Computer Vision and Image Understanding 164 (2017) 16–26

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Speedup of deep learning ensembles for semantic segmentation using a model compression technique[☆]

Andrew Holliday[a], Mohammadamin Barekatain[b], Johannes Laurmaa[c], Chetak Kandaswamy[d], Helmut Prendinger[e,*]

[a] McGill University, Canada
[b] Technical University of Munich, Germany
[c] École Polytechnique Fédérale de Lausanne, Switzerland
[d] INESC Technology and Science, Portugal
[e] National Institute of Informatics, Tokyo

ARTICLE INFO

Article history:
Received 14 June 2016
Revised 22 March 2017
Accepted 11 May 2017
Available online 26 May 2017

Keywords:
Semantic segmentation
Model compression
Transfer learning
Real-time application

ABSTRACT

Deep Learning (DL) has been proven as a powerful recognition method as evidenced by its success in recent computer vision competitions. The most accurate results have been obtained by ensembles of DL models that pool their results. However, such ensembles are computationally costly, making them inapplicable to real-time applications. In this paper, we apply model compression techniques to the problem of semantic segmentation, which is one of the most challenging problems in computer vision. Our results suggest that compressed models can approach the accuracy of full ensembles on this task, combining the diverse strengths of networks of very different architectures, while maintaining real-time performance.

1. Introduction

Since the publication of the AlexNet architecture by Krizhevsky et al. (2012), Deep Learning has become the gold standard method in many areas of computer vision and pattern recognition. This has been enabled by the growing availability of “Big Data” (large, human-annotated databases of various kinds of data) and by the rapid advancement in GPU technology that has made parallel-processing computational hardware increasingly cheap and ubiquitous. DL algorithms have achieved higher accuracy than humans in some vision tasks, such as detection of traffic signs (Sermanet and LeCun, 2011) or playing Atari video games (Mnih et al., 2013). Yearly competitions for computer vision tasks such as the ImageNet Large Scale Visual Recognition Competition (ILSVRC) have shown continual improvement of DL models on a variety of tasks such as image classification. In these competitions, the winning results are routinely obtained not by individual DL networks, but by ensembles of multiple such networks that run individually on each input before finally combining their outputs to produce a result.

Ensembles of this form typically achieve accuracy several percentage points better than those of the individual models that comprise the ensemble. The downside of such ensembles, however, is that because they must run each of the networks individually, their computational costs are the sum of the costs of the individual networks. Modern DL models are already quite costly, with memory requirements on the order of gigabytes and processing times for a single image on the order of 100 to 1000 ms when run on a single high-end GPU. Ensembles thus are prohibitively costly for embedded real-time applications, such as scene understanding for autonomous cars or UAVs (Unmanned Aerial Vehicles).

Model compression techniques provide a possible solution to this issue. Model compression refers to any technique by which the knowledge and understanding of a set of computational models for a problem are compressed into a single model, while trying to minimize the information loss that results from this compression. These techniques have already proven useful in areas of character recognition (Bottou et al., 1994; Ciresan et al., 2012; LeCun et al., 1995) and speech recognition (Dahl et al., 2012; Hinton et al., 2012). However, to the best of our knowledge, no attempts have yet been made to apply these techniques to real-world scene understanding problems such as image classification or semantic segmentation.

Semantic segmentation is the problem of partitioning an image into discrete components that correspond to semantically mean-

☆ This work was conducted while the first four authors were participating in the NII International Internship program.
* Corresponding author.
E-mail addresses: helmut@nii.ac.jp, ahollid@cim.mcgill.ca (H. Prendinger).

http://dx.doi.org/10.1016/j.cviu.2017.05.004
1077-3142/© 2017 Published by Elsevier Inc.

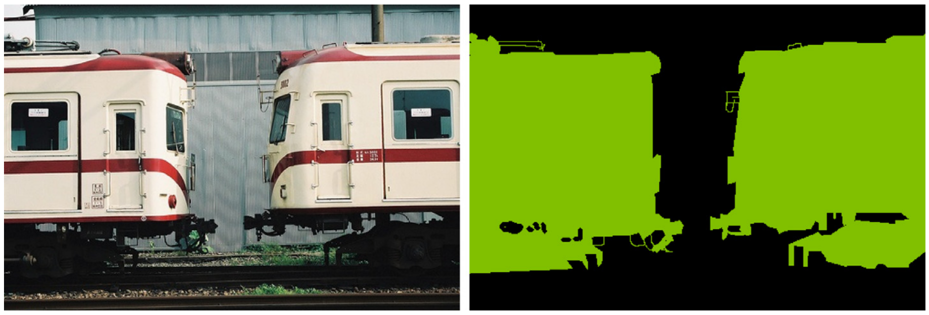


Fig. 1. Simple example of segmentation with only two categories: train (green) and background (black). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

ingful categories (see Fig. 1). It has recently gained considerable research attention (Badrinarayanan et al., 2015; Long et al., 2015; Zheng et al., 2015) as accurate semantic segmentation of an image—understanding which parts of an image correspond to what real-world entities—is fundamental to most real-world scene understanding problems.

The objective of this paper is to show that model compression can be used to produce a semantic segmentation model (SSM) that is light and fast enough for real-time embedded applications, and comparable in accuracy to an ensemble of ordinary SSMs. Our proposed method is a novel synthesis of several existing techniques. We use multiple existing architectures for the individual networks comprising our ensembles, as well as developing a new architecture, FCN-ResNet, by applying the up-sampling method described in Long et al. (2015) to the ResNet architecture detailed in He et al. (2016). To perform model compression, we expand on the ‘distillation’ technique proposed in Hinton et al. (2015) by applying it to ensembles consisting of several entirely different network architectures. We validate our approach on the Pascal VOC and SIFT Flow datasets. In the process, we empirically evaluate the importance of particular aspects of ConvNet architecture design for semantic segmentation.

The remainder of this paper is organized as follows. Section 2 describes some important related work and outlines how our contributions set us apart. In Section 3, we provide a rigorous definition of the semantic segmentation problem and outline the commonalities in recent DL-based attempts at solving it. Section 4 describes the details of our proposed method. Section 5 describes the aspects of our experimental setup common to all experiments. Sections 6 and 7 provide the results obtained by these experiments, and Section 8 provides a discussion on the results. Finally we discuss future work and give concluding remarks in Section 9.

2. Related work

Deep learning in the computer vision field has shown state-of-the-art performance in image classification and semantic segmentation tasks. Long et al. (2015) proposed a general method of adapting a Convolutional Neural Net (ConvNet for short) for image classification into a “Fully Convolutional Network” (FCN) for segmentation tasks. In their paper, they compare the results on the Pascal VOC 2011 validation set for FCNs adapted from AlexNet (Krizhevsky et al., 2012), VGG-16 (Simonyan and Zisserman, 2014) and GoogLeNet (Szegedy et al., 2015). Of these, the adapted VGG-16 architecture (termed FCN-8) performed the best.

Since their publication, however, new ConvNet architectures have been published that achieve better results on the image classification task, such as the Residual Networks proposed by He et al. (2016). They claim that the reason these “ResNets” obtain improved results is the increased network depth that they allow. However, they only demonstrate the network on the image classification problem.

