNet against several other well adopted deep architectures for segmentation such as FCN [2], DeepLab-LargFOV [3] and DeconvNet [4]. Our objective was to understand the performance of these architectures when trained end-to-end on the same datasets. To enable end-to-end training we added batch normalization [51] layers after each convolutional layer. For DeepLab-LargeFOV, we changed the max pooling 3 stride to 1 to achieve a final predictive resolution of 45×60. We restricted the feature size in the fully connnected layers of DeconvNet to 1024 so as to enable training with the same batch size as other models. Here note that the authors of DeepLab-LargeFOV [3] have also reported little loss in performance by reducing the size of the fully connected layers.

In order to perform a controlled benchmark we used the same SGD solver [17] with a fixed learning rate of 10^{-3} and momentum of 0.9. The optimization was performed for more than 100 epochs through the dataset until no further performance increase was observed. Dropout of 0.5 was added to the end of deeper convolutional layers in all models to prevent overfitting (see http://mi.eng.cam.ac.uk/projects/segnet/tutorial.html for example caffe prototxt). For the road scenes which have 11 classes we used a mini-batch size of 5 and for indoor scenes with 37 classes we used a mini-batch size of 4.

3. Our web demo and Caffe implementation is available for evaluation at http://mi.eng.cam.ac.uk/projects/segnet/

4.1 Road Scene Segmentation

A number of road scene datasets are available for semantic parsing [22], [26], [60], [61]. Of these we choose to benchmark SegNet using the CamVid dataset [22] as it contains video sequences. This enables us to compare our proposed architecture with those which use *motion and structure* [28], [29], [30] and video segments [33]. We also combine [22], [26], [60], [61] to form an ensemble of 3433 images to train SegNet for an additional benchmark. For a web demo (see footnote 3) of road scene segmentation, we include the CamVid test set to this larger dataset. Here, we would like to note that another recent and independent segmentation benchmark on road scenes has been performed for SegNet and the other competing architectures used in this paper [62]. However, the benchmark was not controlled, meaning that each architecture was trained with a separate recipe with varying input resolutions and sometimes with a validation set included. Therefore, we believe our more controlled benchmark can be used to complement their efforts.

The qualitative comparisons of SegNet predictions with other deep architectures can be seen in Fig. 4. The qualitative results show the ability of the proposed architecture to segment smaller classes in road scenes while producing a smooth segmentation of the overall scene. Indeed, under the controlled benchmark setting, SegNet shows superior performance as compared to some of the larger models. DeepLab-LargeFOV is the most efficient model and with CRF post-processing can produce competitive results although smaller classes are lost. FCN with learnt deconvolution is clearly better than with fixed bilinear upsampling. DeconvNet is the largest model and the most inefficient to train. Its predictions do not retain small classes.

We also use this benchmark to first compare SegNet with several non deep-learning methods including Random Forests [27], Boosting [27], [29] in combination with CRF based methods [30]. This was done to give the user a perspective of the improvements in accuracy that has been achieved using deep networks compared to classical feature engineering based techniques.

The results in Table 2 show SegNet-Basic, SegNet obtain competitive results when compared with methods which use CRFs. This shows the ability of the deep architecture to extract meaningful features from the input image and map it to accurate and smooth class segment labels. The most interesting result here is the large performance improvement in class average and mIOU metrics that is obtained when a large training dataset, obtained by combining [22], [26], [60], [61], is used to train SegNet. Correspondingly, the qualitative results of SegNet (see Fig. 4) are clearly superior to the rest of the methods. It is able to segment both small and large classes well. We remark here that we used median frequency class balancing [50] in training SegNet-Basic and SegNet. In addition, there is an overall smooth quality of segmentation much like what is typically obtained with CRF post-processing. Although the fact that results improve with larger training sets is not surprising, the percentage improvement obtained using pre-trained encoder network and this training set indicates that this architecture can potentially be deployed for

