Mimicking Very Efficient Network for Object Detection

Quanquan Li[1], Shengying Jin[2], Junjie Yan[1]
[1]SenseTime [2]Beihang University
liquanquan@sensetime.com, jsychffy@gmail.com, yanjunjie@outlook.com

Abstract

Current CNN based object detectors need initialization from pre-trained ImageNet classification models, which are usually time-consuming. In this paper, we present a fully convolutional feature mimic framework to train very efficient CNN based detectors, which do not need ImageNet pre-training and achieve competitive performance as the large and slow models. We add supervision from high-level features of the large networks in training to help the small network better learn object representation. More specifically, we conduct a mimic method for the features sampled from the entire feature map and use a transform layer to map features from the small network onto the same dimension of the large network. In training the small network, we optimize the similarity between features sampled from the same region on the feature maps of both networks. Extensive experiments are conducted on pedestrian and common object detection tasks using VGG, Inception and ResNet. On both Caltech and Pascal VOC, we show that the modified 2.5× accelerated Inception network achieves competitive performance as the full Inception Network. Our faster model runs at 80 FPS for a 1000×1500 large input with only a minor degradation of performance on Caltech.

1. Introduction

Object detection is a fundamental problem in image understanding. It aims to determine where in the image the objects are and which category each object belongs to. Many popular deep convolutional neural network based object detection methods have been proposed, such as Faster R-CNN [28], R-FCN [6] and SSD [25]. Compared with traditional methods such as DPM [12], these CNN based frameworks achieve good performance on challenging dataset.

Since the pioneering work R-CNN [14], CNN based object detectors need a pre-trained ImageNet classification model for initialization to get the desired performance. According to the experiments in [22], the Fast R-CNN [13] with AlexNet trained from scratch gets the 40.4% AP on Pascal VOC 2007, while with ImageNet pre-trained AlexNet gets 56.8% AP. Due to this phenomenon, nearly all the modern detection methods can only train networks which have been trained on ImageNet before and cannot train a network from scratch to achieve comparable results. The result is that we can only use networks designed for classification task such as AlexNet [23], ZFNet [35], VGGNet [30] and ResNet [17], which are not necessarily optimal for detection. Due to the limitation, if we want to sweep different network configurations and find a more efficient network, we will need to pre-train these models on ImageNet classification task and then fine-tune them on detection task. The process is very expensive considering that training a ImageNet classification model needs several weeks even on multiple GPUs. Moreover, in experiments we find that smaller networks always perform poor on ImageNet classification so that fine-tuning them on detection also leads to poor detection performance.

Method	MR_{-2}	Parameters	test time (ms)
Inception R-FCN	7.15	2.5M	53.5
1/2-Inception Mimic R-FCN	7.31	625K	22.8
1/2-Inception finetuned from ImageNet	8.88	625K	22.8

Table 1: The parameters and test time of large and small models. Tested on TITANX with 1000×1500 input. The 1/2-Inception model trained by mimicking outperforms that fine-tuned from ImageNet pre-trained model. Moreover, it obtains similar performance as the large Inception model with only 1/4 parameters and achieves a 2.5× speed-up.

In this paper, we want to train more efficient detection networks without ImageNet pre-training. More importantly, we still need to achieve competitive performance as the large ImageNet pre-trained models. The basic idea is that if we already have a network that achieves satisfying performance for detection, the network can be used to supervise other network training for detection. The question then becomes how to use a detection network to supervise a more efficient network and keeps its accuracy for detection.

Similar ideas have been used in standard classification

task, such as [18, 2]. However, we find that they do not work well for this more complex detection task. The main problems are how and where to add the supervision from detection ground-truth and the one from a different network.

Our solution for mimicking in object detection comes from observation of modern CNN based detectors, including Faster R-CNN [28], R-FCN [6], SSD [25] and YOLO [27]. They all calculate a feature map and then use different methods to decode detection results from the feature map. In this way, detector can actually be divided into the jointly trained feature extractor and the feature decoder. The differences between the large network and the more efficient network lie in the feature extractor. To this end, our philosophy is that the mimicking supervision should be added in the feature map generated by the feature extractor; the ground-truth supervision should be added on the final feature decoder. For the mimicking supervision, we define a transformation on top of the feature map generated by the small network to a new feature. We want to minimize the Euclidean distance between this new feature and the feature generated by the large network.

For the ground-truth supervision, it is the same as the origin detector, such as the joint classification and localization loss in Fast R-CNN. In training, we first extract the feature map of each training image generated by the large network, and then use the feature maps and detection annotations to jointly train the detector with the small network initialized from scratch. One problem is that the feature map is of very high dimension, and we find that directly mimicking the feature map does not converge as expected. Since the feature extractor is region or proposal based, we sample features from regions to optimize, which leads to satisfying results.

