IEEE Access
The journal for rapid open access publishing
xxxx 00, 0000.
Digital Object Identifier 10.1109/ACCESS.2017.DOI
IEEE Xplore Digital Library
http://ieeexplore.ieee.org/Xplore

Fast Deep Neural Networks with Knowledge Guided Training and Predicted Regions of Interests for Real-time Video Object Detection

WENMING CAO [1,2],(Member, IEEE), JIANHE YUAN [1], ZHIHAI HE [2], (Fellow, IEEE), ZHI ZHANG [2], and ZHIQUAN HE [1] (Member, IEEE)

[1]Shenzhen Key Laboratory of Media Security, Shenzhen University, China (e-mails: W. Cao: wmcao@szu.edu.cn, J. Yuan: jyuan1118@gmail.com, Z. He: zhiquan@szu.edu.cn)

[2]Video Processing and Communication Lab, Department of Electrical and Computer Engineering, University of Missouri, Columbia, MO, USA. (e-mails: Zhihai He: hezhi@missouri.edu, Zhi Zhang: zzbhf@mail.missouri.edu)

Corresponding author: Zhiquan He (e-mail: zhiquan@szu.edu.cn).

This work was supported in part by Natioanl Natural Science Foundation of China, No.61771322,61375015, and in part by NSF under grants US Ignite 1647213 and CyberSEES 1539389.

ABSTRACT It has been recognized that deeper and wider neural networks are continuously advancing the state-of-the-art performance of various computer vision and machine learning tasks. However, they often require large sets of labeled data for effective training and suffer from extremely high computational complexity, preventing them from being deployed in real-time systems, for example vehicle object detection from vehicle cameras for assisted driving. In this paper, we aim to develop a fast deep neural network for real-time video object detection by exploring the ideas of knowledge-guided training and predicted regions of interest. Specifically, we will develop a new framework for training deep neural networks on datasets with limited labeled samples using cross-network knowledge projection which is able to improve the network performance while reducing the overall computational complexity significantly. A large pre-trained teacher network is used to observe samples from the training data. A projection matrix is learned to project this teacher-level knowledge and its visual representations from an intermediate layer of the teacher network to an intermediate layer of a thinner and faster student network to guide and regulate the training process. To further speed up the network, we propose to train a low-complexity object detection using traditional machine learning methods, such as Support Vector Machine (SVM). Using this low-complexity object detector, we identify regions of interest that contain the target objects with high confidence. We obtain a mathematical formula to estimate the regions of interest to save the computation for each convolution layer. Our experimental results on vehicle detection from video demonstrated that the proposed method is able to speed up the network by up to 16 times while maintaining the object detection performance.

INDEX TERMS Assisted driving, deep neural networks, knowledge projection, speed optimization, vehicle detection.

I. INTRODUCTION

Deep learning and deep convolution neural network (DCNN) has demonstrated its extraordinary performance on various computer vision and machine learning tasks [1]–[4]. Recent results on large datasets such as ImageNet [5] and MS COCO [6] suggest that *wider* and *deeper* convolutional neural networks tend to achieve better performance. These types of networks often require large sets of labeled data for training and involve high computational complexity. This poses significant challenges for us to develop and deploy deep neural networks in real-time systems, for example the advanced driver-assistance systems (ADAS). As we know, real-time systems and end devices, such as mobile phones, have limited computational resources and memory bandwidth. Recently, great efforts have been made to address the network speed issue. A variety of model compression approaches [7]–[9]

have been proposed to obtain faster networks that mimic the behavior of large networks. Another important issue in practical applications is that we often have access to very limited labeled samples. It is very expensive to obtain human labeled ground-truth samples for training. In some applications domains, it is simply not feasible to accumulate enough training examples for deep networks [10]–[12].

In this paper, we aim to develop a fast deep neural network for real-time video object detection by exploring the ideas of knowledge-guided training and predicted regions of interest. Specifically, we will develop a new framework for training deep neural networks on datasets with limited labeled samples using cross-network knowledge projection which is able to improve the network performance while reducing the overall computational complexity significantly. A large pre-trained teacher network is used to observe samples from the training data. A projection matrix is learned to project this teacher-level knowledge and its visual representations from an intermediate layer of the teacher network to an intermediate layer of a thinner and faster student network to guide and regulate the training process. We carefully design the teacher-student architecture and joint loss function so that the smaller student network can benefit from extra guidance while learning towards specific task targets. Therefore, same level performance can be achieved using a smaller network.

Besides the smaller and faster DCNN model, at the detection side, we predict the salient local regions for objects and design a fast framework of deep neural network which is able to perform analysis on regions of interest to speed up the detection process. Specifically, we propose to train a low-complexity object detection using traditional machine learning methods, such as Support Vector Machines (SVM) [13]. Using this low-complexity object detector, we identify regions of interest that contain the target objects with high confidence. We obtain a mathematical formula to estimate the regions of interest to save the computation for each convolution layer. Our experimental results on vehicle detection from videos demonstrated that the proposed method is able to speed up the network by up to 16 times while maintaining the object detection performance.

The major contributions of this paper are summarized as follows: (1) We propose a new architecture to transfer the knowledge from a pre-trained large teacher network into a thinner and faster student network to guide the training on a smaller dataset. Our approach addresses the issues of network adaptation and model compression at the same time. (2) We have developed a fast method to determine the candidate regions of interest which contain the target objects. (3) We have established an analytic model to estimate the support regions at each convolution layer and integrate this into the existing object detection framework using deep convolutional neural networks. (4) We have conducted extensive experiments to demonstrate that our method is able to significantly reduce the network computational complexity by 16 times while largely maintaining the network performance by a significant margin.

The rest of this paper is organized as follows. Related work is reviewed in Section II. The overview of the proposed method is provided in Section III. The knowledge-guided training and projection matrix learning are presented in Section IV. The proposed SSD-ROI method is explained in Section V. Experimental results are presented in Section VI. Finally, Section VII concludes this paper.