Newer ConvNet-based semantic segmentation algorithms have been extended from the architecture of Long et al. (2015), such as SegNet (Badrinarayanan et al., 2015), DeConvnet (Noh et al., 2015), DeepLab (Chen et al., 2015; Papandreou et al., 2015) and CRF-RNN (Zheng et al., 2015). They are mainly a response to the poor spatial resolution of Fully Convolutional Networks. SegNet uses a more sophisticated up-sampling method that retains the location of the maxima in Max-pooling layers, for a more precise image reconstruction. DeepLab uses Conditional Random Fields, a type of probabilistic graphical model to output a finer segmentation. CRF-RNN goes further by combining ConvNets and CRFs into an end-to-end trainable network. For our model compression technique, we focus on the base architecture of Long et al., and attempt to apply it to the residual networks of He et al. (2016). We wish to note that after this work was conducted, we became aware of a similar synthesis of FCNs and residual networks conducted by Wu et al. (2016).

Bucilu et al. (2006) introduce the concept of model compression: they have shown that large ensembles of models can be compressed to a single model, by training the single model to mimic the outputs (logits) of the ensemble. The resultant compressed model outperforms a model of the same architecture that is trained from hand-labeled data instead of from the ensemble. Many model types were used in these experiments, including shallow neural networks, K-nearest-neighbor classifiers, and support vector machine classifiers, but the experiments did not deal with deep networks or ConvNets. They later showed that complex deep nets can sometimes be equaled in performance by smaller models (Ba and Caruana, 2014). The paper shows that models trained on the prediction targets taken from other models can be even more accurate than models trained on the original labels.

The model compression technique is furthermore extended by Hinton et al. (2015) to give a more generalized way of transferring the knowledge from a teacher network to a student network. They demonstrate their technique by compressing a single teacher ConvNet to a student ConvNet, and by compressing an ensemble of ConvNets with different random weight initializations. However, the ConvNets in their ensemble all had the same architecture. The work of Romero et al. about FitNets (Romero et al., 2014) extends the ideas of Hinton et al. (2015) to ConvNets and the CIFAR-100 dataset. Meanwhile, Chan et al. (2015) show that knowledge transference is possible not only between different deep learning models, but between any models with a common output format. In both Romero et al. (2014) and Chan et al. (2015), only knowledge transfer from a single teacher network is attempted. To our knowledge, our work represents the first attempt to perform model compression from a heterogeneous ensemble of ConvNet architectures to a single ConvNet architecture. It is also unique in applying model compression to the problem of semantic segmentation.

3. Semantic segmentation with ConvNets

For the purposes of this work, we formulate semantic segmentation as a discrete classification problem where each pixel $\mathbf{x}^{(m,\,n)}$ of an image $\mathbf{X}$ is to be assigned a class $y^{(m,n)} \in \{1, \ldots, c\}$, where c is the number of classes, and m and n are the position of the pixel in $\mathbf{X}$.

3.1. 'Extract-upscale' ConvNets

To perform semantic segmentation, we follow the techniques proposed in Long et al. (2015) and train extract-upscale ConvNet architectures. These are ConvNet architectures consisting of an 'extract' stage, a modified object classification ConvNet architecture such as VGGNet or ResNet, followed by a second 'upscale' stage, which up-samples the coarse output of the extract stage to the original resolution of the image. This enables us to make use of the many available object classification algorithms that have shown state-of-the-art performance in competitions such as ImageNet. The whole extract-upscale ConvNet model is trained on the labeled dataset $\mathcal{D} = \{(\mathbf{X}, \mathbf{G})\}_{i=1}^{N}$, a collection of N images and their respective ground-truth labelings.

State-of-the-art Deep Learning networks for object recognition (Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; Szegedy et al., 2016; 2015) all show similar patterns in their architectures. They are composed of multiple convolutional layers for extracting features, pooling layers for dimensionality reduction, and finally a set of fully connected layers that output a classification. To transform these object recognition algorithms into 'extract-upscale' architectures, we transform the final classification layers of the models from fully-connected layers into convolutional layers. The first classification layer is given a relatively large kernel size (such as 7) so as to cover some "patch" of the input image. The subsequent classification layers have kernel size 1, so that each output vector is a function of exactly one input vector from the previous layer, and all classification layers have stride 1. This maintains the spatial relationship of the output vectors to portions of the input image.

At the end of the 'Extract' stage, the image has a very low resolution. The 'Upscale' stage that follows converts this low-resolution segmentation into a segmentation of the same size as the original image. The 'Upscale' stage consists of a series of upscaling layers that perform bilinear interpolation. In some cases, these are interleaved with the skip connections introduced by Long et al. (2015), that connect earlier layers of the 'Extract' stage directly to the output. Features extracted earlier in the network carry more precise spatial information, and using them in this way improves the accuracy of the final segmentation. Two such networks proposed in Long et al. (2015) are FCN-16 and FCN-8, having respectively one and two skip connections.

These 'extract-upscale' networks will give as an output a vector of logits $\boldsymbol{z}^{(m,\,n)}$ for each pixel $\mathbf{x}^{(m,\,n)}$. The value of the ith element, $\boldsymbol{z}^{(m,\,n)}(i)$, corresponds to the ConvNet's confidence that pixel $(m,\,n)$ belongs to class i. The label of each pixel will then be computed as:

$$y^{(m,n)} = \arg\max_{i} \boldsymbol{z}^{(m,n)}(i) \tag{1}$$

3.2. Algorithm comparison

We performed a comparison of the sizes and computational costs of a variety of different ConvNet architectures to assess their suitability for real-time applications on embedded devices (see Table 1).

There is significant variety in the computational costs, number of parameters, and sizes of the different models we consider. Although deeper and larger models tend to have higher accuracy, accuracy also depends strongly on the network's architecture. VGG-16 is the largest and most computationally expensive model of this list. Lighter models such as AlexNet and GoogLeNet (Long et al., 2015) showed poor accuracy for segmentation.

4. Proposed method

This section is organized as follows. In Section 4.1, we describe the modified ResNet architecture that we use for compression, which we call FCN-ResNet. In Section 4.2, we detail the construction of the ensemble model that we use in our experiments. In Section 5.3 we explain the entire compression technique and how it incorporates the ensemble and the FCN-ResNet architecture.

4.1. Adapting residual networks for segmentation

The motivation that brought us to adapt residual networks for semantic segmentation were:

- Their ease of optimization (already the case for object recognition (He et al., 2016)).
- Their low forward computation time and small size, making them suitable for real-time applications.
- Their high accuracy on classification tasks, superior to that of the VGG-16 network that serves as the basis for FCN-8 (Long et al., 2015).

Inspired by the FCN structures described in Long et al. (2015), we transform a ResNet into an FCN-ResNet in the following way. Starting with a set of pre-trained ResNet weights, we remove the final average-pooling, fully-connected, and softmax layers of the network. We replace these with a 1×1 convolution layer with a number of filters corresponding to the number of categories, a 64×64 'deconvolution' layer with stride 32, and a crop layer. The 1×1 convolution layer is randomly initialized with variables drawn from a Gaussian distribution. The weights of the 64×64 'deconvolution' layer are manually set to bilinear interpolation, and it is "frozen" in this arrangement by setting its learning rate to 0.

