Rocket Launching: A unified and effecient framework for training well-behaved light net

Xiaoqiang Zhu
Alibaba Group

Rocket Launching: A Universal and Efficient Framework for Training Well-performing Light Net

Guorui Zhou , Ying Fan , Runpeng Cui , Weijie Bian
Xiaoqiang Zhu and **Kun Gai**

Alibaba Inc.
Beijing, China
{guorui.xgr, fanying.fy, weijie.bwj, xiaoqiang.zxq}@alibaba-inc.com
cuirunpeng@gmail.com
jingshi.gk@taobao.com

Abstract

Models applied on real time response task, like click-through rate (CTR) prediction model, require high accuracy and rigorous response time. Therefore, top-performing deep models of high depth and complexity are not well suited for these applications with the limitations on the inference time. In order to get neural networks of better performance given the time limitations, we propose an unified framework that exploits a booster net to help train the lightweight net for prediction. We dub the whole process rocket launching, where the booster net is used to guide the learning of our light net throughout the whole training process. We analyze different loss functions aiming at pushing the light net to behave similarly to the booster net, and adopt the loss with best performance in our experiments. We use one technique called gradient block to improve the performance of light net and booster net further. Experiments on benchmark datasets and real-life industrial advertisement data show the effectiveness of our proposed method.

Introduction

Deep networks have achieved state-of-the-art results in many areas, such as computer vision (Huang et al. 2016) and nature language processing (Bahdanau, Cho, and Bengio 2014). From AlexNet (Krizhevsky, Sutskever, and Hinton 2012) to recently proposed DenseNet (Huang et al. 2016), better performances are accompanied with deeper and wider networks and more complex and adaptable structures. A more complex structure of neural networks means longer inference time, which is not tolerated in the real-life industrial environment. Networks mentioned above only consider the evaluation criterion of accuracy, while neglect the necessity of real-time response in industrial applications.

At the same time, some nets like DIN (Zhou et al. 2017) and wide & deep model (Cheng et al. 2016) get more and more attention. These net shares some characteristics: nets are shallow, layers are very simple and with less computation cost. In real-life industrial applications, e.g. online advertising systems, models have to make prediction of hundreds of advertisements for one user in several milliseconds, which restricts the complexity of model. Only simple and shallow structure meets the stringent response time requirements in real-life industry.

Accuracy and latency are the two points that we pay attention to. In general, there are two solutions to reduce runtime complexities while keeping a decent performance. Some works factorize or compress to directly simplify the computation, such as matrix SVD (Denton et al. 2014), MobileNet (Howard et al. 2017), and ShuffleNet (Zhang et al. 2017). Other approaches adopt the teacher-student strategy. They use light networks with fewer layers and parameters to decrease the inference time, while the light nets are trained helped by a complicated teacher network that trained in advance, like knowledge distillation (Hinton, Vinyals, and Dean 2015) and FitNet (Romero et al. 2014). These teacher-student methods decrease the runtime complexities, and can be further combined with approaches of the first category. In this work, we propose a novel unified framework to train decent small networks, motivated by the potential of teacher-student methods.

In this work, we develop a novel network training process dubbed rocket launching. The light net is the target network for inference, the booster relates to the deeper and more complex network from the architecture. Both the light and the booster net compose the architecture of rocket network. At the training stage, the light and booster networks are trained simultaneously on the same task. Besides, the light net also keeps getting knowledge learned by the booster through the optimization of the hint loss, which is included in the objective function to make both nets have similar behaviour during training. The booster guides the optimization of the target light network all along the training process. At the inference stage, only the trained light network is used. Different from previous teacher-student methods (Hinton, Vinyals, and Dean 2015; Romero et al. 2014), we make the light model share some lower layers with the cumbersome one and train them simultaneously.

In this paper, we propose a universal method aiming to obtain a well-behaved light net considering limitations on inference time. Our method is suitable to many different network structures. In brief, our contributions can be summarized as follows:

- We propose a novel universal training process called rocket launching, which makes use of the booster net to supervise the learning of the light network through whole training process. We show that a light model can be trained to perform close to deeper and more complex models in experiments.

- We analyze different hint loss functions to transfer the information from booster to the light net.
- In order to push light net to be close to booster net, we use gradient block technique to cancel the effect of hint loss back-propagation on layers of the booster, which gives booster net more freedom to update its parameters based on ground truth, which improve the performance further.

Our method achieves the state-of-the-art results on publicly available benchmarks as well as real-life industrial database. It is notable that our method performs better than other teacher-student approaches. Experimental results present that the performance can be further improved when combining other teacher-student approaches with our framework.

The remainder of this paper starts from a summary of related work. Then we introduce our approach, followed by experiments and conclusions.

Related work

Deep neural networks have drawn increasing attention in recent years due to their overwhelming performance on many research areas. One main trend of network structure design is to develop neural networks with larger depth, more parameters and higher complexity to achieve better performance (Simonyan and Zisserman 2015; Szegedy et al. 2015; He et al. 2016; Zagoruyko and Nikos 2016). However, these top-performing networks with high complexity will result in time consuming systems at the inference phase. Therefore, they are not well suited for applications with inference time limitations.