II. RELATED WORK

In this work, we focus on road object detection for advanced driver-assistance systems (ADAS). Robust and reliable vehicles and road objects detection in real-time is a critical component in ADAS [14]. Active sensors, such as LIDAR, millimeter wave radars, or lasers, have several drawbacks, including low spatial resolution, slow scanning speed, and high cost [15], [16]. Vision-based road object detection using cameras offer a more affordable solution and can be used to detect and track vehicles more accurately and effectively. In traditional vision based vehicle detection methods, image features and machine learning methods such as SVM [17] are widely used. For example, Histogram of oriented gradient (HOG) [18] features have been used in a number of studies [19], [20]. Haar-like features are extensively used in vehicle detection in a number of studies [16], [21]–[23] as Haar-like features are found to be efficient for detecting horizontal, vertical,and symmetric structures. SIFT features [24] and hidden Conditional Random Field classification are combined in [25].

Recently, methods have been developed detect vehicles from videos or static images using deep convolution neural networks [26]–[30]. For example, faster R-CNN [2] proposes candidate regions and uses CNN to verify candidates as valid objects. YOLO [4] uses end-to-end unified fully convolutional network (FCN) frameworks which predict the objectness confidence and the bounding boxes simultaneously over the whole image. SSD [3] outperforms YOLO by discretizing the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. YOLO-2 [31] achieves start-of-art performance in object detection by improving various aspects of its previous version. The work in [26] uses a fully convolutional network for object detetion from three dimensional (3D) range scan data with LIDAR. Wang *et al*. proposes a 2D-DBN architecture [32] which uses second-order planes instead of first-order vectors as inputs and uses bilinear projection for retaining discriminative information to improve the detection rate.

Although DCNN based methods achieve the state-of-art accuracy of detection or classification, they often require intensive computation and large amount of labeled training data. During the past few years, in order to deploy deep neural network economically in real-time applications, a significant amount of efforts have been put to address these two problems [33], [34]. Our proposed method is closely related to domain adaptation and model compression which are reviewed in this section.

Manually labeling the ground-truth training samples is labor intensive and time consuming. In some application domains, it is simply not feasible to do so. In these cases, domain adaptation [10]–[12], [35] can be a powerful tool to enable training a large network without over-fitting. Methods for network domain adaptation [11], [36], [37] have been developed to enable training on new domains with inadequate labeled samples or even unlabeled data. Learning shallow representation models is a promising approach to reduce domain discrepancy. However, without deeply embedding the adaptation in the feature space, the transferability of shallow features will be limited by the task-specific variability. The work [38] embeds domain adaptations in deep learning architecture and outperforms traditional methods by a large margin. There are also some shallow architectures [39], [40] in the context of learning domain-invariant features. Limited by representation capacity of shallow architectures, the performance of shallow networks are often inferior to that of deep networks [37]. Within the context of deep feed-forward neural networks, *fine-tune* is an effective and overwhelmingly popular method [41], [42]. Feature transferability of deep neural networks has been comprehensively studied in [43].

To address the issue of high computational complexity of deep neural networks, researchers have designed smaller and thinner networks from larger pre-trained networks. A typical approach is to prune unnecessary parameters in trained networks while retaining similar outputs. Instead of removing close-to-zero weights in the network, LeCunn *et al*. proposed Optimal Brain Damage (OBD) [44] which uses the second order derivatives to find trade-off between performance and model complexity. Following work of Optimal Brain Surgeon (OBS) [7] by Hassibi *et al*. outperformed the original OBD method, but was more computationally intensive. Han *et al*. [45] developed a method to prune state-of-art CNN models without loss of accuracy. Based on this work, deep compression [46] used ensembles of parameter pruning, trained quantization and Huffman coding, and achieved 3 to 4 times layer-wise speed up and reduced the size of VGG-16 [47] by 49 times. This line of work focuses on pruning unnecessary connections and weights in trained models and optimizing for better computation and storage efficiency.

Various factorization methods have also been proposed to speed up the slow matrix operations commonly used to optimize network performance. Jenderberg *et al*. [8] and Denton *et al*. [48] use SVD-based low rank approximation. Zhang *et al*. [49] successfully compressed VGG-16 [47] to achieve 4 times speed up with 0.3% loss of accuracy based on Generalized Singular Value Decomposition. Gong *et al*. [50] used a clustering-based product quantization to reduce the size of matrices by building an indexing. In contrast to off-line optimization, Ciresan *et al*. [51] trained a sparse network with random connections, providing good performance with better computational efficiency than densely connected networks.

Another line of work trains a smaller network from scratch to mimic the behavior of a much larger network. Starting from the work of Bucila *et al*. [52] and Knowledge Distillation (KD) by Hinton *et al*. [9], the design of smaller yet efficient networks has gained a lot of research interest. It has been demonstrated in [9] that small networks can be trained to generalize in the same way as large networks with proper guidance. FitNets [53] achieved better compression rate than knowledge distillation by designing a deeper but much thinner network using trained models. Training deep networks has proven to be challenging [54] Recently, adding supervision to intermediate layers of deep networks is explored to assist the training process [55], [56]. These methods assume that source and target domains are consistent. It is still unclear whether the guided training is effective when the source and target domains are significantly different.

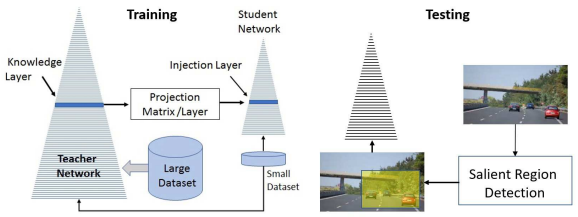


FIGURE 1: Overview of the proposed system.

In this paper, we consider a unique setting of the problem. We use a large network pre-trained on a large dataset (*e.g.* , the ImageNet) to guide the training of a thinner and faster network on a new smaller dataset with limited labeled samples, involving adaptation over different data domains and model compression at the same time. We also incorporate pre-analysis using fast machine learning methods into the existing deep convolutional neural network for fast object detection.