We adapted three residual networks of different depths: ResNet-50, ResNet-101 and ResNet-152, having respectively 50, 101 and 152 layers.

4.2. Ensemble

Each of the Deep Learning networks learn to extract certain types of features. It is a common technique to combine many of these models into an ensemble, so that more types of features can be extracted and the different information contained in each network can be combined. This can lead to a more general system, because for images on which one model is prone to errors, it is likely that others will guess correctly. In this way, we achieve a more general and more accurate model.

We first train several individual semantic segmentation models, with different architectures, different hyperparameters, and different random initializations. We then combine them into an ensemble by merging the logits (outputs from the 'upscale' part of the network) from the R different models into one. Two techniques can be used for merging:

- Arithmetic averaging of logits: $\bar{\boldsymbol{z}}^{(m,n)} = \frac{1}{R}\sum_r \boldsymbol{z_r}^{(m,n)}$
- Geometric averaging of logits: $\bar{\boldsymbol{z}}^{(m,n)} = \sqrt[R]{\prod_r \boldsymbol{z_r}^{(m,n)}}$

The output of the ensemble is then a vector of logit values for each pixel, one value for each category. The semantic segmentation can then be computed simply by calculating the argmax of logits over the categories for each pixel.

Majority voting is another technique for combining outputs of different models. In this technique, the pixel labels are chosen

Table 1
Comparison of deep learning architectures.

Models	Network depth	Model size (MB)	Number of parameters (Millions)	Number of operations (GFLOP)
AlexNet (Krizhevsky et al., 2012)	8	240	61	1.5
VGG-16 (Simonyan and Zisserman, 2014)	16	553	138	15.3
GoogLeNet (Szegedy et al., 2015)	22	35	6.8	1.5
ResNet-50 (He et al., 2016)	50	102	0.8	3.8
Resnet-101 (He et al., 2016)	101	178	1.6	7.6
Resnet-152 (He et al., 2016)	152	241	2.3	11.3

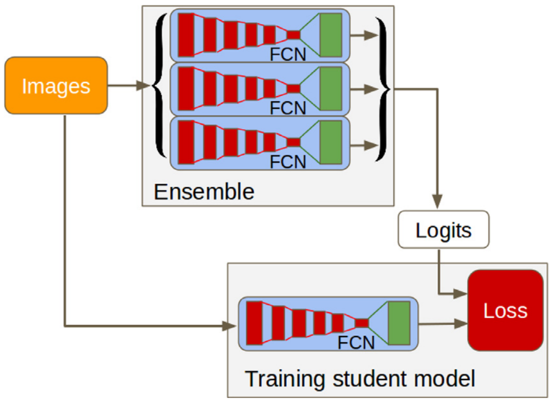


Fig. 2. Block diagram of model compression method for Ensemble of Deep Learning Models for Semantic Segmentation.

based on the logits for each model individually, and then the label chosen for each pixel by the largest number of the networks is provided as the output of ensemble. However, it tends to produce outputs with poorer mean IU (mean intersection over union, defined in Section 5.4) than the above methods. Further, because the outputs are a class label for each pixel instead of a full vector of logits, majority voting is not appropriate for our model compression technique.

4.3. Model compression

Model compression, as described in Bucilu et al. (2006) and Hinton et al. (2015), is the technique of using the outputs of an ensemble of models as the "ground truth" training inputs to a single "student" model (see Fig. 2). The advantage of doing this is that the resultant student model is comparable in size and speed to the individual models in the ensemble, and thus is much smaller and faster than the whole ensemble, while still achieving accuracy comparable to the whole ensemble.

The compression process occurs in two steps: pre-training and knowledge transfer. In the pre-training step, we train a semantic segmentation model S on a set of labeled images, $\mathcal{D}_{train}$, in the usual way. When the training converges, we commence the knowledge transfer step.

In the knowledge transfer step, we use a second dataset $\mathcal{D}_{transfer}$. We produce labels for this data set by running it through a pre-existing ensemble model, and storing the full logit vector produced by the ensemble for each pixel of each image. We then continue training S, but now train it to minimize the difference between its own output logit vectors and those of the ensemble, as in Bucilu et al. (2006). Specifically, the loss function in this stage is defined as the Euclidean distance between the two vectors:

$$Loss = \sum_{m,n} \| (\bar{\boldsymbol{z}}^{(m,n)} - \boldsymbol{v}^{(m,n)})^2 \| \tag{2}$$

$\bar{\boldsymbol{z}}^{(m,n)}$ being the vectors of logits ouputted by the ensemble for the pixel at location (m, n) in image $\mathbf{X}$, and $\boldsymbol{v}^{(m,\, n)}$ being the vectors of logits from S. Hinton et al. (2015) proposes increasing the temperature of the softmax function and then training on the resultant outputs instead of training directly on the logits. The advantage of their method is that it turns the temperature of the softmax function into a hyperparameter that can be tuned to optimize the distillation results, whereas training on the logits does not provide any additional hyperparameters. As their work demonstrates, however, training on the logits is in fact a special case of training on the tuned softmax outputs; for some particular setting of the softmax temperature, the two processes are equivalent. Furthermore, we found in practice that the logit training objective described above provided good results from the compression process. For these reasons, Eq. (2) was used as our loss objective in all experiments.

One powerful property of this model compression technique is that we can use extra unlabeled data as $\mathcal{D}_{transfer}$. In the knowledge transfer step, the student network will learn directly from the output of the ensemble, so the only data we need are input images. This property can be interesting in cases when labeled data is scarce, and therefore overfitting is a problem. The ensemble can be trained with relatively lower overfitting, and this generalization can be transferred to the student model.

5. Experimental setup and evaluation

All experiments were carried out using the Caffe neural network framework and NVIDIA's CUDA and CUDNN libraries. Due to changing availability of computing resources over the course of our evaluation, different hardware configurations were used for different experiments. These are described in separate sections for each experiment.

5.1. Datasets

To evaluate our technique, we conducted experiments using two different datasets of image with semantic segmentation ground truth labels. The first was the SBD dataset (Hariharan et al., 2011), an extended version of the Pascal VOC dataset (Everingham et al., 2012). The second was the SIFT Flow dataset, originally described by Liu et al. (2009).

5.2. Ensemble composition

In all experiments, we used arithmetic averaging rather than geometric averaging to combine the output logits of each model, as we found it gave slightly better results in practice. Shelhamer et al. report results for the FCN-32, FCN-16, and FCN-8 models on both Pascal VOC and SIFT Flow in Long et al. (2015), and they have made the trained weights of the models for each dataset publically available. In all experiments where these models appear in an ensemble, these pre-trained weights were used. For the other models in the ensembles, we adapted GoogLeNet, ResNet-50, ResNet-101 and ResNet-152 into Fully Convolutional Networks following the steps from Section 4.1. Then we trained them after initializing the networks from the publicly-available ImageNet-trained weights of plain GoogLeNet and plain ResNets. FCN-GoogLeNet and all FCN-ResNets were trained with learning rate 10^{-10}, momentum 0.99, and weight decay 0.0005. The training curves of our FCN-ResNet on Pascal VOC can be seen on Fig. 4.