There have been some explorations of model compression by directly simplifying the computation or pruning of the original neural operations. Denton et al. (Denton et al. 2014) use SVD to approximate the convolutional operations in deep CNNs. MobileNets (Howard et al. 2017) are based on a streamlined architecture that uses depthwise separable convolutions to build lightweight deep neural networks. ShuffleNet (Zhang et al. 2017) uses pointwise group convolution and channel shuffle to reduce computation cost. ThiNet (Luo, Wu, and Lin 2017) uses statistic information from next layer to prune filters which accelerates CNN models while maintaining accuracy.

Besides designing delicate net structure, light net can get more information from extra pre-trained model during training phrase. This idea has been emphasized in Learnware (Zhou 2016). There have been some attempts adopting a teacher-student strategy, where a more complex teacher network is employed to teach a lightweight student network on a given task. The teacher network helps the student to get a decent performance at the inference phase. Buciluǎ et al. (Buciluǎ, Caruana, and Niculescu-Mizil 2006) improve compression model, which pioneered this type of learning process. They expound that the knowledge of a large ensemble of models could transfer to a single small model, they use a large ensemble of models to label large amounts of unlabel data, then use the data labeled by the ensemble models to train small model. Furthermore, Ba et al. (Ba and Caruana 2014) train a wider and shallower net called student net to mimic the big model called teacher net via regressing logits before the softmax layer with ℓ_2 loss. They think that matching logits could get more information than the hard label that provided by the cumbersome model. Hinton et al. (Hinton, Vinyals, and Dean 2015) point out that identifying knowledge in a trained model with the learned parameter value is hard. Instead, they make use of one abstract view of the knowledge that is the learned mapping from input vectors to output vectors, they propose the strategy of knowledge distillation which uses the class probabilities produced by the cumbersome model as "soft targets" for training the small model. They prove that they are the general version of matching logits which uesd by (Ba and Caruana 2014).

Besides using the output of the teacher network, people try to get more supervised information from the teacher. FitNets (Romero et al. 2014) use not only the outputs but also the intermediate representations learned by the cumbersome model as the hint to supervise the training process. Zagoruyko et al. (Zagoruyko and Komodakis 2016) use attention as a mechanism of transferring knowledge from one network to another. By properly defining attention for convolutional neural networks, they improve the performance of a student CNN by forcing it to mimic the attention maps of a powerful teacher network.

In previous teacher-student approaches, the cumbersome teacher networks are trained in advance. Instead of only transferring the final stationary outputs of the pretrained model, we let the booster model guide the whole training process of light net in rocket launching. We think that the knowledge learned by the cumbersome model exists not only in the final outputs, but also in the full learning process. The light model gets not only the difference between the target and the temporary outputs, but also the possible path towards the final target provided by a complex model of more learning capability. Another difference of our approach is that the part parameters of the light model and the booster are shared in our framework. We adopt the parameter sharing scheme since the lower-level representation of the same task should be universal. In the proposed architecture, the booster has a much deeper specific layers to ensure the capability to guide the light model to learn the task better.

Training several nets together is often applied on multi input scenes (Andrew et al. 2013; Bromley et al. 1994) or semi-supervised task (Laine and Aila 2016). Parameters sharing has also been used in multi-task (He et al. 2017). However, to the best of our knowledge, there is no attempt using these techniques on training of small net to get better performance. We are the first to utilize these schemes in model compression attempts, and experimental results present the effectiveness of our method.

Our approach

In this section, we will describe our proposed rocket net training process in detail. We will further analyze the highlights of our method and compare different hint loss functions.

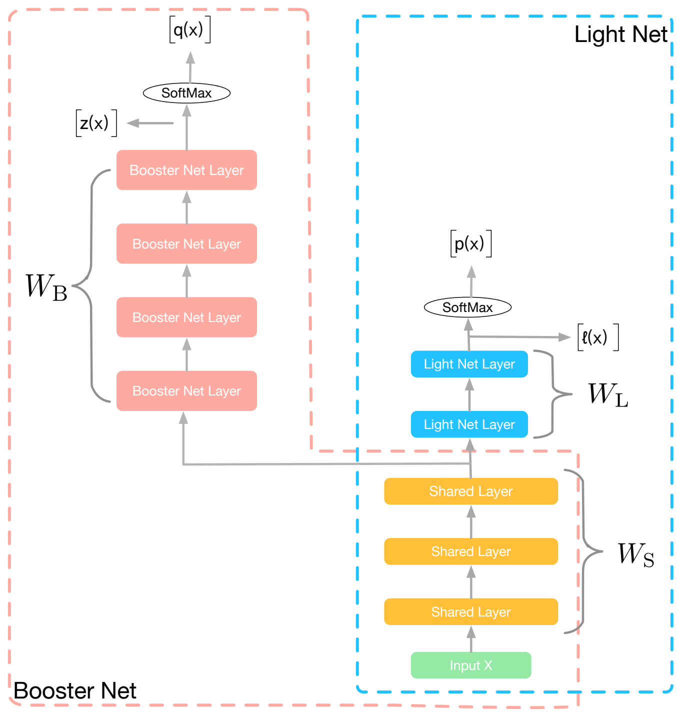


Figure 1: Whole Net Structure, blue dashed circle represents light net, pink dashed circle represents booster net. Yellow layers are shared by light net and booster net.