III. OVERVIEW OF THE PROPOSED METHOD

Fig. 1 provides an overview of the proposed method. On the training side, we develop a knowledge-guided framework where a large teacher network pre-trained on a larget dataset is used to guided the training of a small yet fast student network. Specifically, as the teacher network is analyzing the training image, the feature vector from one layer of the teacher network is projected into the student network to regulate its training process. The projection is performed by a project matrix which is learning during training. Once properly training, during the test stage, we first develop a fast method to detect the regions of interest which will contain the target object with high probability. We then integrate the predicted regions of interest into the deep convolutional neural networks to speed up the detection process.

IV. CROSS-NETWORK FEATURE PROJECTION FOR KNOWLEDGE-GUIDED TRAINING

In this section, we present the proposed method of cross-network feature projection for knowledge-guided training (KGT). An example pipeline of knowledge-guided training

is illustrated in Fig. 2. Starting from a large teacher network pre-trained on a large dataset, a student network is designed to predict desired outputs for the target problem with a certain level of guidance from the teacher network. The student network aims to resemble the behavior of its teacher. Being trained on a large dataset, the powerful teacher network is able to learn and extract very effective visual representation of the input images. In our knowledge-guided training design, the teacher network and the student network are examining the input image simultaneously. In the early stage, visual features generated by the teacher network will be more useful and effective than those from the student network. We use the features from one specific middle layers of the teacher network to guide the training process of the student network. To this end, we propose to map the feature $\mathcal{F}_\mathcal{T}$ of size N learned at one specific layer of the teacher network into a feature vector $\mathcal{F}_\mathcal{S}$ of size M and inject it into the student network to guide its training process. For the mapping, we choose linear projection

$$\mathcal{F}_\mathcal{S} = \mathcal{P} \cdot \mathcal{F}_\mathcal{T}, \tag{1}$$

where $\mathcal{P}$ is an $N \times M$ matrix. In deep convolutional neural networks, this linear projection matrix $\mathcal{P}$ can be learned by constructing a convolution layer between the teacher and student network. Specifically, we use a convolutional layer to bridge teacher's *knowledge* layer and student's *injection* layer. A *knowledge* layer is defined as the output of a teacher's hidden convolutional layer responsible for guiding the student's learning process by regularizing the output of student's *injection* convolutional layer.

To achieve reduced computational complexity, the student network is designed to be thinner (in terms of feature maps) but deeper to effectively reduce network capacity while preserves enough representation power [53], [57]. In our knowledge-guided training, the student network is trained by optimizing the following joint loss function:

$$W_s^* \leftarrow \arg\min_{W_s}(\lambda \cdot \mathcal{L}_{KP}(W_s, W_k) + \mathcal{L}_p(W_s) + \mathcal{R}), \tag{2}$$

where $\mathcal{L}_{KP}$ and $\mathcal{L}_p$ are loss from the knowledge projection layer and problem specific loss, respectively. For example, for the problem-specific loss, we can choose the cross-entropy loss in many object recognition tasks. λ is the weight parameter decaying during training, W_k is the trained teacher network, $\mathcal{R}$ is a L2 regularization term, and W_s^* is the trained parameters in the student network. Unlike traditional supervised training, the knowledge projection loss $\mathcal{L}_{KP}$ plays an important role in guiding the training direction of KPN, which will be discussed in more detail in the following section.

Let O_h^t, O_w^t and O_c^t be the spatial height, spatial width, and number of channels of the knowledge layer output in the teacher network, respectively. Let O_h^s, O_w^s and O_c^s be the corresponding sizes of student's injection layer output, respectively. Note that there are a number of additional layers in the student network to further analyze the feature information acquired in the inject layer and contribute to the final network output. We introduce the following loss function:

$$\mathcal{L}_{KP(W_s,W_k)} = h[\mu(x;W_k)] \cdot |r[\mu(x;W_k);W_{KP}] - v[x;W_s]|\,, \tag{3}$$

$$h(x) = \begin{cases} 1, & \text{if x} \geq 0, \\ \eta, & \text{otherwise} \end{cases} \tag{4}$$

where μ and v represent the deep nested functions (stacks of convolutional operations) up to the knowledge and injection layer with network parameters W_k and W_s, respectively. $r[\cdot]$ is the knowledge projection function applied on $\mu[\cdot]$ with parameter W_{KP} which is another convolution layer in this work. μ, v and r must be comparable in terms of spatial dimensionality.

The knowledge projection layer is designed as a convolutional operation with a 1×1 kernel in the spatial domain. As a result, W_{KP} is a $O_c^t \times O_c^s \times 1 \times 1$ tensor. As a comparison, a fully connected adaptation layer will require $O_h^t \times O_w^t \times O_c^t \times O_h^s \times O_w^s \times O_c^s$ parameters which is not feasible in practice especially when the spatial size of output is relatively large in the early layers. The output of the knowledge projection layer will guide the training of student network by generating a strong and explicit gradient applied to backward path to the *injection* layer in the following form

$$\Delta W_{s,i} = -\lambda \cdot \frac{\partial \mathcal{L}_{KP}}{\partial W_{s,i}}, \tag{5}$$

where $W_{s,i}$ is the weight matrix of injection layer in student network. Note that in (3), $h[\mu(x;W_k)]$ is applied to $\mathcal{L}_{KP}$ with respect to the hidden output of knowledge projection layer as a relaxation term. For negative responses from $\mu(x;W_k)$, $\mathcal{L}_{KP}$ is effectively reduced by the slope factor η, which is set to 0.25 by cross-validation. Overall, $\mathcal{L}_{KP}$ acts as a relaxed $L1$ loss. Compared to $L2$ loss, $\mathcal{L}_{KP}$ is more robust to outliers, but still has access to finer level representations in $r[\mu(x;W_k);W_{KP}]$.