5.3. Model compression

In the experiments for each dataset, an ensemble of the form described in Section 5.2 served as our teacher model. For the student model, FCN-8 and FCN-ResNet-152 were reasonable candidates, and we attempted to compress the ensemble to both models for each dataset. In every case, the student model was initialized with the weights of the same model trained directly on the dataset, whether obtained from public sources (as in the case of FCN-8) or trained ourselves. For the knowledge transfer stage, we kept the hyperparameters described in Section 5.2, but lowered the learning rate to 10^{-12}, and used the Euclidian loss from Eq. (2), except where stated otherwise in the sections on each experiment. The training curves can be seen on Fig. 5.

5.4. Evaluation metrics

Our evaluation metrics are based on a confusion matrix, which crosses the predicted segmentation of $\mathbf{Y}$ of the network with the ground truth $\mathbf{G}$, to compute their similarity. Let $\mathcal{C}_{ij}$ be the number of pixels in all images in a dataset that in reality belong to class $i \in 1, \ldots, c$ and which are predicted by the model to belong to class j (in $\mathbf{Y}$). For example, if class 3 stands for "dog" and class 4 for "cat", then $\mathcal{C}_{34}$ is the number of pixels across the entire dataset which in reality are "dog" pixels, but which were labeled by the model as being "cat" pixels.

$$\mathcal{C}_{ij} = \sum_{\{(\mathbf{X},\mathbf{G})\} \in \mathcal{D}_{val}} \sum_{(m,n)} \{g^{(m,n)} = i \cap y^{(m,n)} = j\} \tag{3}$$

We can, then, define the following terms:

- Pixels that "truly" belong to class i : $K_i = \sum_{j=1}^{c} \mathcal{C}_{ij}$
- Pixels predicted to belong to class j : $L_j = \sum_{i=1}^{c} \mathcal{C}_{ij}$
- Total number of pixels in the dataset: $n_{pixels} = \sum_{i=1}^{c} \sum_{j=1}^{c} \mathcal{C}_{ij}$

To evaluate the performance of our models, we compute two different metrics:

- The overall pixel accuracy, which is the fraction of all pixels that are correctly labeled by the model:

$$pixel\ accuracy = \frac{\sum_{i=1}^{c} \mathcal{C}_{ii}}{n_{pixels}} \tag{4}$$

- The Mean Intersection over Union, or mean IU:

$$mean\ IU = \frac{1}{c} \sum_{i=1}^{c} \frac{\mathcal{C}_{ii}}{L_i + K_i - \mathcal{C}_{ii}}$$

  This is simply the unweighted average of the Jaccard indices of each category. This is the most widely-used metric for semantic segmentation.

Both of these metrics give a good sense of the overall accuracy of the system, but both may be misleading if the network is not equally accurate for all categories throughout the dataset. This may occur if the distribution of pixels of different categories in the dataset is uneven.

6. Pascal VOC experiments

6.1. Dataset

The segmentation ConvNets we use are trained on 8498 images from the SBD training set (Hariharan et al., 2011), an extended version of Pascal VOC dataset (Everingham et al., 2012). We refer to this as our labeled training dataset $\mathcal{D}_{train}$. The compression training uses the same dataset $\mathcal{D}_{transfer} = \mathcal{D}_{train}$. To evaluate the model accuracy, we validate on $\mathcal{D}_{val}$, which consists of 736 images from PASCAL VOC 2011, none of which are contained in $\mathcal{D}_{train}$. These are the same training and validation datasets used in Long et al. (2015).

6.2. Hardware

The Pascal VOC experiments were carried out on an NVIDIA Maxwell Titan X GPU.

6.3. Ensemble composition

In order to have as general of an ensemble as possible, we used a large variety of models: our ensemble is composed of FCN-32, FCN-16, FCN-8, FCN-GoogLeNet, FCN-ResNet-50, FCN-ResNet-101, and FCN-ResNet-152, all with equal weighting. The architecture and weights for all models in the ensemble were obtained as described in Section 5.2.

Accuracy could possibly be improved even further by adding more models trained with different hyperparameters/random initialization, or discarding the least accurate models. However, our goal was mainly to show that a collection of diverse models helps for generalization, and can be used to teach a single model to work better.

6.4. Model compression results

In practice, we found that FCN-ResNet-152 was easier to train as a student model of the ensemble than FCN-8 (see Section 4.1 for details on its construction). The accuracy of FCN-ResNet-152 improved when we trained it on the ensemble's logits using the hyperparameters described in Section 5.3, while we were unable to find hyperparameter settings that would allow FCN-8 to converge with this training objective.

In Table 2, we compare the accuracy and forward computation time of various trained or pre-trained single models, an ensemble of these models together and a compressed model from that ensemble. We observe that even though the compressed network does not achieve a mean IU and pixel accuracy as high as the ensemble, it is better than any of the single models in the ensemble, and its forward computation time is less than $\frac{1}{10}^{th}$ that of the ensemble. Thus our method has allowed us to capture some of the ensemble's superior performance, while maintaining the speed of a single FCN-ResNet.

Applying model compression to semantic segmentation provides new insights into how ensembles and model compression provide the benefits that they do. One can directly see how confusing are certain parts of an image to the ensemble's constituent models, and the extent to which the student network captures both their shared knowledge and some of their confusion. When analyzing the predictions output by our networks on Pascal VOC, we observe that knowledge transfer helps avoid some class confusions in certain pixel areas, as we can see in the horse's neck and the front of the train in Fig. 3. In these case, the ensemble has largely corrected for what one of its component networks perceived as an ambiguity by averaging its output with that of other networks that had correctly classified this part of the image. Learning from the ensemble in this regard, the compressed network also knows how to handle these ambiguities. The case of the sheep on the second row of Fig. 3 is particularly interesting. Here the student model succeeds in correcting an error preserved by the ensemble at the end of the sheep's hind leg. In Table 3, we can see the difference in per-class IU for the single FCN-ResNet-152 model, the ensemble and the compressed FCN-ResNet-152.

Table 2
Model accuracies on Pascal VOC.

FCN models	Mean IU (%)	Pixel acc. (%)	Forward time (s)
VGG-16 (32s)	63.6	90.5	0.1
VGG-16 (16s)	65.0	91.0	0.1
VGG-16 (8s)	65.5	91.2	0.1
GoogLeNet	54.7	88.4	**0.06**
ResNet-50	60.6	90.4	**0.06**
Resnet-101	64.0	91.2	0.11
Resnet-152	65.0	91.5	0.11
Ensemble	**67.7**	**92.1**	1.29
Compressed model (Resnet-152)	66.1	91.7	0.11

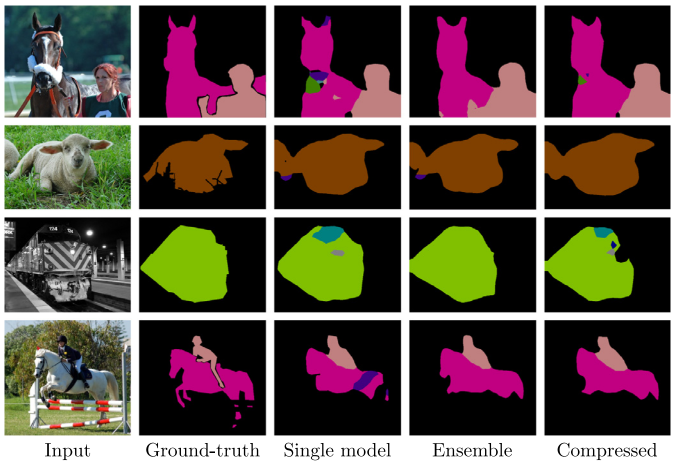


Fig. 3. Comparison between output labels on a sample Pascal VOC image for the single and compressed FCN-ResNet-152 models, the ensemble and the ground-truth. In most cases, the compressed model is getting closer to the segmentation quality of the ensemble.