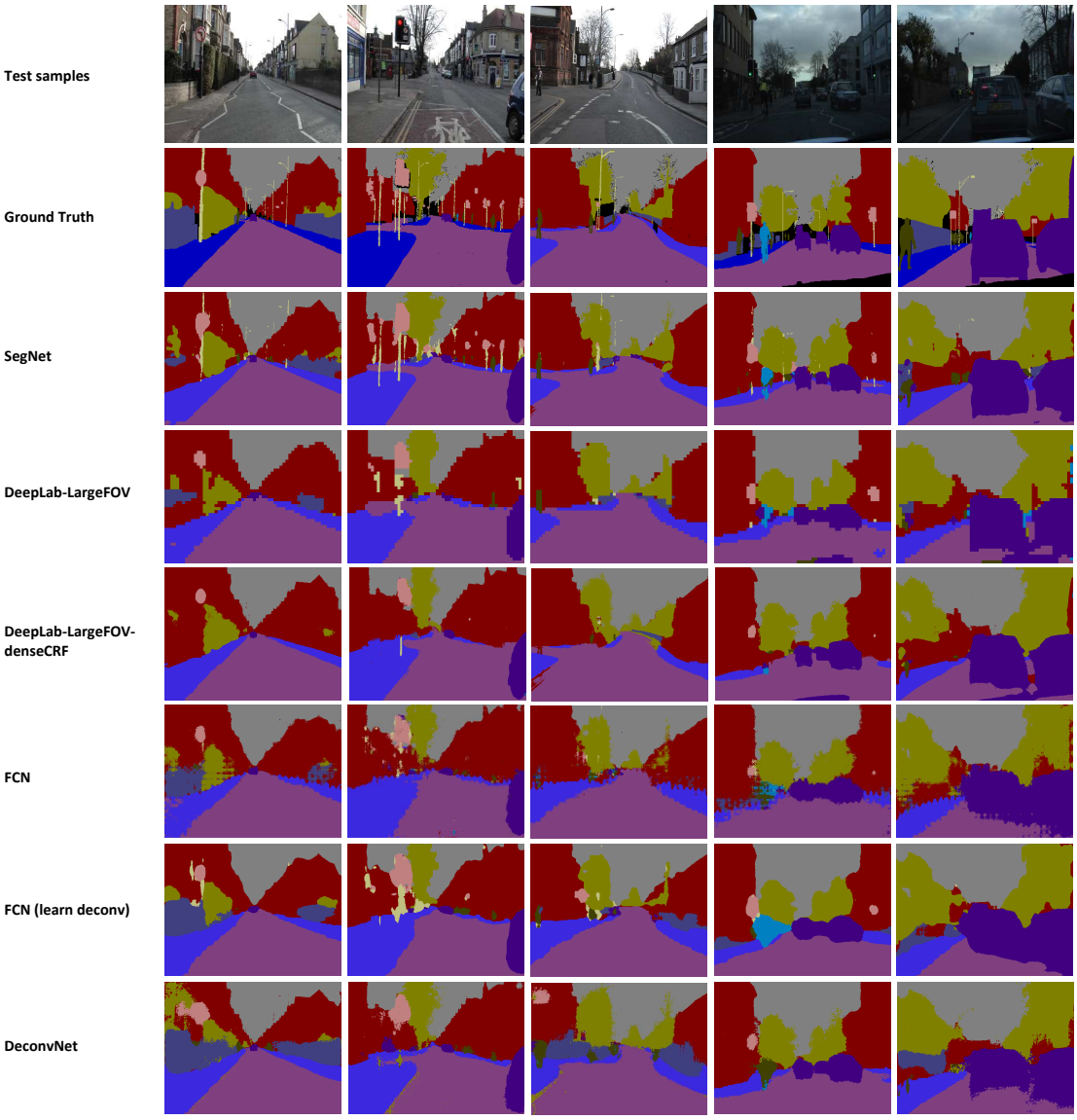

Fig. 4. Results on CamVid day and dusk test samples. SegNet shows superior performance, particularly with its ability to delineate boundaries, as compared to some of the larger models when all are trained in a controlled setting. DeepLab-LargeFOV is the most efficient model and with CRF post-processing can produce competitive results although smaller classes are lost. FCN with learnt deconvolution is clearly better. DeconvNet is the largest model with the longest training time, but its predictions loose small classes. Note that these results correspond to the model corresponding to the highest mIoU accuracy in Table 3.

practical applications. Our random testing on urban and highway images from the internet (see Fig. 1) demonstrates that SegNet can *absorb* a large training set and generalize well to unseen images. It also indicates the contribution of the prior (CRF) can be lessened when sufficient amount of training data is made available.

In Table 3 we compare SegNet's performance with now widely adopted fully convolutional architectures for segmentation. As compared to the experiment in Table 2, we did not use any class blancing for training any of the deep architectures including SegNet. This is because we found it difficult to train larger models such as DeconvNet with median frequency balancing. We benchmark performance at 40K, 80K and $>$80K iterations which given the mini-batch size and training set size approximately corresponds to $50, 100$ and $>$100 epochs. For the last test point we also report the maximum number of iterations (here atleast 150 epochs) beyond which we observed no accuracy improvements or when over-fitting set in. We report the metrics at three stages in the training phase to reveal how the metrics varied with training time, particularly for larger networks. This is important to understand if additional training time is justified when set against accuracy increases. Note also that for each evaluation we performed a complete run through the dataset to obtain batch norm statistics and then evaluated the test model with this statistic (see http://mi.eng.cam.ac.uk/projects/segnet/tutorial.html for code.). These evaluations are expensive to perform on large training sets and hence we only report metrics at three time points in the training phase.