In the student network, layers after the *injection* layer are responsible for adapting the projected feature to the final network output. This adaptation must be memorized through out the training process. Those network layers before the injection layer aim to learn distinctive low level features.

Therefore, in our KPN framework, the student network and knowledge projection layer are randomized and trained in two stages: **initialization stage** and end to end **joint training stage**. In the initialization stage, path ② in Fig. 2 is disconnected, *i.e.* the knowledge projection layer together with the lower (after injection layer) part of student network is trained to adapt the intermediate output of teacher's knowledge layer to the final target by minimizing $\mathcal{L}_p$. The upper (before injection layer) part of student network is trained sorely by minimizing $\mathcal{L}_{KP}$. In this stage, we use the projection matrix as an implicit connection between upper and lower parts in the student network. The upper student network layers are always optimized towards features interpreted by

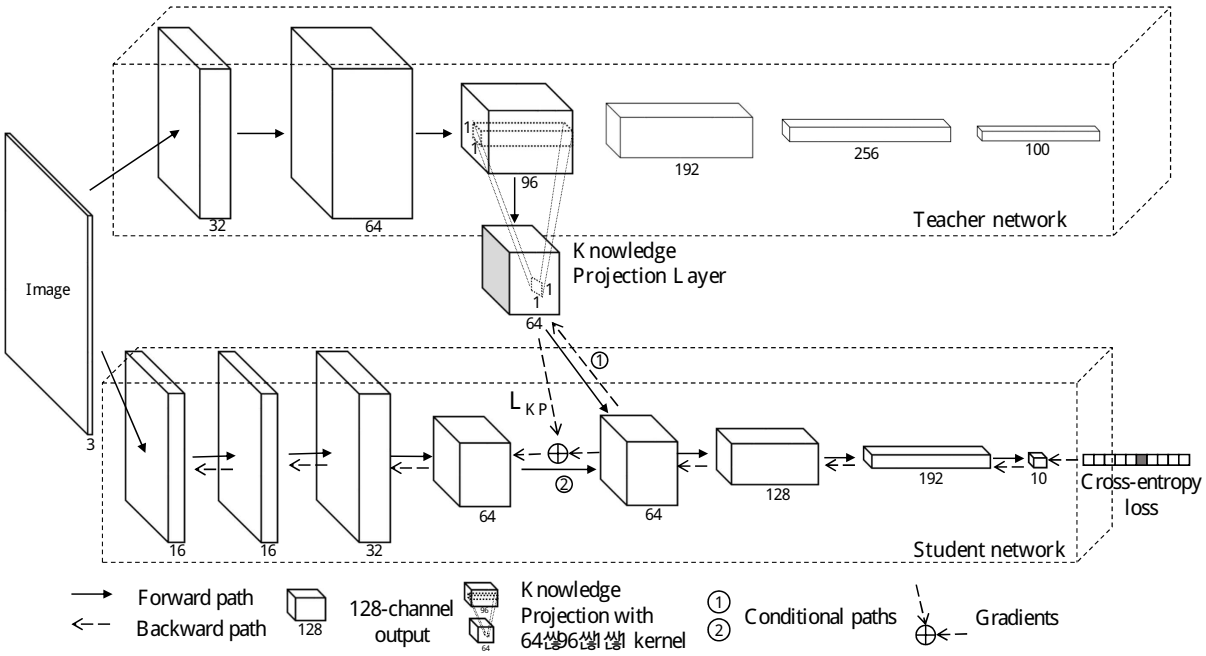


FIGURE 2: KPN architecture. Solid arrows showing the forward data-flow, dotted arrows showing the paths for gradients.

the projection matrix, and have no direct access to targets. This strategy prevents student network to over-fit quickly during the early training stage which is very hard to correct afterwards.

After the initialization stage, we then disconnect path ① and reconnect path ②, the training now involves jointly minimizing the objective function described in (2). Using the results from stage 1 as the initialization, the joint optimization process aims to establish smooth transitions inside the student network from the input to the final output. The loss $\mathcal{L}_{KP}$ injected into student network continues to regularize the training process. In this way, the student network is trained based on a multi-loss function which has been used in the literature to regulate deep networks [58].

V. DEEP CONVOLUTIONAL NEURAL NETWORK WITH REGIONS OF INTEREST

In this section, we introduce the DCNN-ROI method which predicts and incorporates regions of interest into the deep convolutional neural network analysis framework for fast object detection.

A. SALIENT REGION FOR OBJECT DETECTION

To detect an object in an image, SSD proposes a number of bounding boxes and estimates the likelihood for the box containing the target object. To estimate the salient region, let's simplify the model architecture and suppose the model has k layers with filters $(F_i, S_i, P_i), 1 \leq i \leq k$ where F_i, S_i, P_i are the kernel size, stride and padding of each filter, and each filter is followed by a pooling layer with size D_i. For a feature map with size W_i, filter i produces a new feature map with size of $\frac{W_i - F_i + 2P_i}{S_i} + 1$. The subsequent pooling operation will further reduce its size to $1/D_i$. So,

$$W_{i+1} = \frac{W_i - F_i + 2P_i + S_i}{S_i D_i}. \tag{6}$$

We have

$$W_i = W_{i+1} * S_i * D_i - S_i - 2P_i + F_i \tag{7}$$

Based on this backward recursive formula, we can calculate the minimum region containing all the information for the object in input image.

B. FAST VEHICLE DETECTION WITH ROI PREDICTION

Besides the smaller and faster DCNN model, at the detection side, we design a fast framework of deep neural network which is able to perform analysis on regions of interest and predict the salient local regions for vehicles to speed up the detection process. In driving practice, we are interested in the areas inside or close to the driving lane. As shown in Fig. 3, we detect the lanes in the frame, propose a number of potential candidate regions in the lane area. For each candidate region, we extract histogram of oriented gradients (HOG) features [18] and train a linear SVM model to assign a probability score to each candidate region. The score is the likelihood of the region containing a vehicle. We select those candidate regions with score higher than a threshold and combine them into a larger one. The combined region is the estimated salient region we are interested in. When testing a frame using SSD, the pixels outside the ROI region is not involved in computation in each convolutional layer, which speeds up the detection process.