The feature map mimicking technique proposed in the paper can naturally be extended. The first extension is that we can mimic across scales. In CNN based detection, we only need $1/4$ computation if we can reduce the width and height of the input image by half. However, it usually leads to significantly performance drop. We show that we can define a simple transformation to up-sample the feature map to a large scale and then mimic the transformed feature map. Another extension is that we can extend the mimicking technique to a two-stage procedure that further improves the performance.

We conduct experiments on Caltech pedestrian detection and Pascal VOC object detection using R-FCN and Faster R-CNN. On both Caltech and Pascal VOC, we show that the mimicked models demonstrate superior performance than the models fine-tuned from ImageNet pre-trained model. As shown in Table 1, the model with $1/4$ parameters achieves similar performance as the full Inception Network and the faster model achieves 4.5$\times$ acceleration and 16$\times$ compression with only a minor degradation of performance on Caltech detection tasks.

2. Related Work

The related work includes recent CNN based object detections, network mimicking and network training, as well as network acceleration.

A seminal CNN based object detection method is R-CNN [14], which uses the fine-tuned CNN to extract features from object proposals and SVM to classify them. The spatial pyramid pooling [16] and Fast R-CNN [13] extract features on top of a shared feature map to speed up the R-CNN. Faster R-CNN [28] further improves by predicting region proposals and classifying proposals in the shared feature map. A very recent work R-FCN [6] proposes the position-sensitive score map to share more computation. The R-CNN series takes object detection as a two-shot problem, including region proposal generation and region classification. Recently, one-shot methods have been proposed, such as YOLO and SSD. All these methods need to calculate the feature map which takes most of the computation. The mimicking technique we proposed is validated on Faster R-CNN and R-FCN, but it can be naturally extended to SSD, YOLO and other CNN feature map based methods.

Network mimicking or distilling are recently introduced model acceleration and compression approaches by [18, 2] aiming to train a more compact model that can learn from the output of a large model. [29] further improves this method by way of the implementation of deeper student models and hints from the intermediate representation learned by the teacher network. However, all these mimicking works, to our best knowledge, have only been validated on easy classification tasks [18, 2, 29, 33]. In this paper, we show how to extend the mimicking techniques for more challenging object detection tasks, and how to use it to train more efficient object detector.

Some works have been proposed to give better initialization or replace the ImageNet pre-train. [22] sets the weights of a network such that all units in the network train at roughly the same rate to avoid vanishing or exploding gradients. [1] and [8] learn an unsupervised representation from videos, and [8] uses spatial context as the source to provide supervision signal for training. These methods perform much better than being trained randomly from scratch, but they still have a large performance gap between the pre-trained method from ImageNet. The recent work [19] analyzes the ImageNet features in detail.

Our work is also related to works of network acceleration. [7, 20, 24] accelerate single layer of CNN through linear decomposition , while [38] considers the nonlinear approximation. [34] uses the quantization to accelerate the convolution. [15] combines pruning, quantization and Huffman coding to compress parameters. [31, 10, 5, 26] propose to approximate networks by binary weights. These methods

accelerate or compress a given network, but does not change the network structure itself. The mimicking techniques proposed in our paper are orthogonal to these methods and can be combined for further acceleration.

3. Mimicking for Object Detection

3.1. Logits Mimic Learning

The main idea of mimicking is to train a small neural network from the soft targets or logits (predictions before softmax) of a large model or an ensemble of large models. The soft targets carrying helpful information learned by the large complex model allow the small network to approximate the complex functions of the large network. As described in [2], the objective loss function we want the small network to optimize is logits regression with L2 loss given training data $\{(x^{(1)},z^{(1)}),...,(x^{(T)},z^{(T)})\}$

$$\mathcal{L}(W) = \frac{1}{2T}\sum_{t}\|g(x^{(t)};W) - z^{(t)}\|_2^2, \tag{1}$$

where W is the weights of the neural network, $\mathrm{g}(x^{(t)};W)$ is the model prediction of the t^{th} training data. By mimicking logits of the large model, the knowledge learned by the large complex model can be transferred to the small faster model so that it could achieve as accurate results as the large model.

To the best of our knowledge, mimic method, as a technique for model acceleration and compression, has only been applied on classification tasks [2, 18]. We want to extend this idea to object detection tasks to train small and faster object detector. Different from the single class score prediction in classification task, the object detection network usually predicts both the object scores and the object locations for the entire image. An intuitive idea would be to train the small network matching both outputs of the large network. Our experiments show that it is difficult to transfer the knowledge to the small model by this naive logits matching method in object detection frameworks. The mimic model performs worse than the model fine-tuned from ImageNet pre-trained models on the training data that supervised by the ground-truths only.