From Table 3 we immediately see that SegNet, DeconvNet achieve the highest scores in all the metrics as compared to other models. DeconvNet has a higher boundary delineation accuracy but SegNet is much more efficient as compared to DeconvNet. This can be seen from the compute statistics in Table 6. FCN,

Method	Building	Tree	Sky	Car	Sign-Symbol	Road	Pedestrian	Fence	Column-Pole	Side-walk	Bicyclist	Class avg.	Global avg.	mIoU	BF
SfM+Appearance [28]	46.2	61.9	89.7	68.6	42.9	89.5	53.6	46.6	0.7	60.5	22.5	53.0	69.1	n/a*	
Boosting [29]	61.9	67.3	91.1	71.1	58.5	92.9	49.5	37.6	25.8	77.8	24.7	59.8	76.4	n/a*	
Dense Depth Maps [32]	85.3	57.3	95.4	69.2	46.5	**98.5**	23.8	44.3	22.0	38.1	28.7	55.4	82.1	n/a*	
Structured Random Forests [31]	n/a											51.4	72.5	n/a*	
Neural Decision Forests [64]	n/a											56.1	82.1	n/a*	
Local Label Descriptors [65]	80.7	61.5	88.8	16.4	n/a	98.0	1.09	0.05	4.13	12.4	0.07	36.3	73.6	n/a*	
Super Parsing [33]	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*	
SegNet (3.5K dataset training - 140K)	**89.6**	**83.4**	96.1	**87.7**	52.7	96.4	**62.2**	**53.45**	**32.1**	**93.3**	**36.5**	**71.20**	**90.40**	60.10	46.84
CRF based approaches															
Boosting + pairwise CRF [29]	70.7	70.8	94.7	74.4	55.9	94.1	45.7	37.2	13.0	79.3	23.1	59.9	79.8	n/a*	
Boosting+Higher order [29]	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 [30]	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*	

TABLE 2
Quantitative comparisons of SegNet with traditional methods on the CamVid 11 road class segmentation problem [22]. SegNet outperforms all the other methods, including those using depth, video and/or CRF's on the majority of classes. In comparison with the CRF based methods SegNet predictions are more accurate in 8 out of the 11 classes. It also shows a good $\approx 10\%$ improvement in class average accuracy when trained on a large dataset of 3.5K images. Particularly noteworthy are the significant improvements in accuracy for the smaller/thinner classes. * Note that we could not access predictions for older methods for computing the mIoU, BF metrics.

DeconvNet which have fully connected layers (turned into convolutional layers) train much more slowly and have comparable or higher forward-backward pass time with reference to SegNet. Here we note also that over-fitting was not an issue in training these larger models, since at comparable iterations to SegNet their metrics showed an increasing trend.

For the FCN model learning the deconvolutional layers as opposed to fixing them with bi-linear interpolation weights improves performance particularly the BF score. It also achieves higher metrics in a far lesser time. This fact agrees with our earlier analysis in Sec. 3.3.

Surprisingly, DeepLab-LargeFOV which is trained to predict labels at a resolution of 45×60 produces competitive performance given that it is the smallest model in terms of parameterization and also has the fastest training time as per Table 6. However, the boundary accuracy is poorer and this is shared by the other architectures. DeconvNet's BF score is higher than the other networks when trained for a very long time. Given our analysis in Sec. 3.3 and the fact that it shares a SegNet type architecture.

The impact of dense CRF [63] post-processing can be seen in the last time point for DeepLab-LargeFOV-denseCRF. Both global and mIoU improve but class average diminshes. However a large improvement is obtained for the BF score. Note here that the dense CRF hyperparameters were obtained by an expensive grid-search process on a subset of the training set since no validation set was available.

4.2 SUN RGB-D Indoor Scenes

SUN RGB-D [23] is a very challenging and large dataset of indoor scenes with 5285 training and 5050 testing images. The images are captured by different sensors and hence come in various resolutions. The task is to segment 37 indoor scene classes including wall, floor, ceiling, table, chair, sofa etc. This task is made hard by the fact that object classes come in various shapes, sizes and in different poses. There are frequent partial occlusions since there are typically many different classes present in each of the test images. These factors make this one of the hardest segmentation challenges. We only use the RGB modality for our training and testing. Using the depth modality would necessitate architectural modifications/redesign [2]. Also the quality of depth images from current cameras require careful post-processing to fill-in missing measurements. They may also require using fusion of many frames to robustly extract features for segmentation. Therefore we believe using depth for segmentation merits a separate body of work which is not in the scope of this paper. We also note that an earlier benchmark dataset NYUv2 [25] is included as part of this dataset.

Road scene images have limited variation, both in terms of the classes of interest and their spatial arrangements. When captured from a moving vehicle where the camera position is nearly always parallel to the road surface limiting variability in view points. This makes it easier for deep networks to learn to segment them robustly. In comparison, images of indoor scenes are more complex since the view points can vary a lot and there is less regularity in both the number of classes present in a scene and their spatial arrangement. Another difficulty is caused by the widely varying sizes of the object classes in the scene. Some test samples from the recent SUN RGB-D dataset [23] are shown in Fig. 5. We observe some scenes with few large classes and some others with dense clutter (bottom row and right). The appearance (texture and shape) can also widely vary in indoor scenes. Therefore, we believe this is the hardest challenge for segmentation architectures and methods in computer vision. Other challenges, such as Pascal VOC12 [21] salient object segmentation have occupied researchers more [66], but we believe indoor scene segmentation is more challenging and has more current practical applications such as in AR and robotics. To encourage more research in this direction we compared well known deep architectures on the large SUN RGB-D dataset.