7. SIFT Flow Experiments

7.1. Dataset

The SIFT Flow dataset (Liu et al., 2009) consists of 2688 images with full pixel-wise semantic labelings of 33 object categories and one "unlabeled" category. 2488 of these images are designated as training data, which form $\mathcal{D}_{train}$ in this experiment. As in the Pascal VOC experiment, the compression training uses the same dataset: $\mathcal{D}_{transfer} = \mathcal{D}_{train}$. The remaining 200 images are used as the validation set, $\mathcal{D}_{val}$. Of the 33 object categories, 3 are not actually present in the dataset, so while our network was trained to identify all 33 categories, our evaluation ignores the three absent categories. The SIFT Flow dataset also contains a separate set of geometric pixel-wise labels of the images, which consist of three classes: horizontal, vertical, and sky. As the pixel-wise geometric labeling problem is in essence the same as the semantic labeling problem, just with a smaller number of classes, we report our results on the geometric labels as well.

7.2. Hardware

The SIFT Flow experiments were carried out on an NVIDIA Tesla K40c GPU, with the exception of the training and testing of the FCN-ResNet-skip networks (described in Section 7.4). These were conducted on an NVIDIA GTX 980 GPU.

7.3. Ensemble composition

The FCN-GoogLeNet model was excluded from these experiments. It was observed to perform substantially worse than all other models in the ensemble used for Pascal VOC, and in practice, we were able to obtain good accuracy without including this model, so we deemed it unnecessary for this experiment. Its exclusion allowed us time to conduct a wider range of experiments than with Pascal VOC. The ensemble used for these experiments was composed of FCN-32, FCN-16, FCN-8, FCN-ResNet-50, FCN-ResNet-101, and FCN-ResNet-152 models. The publically-available FCN models for the SIFT Flow dataset use a branched network structure to learn on, and output results for, the semantic and geometric labels of the dataset simultaneously. To ensure that the

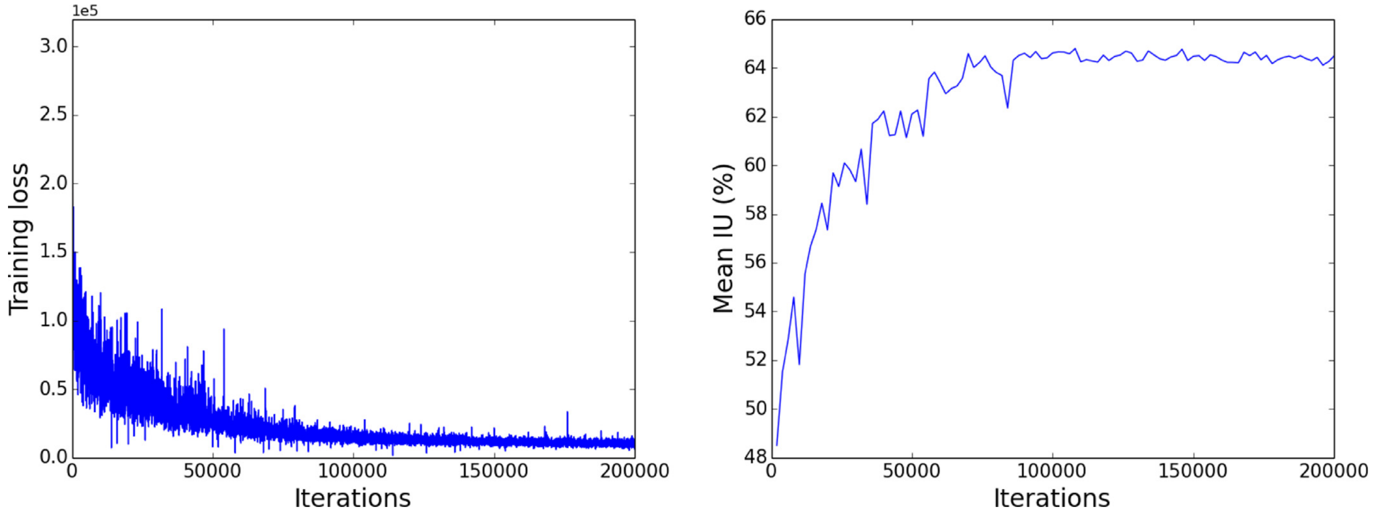
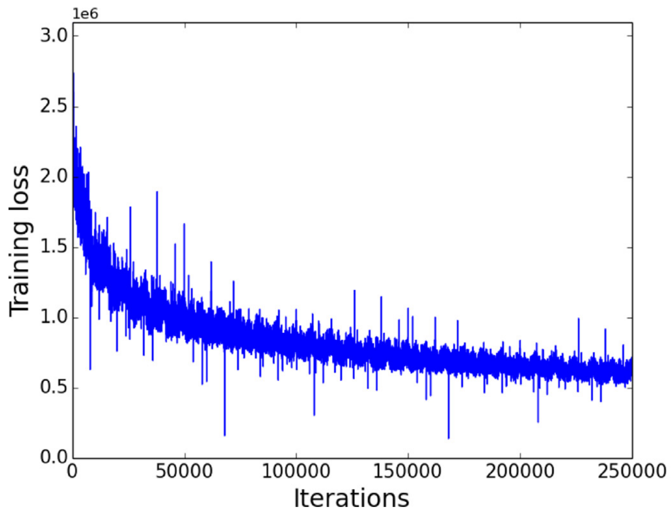


Fig. 4. Training curves of FCN-ResNet on Pascal VOC (Multinomial Logistic loss and Mean IU), finetuned from ResNet model for image classification.



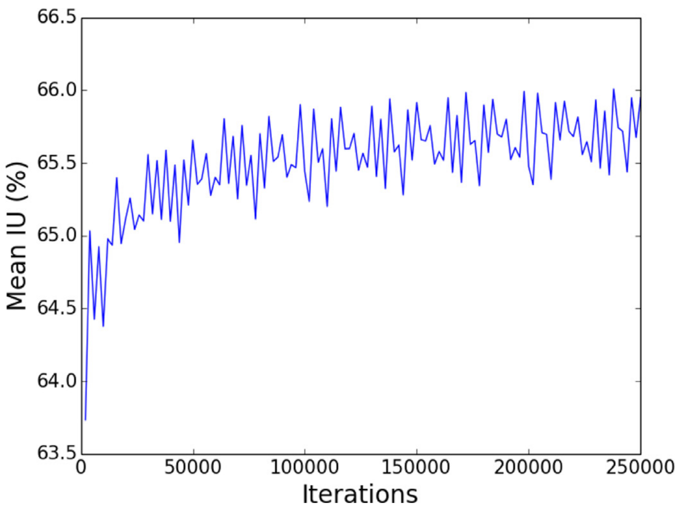


Fig. 5. Training curves (Euclidean loss and Mean IU) of knowledge transfer to FCN-ResNet from the Pascal VOC ensemble described in Section 6.3. We observe an increased mean IU compared to our FCN-ResNet trained under normal conditions (Fig. 4).