3.2. Feature Map Mimic Learning

Recent state of the art object detection algorithms [28, 25, 27] are fully convolutional networks in which the entire image is forwarded through the deep convolution networks once, afterwards the features from candidate windows are extracted on the feature maps. Feature matters in object detection since both the objectness scores and locations are predicted based on the feature map. Therefore, it is more reasonable to mimic the output feature maps between the two detection networks.

We present a feature map mimic method aiming to train the small model to mimic the feature map activations of the large model in a unified fully convolutional network object detection pipeline. The features come from the last convolution layer of the neural network involve the information not only of the strength of the responses but also of their spatial positions [16]. But unlike logits before the softmax in classification network whose dimensions are related to the number of categories, the features of the fully convolutional network dependent on input size and network architecture are high-dimensional. For a typical 600x1000 Pascal VOC image forwarded through the VGG16 model, the output of the network is a feature with dimension of million magnitude. It is difficult to perform regression between the two output feature maps of this high-dimension directly. Our experiment results show that the model is hard to converge during training time. Moreover, the feature map contains response information across the entire image. In the case of an image containing only few objects or in which the scales of objects are all small, most of the regions on the feature map will only have weak responses. The object correspondence information might be ignored or degraded if we perform global mimic learning on the whole feature map directly. For a fully convolutional network object detector, instead of the global context features, the features of local regions where the objects locate contains more representative information for object detection.

Therefore, we propose a new fully convolutional network feature mimic method by mimicking the features sampled from regions of proposals to solve the high-dimensional regression problem of fully convolutional feature map. The feature mimic method based on proposals sampling could also make the small network focus more on learning the region of interests features from the large model rather than the global context features. Local region features can be sampled by bounding boxes of different ratios and sizes from the feature maps of both the small network and the large network using spatial pyramid pooling [16]. Then the sampled features from the feature map of small network are regressed to the same dimension as the large model by a following transformation layer [29]. The loss function that small network intends to minimize is defined as

$$\mathcal{L}(W) = \lambda_1\mathcal{L}_m(W) + \mathcal{L}_{gt}(W), \tag{2}$$

$$\mathcal{L}_m(W) = \frac{1}{2N}\sum_{i}\|u^{(i)} - r(v^{(i)})\|_2^2, \tag{3}$$

$$\mathcal{L}_{gt}(W) = \mathcal{L}_{cls}(W) + \lambda_2\mathcal{L}_{reg}(W), \tag{4}$$

in which $\mathcal{L}_m$ is the L2 loss of feature mimic, $\mathcal{L}_{gt}$ is the classification and regression loss function of region proposal network described in [13], λ_1 and λ_2 are the loss weight balance parameters. N is the total number of the proposals we sampled, $u^{(i)}$ is the feature sampled by spatial pyramid

pooling from the feature map of the large model based on the i^{th} proposal. $v^{(i)}$ is the sampled output feature of the small network and r is the regression function to transform $v^{(i)}$ to the same dimension as $u^{(i)}$. By optimizing this loss function, the small network can be trained under both the ground-truths and the additional supervision from the large models.

A potential problem about the mimic loss is that its value could be large during the training time so that the balancing weight parameter has to be carefully set between the supervision of features and ground-truths to better optimize the network. Besides, the spatial pyramid pooling might cause information loss when performing the pooling process.

Thus we further improve the mimic objective loss function by implementing the L2 loss of the features extracted from the region proposal and perform normalization when calculate the loss. After that, the training process becomes more stable and we can simply set the balance weight to 1. The new mimic loss function is defined as below

$$\mathcal{L}_m(W) = \frac{1}{2N}\sum_i \frac{1}{m_i}\|u^{(i)} - r(v^{(i)})\|_2^2, \tag{5}$$

in which m_i is the dimension of the features extracted by the i^{th} region proposal. Different from the features extracted and pooled to same dimension in Equation (3), the features here are extracted from the feature map directly and are different in dimension for every region proposal.

3.3. Network Architecture and Implementation Details

We integrate the feature map mimic on Faster-RCNN [28] and R-FCN [6] that are state of the art object detection framework. The training process can be separated into two stages. The first stage is to train a Region Proposal Network[28] by feature mimic method. The RPN itself can be viewed as an effective proposal generator as well as a single category object detector. For general object detection tasks we can simply train the RPN first and jointly fine-tune a Faster-RCNN network or a R-FCN network on it at the second stage.