The qualitative results of SegNet on samples of indoor scenes of different types such as bedroom, living room, laboratory, meeting room, bathroom are shown in Fig. 5. We see that SegNet obtains reasonable predictions when the size of the classes are large under different view points. This is particularly interesting since the input modality is only RGB. RGB images are also useful to segment thinner structures such as the legs of chairs and tables, lamps which is difficult to achieve using depth images from currently available sensors. This can be seen from the results of SegNet, DeconvNet in Fig. 5. It is also useful to segment decorative objects such as paintings on the wall for AR tasks. However as compared to outdoor scenes the segmentation quality

Network/Iterations	40K				80K				>80K				Max iter
	G	C	mIoU	BF	G	C	mIoU	BF	G	C	mIoU	BF	
SegNet	88.81	59.93	50.02	35.78	89.68	69.82	57.18	42.08	90.40	71.20	60.10	46.84	140K
DeepLab-LargeFOV [3]	85.95	60.41	50.18	26.25	87.76	62.57	53.34	32.04	88.20	62.53	53.88	32.77	140K
DeepLab-LargeFOV-denseCRF [3]	not computed								89.71	60.67	54.74	40.79	140K
FCN	81.97	54.38	46.59	22.86	82.71	56.22	47.95	24.76	83.27	59.56	49.83	27.99	200K
FCN (learnt deconv) [2]	83.21	56.05	48.68	27.40	83.71	59.64	50.80	31.01	83.14	64.21	51.96	33.18	160K
DeconvNet [4]	85.26	46.40	39.69	27.36	85.19	54.08	43.74	29.33	89.58	70.24	59.77	52.23	260K

TABLE 3
Quantitative comparison of deep networks for semantic segmentation on the CamVid test set when trained on a corpus of 3433 road scenes *without class balancing*. When end-to-end training is performed with the same and fixed learning rate, smaller networks like SegNet learn to perform better in a shorter time. The BF score which measures the accuracy of inter-class boundary delineation is significantly higher for SegNet, DeconvNet as compared to other competing models. DeconvNet matches the metrics for SegNet but at a much larger computational cost. Also see Table 2 for individual class accuracies for SegNet.

Network/Iterations	80K				140K				>140K				Max iter
	G	C	mIoU	BF	G	C	mIoU	BF	G	C	mIoU	BF	
SegNet	70.73	30.82	22.52	9.16	71.66	37.60	27.46	11.33	72.63	44.76	31.84	12.66	240K
DeepLab-LargeFOV [3]	70.70	41.75	30.67	7.28	71.16	42.71	31.29	7.57	71.90	42.21	32.08	8.26	240K
DeepLab-LargeFOV-denseCRF [3]	not computed								66.96	33.06	24.13	9.41	240K
FCN (learnt deconv) [2]	67.31	34.32	24.05	7.88	68.04	37.2	26.33	9.0	68.18	38.41	27.39	9.68	200K
DeconvNet [4]	59.62	12.93	8.35	6.50	63.28	22.53	15.14	7.86	66.13	32.28	22.57	10.47	380K

TABLE 4
Quantitative comparison of deep architectures on the SUNRGB-D dataset when trained on a corpus of 5250 indoor scenes. Note that only the RGB modality was used in these experiments. In this complex task with 37 classes all the architectures perform poorly, particularly because of the smaller sized classes and skew in the class distribution. DeepLab-Large FOV, the smallest and most efficient model has a slightly higher mIoU but SegNet has a better G,C,BF score. Also note that when SegNet was trained with *median frequency class balancing* it obtained 71.75, 44.85, 32.08, 14.06 (180K) as the metrics.

is clearly more noisy. The quality drops significantly when clutter is increased (see the result sample in the middle column).

The quantitative results in Table 4 show that all the deep architectures share low mIoU and boundary metrics. The global and class averages (correlates well with mIou) are also small. SegNet outperforms all other methods in terms of G,C, BF metrics and has a slightly lower mIoU than DeepLab-LargeFOV. As a stand alone experiment we trained SegNet with median frequency class balancing [67] and the metrics were higher (see Table 4) and this agrees with our analysis in Sec. 3.3. Interestingly, using the grid search based optimal hyperparameters for the dense-CRF worsened all except the BF score metric for DeepLab-LargeFOV-denseCRF. More optimal settings could perhaps be found but the grid search process was too expensive given the large inference time for dense-CRFs.