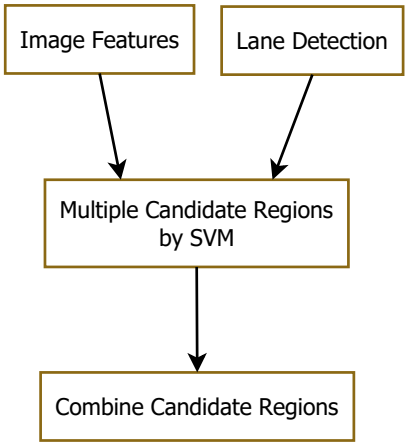


FIGURE 3: Framework of salient region of interest (ROI) prediction.

VI. EXPERIMENTAL RESULTS

In this section, we perform comprehensive evaluations of our proposed method using benchmark datasets. A large dataset D_t is used to train the teacher network and a smaller dataset D_s to train the student network. The large dataset is often available from existing research efforts, for example, the ImageNet. Both the large and the small datasets share the following properties: 1) Image dimensions are same, so that pre-trained models are compatible with each other in terms of shape. 2) Training examples are similar while class labels are different, which ensures transferred patterns are implicitly learned. 3) Training examples are exclusive, to make sure results are comparable. We use the existing teacher network model already trained by other researchers on the public dataset D_t. We compare various algorithms on the benchmark dataset D_s where state-of-the-art results have been reported. Performance reports on small datasets are rare, thus we choose existing large well-known benchmark datasets in following experiments, and aggressively reduce the size of training set to simulate the shortage of labeled data in real world scenarios.

A. NETWORK TRAINING

We build our KPN using the MXNet [59], a deep learning framework designed for both efficiency and flexibility. The dynamically generated computational graph in MXNet allows us to modify network structures during run time. The KPNs are trained on NVidia Titan X 12GB with CUDNN v5.1 enabled. Batch-sizes vary from 16 to 128 depending on the KPN group size. For all experiments, we train using the Stochastic Gradient Descend (SGD) with momentum 0.9 and weight decay 0.0001 except the knowledge projection layers. The weight decay for all knowledge projection layers is 0.001 in the initialization stage and 0 for the joint training stage. 40% of iterations are used for the initialization stage, and the rest goes to be joint training stage. The weight controller parameter λ for joint loss is set to be 0.6, and gradually decays to 0. The pruning frequency is 10000 and we also randomly revoke the initialization stage during joint training stage, to repetitively adjusting network guidance strength.

For fine-tuning, we test with a wide variety of experimental settings. Starting from pre-trained networks, we adjust the last layer to fit to the new dataset, and randomly initialize the last layer. The reshaped network is trained with standard back-propagation with respect to labels on the new dataset, and *unfreeze* one more layer from the bottom one at a time. The best result from all configurations was recorded. To make sure all networks are trained using the optimal hyper-parameter set, we extensively try a wide range of learning rates, and repeat experiments on the best parameter set for at least 5 times. The average performance of the best 3 runs out of 5 will be reported. Data augmentation is limited to random horizontal flip if not otherwise specified.

B. RESULTS ON THE CIFAR-10 DATASET

We evaluate the performance of our method on the CIFAR-10 dataset guided by a teacher network pre-trained on a much larger CIFAR-100 dataset. The CIFAR-10 and CIFAR-100 datasets [60] have 60000 32×32 color images with 10 and 100 classes, respectively. They are both split into 50K-10K sets for training and testing. To validate our approach, we train a 38-layer resnet on the CIFAR-100 as reported in [61], and use it to guide a 50-layer but significantly slimmer resnet on the CIFAR-10. Table 1 summarizes the results, with comparisons against the state-of-the-art results. We do not apply specific optimization techniques used in the state-of-the-art methods due to some structures not reproducible in certain conditions. To compare, we train a standard 38-layer Residue Network, a 50-layer slimmer version of ResNet (each convolutional layer is half the capacity of the vanilla ResNet) and a fine-tuned model of 38-layer ResNet (from CIFAR-100) on CIFAR-10 with different amount of training samples. With all 50000 training data, our proposed method outperforms direct training and best fine-tuning results and still match the state-of-the-art performance. We believe the performance gain specified in [62], [63] can be also applied to our method, i.e. , ensemble of multiple techniques could achieve better performance. The proposed KPN method has improved the accuracy by up to 1.2% while significantly reducing the network size by about 11 times, from 3.1M network parameters to 273K parameters. It also demonstrates strong robustness against aggressive reduction of labeled training samples.

C. DETECTION WITH REGION OF INTEREST

We test the fast detection frame with region of interest on 6 videos and compare the detection performance in terms of accuracy and time efficiency (seconds per frame) between our method and original SSD. Vehicle detection with region of interest needs less time than the original because the image area outside of the ROI is not involved in the convolution. To verify this, we select 200 frames, and for each frame, we

TABLE 1: CIFAR-10 accuracy and network capacity comparisons with state-of-the-art methods. Results using randomly sampled subsets from training data are also reported. Number of network parameters are calculated based on reports in related work.