The sketch of our method

Fig. 1 represents the general structure of our architecture, it is composed of two parts: the light net and the booster layers. These two networks share some lower-level layers (annotated in yellow), and they both have their specific layers for the learning and prediction on the same task.

We let x and y denote the inputs and ground truth labels of our neural architecture. Let L be the light net with an output softmax as $p(x) = \mathrm{softmax}(l(x))$, where $l(x)$ is the weighted sum before the softmax activation. Parameters for the light net consist of two components: parameters in shared layers W_{S} and parameters in its lightweight particular layers for prediction W_{L}. We let B denote the booster network with shared parameters W_{S} and its particular weights W_{B} to get the final output. Similar to the light net, we have $q(x) = \mathrm{softmax}(z(x))$ as the output softmax for the booster, where $z(x)$ is the weighted sum before the softmax activation. We expect that the light net is trained similar to the true labels y, as well as approximate to the knowledge learned by the booster net with much more representation capability. To solve this problem, we take hint loss in the training objective in order to convey knowledge from the booster net to the light net. The objective function for rocket launching is defined as follows:

$$\begin{aligned}\mathcal{L}(x; W_{\mathrm{S}}, W_{\mathrm{L}}, W_{\mathrm{B}}) = \mathcal{H}(y, p(x)) + \mathcal{H}(y, q(x)) \\ + \lambda \| l(x) - z(x) \|_2^2,\end{aligned} \tag{1}$$

where the last term is the hint loss function as the mean square error (MSE) between the logits $z(x)$ and $l(x)$, $\mathcal{H}(p, q) = -\sum_i p_i \log q_i$ is the cross-entropy, λ is the parameter to balance the cross entropy and the hint loss. Here we use the cross-entropy terms for the booster and light nets to learn the true labels, and the hint loss function to exploit the knowledge learned by the booster to guide the learning process of the light network.

Characters of our method

There are some highlights in our method, which have notable effects on the training process and distinguish our method from other teacher-student approaches.

Parameter sharing In our approach, the light net shares parameter with the booster net. This scheme helps the light net get direct thrust from the booster, which pushes it get better performance.

The technique of parameter sharing is not new in deep learning. In computer vision areas, it is a common scheme to train deep convolutional neural networks in a multi-task manner. We assume that these tasks can be built on some shared low-level representations of the images. Given this assumption, we could reduce the parameters in neural network and improve its generalization capability. It is noticeable that, in real-life industrial applications, e.g. CTR prediction, reusing the embedding layers from other tasks helps a new task converge more easily and get a better performance.

Simultaneous training In most teacher-student methods, the teacher network is trained on the target database in advance, and its parameters are fixed when guiding the training process of the student net. Different from these approaches, we have the light and booster nets trained simultaneously, the whole learning process of the target light net is guided by the booster net. The light model can learn from not only the difference between the target and its temporary outputs, but also the possible path towards the final target provided by a complex model with more learning capability.

Notice that instead of training the teacher and student nets separately, the whole training time of our proposed architecture is shortened. Therefore, the compressed model can be trained more efficiently to meet the requirement in industrial applications that the inference model be updated frequently.

Hint loss functions In our approach, we transfer the booster net's knowledge to the light net by minimizing the hint loss. Several different hint loss functions are considered in this work:

- MSE of final softmax: $\mathcal{L}_{\mathrm{MSE}}(x) = \| p(x) - q(x) \|_2^2$,
- MSE of logits before softmax activation, which is also adopted in SNN-MIMIC (Ba and Caruana 2014): $\mathcal{L}_{\mathrm{mimic}}(x) = \| l(x) - z(x) \|_2^2$,
- knowledge distillation (Hinton, Vinyals, and Dean 2015): $\mathcal{L}_{\mathrm{KD}}(x) = \mathcal{H}(\frac{p(x)}{T}, \frac{q(x)}{T})$, where T is the temperature.

For the MSE of final softmax $\mathcal{L}_{\mathrm{MSE}}$, we have the derivative of the hint loss with respect to $l(x)$:

$$\frac{\partial \mathcal{L}_{\mathrm{MSE}}(x)}{\partial l_i(x)} = 2p_i(x)\Big[p_i(x) - q_i(x) + \sum\nolimits_k p_k(x)(q_k(x) - p_k(x))\Big]. \tag{2}$$

Notice that the gradient is proportional to the prediction outputs of the light net. If $l_i(x)$ is very negative, causing the gradient to vanish, the MSE of final softmax may fail to learn the difference in outputs, even when the light net makes radically different outputs from the booster net.