One reason for the overall poor performance is the large number of classes in this segmentation task, many of which occupy a small part of the image and appear infrequently. The accuracies reported in Table 5 clearly show that larger classes have reasonable accuracy and smaller classes have lower accuracies. This can be improved with larger sized datasets and class distribution aware training techniques. Another reason for poor performance could lie in the inability of these deep architectures (all are based on the VGG architecture [6]) to large variability in indoor scenes . This conjecture on our part is based on the fact that the smallest model DeepLab-LargeFOV produces the best accuracy in terms of mIoU and in comparison, larger parameterizations in DeconvNet, FCN did not improve perfomance even with much longer training (DeconvNet). This suggests there could lie a common reason for poor performance across all architectures. More controlled datasets [68] are needed to verify this hypothesis.

5 Discussion and Future Work

Deep learning models have often achieved increasing success due to the availability of massive datasets and expanding model depth and parameterisation. However, in practice factors like memory and computational time during training and testing are important factors to consider when choosing a model from a large bank of models. Training time becomes an important consideration particularly when the performance gain is not commensurate with increased training time as shown in our experiments. Test time memory and computational load are important to deploy models on specialised embedded devices, for example, in AR applications. From an overall efficiency viewpoint, we feel less attention has been paid to smaller and more memory, time efficient models for real-time applications such as road scene understanding and AR. This was the primary motivation behind the proposal of SegNet, which is significantly smaller and faster than other competing architectures, but which we have shown to be efficient for tasks such as road scene understanding.

Segmentation challenges such as Pascal [21] and MS-COCO [42] are object segmentation challenges wherein a few classes are present in any test image. Scene segmentation is more challenging due to the high variability of indoor scenes and a need to segment a larger number of classes simultaneously. The task of outdoor and indoor scene segmentation are also more practically oriented with current applications such as autonomous driving, robotics and AR.

The metrics we chose to benchmark various deep segmentation architectures like the boundary F1-measure (BF) was done to complement the existing metrics which are more biased towards region accuracies. It is clear from our experiments and other independent benchmarks [62] that outdoor scene images captured from a moving car are easier to segment and deep architectures perform robustly. We hope our experiments will encourage researchers to engage their attention towards the more challenging indoor scene