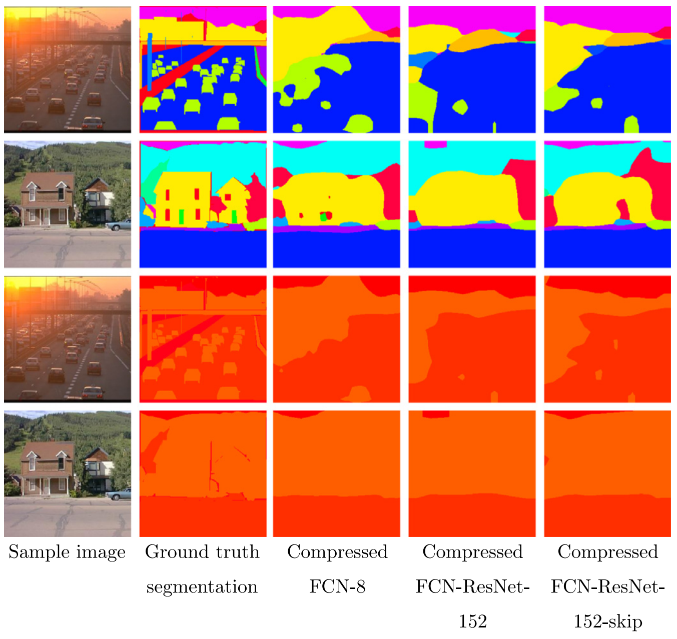

Sample image
Ground truth segmentation
Compressed FCN-8
Compressed FCN-ResNet-152
Compressed FCN-ResNet-152-skip

Fig. 6. Typical SIFT Flow images, their ground-truth labels, and the predictions of various compressed networks on those images. The top rows show semantic labels, the bottom rows show geometric labels.

accuracies and forward times were comparable, our FCN-ResNets were adapted to use the same branched structure as the FCN-32 SIFT Flow architecture, and were also trained on the semantic and geometric labels simultaneously. To perform model compression, our student networks were similarly trained on a training objective that compared trained the logits of each of the student's output branches to the logits of the ensemble's two output branches. The separate losses from each branch were given equal weight during training of all networks.

7.4. Model compression results

The performance of all ensemble and compressed models on SIFT Flow are presented in Table 4, while Table 5 contains per-class IU scores on the semantic labels. Example segmentations output by our network are shown in Fig. 6. The mean IU is reported on the semantic labels only, as the set of labels on the geometric set is too small for this to be a meaningful metric. Using the ensemble described above as our teacher model, we began by training both FCN-ResNet-152 and FCN-8 as student models. Whereas on Pascal VOC, we were unable to compress the ensemble network to an FCN-8 network with any success, on SIFT Flow the FCN-8 compressed model outperformed the FCN-ResNet-152 compressed model by a considerable margin: its mean IU was better by 1.4%, and its semantic pixel accuracy was better by 0.6%. The non-compressed FCN-ResNet-152 was outperformed by non-compressed FCN-8 by a similar margin.

We hypothesize that this may be because the FCN-ResNet networks simply perform bilinear interpolation to upscale the low-resolution classification map of the final convolutional layer. The FCN-8 network, by contrast, contains “skip” connections between the upscaling layers and earlier, higher-resolution convolutional layers, allowing this high-res information to directly inform the final segmentation, making it more accurate.

To test this hypothesis, we developed a new network architecture, FCN-ResNet-skip, which added a skip connection between the eleventh ReLU layer back from the final ReLU layer. That is, for FCN-ResNet-152-skip, a connection is made between the 140th ReLU output and the output of the first up-sampling layer; for FCN-ResNet-50-skip, the connection is made between the output of the 38th ReLU layer and that of the first up-sampling layer. These skip connections function in the same way as those of FCN-8, as described in Section 3.1.

We trained these FCN-ResNet-skip networks directly on SIFT Flow, and also used them as student networks of the SIFT Flow ensemble for compression. As shown in Table 4, the addition of a single skip connection was enough for the SIFT Flow-trained FCN-ResNet-152-skip network to outperform all VGG-based models, including FCN-8, which uses two skip connections. This network even outperforms the compressed FCN-8 model by 0.4% in mean IU and geometric pixel accuracy, and matches its semantic pixel accuracy, while having a lower forward time. Compressing the ensemble to an FCN-ResNet-152-skip network achieved the highest accuracy of any single network we tested on SIFT Flow. Remarkably, it even exceeds the mean IU of the ensemble network used to train it by 0.9%, though the ensemble's pixel accuracy is still 0.4% higher on the semantic labels and 0.3% higher on the geometric labels.

Equally significant is the fact that the SIFT Flow-trained FCN-ResNet-50-skip network exceeds the performance of the SIFT Flow-trained FCN-ResNet-101 network. Whereas FCN-ResNet-101 outper-

Table 3
IU scores on each object class in the Pascal VOC dataset. Evidently, some classes of objects are easier to correctly segment than others.

IU per class	Background	Aeroplane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Dining table	Dog	Horse	Motorbike	Person	Potted plant	Sheep	Sofa	Train	TV/Monitor
Single model	91.3	78.5	34.2	83.0	50.0	74.7	75.6	74.4	82.5	27.5	71.2	33.5	73.5	63.6	77.8	80.3	46.1	70.7	34.3	78.0	64.4
Compressed model	91.3	79.1	34.9	**83.8**	50.4	74.1	**76.7**	75.5	83.4	28.2	**72.0**	**39.9**	76.3	64.8	76.8	80.9	48.7	71.0	**36.7**	78.7	63.8
Ensemble	**91.5**	**82.2**	**37.9**	82.5	**59.0**	**78.0**	76.5	**77.9**	**84.4**	**28.8**	71.7	34.0	**78.1**	**70.8**	**79.1**	**82.3**	**50.6**	**77.4**	33.9	**81.5**	**65.0**

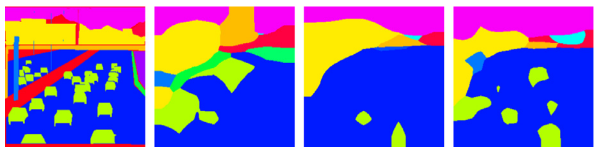


Fig. 7. A comparison of the base FCN-ResNet-101 output on a SIFT Flow test image, and those of two compressed nets of half the depth.

forms regular FCN-ResNet-50 by 0.6% in both mean IU and semantic accuracy and by 0.2% in geometric accuracy, FCN-ResNet-50-skip outperforms FCN-ResNet-50 by 1.7% in mean IU, 1.5% in semantic accuracy, and 0.6% in geometric accuracy. The improvement from adding the skip connection was more than double the improvement gained by increasing the depth of the network by a factor of 2.