SNN-MIMIC learning (Ba and Caruana 2014) uses the formulation of $\mathcal{L}_{\mathrm{mimic}}$ between the teacher and student networks. We have the derivative w.r.t. $l_i(x)$:

$$\frac{\partial \mathcal{L}_{\mathrm{mimic}}(x)}{\partial l_i(x)} = l_i(x) - z_i(x). \tag{3}$$

We observe that the update directly reduces the difference between the logits before softmax, which relieves the problem in gradient with $\mathcal{L}_{\mathrm{MSE}}$. Experimental results also present that training with $\mathcal{L}_{\mathrm{mimic}}$ achieves the best performance among these different hint loss formulations. We also analyzed this kind of hit loss from the perspective of maximum likelihood in the Appendix.

Knowledge distillation (Hinton, Vinyals, and Dean 2015) uses cross-entropy to restrict the probability outputs of two models. In their work, a temperature T is introduced to produce a softer probability distribution among classes. They think that knowledge distillation is the general case of matching logits. They prove that with a high temperature, the gradient w.r.t. $l_i(x)$ is:

$$\frac{\partial \mathcal{L}_{\mathrm{KD}}(x)}{\partial l_i(x)} \approx \frac{1}{NT^2}(l_i(x) - z_i(x)), \tag{4}$$

where N is the number of classes, and approximation $\mathrm{e}^{l_i(x)/T} \approx 1 + l_i(x)/T$ is used. Their approximation neglects the term $(l_i(x)/T)^2$ in Taylor series when the temperature is high enough compared with the magnitude of the logits. Notice that the approximate gradient is the same order of infinitesimal to the neglected term, we argue that this approximation is problematic. But we approve the temperature's effect that it can soften class probability, which makes the distillation pays more attention to matching the negative logits below the average. In practice, Hinton et al. (Hinton, Vinyals, and Dean 2015) suggest that intermediate temperatures work best, which ignores the very negative logits that might be noisy. While in this work, we find that the optimization of all the logits' difference in our framework outperforms using the formulation of $\mathcal{L}_{\mathrm{KD}}$. We think that some very negative logits may conveys useful knowledge acquired by the cumbersome net, which helps the student network to get a better performance.

Gradient block In our proposed training process, the light net shares parameters and is trained together with the booster net. This simultaneous training scheme has an inevitable influence on the performance of the booster booster. Using both second term and hint loss term to update booster net's parameters will lead that booster net's probability and light net's probability are close to one result between them, which means booster net will be hindered from direct learning from the true labels, and its probability predictions are also strongly affected by the light net. Since the learning capability of the light model is limited, the performance of the booster net will be inevitably deteriorated. Notice that the light model learns from the knowledge conveyed by the booster net during training, this deterioration in the booster model's learning will further diminish the learning potential of the light network.

In order to solve this problem, during the training process, we develop the gradient block scheme to prevent the booster model from minimizing the hint loss objective. During the back-propagation of the hint loss term, we fix the gradient of booster net's specific parameters(W_{B}), and use this moment booster net's probability as target to supervise light net's study.

This operation makes the specific parameters W_{B} in booster net away from influence given by the light model, thus the booster can directly learn from the ground truth labels to achieve its best performance. For the light net, the parameters are normally updated to optimize the objective function in Eq. 1. Both the supervisory information and the booster's knowledge are the targets for the light model to learn from.

Experiments

In this section, we evaluate our rocket launching on several classification datasets and Alibaba advertisement database. Experimental results present that our proposed approach achieves notable improvements in the light net's performance and outperforms other teacher-student methods. In experiments on public benchmarks, we compare our method with KD (Hinton, Vinyals, and Dean 2015) and attention transfer (Zagoruyko and Komodakis 2016).

Experiments on CIFAR-10

The CIFAR-10 dataset (Krizhevsky and Hinton 2009) consists of 32×32 color images from 10 class. These images are split into $50,000$ training samples and $10,000$ testing samples. We preprocess the data with the same operations as in (Zagoruyko and Komodakis 2016). All the experiments are repeated 3 times with different seed, and we take the median of error rates as the final results. All the experiments, we use the same learning rate tuning and epochs as in (Zagoruyko and Komodakis 2016). We use the initial learning rate to 0.1 with momentum to 0.9, while we drop learning rate by 0.2 at [60,120,160] epochs and train for total 200 epochs.

We employ wide residual net (Zagoruyko and Nikos 2016) to be the instantiation of rocket launching on CIFAR-10 datasets. Wide residual net (WRN) has three groups of block, each block has two convolutional layers with larger width in contrast with the original ResNet. The wider layers are accompanied with more parameters, which could offer more representation capability. Fig. 2(a) shows the schematics of the rocket net structure based on wide residual networks. Layers in red are shared by the light net and the

Table 1: Comparisons of classification performance(test error) on CIFAR-10

light	booster	base[1]	AT	KD	rocket[2]	rocket+KD[3]	booster[4]	booster only[5]
WRN-16-1, 0.2M(a)	WRN-40-1, 0.6M	8.77	8.25	8.39	7.87	7.52	6.64	6.58
WRN-16-2, 0.7M(a)	WRN-40-2, 2.2M	6.31	5.85	6.08	5.67	5.64	5.20	5.23
WRN-16-1, 0.2M(b)	WRN-40-1, 0.6M	8.69	-[6]	8.34	7.85	7.86	7.27	6.58

[1] base means WRN-16 trains individually. [2] rocket means light net's result in rocket launching. [3] rocket+KD means light net's result using rocket launching combined with KD. [4] booster means booster net's result in rocket launching. [5] booster only means WRN-40 trains individually. [6] WRN-16-1, 0.2M(b) can't be applied on AT directly, so we are not report this result.

booster. The yellow part is the specific structure designed for light net to make prediction. The blue part is the specific layers of the booster, which is removed at inference.