The framework of mimic training is illustrated in Figure 1. A large Faster-RCNN or R-FCN network is trained in usual way to optimize softmax loss and smooth L1 loss by the supervision of the ground-truths on the training data. The feature mimic architecture contains two networks. The large network is initialized by the weights of the well-trained detection network and the layers are fixed during the training process. The small network is randomly initialized. A Region Proposal Network is added at the end of the small network to generate object proposals. Initially, an entire image is forwarded through both of the networks to produce two different feature maps. Given the proposal regions generated by the RPN on small network, the sampler

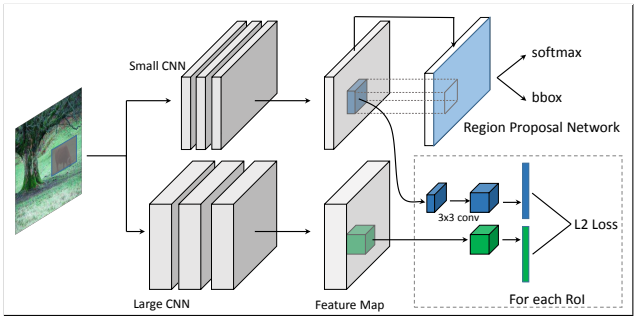

Figure 1: Overall architecture of feature mimic by proposal sampling. A Region Proposal Network generates candidate RoIs, which then used to extract features from the feature maps.

will extract the features at the proposal regions on the feature map of both the large network and the small network. The entire framework is trained end to end by the loss defined in Equation (5). All loss weights (λ_1 and λ_2) are set to 1 during our training.

In the second stage we fine-tune a Faster R-CNN or a R-FCN detection network from the region proposal network on the training dataset and we randomly initialize all the new added layers. The inference process is the same as the original Faster-RCNN or R-FCN without any increased parameters. Our implementation uses Caffe [21].

3.4. Two-stage Mimic

Fine-tuned by the ground-truth supervision only in the second stage might degrade the feature learned by mimicking at the first stage. The prediction of the detector in Faster-RCNN or R-FCN detector can be regarded as a classification task. Therefore we could add logits matching supervision to the second stage of detection pipeline to further transfer the knowledge of the large detection models to the small models. Moreover, by two-stage mimic, not only the proposal related information but also the category classification information learned by the large model can be passed to the small network. Richer information from the large model can further help the small network mimic the large model. We can simply fine-tune the detection network from the mimicked region proposal network and add a L2 regression loss of the predicted classification logits and bounding box regression values. Experiments results show that two-stage mimic further improves the performance of the small network than fine-tuning the mimicked RPN model in stage 2.

3.5. Mimic over Scales

We further extend the feature map mimic to improve detection performance when the input size is reduced. Besides the network complexity, the input image scale is another key factor to the speed of object detection framework.

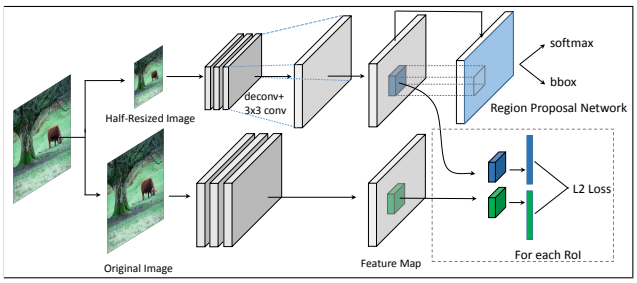

Figure 2: Overall architecture of feature mimic by proposal sampling on different input scales. A Region Proposal Network generates candidate RoIs, which then used to extract features from the up-sampled feature map and the feature map of the network with large input.

Object detectors perform worse on objects of small scales [9]. Experiments on Caltech demonstrate that the detector trained and tested on 480×640 pixels are much worse than the one trained and tested on 1000×1500 pixels. Techniques like multi-scale test [13], hierarchical feature fusion [3] and hole algorithms [4] are presented to improve detection performance for small objects but also bring large increase of time cost during the inference time.

The main reason for the degraded performance of small scales in detection is that the the features of small objects are very small in the final feature map after the downsampling by the convolutional network. The feature mimic is intrinsically designed to train by feature regression to achieve kind of activation similarity between two feature maps. We can simply add a deconvolution layer on top of the final feature map to enlarge the feature map and then mimic the feature map for the large input network to improve the performance.

As illustrated in Figure 2, the image is forwarded through a network with stride 16 and produces a feature map; the same image half-resized is forwarded through the network with stride 8 and produces a similar-sized feature map. Then we perform feature map mimic method on the two feature maps during training. Our experiments show that given the input of small scales, the mimic method can significantly increase the performance of the detector.

4. Experiments

We comprehensively evaluate our method on the Caltech pedestrian detection benchmark [9] and PASCAL VOC 2007 object detection benchmark [11].