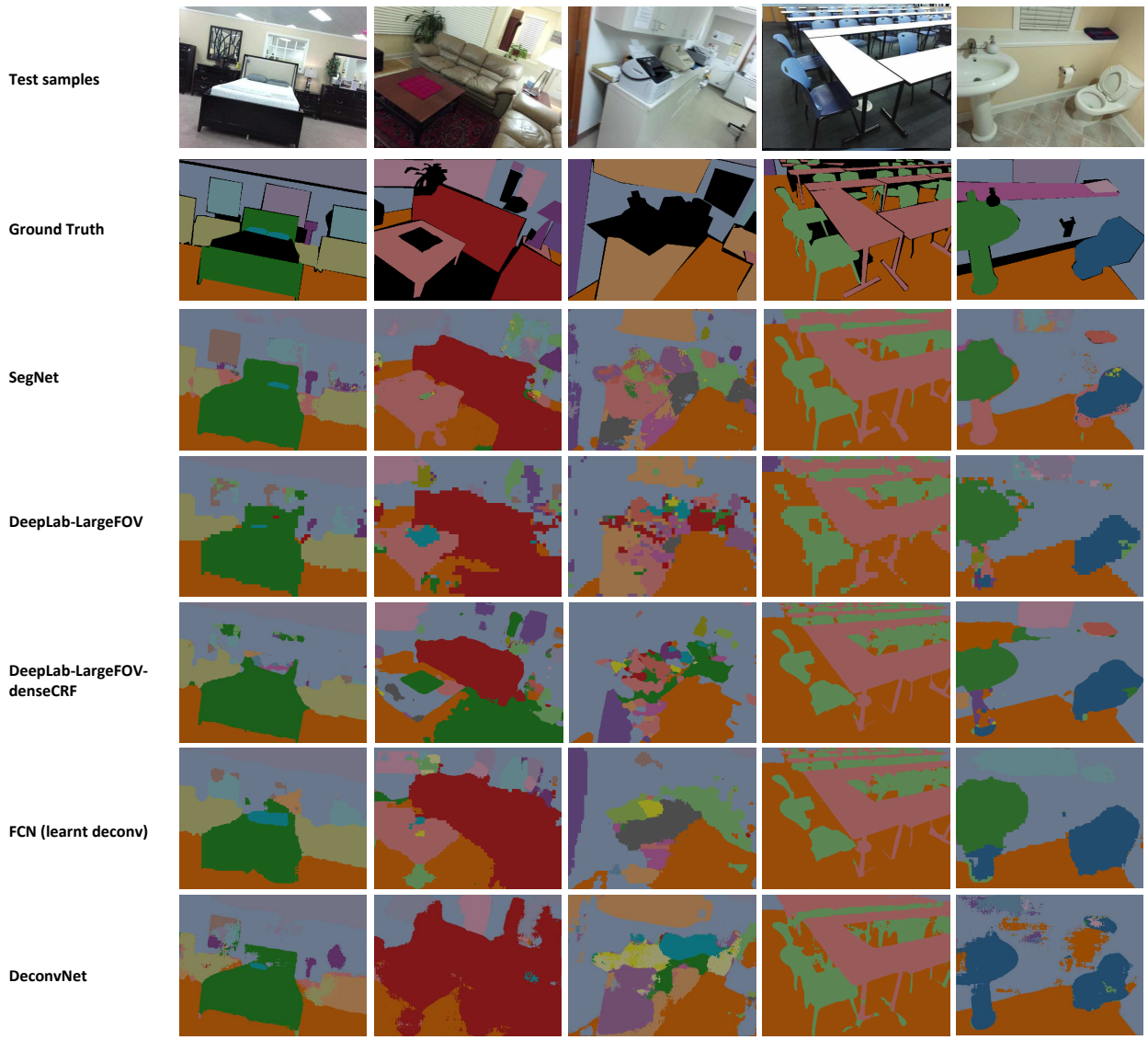

Fig. 5. Qualitative assessment of SegNet predictions on RGB indoor test scenes from the recently released SUN RGB-D dataset [23]. In this hard challenge, SegNet predictions delineate inter class boundaries well for object classes in a variety of scenes and their view-points. Overall rhe segmentation quality is better when object classes are reasonably sized but is very noisy when the scene is more cluttered. Note that often parts of an image of a scene do not have ground truth labels and these are shown in black colour. These parts are not masked in the corresponding deep model predictions that are shown. Note that these results correspond to the model corresponding to the highest mIoU accuracy in Table 4.

segmentation task.

An important choice we had to make when benchmarking different deep architectures of varying parameterization was the manner in which to train them. Many of these architectures have used a host of supporting techniques and multi-stage training recipes to arrive at high accuracies on datasets but this makes it difficult to gather evidence about their true performance under time and memory constraints. Instead we chose to perform a controlled benchmarking where we used batch normalization to enable end-to-end training with the same solver (SGD). However, we note that this approach cannot entirely disentangle the effects of model versus solver (optimization) in achieving a particular result. This is mainly due to the fact that training these networks involves gradient back-propagation which is imperfect and the optimization is a non-convex problem in extremely large dimensions. Acknowledging these shortcomings, our hope is that this controlled analysis complements other benchmarks [62] and reveals the practical trade-offs involved in different well known architectures.

For the future, we would like to exploit our understanding of segmentation architectures gathered from our analysis to design more efficient architectures for real-time applications. We are also interested in estimating the model uncertainty for predictions from deep segmentation architectures [69], [70].

6 Conclusion

We presented SegNet, a deep convolutional network architecture for semantic segmentation. The main motivation behind SegNet was the need to design an efficient architecture for road and indoor scene understanding which is efficient both in terms of memory and computational time. We analysed SegNet and compared it with other important variants to reveal the practical trade-offs involved in designing architectures for segmentation, particularly training time, memory versus accuracy. Those architectures which

Wall	Floor	Cabinet	Bed	Chair	Sofa	Table	Door	Window	Bookshelf	Picture	Counter	Blinds
83.42	93.43	63.37	73.18	75.92	59.57	64.18	52.50	57.51	42.05	56.17	37.66	40.29
Desk	Shelves	Curtain	Dresser	Pillow	Mirror	Floor mat	Clothes	Ceiling	Books	Fridge	TV	Paper
11.92	11.45	66.56	52.73	43.80	26.30	0.00	34.31	74.11	53.77	29.85	33.76	22.73
Towel	Shower curtain	Box	Whiteboard	Person	Night stand	Toilet	Sink	Lamp	Bathtub	Bag		
19.83	0.03	23.14	60.25	27.27	29.88	76.00	58.10	35.27	48.86	16.76		

TABLE 5
Class average accuracies of SegNet predictions for the 37 indoor scene classes in the SUN RGB-D benchmark dataset. The performance correlates well with size of the classes in indoor scenes. Note that class average accuracy has a strong correlation with mIoU metric.

Network	Forward pass(ms)	Backward pass(ms)	GPU training memory (MB)	GPU inference memory (MB)	Model size (MB)
SegNet	422.50	488.71	6803	**1052**	117
DeepLab-LargeFOV [3]	**110.06**	**160.73**	**5618**	1993	83
FCN (learnt deconv) [2]	317.09	484.11	9735	1806	539
DeconvNet [4]	474.65	602.15	9731	1872	877

TABLE 6
A comparison of computational time and hardware resources required for various deep architectures. The caffe time command was used to compute time requirement averaged over 10 iterations with mini batch size 1 and an image of 360×480 resolution We used nvidia-smi unix command to compute memory consumption. For training memory computation we used a mini-batch of size 4 and for inference memory the batch size was 1. Model size was the size of the caffe models on disk. SegNet is most memory efficient during inference model.

store the encoder network feature maps in full perform best but consume more memory during inference time. SegNet on the other hand is more efficient since it only stores the max-pooling indices of the feature maps and uses them in its decoder network to achieve good performance. On large and well known datasets SegNet performs competitively, achieving high scores for road scene understanding. End-to-end learning of deep segmentation architectures is a harder challenge and we hope to see more attention paid to this important problem.