7.5. Compression to small networks

We were curious to what extent the accuracy of the ensemble could be preserved by compressing to a very small model. To test this, we performed compression of the ensemble described in Section 7.3 to an FCN-ResNet-50 network, and to an FCN-ResNet-50-skip network. The results are included in Table 4. The FCN-ResNet-50-skip network achieved accuracy approximately equivalent to the most accurate models inside the ensemble, while being smaller than all but smallest model in the ensemble, although its accuracy was significantly worse than the ensemble as a whole. Much of its benefit was likely derived from the addition of its skip layer, as detailed in 7.4.

The compressed FCN-ResNet-50 network is perhaps more illustrative. Although its performance is worse still than that of the compressed FCN-ResNet-50-skip network, only two of the ensemble's models (FCN-8 and FCN-ResNet-152) clearly exceed its performance. The compressed model improves substantially on the performance of the same architecture in the ensemble (+0.5% mean IU, +1.0% semantic accuracy, +0.3% geometric accuracy), and performs comparably well with the ensemble's FCN-ResNet-101 model (-0.1% mean IU, +0.4% semantic accuracy, +0.1% geometric accuracy), which has twice as many layers as the student network. A sample of the outputs of these three networks in comparison with the ground truth can be seen in Fig. 7.

This result has practical implications for scenarios, such as embedded applications, in which run-time computational resources are at a premium. By training a small network on the outputs of an ensemble of larger networks, the accuracy of the small model can, with no additional cost in memory or evaluation time when running the trained network, be meaningfully improved in accuracy, even equaling the performance of much deeper networks.

8. Discussion

Our experiments show that an ensemble of ConvNets with diverse architectures and different strengths and weaknesses can be compressed to a single network that combines the strengths of all of the ensemble's constituents as the ensemble does, without loss of performance. This is particularly evident from the results of compressing the SIFT Flow ensemble to an FCN-ResNet-152-skip network. The fact that this compressed network significantly exceeds the ensemble in mean IU suggests that when an ensemble of simple models is compressed to a more complex student model, the student model can learn not just to replicate the ensemble's

Table 4
Model accuracies on SIFT Flow.

FCN models	Semantic mean IU (%)	Semantic pixel acc. (%)	Geometric pixel acc. (%)	Forward time (s)
VGG-16 (32s)	33.9	84.3	93.6	0.14
VGG-16 (16s)	38.0	84.9	93.8	0.14
VGG-16 (8s)	37.5	85.9	94.6	0.14
ResNet-50	34.9	83.9	93.8	**0.07**
ResNet-101	35.5	84.5	94.0	0.09
ResNet-152	36.7	84.9	94.3	0.12
ResNet-50-skip (Not part of ensemble)	36.6	85.4	94.4	**0.07**
ResNet-152-skip (Not part of ensemble)	38.8	86.3	95.0	0.12
Ensemble	39.1	**86.9**	**95.1**	1.25
Compressed models				
VGG-16 (8s)	38.4	86.3	94.6	0.14
FCN-ResNet-152	37.0	85.7	94.5	0.12
FCN-ResNet-152-skip	**40.0**	86.5	94.8	0.12
FCN-ResNet-50	35.4	84.9	94.1	**0.07**
FCN-ResNet-50-skip	37.9	85.2	94.2	**0.07**

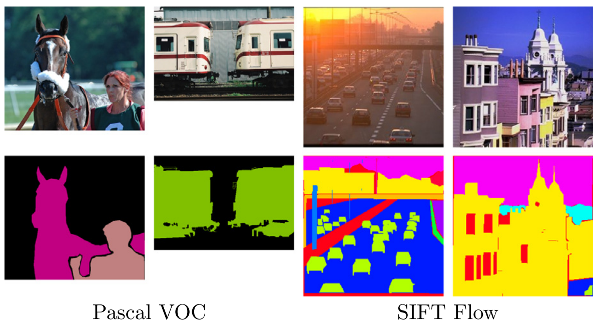


Fig. 8. Comparison of images and ground truth labels of Pascal VOC and SIFT Flow.

outputs, but to discern meaningful scene structure based on the ensemble's outputs that was not evident to the ensemble itself.

Another interesting observation is the difference between the relative performance of the FCN-32s, -16s, and -8s networks and the FCN-ResNets on the two different datasets. While FCN-ResNet-152 achieved comparable performance to FCN-8 when trained directly on Pascal VOC, on the SIFT Flow dataset FCN-8 performed considerably better than FCN-ResNet-152, both when trained directly on the labels and when trained on the logits of an ensemble. This is despite the fact that the base ResNet-152 architecture has superior classification performance to the VGG-16 architecture. As described in Section 7.4, we showed that this improvement was due to the superior up-sampling mechanism of FCN-8, which incorporates high-resolution structure in the image into the up-sampling process. Incorporating this mechanism to FCN-ResNet-152 brought its performance above that of FCN-8.

One major difference between the Pascal VOC and SIFT Flow datasets appears to account for this. Most of the images in Pascal VOC contain only a few (1 to 3) semantically labeled objects that fill a large fraction of the image, with the rest of the pixels being unlabeled. Many of the images in SIFT Flow, on the other hand, contain a large number of small labeled objects, and most pixels are labeled. Typical images from each dataset can be seen in Fig. 8.

This implies that the relative importance of the ability to discriminate between classes, and the ability to produce a highly detailed segmentation, depend strongly on the complexity of the image. The more complex an image is in terms of the number of objects it contains and to what extent different object categories are interleaved in the scene, the more important that detail becomes.

Practically speaking, for complex real-world scenes such as those contained in SIFT Flow, increasing the fine-grained accuracy of the segmentation by adding skip connections to a network may yield greater improvements to accuracy, and for a smaller increase in computational cost, than increasing the network's discriminative power by increasing its depth. This may be true even when the number of object categories is large.

We were surprised that although no hyperparameter setting could be found that would allow compression of the Pascal VOC ensemble to converge with an FCN-8 student network on Pascal VOC, the same hyperparameters used to train FCN-8 on SIFT Flow (with the learning rate reduced by a factor of 10^2) led to successful compression of the SIFT Flow ensemble.

Surprisingly, while compression of the Pascal VOC ensemble to an FCN-8 network could not be made to converge with any hyperparameter setting, compression of the SIFT Flow ensemble to this network architecture converged successfully with our default compression hyperparameters. Three possible causes suggest themselves. One is that the Pascal VOC dataset is in some sense more challenging than SIFT Flow. This seems unlikely, as SIFT Flow has a larger set of classes than Pascal VOC, and its labels are generally more intricate. Another possibility is that the difference was due to the split output of the networks trained on SIFT Flow, which were used to learn and output both the semantic and geometric labels simultaneously. This split would have a substantial effect on the magnitudes of the gradients received by the network during back-propagation.

The final possibility is that the inclusion of the FCN-GoogLeNet architecture in the Pascal VOC ensemble was responsible for FCN-8's failure to converge. As stated in Section 7.3, no FCN-GoogLeNet was included in the SIFT Flow ensemble. The FCN-GoogLeNet's performance was substantially worse than that of any other network in the Pascal VOC ensemble. It is possible that it degraded the quality of the ensemble's logit outputs enough that FCN-8 was not easily able to learn from them, even though our FCN-ResNet was able to. This is consistent with our observation in 4.1 that the training of FCN-8 networks is highly sensitive to the hyperparameter settings, while ResNets train properly over a much wider range of hyperparameters.