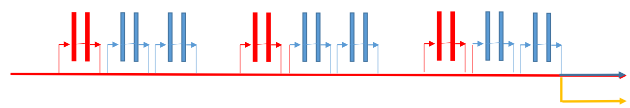
We explore rocket launching on light and booster net with different network depths and widths (e.g. WRN-16-1(a),0.2M means wide residual network with depth of 16 and widening factor of 1, using the layer sharing way like Fig. 2(a), its parameters' size is 0.2M). As shown in Table 1, our approach achieves consistently notable improvement compared to the base light net with different experimental settings. Taking the first line of Table 1 as example, using the same WRN-16-1(a) net structure, our rocket launching get 0.9% improvement compared with this net trained individual. We also observe that our approach outperforms other teacher-student methods, such as knowledge distillation(KD) (Hinton, Vinyals, and Dean 2015) and attention transfer (Zagoruyko and Komodakis 2016).

Besides comparing with other approaches, we also try to combine KD with our method by adding $\mathcal{L}_{\mathrm{KD}}$ to the objective in Eq. 1. It's notable that we use probability that pretrained by booster net in this term, which means light net can also additional guidance of a pre-trained booster network. From Table 1 we see that the performance can be further improved with the application of KD, which means our rocket launching has different effect on the light net with KD. The light net benefits from both the supervisory information brought by the pre-trained teacher network, and the knowledge conveyed by the booster network during the training process.

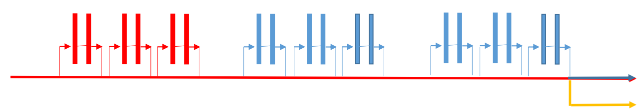
We also investigate our framework with different hint loss formulations. From Table 3 we see that the adopted hint loss to match the logits achieves the best performance among the different objectives. While hint loss to match the probability performs worst, which means gradient vanishing affects the training process. The experimental results are in accordance with our previous analysis.

Experiments are also carried out on CIFAR-10 to evaluate our framework design (see Fig. 2). We observe that both layer sharing and gradient block contribute to the improvements of our approach. For WRN-16-1(a), using gradient block(GB) gets 0.63% improvement compared with rocket (no GB); Using parameter sharing gets 0.2% improvement compared with rocket (no sharing).

Generalizability to different structures Different ways of parameter sharing may affect the performance of cumbersome net. Fig. 2(a) shows that the light net share some lower blocks with the booster in each group. We try to change the layers that shared by the light model and the booster, then observe the effect of sharing parameter on both networks.



(a) interval rocket net on wide residual net



(b) bottom rocket net on wide residual net

Figure 2: Proposed network structures for rocket net.

As shown in Fig. 2(b), we develop a different structure by sharing layers (layers in red) in the lower group of wide residual net. From Table 1 we see that our approach consistently works well on the different structure and outperforms other methods, which presents that the performance of our approach can well generalize to other structures.

Performance with different depths In this part, we investigate the learning capability of the light model with different depths and parameter sizes. By changing the number of shared blocks in different groups, we tune the number of shared layers from 16 to 28. From Fig. 3, we see that the light model gets a better performance and more learning capability with the increasing depth. Notice that the light get slight better result than booster net when its depth increases to 28. We think that in this case the light and booster nets share most of their layers, which may limit the learning capacity of the booster.

Visualization of rocket launching and attention transfer In order to explain our method intuitively. We visualize each group's output of light net and booster net respectively. To be consist with previous part, we use Fig. 2(a) as the basic net. For comparison, we visualize the corresponding results of spatial attention mapping. As we can see from Fig. 4, for both rocket launching and attention transfer (AT), the feature map generated from lower groups are similar between student and teacher. It indicates that parameter sharing and

Table 2: Comparisons of different framework design's result(test error) on CIFAR-10.

light	booster	base	rocket (no GB)[1]	rocket (no sharing)[2]	rocket
WRN-16-1(a)	WRN-40-1	8.77	8.50	8.06	7.87
WRN-16-1(b)	WRN-40-1	8.69	8.30	8.40	7.74

[1] rocket (no GB) means rocket launching without gradient block.
[2] rocket (no sharing) means rocket launching without parameter sharing.