Methods	Accuracy at # training samples				# of params	MultAdds
	50000	5000	1000	500		
Our methods						
ResNet-50 slim	87.53	71.92	55.86	48.17	**0.27M**	**31M**
ResNet-38	90.86	75.28	61.74	51.62	3.1M	113M
ResNet-38 fine-tune	91.15	89.61	86.26	83.45	3.1M	113M
Our method	**92.37**	**90.35**	**88.73**	**87.61**	**0.27M**	**31M**

Methods	Acc@50000	# of params	MultAdds
State-of-the-art methods			
Maxout [64]	90.62	9.0M	379M
FitNets-11 [53]	91.06	**0.86M**	**53M**
FitNets [53]	91.61	2.5M	107M
GP CNN [62]	93.95	3.5M	362M
ALL-CNN-C [65]	92.7	1.0M	257M
Good Init [63]	**94.16**	2.5M	166M

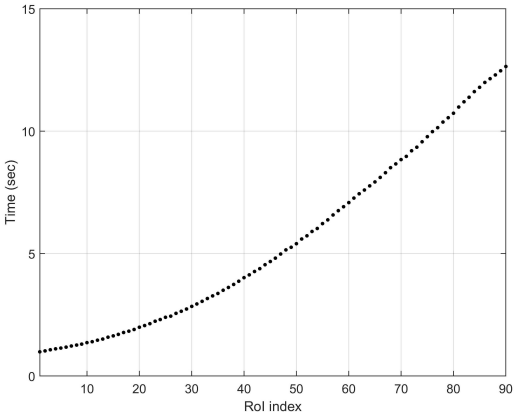


FIGURE 4: Average prediction time of different ROI size.

generate a set of ROIs with increasing sizes, starting from the size of the ground truth bounding box. Fig. 4 shows the average prediction time increasing with ROIs of increasing sizes. When the ROI region is the entire frame, the prediction time is same as the original one.

Table 2 shows the vehicle detection performance between our SSD-ROI method and the original SSD. From the average measures in the last row, we can see that SSD-ROI achieves about 1.6 times faster than the original SSD (column 3 divided by column 5), while keeping the accuracy almost the same. On the other hand, smaller models can speedup the computation significantly. In Table 1, the speedup is more than 10. So we have the last column in Table 2, which is the speedup from ROI detection times the speedup from smaller models. Fig. 5 shows several examples with the detection accuracy gain and speedup, where accuracy gain is the IOU score from SSD-ROI minus that from original SSD, and the speedup is the time cost of original divided by the time used by our method. From these figures, we can see that the detection with ROI prediction is about 1.6 times faster than the original. On the other hand, the detection accuracy is only slightly affected by ROI prediction, as the ROI is applied as a mask to SSD algorithm.

D. DISCUSSION AND FUTURE WORK

Our KPN is designed in a highly modular manner. The training of projection layers is removed during actutal network testing, and the network capacity is highly configurable for performance/speed trade-off. This KPN method can be easily extended to other problems such as object detection, object segmentation, and pose estimation by replacing softmax loss layer used in the classification problems. Since the deployed network is a pure standard network, another research direction is to apply KPN as a building block in traditional model compression techniques to reshape the network in a new perspective. Although we have focused on the advantage of KPN with thinner networks on smaller datasets, there are potential benefits to apply KPN on large network and relatively large datasets, for example, performance oriented situations where speed is not an issue.

Detection of region of interest with a fast and low complexity SVM model speeds up the vehicle detection about 1.6 times, while keeping the accuracy almost the same. With the faster deep neural network, the overall performance can speed up 16 times. Because SSD algorithm produces predictions from feature maps of different scales, even though the estimated ROI is good enough, the predicted bounding box is not the same as the original one, but on the average, the accuracy is almost the same, which can be seen from Table 2,

The estimation of salient region is crucial to the final detection performance. From Fig. 5 and Eqn. 7, we can see that if the margin on four sides between the estimated region and the target object is not big enough, the detection accuracy will be affected. For example, in Fig. 5 (d), the vehicle is not well centered at the estimated region, which causes the accuracy loss. One way to improve the salient region estimation is to generate more candidate bounding boxes and train a better SVM model. In future, we are also going to extend our work to other deep learning frameworks.

VII. CONCLUSION

We have developed a novel knowledge projection framework for deep neural networks the address the issues of domain adaptation and model compression in training simultaneously. We exploit the distinctive general features produced by the teacher network trained on large dataset, and use a learned matrix to project them into domain relevant representations to be used by the student network. A smaller and faster student network is trained to minimize joint loss designed for domain adaptation and knowledge distillation simultaneously. Besides the smaller and faster DCNN model, at the detection side, we have developed a fast method to determine the candidate regions of interest which contain the target objects and established an analytic model to compute the support regions at each convolution layer and integrated

TABLE 2: Vehicle detection performance comparison without GPU. ROI speedup is the speedup from ROI detection only, which is column 3 divided by column 5. The overall speedup is the ROI speedup times the speedup from smaller models shown in Table 1, which is about 10.

	Original SSD		SSD-ROI			
Videos	**Accuracy**	**Time (sec)**	**Accuracy**	**Time (sec)**	**ROI Speedup**	**Overall Speedup**
video1	0.761	31.328	0.748	18.984	1.65	16.5
video2	0.651	31.47	0.65	20.286	1.55	15.5
video3	0.613	31.479	0.627	19.687	1.60	16
video4	0.554	31.31	0.55	19.528	1.60	16
video5	0.631	31.288	0.638	19.673	1.60	16
video6	0.631	30.817	0.627	22.466	1.37	13.7
Mean	**0.669**	**31.318**	**0.667**	**19.847**	**1.58**	**15.8**

this into the existing object detection framework using deep convolution neural networks. Our experimental results on vehicle detection from videos demonstrated that the proposed method is able to speed up the network by up to 16 times while maintaining the object detection performance.

REFERENCES

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

[2] 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, 2015, pp. 91–99.

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in European Conference on Computer Vision. Springer, 2016, pp. 21–37.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.

[5] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," International Journal of Computer Vision (IJCV), vol. 115, no. 3, pp. 211–252, 2015.

[6] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," CoRR, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[7] B. Hassibi, D. G. Stork et al., "Second order derivatives for network pruning: Optimal brain surgeon," Advances in neural information processing systems, pp. 164–164, 1993.

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

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

[10] K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang, "Domain adaptation under target and conditional shift." in ICML (3), 2013, pp. 819–827.