REFERENCES

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[2] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, pp. 3431–3440, 2015.

[3] C. Liang-Chieh, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," in *ICLR*, 2015.

[4] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *ICCV*, pp. 1520–1528, 2015.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, pp. 1–9, 2015.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[7] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE PAMI*, vol. 35, no. 8, pp. 1915–1929, 2013.

[8] N. Hft, H. Schulz, and S. Behnke, "Fast semantic segmentation of rgb-d scenes with gpu-accelerated deep neural networks," in *KI 2014: Advances in Artificial Intelligence* (C. Lutz and M. Thielscher, eds.), vol. 8736 of *Lecture Notes in Computer Science*, pp. 80–85, Springer International Publishing, 2014.

[9] R. Socher, C. C. Lin, C. Manning, and A. Y. Ng, "Parsing natural scenes and natural language with recursive neural networks," in *ICML*, pp. 129–136, 2011.

[10] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1529–1537, 2015.

[11] W. Liu, A. Rabinovich, and A. C. Berg, "Parsenet: Looking wider to see better," *CoRR*, vol. abs/1506.04579, 2015.

[12] V. Badrinarayanan, A. Handa, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling," *CoRR*, vol. abs/1505.07293, 2015.

[13] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *ICCV*, pp. 2650–2658, 2015.

[14] G. Papandreou, L.-C. Chen, K. Murphy, and A. L. Yuille, "Weakly-and semi-supervised learning of a dcnn for semantic image segmentation," *arXiv preprint arXiv:1502.02734*, 2015.

[15] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[16] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, pp. 234–241, Springer, 2015.

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

[18] S. Hong, H. Noh, and B. Han, "Decoupled deep neural network for semi-supervised semantic segmentation," in *NIPS*, pp. 1495–1503, 2015.

[19] M. Ranzato, F. J. Huang, Y. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *CVPR*, 2007.

[20] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, *et al.*, "The role of context for object detection and semantic segmentation in the wild," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pp. 891–898, IEEE, 2014.

[21] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136.

[22] G. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *PRL*, vol. 30(2), pp. 88–97, 2009.

[23] S. Song, S. P. Lichtenberg, and J. Xiao, "Sun rgb-d: A rgb-d scene understanding benchmark suite," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 567–576, 2015.

[24] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *Computer Vision–ECCV 2014*, pp. 391–405, Springer, 2014.

[25] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, pp. 746–760, Springer, 2012.

[26] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *CVPR*, pp. 3354–3361, 2012.

[27] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *CVPR*, 2008.

[28] G. Brostow, J. Shotton, J., and R. Cipolla, "Segmentation and recognition using structure from motion point clouds," in *ECCV, Marseille*, 2008.

[29] P. Sturgess, K. Alahari, L. Ladicky, and P. H.S.Torr, "Combining appearance and structure from motion features for road scene understanding," in *BMVC*, 2009.

[30] L. Ladicky, P. Sturgess, K. Alahari, C. Russell, and P. H. S. Torr, "What, where and how many? combining object detectors and crfs," in *ECCV*, pp. 424–437, 2010.

[31] P. Kontschieder, S. R. Bulo, H. Bischof, and M. Pelillo, "Structured class-labels in random forests for semantic image labelling," in *ICCV*, pp. 2190–2197, IEEE, 2011.

[32] C. Zhang, L. Wang, and R. Yang, "Semantic segmentation of urban scenes using dense depth maps," in *ECCV*, pp. 708–721, Springer, 2010.

[33] J. Tighe and S. Lazebnik, "Superparsing," *IJCV*, vol. 101, no. 2, pp. 329–349, 2013.

[34] X. Ren, L. Bo, and D. Fox, "Rgb-(d) scene labeling: Features and algorithms," in *CVPR*, pp. 2759–2766, IEEE, 2012.

[35] A. Hermans, G. Floros, and B. Leibe, "Dense 3D Semantic Mapping of Indoor Scenes from RGB-D Images," in *ICRA*, 2014.

[36] S. Gupta, P. Arbelaez, and J. Malik, "Perceptual organization and recognition of indoor scenes from rgb-d images," in *CVPR*, pp. 564–571, IEEE, 2013.

[37] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Scene parsing with multiscale feature learning, purity trees, and optimal covers," in *ICML*, 2012.

[38] D. Grangier, L. Bottou, and R. Collobert, "Deep convolutional networks for scene parsing," in *ICML Workshop on Deep Learning*, 2009.

[39] C. Gatta, A. Romero, and J. van de Weijer, "Unrolling loopy top-down semantic feedback in convolutional deep networks," in *CVPR Workshop on Deep Vision*, 2014.