Table 3: Comparisons of different hint loss functions on CIFAR-10

light	booster	$\mathcal{L}_{\text{mimic}}$	$\mathcal{L}_{\text{MSE}}$	$\mathcal{L}_{\text{KD}}$
WRN-16-1 (a)	WRN-40-1	7.86	8.32	7.98
WRN-16-1 (b)	WER-40-1	7.74	8.36	8.26

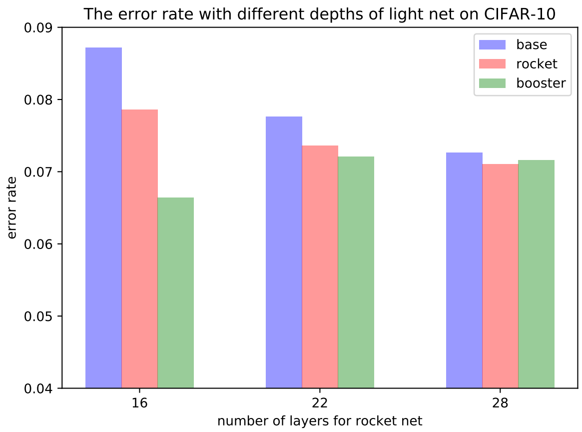
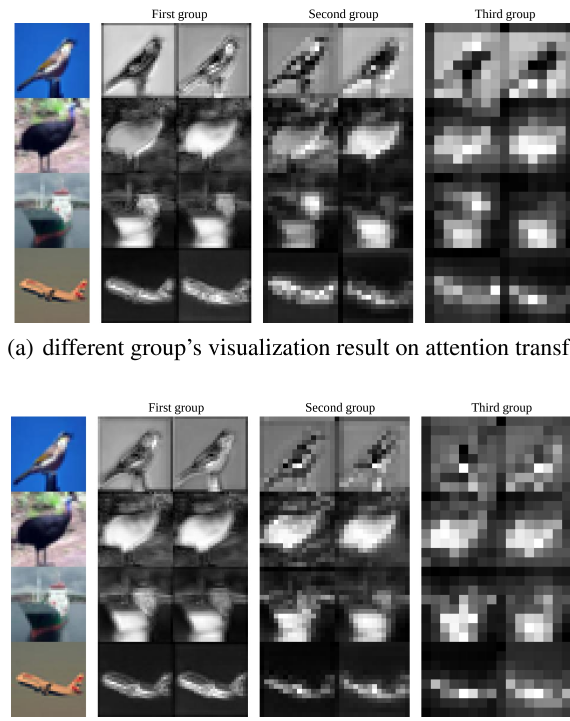


Figure 3: The error rates with different depths of light net on CIFAR-10



(a) different group's visualization result on attention transfer

(b) different group's visualization result on rocket launching

Figure 4: The visualization results on both rocket launching and attention transfer, in each group, the first and second picture in each group stands for booster net and light net respectively

attention have similar effect on lower layers. It can also show that these methods can learn the feature representation from booster net in low layer.

Experiments on SVHN and CIFAR-100

In order to verify the effectiveness of rocket launching further, we apply our method on CIFAR-100 and SVHN respectively. In order to compare with AT (which is based on WRN), we still use WRN as basic net structure, and we use the sharing method like Fig. 2(a) .

The CIFAR-100 datasets (Krizhevsky and Hinton 2009) consist of 32×32 color images from 100 classes. Like CIFAR-10, these images are still split into $50,000$ training and $10,000$ testing samples. The experiment setting of CIFAR-100 is same as CIFAR-10.

The SVHN database (Netzer et al. 2011) is obtained from house numbers in Google Street View images. It contains 32×32 images with RGB color channels in 10 class. There are $73,257$ images in the training set, $26,032$ images in test and $531,131$ samples in extra set. We follow the same evaluation procedure as Sermanet et al. (Sermanet, Chintala, and LeCun 2012) to compose our training, validation and test sets. For this dataset, we use validation dataset to choose the final model. In our experiment, we use Adam (Kingma and Ba 2014) with initial learning rate 0.001, while we drop learning rate by 0.2 at [20,40,60] epochs. Because this dataset is easy to learn, we add dropout after each booster specific layer with dropout rate 20% to prevent overfitting. For booster trained alone, same dropout layers are added to keep consistent.

Error rate on above two dataset is shown in Table 4. We observe that our approach get 1.29% improvement on SVHN and 10.4% improvement on CIFAR-100 compared with base. Further, rocket launching outperforms other teacher-student methods on all settings.

Table 4: Comparisons of classification performance(test error) on CIFAR-100 and SVHN

dataset	light	booster	base	AT	KD	rocket	rocket+KD
SVHN	WRN-16-1, 0.2M(a)	WRN-40-1, 0.6M	3.58	2.99	2.31	2.29	2.20
CIFAR-100	WRN-16-1, 0.2M(a)	WRN-40-1, 0.6M	43.7	34.1	36.4	33.3	33.0

Table 5: Experiments on Alibaba Advertisement Dataset

model	# parameters in FC layers	# multiplications in FC layers	inference time of FC Layers	GAUC
base	576 * 200 * 80 * 2	131360	4 ms	0.632
rocket	576 * 200 * 80 * 2	131360	4 ms	0.635
booster only	576 * 720 * 360 * 240 * 180 * 90 * 2	837900	- ms	0.637

Experiments on Alibaba Advertisement Dataset

In order to verity the effectiveness of rocket launching further, we test our method on huge real industry dataset. The dataset comes from productive display advertising system, we use rocket launching to predict if user clicks given product. The size of training dataset is 4 billion, the test dataset is 0.285 billion.