4.1. Experiments on Caltech

On Caltech, we train our models on the new annotated $10\times$ training data provided by [37] and select only the images that contains ground-truth bounding boxes in the training dataset which has about 9k images in total. The standard test set consisting of 4024 frames are used for evaluation on the new annotations under reasonable evaluation settings. Following [37], the evaluation metrics we use are log average Miss Rate on False Positive Per Image (FPPI) in $[10^{-2}, 10^{0}]$ (denoted as MR_{-2}) and log-average Miss Rate on FPPI in $[10^{-4}, 10^{0}]$ (denoted as MR_{-4}).

The performance of the small network directly depends on the large model in mimic learning pipeline. Therefore it is important to train a large model with high performance in the experiments. We implement Region Proposal Network and R-FCN detection framework and achieve competitive detection results on Caltech based on GoogLeNet Inception Network Architecture[32].

Training the Region Proposal Network and R-FCN detection network jointly on Caltech dataset with the original image size 640×480 pixels is difficult to achieve a comparable performance as the current state-of-the-art [36]. The method has an MR_{-2} of only 14.64%. Considering that the height of most pedestrians in Caltech dataset is under 80 pixels which is quite challenging to the detector, we rescale the images such that their shorter side is 1000 pixels. We train and test both RPN and R-FCN networks on images of single scale. For RPN anchors, we use 2 aspect ratios of 2:1, 3:1 and 3 scales with box areas of 4^2, 8^2, 16^2. The RPN and R-FCN network are jointly trained on a pre-trained model for ImageNet classification as in standard practice. We fine-tune all layers of the Inception network and all the new layers are randomly initialized as described in [28]. We use a leaning rate of 0.001 for 30k mini-batches and 0.0001 for the next 10k mini-batches. The momentum is set to 0.9 and the weight decay is 0.0005. The final detection result is MR_{-2} = 7.15%, as shown in Table 2. The Region Proposal Network can be regarded as a single category detector, therefore we also reported the performance of the RPN stand-alone in the experiments.

Method	MR_{-2}	MR_{-4}
Inception RPN stand-alone	8.68	21.74
Inception RPN+ R-FCN	7.15	19.18

Table 2: Detection results of Inception RPN and R-FCN.

Considering that we are not to examine which kind of network architectures perform better on object detection task, instead we want to examine the feature mimicking method can improve the performance of small networks. We use simple design for small network structures. The small networks we use in the experiments are modified Inception networks. $1/n$-Inception network represents the network that has the same depth as the Inception network but each convolutional layer of it only contains $1/n$ of filters of the Inception Network. The feature map we mimicked in the experiments is the output of Inception-4d layer of the network. The test time for different R-FCN networks are

listed in Table 3.

First we evaluate the method of naive logits mimic and the entire feature map regression mimic mentioned in Section 3.1 and 3.2. The naive logits mimic on object detection is to optimize L2 loss of the soft targets and the predicted regression values between two Region Proposal Networks. The entire feature map regression is to train the small network as a regression problem of two high dimensional features. Results of logits mimic and entire feature map regression mimic are shown in Table 4. The mimicked models have a large decrease on performance compared with the large model. The small network is difficult to mimic the large model by simple logits matching or global feature regression.

Method	test time (ms)
Inception	53.45
VGG	232.57
1/2-Inception	22.76
1/4-Inception	12.39
1/8-Inception	9.48

Table 3: Test time of R-FCN for different networks. Tested for input scale of 1000×1500 pixels on TITANX.

Method	MR_{-2}	MR_{-4}
Inception RPN stand-alone	8.68	21.74
1/2-Inception logits mimic	68.75	81.32
1/2-Inception feature map regression	64.22	76.99

Table 4: Detection results of Naive Mimic and Entire Feature Regression Mimic.

Next we follow the same architecture of feature map mimic method based on proposal sampling as described in Section 3.3 to train the 1/2-Inception Network on Caltech end-to-end. In the first stage, we use the 1/2-Inception Network to mimic the features extracted from the pre-trained Inception RPN feature map by the region proposals the 1/2-Inception Network generated during training. We use 128 RoIs to sample the features on the feature maps of both networks and the ratio of positive and negative samples of RoIs are 1:1. The comparison of the results for the Inception RPN and the mimicked 1/2-Inception RPN are shown in Table 5. The mimicked model with only 1/4 parameters achieves competitive performance as the large model but 2.5× faster.