9. Conclusions and future work

We have demonstrated a novel synthesis of techniques whereby a single semantic segmentation ConvNet can be trained to replicate some of the accuracy of a ensemble of ConvNets with heterogeneous architectures, retaining the real-time performance of an individual net while outperforming all of the individual ConvNets in the ensemble, and in some cases even the ensemble as a whole. We have also discovered that for complex, difficult-to-segment scenes, the overall performance of a ConvNet depends more strongly on the sophistication of its up-sampling scheme than on its depth.

Table 5
IU scores on each object class in the SIFT Flow dataset. While the ensemble performs best on the majority of classes, there are several, such as Grass and Rock, where a compressed model performs very significantly better.

IU per class	Awning	Balcony	Bird	Boat	Bridge	Building	Bus	Car	Crosswalk	Door	Fence	Field	Grass	Mountain	Person	Plant	Pole	River	Road	Rock
Single FCN-8	**34.3**	**5.2**	0.0	31.4	**22.2**	83.7	**2.4**	69.8	32.1	24.9	31.5	48.4	36.1	75.5	**23.4**	28.7	**13.2**	63.9	84.0	22.7
Compressed FCN-8	9.5	0.0	0.0	**34.9**	20.1	83.3	0.0	64.6	6.1	25.6	31.4	53.9	**39.5**	76.6	**23.4**	**30.0**	0.0	**68.2**	83.1	**30.0**
Ensemble	13.9	0.0	0.0	24.3	20.2	**84.8**	0.0	**70.3**	**34.0**	**29.9**	33.8	48.4	39.2	**79.2**	21.7	16.9	0.0	62.6	**85.3**	23.1

IU per class (cont'd)	Sand	Sea	Sidewalk	Sign	Sky	Staircase	Streetlight	Sun	Tree	Window
Single FCN-8	51.5	77.6	49.2	21.5	93.0	59.1	**3.7**	49.8	69.4	30.9
Compressed FCN-8	51.1	76.2	51.3	15.6	92.3	64.2	0.0	60.1	70.0	29.7
Ensemble	**56.8**	**78.2**	**52.0**	**30.7**	**93.4**	**68.2**	0.1	**83.3**	**71.0**	**31.1**

There are various aspects of this work that we believe deserve further study. We could expect that the more different the ensemble accuracy and single model accuracies are (e.g. for small datasets), the more useful model compression will be. To this end, we would like to study the case of training the student network from an ensemble consisting of multiple class-specific models of various architectures in combination.

We wish to perform further experiments to determine why compression to an FCN-8 network failed on Pascal VOC but succeeded on SIFT Flow. The two most likely probable causes, described in 8, are both amenable to experimentation, by modifying the contents of the ensembles and by modifying the networks themselves.

Another area worth investigating is the ability of some compressed models to outperform the teacher ensemble's accuracy, such as in FitNets (Romero et al., 2014). This may be because a logit vector is a more information-rich representation of "ground-truth" than a single label value. In our SIFT Flow experiments, our FCN-ResNet-152-skip student model was able to exceed the ensemble's performance on one of two metrics; we would like to apply this compression to this model with other datasets to determine how general this effect may be.

As mentioned previously, the knowledge transfer stage of the compression process does not require any labeled data. As long as the image distribution is the same, it is possible to use huge databases of unlabeled data for this process, which is considerably more convenient than labeled data. We experimented with this (see Section 6.1) but did not observe any significant improvement in the results. We suspect the reason for this is that, because the SBD training dataset is relatively large, there was little improvement to be made after training with it. We are planning to explore further cases where unlabeled data might be essential, and where we can fully realize the potential of our model compression technique.

Acknowledgments

The authors would like to thank the support of the National Institute of Informatics, Tokyo. We would also like to thank Dr. Kai Yan from LabRomance and enRoute Co., Ltd. for their financial support. In addition, we wish to thank Professor Yutaka Matsuo for his financial support through the Grant-in-Aid for Scientific Research on Innovative Areas.

References

Ba, J., Caruana, R., 2014. Do deep nets really need to be deep? In: Advances in Neural Information Processing Systems, pp. 2654–2662.
Badrinarayanan, V., Handa, A., Cipolla, R., 2015. Segnet: a deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. arXiv preprint arXiv:1510.07818v1.
Bottou, L., Cortes, C., Denker, J.S., Drucker, H., Guyon, I., Jackel, L.D., LeCun, Y., Muller, U.A., Sackinger, E., Simard, P., et al., 1994. Comparison of classifier methods: a case study in handwritten digit recognition. In: International Conference on Pattern Recognition. IEEE Computer Society Press. 77–77.
Bucilu, C., Caruana, R., Niculescu-Mizil, A., 2006. Model compression. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 535–541.
Chan, W., Ke, N.R., Lane, I., 2015. Transferring knowledge from a RNN to a DNN. INTERSPEECH.
Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L., 2015. Semantic image segmentation with deep convolutional nets and fully connected crfs. In: International Conference on Learning Representations (ICLR).
Ciresan, D., Meier, U., Schmidhuber, J., 2012. Multi-column deep neural networks for image classification. In: Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp. 3642–3649.
Dahl, G.E., Yu, D., Deng, L., Acero, A., 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. Trans/ Audio Speech Lang/ Process 20 (1), 30–42.
Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A., The PASCAL Visual Object Classes Challenge (VOC2012) 2012.

Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., Malik, J., 2011. Semantic contours from inverse detectors. In: Conference on Computer Vision (ICCV). IEEE, pp. 991–998.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Conference on Computer Vision and Pattern Recognition (CVPR). IEEE.

Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.-R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., et al., 2012. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. Signal Process. Mag. IEEE 29 (6), 82–97.

Hinton, G., Vinyals, O., Dean, J., 2015. Distilling the knowledge in a neural network. Deep Learning and Representation Learning Workshop.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105.

LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Muller, U., Sackinger, E., et al., 1995. Comparison of learning algorithms for handwritten digit recognition. In: International Conference on Artificial Neural Networks, vol. 60, pp. 53–60.

Liu, C., Yuen, J., Torralba, A., 2009. Nonparametric scene parsing: Label transfer via dense scene alignment. In: 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20–25 June 2009, Miami, Florida, USA, pp. 1972–1979. doi:10.1109/CVPRW.2009.5206536.

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

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M., 2013. Playing atari with deep reinforcement learning. Workshop on Deep Learning, NIPS.

Noh, H., Hong, S., Han, B., 2015. Learning deconvolution network for semantic segmentation. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1520–1528.

Papandreou, G., Chen, L.-C., Murphy, K., Yuille, A.L., 2015. Weakly- and semi-supervised learning of a DCNN for semantic image segmentation.

Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y., 2014. Fitnets: hints for thin deep nets. CoRR abs/1412.6550.

Sermanet, P., LeCun, Y., 2011. Traffic sign recognition with multi-scale convolutional networks. In: The 2011 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 2809–2813.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556.

Szegedy, C., Ioffe, S., Vanhoucke, V., 2016. Inception-v4, inception-resnet and the impact of residual connections on learning. ICLR 2016 Workshop.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9.

Wu, Z., Shen, C., van den Hengel, A., 2016. High-performance semantic segmentation using very deep fully convolutional networks. CoRR abs/1604.04339.

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