The network that we use is showed in DIN (Zhou et al. 2017). In the online system, most calculations focus on the fully connection layers after the embedding layers. So we try to use a booster net with more complex fully connection layers to guide our light net. The light net shares embedding layers with booster net. The booster net has seven wide hidden layers using complicated operation like batch normalization(Ioffe and Szegedy 2015), light net's specific layers with less hidden units and has only fully connection layers. The light net in the huge real data gets 0.3% improvement on GAUC (the generalization of AUC) (Zhou et al. 2017) with the same latency as the base model. The booster net gets the best performance on the offline metric, but the inference time of booster net is unacceptable for online system. Our approach can get an health improvement on a model with fixed structure and parameter quantity. And this experiment proves that one can use our approach to break the boundary brought by the latency limitation to some degree.

Conclusion

We propose a general framework named rocket launching to get a efficient well-performing light model with the help of a cumbersome booster net. In order to get as much as information from the booster model, we make the booster and the light net train on the same task together with the hint loss objective, which pushes the booster model to supervise the whole training process of the light one. Besides, the light model shares parameter with the booster to make the light net get low-level representation directly from the booster. We also analyze different hint loss functions that can convey knowledge from the booster to the light model. Moreover, we develop the gradient block scheme to prevent the booster net from deterioration. For future work, we would like to explore training networks with not only smaller depths but also fewer neurons in each layer to further improve the inference efficiency. Another promising research path is to develop methods to implement attention-based models with less inference time.

References

[Andrew et al. 2013] Andrew, G.; Arora, R.; Bilmes, J.; and Livescu, K. 2013. Deep canonical correlation analysis. In *Proceedings of the 30th International Conference on Machine Learning*, 1247–1255.

[Ba and Caruana 2014] Ba, J., and Caruana, R. 2014. Do deep nets really need to be deep? In *Advances in neural information processing systems 27*, 2654–2662. Cambridge, MA.: MIT Press.

[Bahdanau, Cho, and Bengio 2014] Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Bromley et al. 1994] Bromley, J.; Guyon, I.; LeCun, Y.; Säckinger, E.; and Shah, R. 1994. Signature verification using a siamese time delay neural network. In *Proceedings of the 12th Advances in Neural Information Processing Systems*, 737–744.

[Buciluǎ, Caruana, and Niculescu-Mizil 2006] Buciluǎ, C.; Caruana, R.; and Niculescu-Mizil, A. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 535–541.

[Cheng et al. 2016] Cheng, H.-T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, 7–10. Boston, USA: ACM.

[Denton et al. 2014] Denton, E. L.; Zaremba, W.; Bruna, J.; LeCun, Y.; and Fergus, R. 2014. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems 27*, 1269–1277. Cambridge, MA.: MIT Press.

[He et al. 2016] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

[He et al. 2017] He, K.; Gkioxari, G.; Dollar, P.; and Girshick, R. 2017. Mask r-cnn. *arXiv preprint arXiv:1703.06870*.

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

[Howard et al. 2017] Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[Huang et al. 2016] Huang, G.; Liu, Z.; Weinberger, K. Q.; and van der Maaten, L. 2016. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*.

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

[Kingma and Ba 2014] Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Krizhevsky and Hinton 2009] Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images.

[Krizhevsky, Sutskever, and Hinton 2012] Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems 25*, 1097–1105.

[Laine and Aila 2016] Laine, S., and Aila, T. 2016. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*.

[Luo, Wu, and Lin 2017] Luo, J.-H.; Wu, J.; and Lin, W. 2017. Thinet: A filter level pruning method for deep neural network compression. *arXiv preprint arXiv:1707.06342*.

[Netzer et al. 2011] Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, 5.

[Romero et al. 2014] Romero, A.; Ballas, N.; Kahou, S. E.; and Chassang, A. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.655d0*.

[Sermanet, Chintala, and LeCun 2012] Sermanet, P.; Chintala, S.; and LeCun, Y. 2012. Convolutional neural networks applied to house numbers digit classification. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, 3288–3291. IEEE.

[Simonyan and Zisserman 2015] Simonyan, K., and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the 3th International Conference on Learning Representations*, 621–630.

[Szegedy et al. 2015] Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

[Zagoruyko and Komodakis 2016] Zagoruyko, Z., and Komodakis, K. 2016. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*.

[Zagoruyko and Nikos 2016] Zagoruyko, S., and Nikos, K. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146*.

[Zhang et al. 2017] Zhang, X.; Zhou, X.; Lin, M.; and Sun, J. 2017. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*.

[Zhou et al. 2017] Zhou, G.; Song, C.; Zhu, X.; Ma, X.; Yan, Y.; Dai, X.; Zhu, H.; Jin, J.; Li, H.; and Gai, K. 2017. Deep interest network for click-through rate prediction. *arXiv preprint arXiv:1706.06978*.

[Zhou 2016] Zhou, Z.-H. 2016. Learnware: on the future of machine learning. *Frontiers of Computer Science* 10(4):589–590.