In stage 2, we fine-tune the R-FCN network from the Region Proposal Network trained in stage 1. Since the RPN layers have been trained well enough in stage 1, the learning rate for the RPN layers are set as 1/10 of the new added layers. The RoI output size in R-FCN is set to 7 × 7. The stage 2 fine-tune results can be found in Table 6. The mimic

Method	MR_{-2}	MR_{-4}
Inception RPN	8.68	21.74
1/2-Inception mimic RPN	9.16	21.82

Table 5: Detection results of Incepion-RPN trained on training data and 1/2-Inception RPN trained by mimic method.

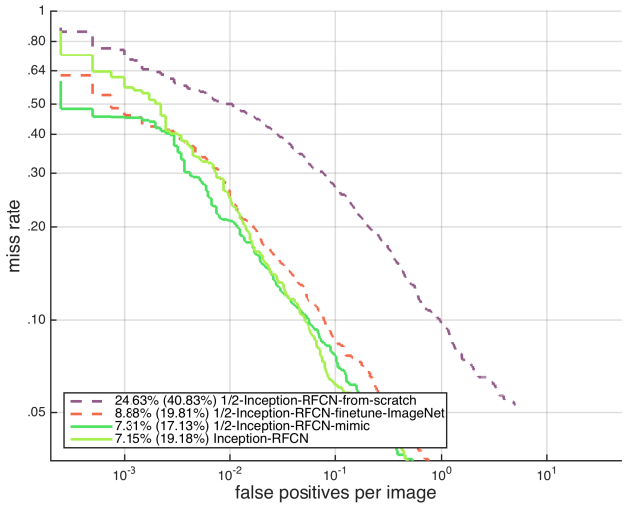

Figure 3: Comparison of detection results evaluated on the new annotations (MR_{-2}(MR_{-4})).

+ finetune network achieve similar performance as the large network for MR_{-2} and even get a better performance on MR_{-4}.

Method	MR_{-2}	MR_{-4}
Inception RPN + R-FCN	7.15	19.18
1/2-Inception-from-scratch	24.63	40.83
1/2-Inception-finetune-ImageNet	8.88	19.81
1/2-Inception mimic + fine-tune R-FCN	7.55	17.59

Table 6: Detection results of 1/2-Inception R-FCN models trained by ground truth supervision and by mimic method.

In order to compare feature mimic method with traditional method that training the model directly on the training dataset by ground-truth supervision, we train a 1/2-Inception R-FCN from scratch. As described in [14], models that fine-tuned from a pre-trained model on ImageNet classification tasks performs better than those trained from scratch. Therefore we also pre-train a 1/2-Inception network on ImageNet dataset and fine-tune the R-FCN detection network from the pre-trained model. The results are compared in Table 6 and Figure 3. The results shows the model trained by feature mimic learning demonstrate superior performance than both models trained from scratch and fine-tuned from ImageNet pre-trained model on MR_{-2} and MR_{-4}.

We further conduct experiments on different smaller net-

works. The experiment results on Table 7 provide evidence that the mimic learning method can generally improve performance on different small networks. There is a large performance gap between mimicked model and the model trained from scratch. The smaller networks like $1/4$-Inception and $1/8$-Inception networks are difficult to train on the ImageNet dataset. Training smaller network by feature mimic is more efficient for implementation. The $1/4$-Inception Network, which has about $1/16$ parameters than the Inception Network, still achieves comparable performance as the large network. We also train another large R-FCN detection based on VGG network and a Faster R-CNN model based on ResNet network to evaluate the generalization of the mimic learning method. The results shown in Table 8 and Table 14 demonstrate that the feature mimic method can help train a small network to mimic a large model with totally different network architectures. The mimicked $1/2$-Inception model achieves similar performance as the large model but are $10\times$ faster during inference. Also for the same small network, the performance of the large model is important for the mimic learning. The same model mimicked from Inception outperforms that mimicked from VGG model as the Inception model outperforms VGG model.

Method	MR_{-2}	MR_{-4}
$1/4$-Inception-from-scratch	30.36	45.73
$1/4$-Inception-mimic + finetune	10.02	21.84
$1/8$-Inception-from-scratch	42.64	58.21
$1/8$-Inception-mimic + finetune	16.86	32.46

Table 7: Detection results of mimic learning on different smaller network structures.

Method	MR_{-2}	MR_{-4}	test time (ms)
VGG RPN + R-FCN	7.68	18.59	232.57
$1/2$-Inception mimic -from-VGG + finetune	8.47	18.57	22.76

Table 8: Detection results and test time of $1/2$-Inception R-FCN mimicked from VGG model.

Instead of fine-tuning the R-FCN from the mimicked model we get at stage 1, we train the R-FCN of small network under the supervision of ground-truths as well as the logits of the large network at the same time during training. Experiments demonsntrate that two-stage mimic further improves the performance of small model in Table 9.