[40] P. Pinheiro and R. Collobert, "Recurrent convolutional neural networks for scene labeling," in *ICML*, pp. 82–90, 2014.

[41] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, pp. 1–42, April 2015.

[42] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014*, pp. 740–755, Springer, 2014.

[43] A. G. Schwing and R. Urtasun, "Fully connected deep structured networks," *arXiv preprint arXiv:1503.02351*, 2015.

[44] G. Lin, C. Shen, I. Reid, *et al.*, "Efficient piecewise training of deep structured models for semantic segmentation," *arXiv preprint arXiv:1504.01013*, 2015.

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

[46] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich, "Feedforward semantic segmentation with zoom-out features," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3376–3385, 2015.

[47] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *CVPR*, pp. 2528–2535, IEEE, 2010.

[48] K. Kavukcuoglu, P. Sermanet, Y. Boureau, K. Gregor, M. Mathieu, and Y. LeCun, "Learning convolutional feature hierarchies for visual recognition," in *NIPS*, pp. 1090–1098, 2010.

[49] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *ECCV*, pp. 184–199, Springer, 2014.

[50] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *NIPS*, pp. 2366–2374, 2014.

[51] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[52] V. Badrinarayanan, B. Mishra, and R. Cipolla, "Understanding symmetries in deep networks,"

[53] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," *CoRR*, vol. abs/1505.04366, 2015.

[54] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?," in *ICCV*, pp. 2146–2153, 2009.

[55] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *ICCV*, pp. 1026–1034, 2015.

[56] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 675–678, ACM, 2014.

[57] G. Csurka, D. Larlus, F. Perronnin, and F. Meylan, "What is a good evaluation measure for semantic segmentation?.," in *BMVC*, 2013.

[58] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *https://arxiv.org/pdf/1605.06211v1.pdf*, 2016.

[59] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 5, pp. 530–549, 2004.

[60] S. Gould, R. Fulton, and D. Koller, "Decomposing a scene into geometric and semantically consistent regions," in *ICCV*, pp. 1–8, IEEE, 2009.

[61] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: a database and web-based tool for image annotation," *IJCV*, vol. 77, no. 1-3, pp. 157–173, 2008.

[62] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," *arXiv preprint arXiv:1604.01685*, 2016.

[63] V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," in *In: NIPS (2011*, 2011.

[64] Bulo, S. Rota, and P. Kontschieder, "Neural decision forests for semantic image labelling.," in *CVPR*, 2014.

[65] Y. Yang, Z. Li, L. Zhang, C. Murphy, J. Ver Hoeve, and H. Jiang, "Local label descriptor for example based semantic image labeling," in *ECCV*, pp. 361–375, Springer, 2012.

[66] Z. Liu, X. Li, P. Luo, C.-C. Loy, and X. Tang, "Semantic image segmentation via deep parsing network," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1377–1385, 2015.

[67] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," *arXiv preprint arXiv:1411.4734*, 2014.

[68] A. Handa, V. Patraucean, V. Badrinarayanan, S. Stent, and R. Cipolla, "Scenenet: Understanding real world indoor scenes with synthetic data," in *CVPR*, 2016.

[69] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Insights and applications," in *Deep Learning Workshop, ICML*, 2015.

[70] A. Kendall, V. Badrinarayanan, and R. Cipolla, "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding," *arXiv preprint arXiv:1511.02680*, 2015.

Vijay Badrinarayanan obtained his Ph.D from INRIA Rennes, France in 2009. He was a senior post-doctoral research associate at the Machine Intelligence Laboratory, Department of Engineering, University of Cambridge, U.K. He currently works as a Principal Engineer, Deep Learning at Magic Leap, Inc. in Mountain View, CA. His research interests are in probabilistic graphical models, deep learning applied to image and video based perception problems.

Alex Kendall graduated with a Bachelor of Engineering with First Class Honours in 2013 from the University of Auckland, New Zealand. In 2014 he was awarded a Woolf Fisher Scholarship to study towards a Ph.D at the University of Cambridge, U.K. He is a member of the Machine Intelligence Laboratory and is interested in applications of deep learning for mobile robotics.

Roberto Cipolla obtained a B.A. (Engineering) degree from the University of Cambridge in 1984, an M.S.E. (Electrical Engineering) from the University of Pennsylvania in 1985 and a D.Phil. (Computer Vision) from the University of Oxford in 1991. from 1991-92 was a Toshiba Fellow and engineer at the Toshiba Corporation Research and Development Centre in Kawasaki, Japan. He joined the Department of Engineering, University of Cambridge in 1992 as a Lecturer and a Fellow of Jesus College. He became a Reader in Information Engineering in 1997 and a Professor in 2000. He became a Fellow of the Royal Academy of Engineering (FREng) in 2010. His research interests are in computer vision and robotics. He has authored 3 books, edited 9 volumes and co-authored more than 300 papers.