[11] X. Wang and J. Schneider, "Flexible transfer learning under support and model shift," in Advances in Neural Information Processing Systems, 2014, pp. 1898–1906.

[12] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 4068–4076.

[13] C. Cortes and V. Vapnik, "Support vector machine," Machine learning, vol. 20, no. 3, pp. 273–297, 1995.

[14] Z. Sun, R. Miller, G. Bebis, and D. DiMeo, "A real-time precrash vehicle detection system," in Applications of Computer Vision, 2002.(WACV 2002). Proceedings. Sixth IEEE Workshop on. IEEE, 2002, pp. 171–176.

[15] W. D. Jones, "Building safer cars," IEEE Spectrum, vol. 39, no. 1, pp. 82–85, 2002.

[16] I. EL JAAFARI, M. EL ANSARI, L. KOUTTI, A. ELLAHYANI, and S. CHARFI, "A novel approach for on-road vehicle detection and tracking," International Journal of Advanced Computer Science & Applications, vol. 1, no. 7, pp. 594–601.

[17] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," Neural processing letters, vol. 9, no. 3, pp. 293–300, 1999.

[18] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1. IEEE, 2005, pp. 886–893.

[19] W. Liu, X. Wen, B. Duan, H. Yuan, and N. Wang, "Rear vehicle detection and tracking for lane change assist," in Intelligent Vehicles Symposium, 2007 IEEE. IEEE, 2007, pp. 252–257.

[20] M. Cheon, W. Lee, C. Yoon, and M. Park, "Vision-based vehicle detection system with consideration of the detecting location," IEEE transactions on intelligent transportation systems, vol. 13, no. 3, pp. 1243–1252, 2012.

[21] Z. Sun, G. Bebis, and R. Miller, "Monocular precrash vehicle detection: Features and classifiers," IEEE transactions on image processing, vol. 15, no. 7, pp. 2019–2034, 2006.

[22] J. Cui, F. Liu, Z. Li, and Z. Jia, "Vehicle localisation using a single camera," in Intelligent Vehicles Symposium (IV), 2010 IEEE. IEEE, 2010, pp. 871–876.

[23] S. Sivaraman and M. M. Trivedi, "Active learning based robust monocular vehicle detection for on-road safety systems," in Intelligent Vehicles Symposium, 2009 IEEE. IEEE, 2009, pp. 399–404.

[24] D. G. Lowe, "Object recognition from local scale-invariant features," in Computer vision, 1999. The proceedings of the seventh IEEE international conference on, vol. 2. Ieee, 1999, pp. 1150–1157.

[25] X. Zhang, N. Zheng, Y. He, and F. Wang, "Vehicle detection using an extended hidden random field model," in Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on. IEEE, 2011, pp. 1555–1559.

[26] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3d lidar using fully convolutional network," arXiv preprint arXiv:1608.07916, 2016.

[27] Z. Dong, Y. Wu, M. Pei, and Y. Jia, "Vehicle type classification using a semisupervised convolutional neural network," IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 4, pp. 2247–2256, 2015.

[28] X. Chen, S. Xiang, C.-L. Liu, and C.-H. Pan, "Vehicle detection in satellite images by hybrid deep convolutional neural networks," IEEE Geoscience and remote sensing letters, vol. 11, no. 10, pp. 1797–1801, 2014.

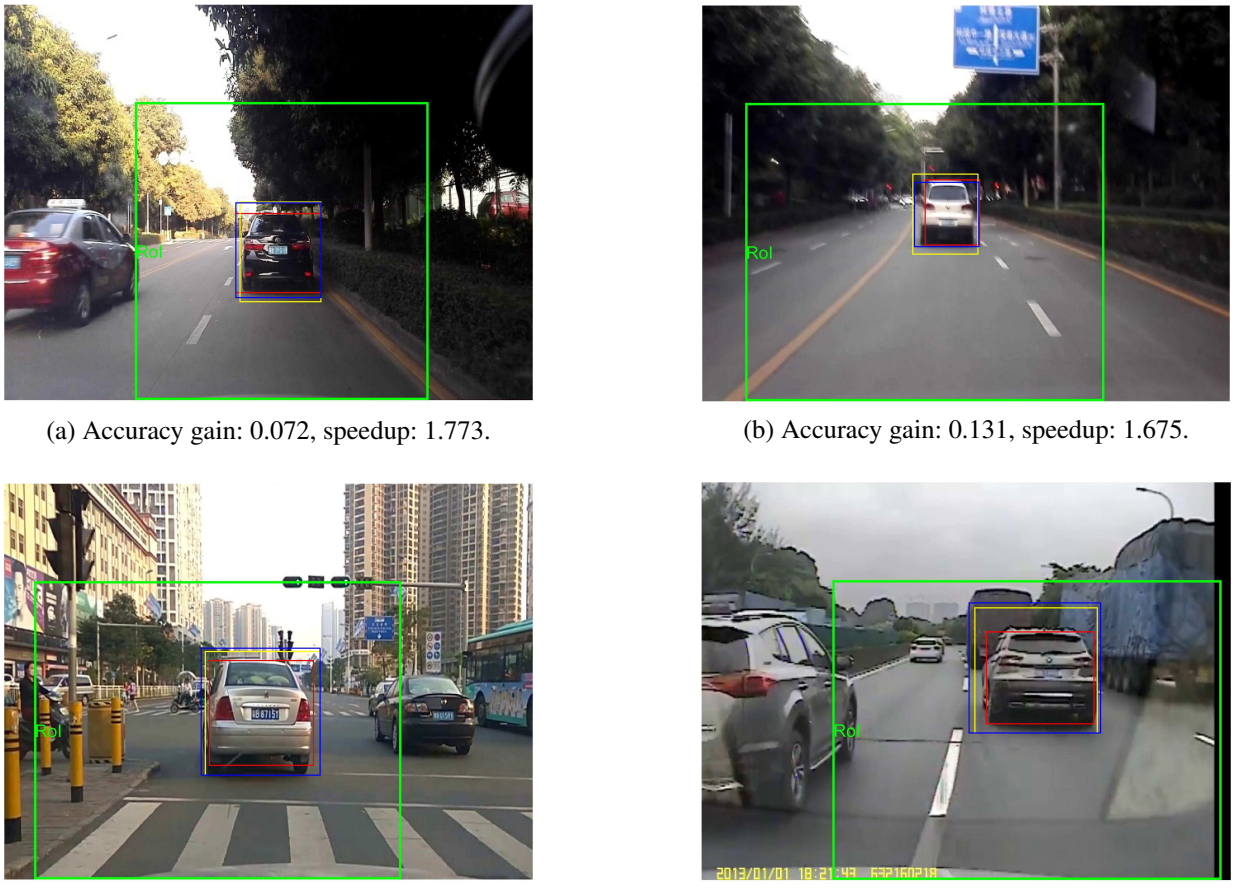
[29] ——, "Vehicle detection in satellite images by parallel deep convolutional neural networks," in Pattern Recognition (ACPR), 2013 2nd IAPR Asian Conference on. IEEE, 2013, pp. 181–185.