The size of input is essential to the performance of the object detector. The MR_{-2} of Inception R-FCN model on Caltech is only 14.64% given input image with size $480\times$ 640 as shown in Table 10. Simply adding a deconvolution layer at top of the final feature map cannot bring any performance improvement. We implement feature map mimic method on the features of the network with large input and the network with reduced input but added a deconvolutional layer for upsampling on top of the feature map. Experiments result shows a significant MR_{-2} decrease with only a little time cost increase (3.5 ms).

Method	MR_{-2}	MR_{-4}
$1/2$-Inception-mimic + finetune	7.55	17.59
$1/2$-Inception-two-stage-mimic	7.31	17.13
$1/4$-Inception-mimic + finetune	10.02	21.84
$1/4$-Inception-two-stage-mimic	9.75	20.23
$1/8$-Inception-mimic + finetune	16.86	32.46
$1/8$-Inception-two-stage-mimic	15.32	31.46
$1/2$-Inception-mimic-VGG + finetune	8.47	18.57
$1/2$-Inception-two-stage-mimic-VGG	8.33	18.46

Table 9: Detection results of R-FCN models fine-tuned in stage 2 and models trained by two-stage mimic.

Method	MR_{-2}	MR_{-4}	test time (ms)
Inception R-FCN	14.64	27.81	15.63
Inception-upsample R-FCN	15.28	29.06	19.18
Inceptipn mimic + fine-tune	11.14	23.35	

Table 10: Detection results and test time with input of 480×640 pixels. The Inception-upsample is the modified network with stride 8 trained on training dataset. Inception mimic is the same network as Inception-upsample but trained by mimic learning.

4.2. Experiments on PASCAL VOC

To evaluate our feature mimic learning method on different detection frameworks and more complicated common object detection task, in this section we introduce our mimic learning method based on the Faster R-CNN framework for PASCAL VOC [11] common object detection benchmark. The experiment results show that our feature mimic method can be generalized well on different detection frameworks and object detection tasks.

Following [28], for the PASCAL VOC 2007 test set, we use the 5k trainval images in VOC 2007 and 16k trainval images in VOC 2012 for training (07+12). The hyper-parameters for training Faster-RCNN are the same as [28]. We train a Region Proposal Network and Fast R-CNN model jointly on the training dataset as the large model we want to mimic. Limited by the space, we only report mAP of the experiments. Further detailed results can be found in the Appendix. The large model we trained based on Inception network architecture achieves 75.7% mAP on VOC 2007 test set. During the mimic training, firstly we mimic a small RPN from the large model in stage 1 and then finetune the Faster R-CNN from this pre-trained model in stage 2 or use two-stage mimic techniques to train models for common object detection.

The mimic experiments on PASCAL VOC are similar as Caltech except that the PASCAL VOC is a common object detection task. The Region Proposal Network we mimic in the first stage can only predict class-agnositic proposals, we therefore evaluate the performance of the RPN trained using mimic method by the recall given up to 300 proposals per image. Aiming for a more complete evaluation, we also report the recall when the IoU threshold is set to 0.7 for true positives. We expect more strict evaluation can be used for better performance comparison between different proposal models. And mAP is used to evaluate the object detector for the stage 2 training. The mimicked RPN model in the first stage achieves similar recall as that of large model and outperforms the model trained from scratch or fine-tuned from ImageNet pre-trained model (Table 11).

Method	Recall@.5	Recall@.7
Inception RPN	97.26%	85.36%
$1/2$-Inception-from-scratch	91.18%	70.66%
$1/2$-Inception-finetune-ImageNet	96.8%	83%
$1/2$-Inception-mimic	97.13%	85.58%

Table 11: RPN results on PASCAL VOC 2007 given up to 300 proposals per image. Recall@.7 means the IoU threshold to determine true positives is set to 0.7.

In stage 2 we fine-tune a Faster R-CNN based on the RPN model trained in stage 1. The $1/2$-Inception Faster-RCNN achieves better performance than the models fine-tuned from ImageNet pre-trained model and far better than the model trained from scratch. Results are shown in Table 12.

Method	mAP
Inception Faster R-CNN	75.70
$1/2$-Inception-from-scratch RPN	49.21
$1/2$-Inception-finetune-ImageNet RPN	72.37
$1/2$-Inception-mimic + finetune RPN	72.79

Table 12: Detection results PASCAL VOC Faster-RCNN Results.

We experiment further for smaller models. It is difficult to train the small models on ImageNet classification tasks. Therefore we compare our mimicked models with the models trained from scratch on the training dataset. The models trained by mimic method outperform much more than the models trained from scratch, which is shown in Table 13. Also the experiments demonstrate that the two-stage mimic can further improve the performance of the mimic method.