[30] Y.-K. Park, J.-K. Park, H.-I. On, and D.-J. Kang, "Convolutional neural network-based system for vehicle front-side detection," Journal of Institute of Control, Robotics and Systems, vol. 21, no. 11, pp. 1008–1016, 2015.

[31] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," arXiv preprint arXiv:1612.08242, 2016.

[32] H. Wang, Y. Cai, and L. Chen, "A vehicle detection algorithm based on deep belief network," The scientific world journal, vol. 2014, 2014.

[33] K. Kim, S. Lee, J.-Y. Kim, M. Kim, and H.-J. Yoo, "A configurable heterogeneous multicore architecture with cellular neural network for real-time object recognition," IEEE Transactions on Circuits and Systems for Video Technology, vol. 19, no. 11, pp. 1612–1622, 2009.



(a) Accuracy gain: 0.072, speedup: 1.773.

(b) Accuracy gain: 0.131, speedup: 1.675.

(c) Accuracy gain: -0.046, speedup: 1.567.

(d) Accuracy gain: -0.068, speedup: 1.582.

FIGURE 5: Compare original SSD and SSD-ROI in terms of accuracy gain (IOU score of SSD-ROI minus that of original SSD) and speedup (time cost used by original SSD divided by that of SSD-ROI). Colors of bounding boxes: green: ROI region, red: ground truth, yellow: predicted by original SSD, blue: predicted by SSD-ROI.

[34] N. Sudha, A. Mohan, and P. K. Meher, "A self-configurable systolic architecture for face recognition system based on principal component neural network," IEEE transactions on circuits and systems for video technology, vol. 21, no. 8, pp. 1071–1084, 2011.

[35] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," IEEE Transactions on Neural Networks, vol. 22, no. 2, pp. 199–210, 2011.

[36] S. J. Pan and Q. Yang, "A survey on transfer learning," IEEE Transactions on knowledge and data engineering, vol. 22, no. 10, pp. 1345–1359, 2010.

[37] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks." in ICML, 2015, pp. 97–105.

[38] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," arXiv preprint arXiv:1409.7495, 2014.

[39] H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, and M. Marchand, "Domain-adversarial neural networks," arXiv preprint arXiv:1412.4446, 2014.

[40] M. Ghifary, W. B. Kleijn, and M. Zhang, "Domain adaptive neural networks for object recognition," in Pacific Rim International Conference on Artificial Intelligence. Springer, 2014, pp. 898–904.

[41] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in European conference on computer vision. Springer, 2014, pp. 818–833.

[42] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 1717–1724.

[43] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in Advances in neural information processing systems, 2014, pp. 3320–3328.

[44] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in Advances in Neural Information Processing Systems 2, D. S. Touretzky, Ed. Morgan-Kaufmann, 1990, pp. 598–605. [Online]. Available: http://papers.nips.cc/paper/250-optimal-brain-damage.pdf

[45] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in Advances in Neural Information Processing Systems, 2015, pp. 1135–1143.

[46] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," arXiv preprint arXiv:1510.00149, 2015.

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

[48] 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, 2014, pp. 1269–1277.

[49] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," IEEE transactions on pattern analysis and machine intelligence, vol. 38, no. 10, pp. 1943–1955, 2016.

[50] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," arXiv preprint arXiv:1412.6115, 2014.

[51] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "High-performance neural networks for visual object classification," arXiv preprint arXiv:1102.0183, 2011.

[52] C. BuciluǍ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2006, pp. 535–541.
[53] 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.
[54] D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent, "The difficulty of training deep architectures and the effect of unsupervised pre-training." in AISTATS, vol. 5, 2009, pp. 153–160.
[55] C.-Y. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets." in AISTATS, vol. 2, no. 3, 2015, p. 5.
[56] 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, 2015, pp. 1–9.
[57] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in Advances in neural information processing systems, 2014, pp. 2654–2662.
[58] C. Xu, C. Lu, X. Liang, J. Gao, W. Zheng, T. Wang, and S. Yan, "Multi-loss regularized deep neural network," IEEE Transactions on Circuits and Systems for Video Technology, vol. 26, no. 12, pp. 2273–2283, 2016.
[59] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," arXiv preprint arXiv:1512.01274, 2015.
[60] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Master's thesis, Department of Computer Science, University of Toronto, 2009.
[61] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[62] C.-Y. Lee, P. W. Gallagher, and Z. Tu, "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree," in International conference on artificial intelligence and statistics, 2016.
[63] D. Mishkin and J. Matas, "All you need is a good init," arXiv preprint arXiv:1511.06422, 2015.
[64] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio, "Maxout networks." ICML (3), vol. 28, pp. 1319–1327, 2013.
[65] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," arXiv preprint arXiv:1412.6806, 2014.