We also conduct experiments of mimic at different locations of extractor are shown in Table 15. The results indicate that mimicking the last shared feature map is the best choice.

Method	mAP
$1/4$-Inception-from-scratch	42.08
$1/4$-Inception-mimic + finetune	65.76
$1/4$-Inception-two-stage-mimic	67.66
$1/8$-Inception-from-scratch	34.77
$1/8$-Inception-mimic + finetune	53.80
$1/8$-Inception-two-stage-mimic	56.14

Table 13: Comparison of detection results of Faster R-CNN models trained by mimic learning and trained by ground-truth supervision only.

Method	0.5	0.6	0.7	0.8	0.9	mAP
$1/4$-vgg16-scratch	89.0	82.5	63.9	27.4	2.9	43.5
$1/4$-vgg16-mimic	93.5	88.7	74.6	34.3	3.3	48.7

Table 14: $1/2$-VGG16 mimic ResNet-50 results. RPN evaluation by recall rate (with regard to different IoUs) and Faster-RCNN detection mAP on VOC07 test set.

Mimic layer	0.5	0.6	0.7	0.8	0.9	mAP
No-mimic	87.4	80.8	64.7	28.4	2.8	42.08
Inception-3b	87.4	81.1	65.6	29.7	3.2	42.46
Inception-4b	92.9	88.4	75.0	36.5	3.9	58.57
Inception-4d	95.1	91.6	80.8	42.2	4.5	65.76

Table 15: $1/4$-Inception RPN and Faster-RCNN mimic Inception model at different layers. Evaluation by recall rate with regard to different IoUs (300 proposals each image) and detection mAP on VOC07 test set.

5. Conclusion

In this paper we propose a feature mimic method to further extend the mimic approach to object detection tasks. By supervision of the features from the large network, we can train networks from scratch to achieve superior performance than fine-tuning from ImageNet pre-trained models. Moreover, our approach makes it possible to train faster and compact detection models as accurate as the large models. In our experiments on Caltech and PASCAL VOC, the $2.5\times$ faster models with $1/4$ parameters trained by mimicking achieves similar performance as the large Inception models. The feature map based mimicking can possibly be extended to other fully convolution network based tasks, such as semantic segmentation, which will be taken as a future work.

References

[1] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 37–45, 2015.

[2] J. Ba and R. Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

[3] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. *arXiv preprint arXiv:1512.04143*, 2015.

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[5] M. Courbariaux, Y. Bengio, and J.-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, pages 3123–3131, 2015.

[6] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. *arXiv preprint arXiv:1605.06409*, 2016.

[7] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, pages 1269–1277, 2014.

[8] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015.

[9] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *PAMI*, 34, 2012.

[10] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha. Backpropagation for energy-efficient neuromorphic computing. In *Advances in Neural Information Processing Systems*, pages 1117–1125, 2015.

[11] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[12] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[13] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

[15] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR, abs/1510.00149*, 2, 2015.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pages 346–361. Springer, 2014.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[18] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[19] M. Huh, P. Agrawal, and A. A. Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

[20] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

[21] 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*, pages 675–678. ACM, 2014.

[22] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell. Data-dependent initializations of convolutional neural networks. *arXiv preprint arXiv:1511.06856*, 2015.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[24] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014.

[25] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed. Ssd: Single shot multibox detector. *arXiv preprint arXiv:1512.02325*, 2015.

[26] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. *arXiv preprint arXiv:1603.05279*, 2016.

[27] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2015.

[28] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[29] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

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

[31] D. Soudry, I. Hubara, and R. Meir. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In *Advances in Neural Information Processing Systems*, pages 963–971, 2014.

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

[33] G. Urban, K. J. Geras, S. E. Kahou, O. Aslan, S. Wang, R. Caruana, A. Mohamed, M. Philipose, and M. Richardson. Do deep convolutional nets really need to be deep (or even convolutional)? *arXiv preprint arXiv:1603.05691*, 2016.

[34] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng. Quantized convolutional neural networks for mobile devices. *arXiv preprint arXiv:1512.06473*, 2015.

[35] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[36] L. Zhang, L. Lin, X. Liang, and K. He. Is faster r-cnn doing well for pedestrian detection? In *European Conference on Computer Vision*, pages 443–457. Springer, 2016.

[37] S. Zhang, R. Benenson, M. Omran, J. Hosang, and B. Schiele. How far are we from solving pedestrian detection? *arXiv preprint arXiv:1602.01237*, 2016.

[38] X. Zhang, J. Zou, K. He, and J. Sun. Accelerating very deep convolutional networks for classification and detection. *arXiv preprint arXiv:1505.06798*, 2015.