Appendix A

We try to improve light net's performance by minimizing the hint loss between output of the booster net and the light net. Now we try to analyze why MSE of logits before softmax activation get good performance in our approach from the perspective of maximum likelihood. We let Y_b and Y_l denote the weighted sum before the softmax activation. In order to simplify the complexity of the problem, we made two assumptions. We assume that Y_b and Y_l are uncorrelated. And when one choose ReLU as activation, we assume that Y_b and Y_l are truncated gaussian. $Y_b = TN(\mu_b, \sigma_b; 0, \infty)$ and $Y_l = TN(\mu_l, \sigma_l; 0, \infty)$, where $TN(\mu, \sigma; a, b)$ is a truncated normal distribution where a and b are the the lower and upper bounds of the truncation respectively. Now the target error we want to optimize is $Z = Y_b - Y_l$. The likelihood L is:

$$L = \prod_{i=0}^{n} P(Z_i). \tag{.1}$$

Where n is the number of samples. Under the assumption that the samples are independent identically distributed:

$$log(L) = \sum_{i=0}^{n} log(P(Z_i)). \tag{.2}$$

Now we should analysis the formulation of $P(Z)$. $Z = Y_b - Y_l$, where Y_b and Y_l are truncated gaussian. In a more general condition, we set $z = x + y$, where $x \sim N(\mu_x, \sigma_x^2)$ be doubly truncated (below and above) at (ax, bx) and $y \sim N(\mu_y, \sigma_y^2)$ be doubly truncated (below and above) at (ay, by). Now:

$$\begin{aligned} P_x(x) &= (x \le a_x)\Phi_(a_x, \mu_x, \sigma_x) + (a_x < x < b_x)\int_{a_x}^{x} \rho(x, \mu_x, \sigma_x)dx + (x > b_x)(1 - \Phi(b_x, \mu_x, \sigma_x)). \\ P_y(y) &= (y \le a_y)\Phi_(a_y, \mu_y, \sigma_y) + (a_y < y < b_y)\int_{a_y}^{y} \rho(y, \mu_y, \sigma_y)dy + (y > b_y)(1 - \Phi(b_y, \mu_y, \sigma_y)). \end{aligned} \tag{.3}$$

So the mean operator E is:

$$E(f(X)) = \Phi(a_x, \mu_x, \sigma_x)f(a_x) + \int_{a_x}^{b_x} f(x)\rho(x, \mu_x, \sigma_x)dx + f(b_x)(1 - \Phi(b_x, \mu_x, \sigma_x)). \tag{.4}$$

Now we define $f(x) = P_y(z - x)$, the we get:

$$P_z(z) = \Phi(a_x, \mu_x, \sigma_x)P_y(z - a_x) + \int_{a_x}^{b_x} P_y(z - x)\rho(x, \mu_x, \sigma_x)dx + P_y(z - b_x)(1 - \Phi(b_x, \mu_x, \sigma_x)). \tag{.5}$$

Obviously :

$$P_y(y) = \frac{1}{\Phi(b_y, \mu_y, \sigma_y) - \Phi(a_y, \mu_y, \sigma_y)} \int_{\max(y, a_y)}^{\min(y, b_y)} \rho(y, \mu_y, \sigma_y)dy = (y > a_y)\frac{\Phi(\min(y, b_y), \mu_y, \sigma_y) - \Phi(a_y, \mu_y, \sigma_y)}{\Phi(b_y, \mu_y, \sigma_y) - \Phi(a_y, \mu_y, \sigma_y)} \tag{.6}$$

and

$$E(f(X)) = \frac{1}{\Phi(b_x, \mu_x, \sigma_x) - \Phi(a_x, \mu_x, \sigma_x)} \int_{a_x}^{b_x} f(x)\rho(x, \mu_x, \sigma_x)dx \tag{.7}$$

So:

$$P_z(z) = \frac{1}{\Phi(b_x, \mu_x, \sigma_x) - \Phi(a_x, \mu_x, \sigma_x)} \int_{a_x}^{b_x} P_y(z - x)\rho(x, \mu_x, \sigma_x)dx \tag{.8}$$

and

$$\rho_z(z) = \frac{\int_{a_x}^{b_x} (a_y < z - x < b_y)\rho(z - x, \mu_y, \sigma_y)\rho(x, \mu_x, \sigma_x)dx}{(\Phi(b_x, \mu_x, \sigma_x) - \Phi(a_x, \mu_x, \sigma_x))(\Phi(b_y, \mu_y, \sigma_y) - \Phi(a_y, \mu_y, \sigma_y))} \tag{.9}$$

which yield 4-parts:

- $z < a_y + b_x$ & $z \le a_x + b_y$,
- $z \ge a_y + b_x$ & $z \le a_x + b_y$,
- $z < a_y + b_x$ & $z > a_x + b_y$,
- $z \ge a_y + b_x$ & $z \ge a_x + b_y$.

In our case, let x and y correspond to Y_b and $-Y_l$, so $\mu_x = -\mu_y$ and $a_x = 0, b_x = \infty, a_y = -\infty, b_y = 0$. The 4-parts are symmetry, so we analyze one of them, $z \geq a_y + b_x$ & $z \geq a_x + b_y$. Then we get:

$$PDF(z) = \frac{e^{\frac{(-z+\mu_x+\mu_y)^2}{2(\sigma_x^2+\sigma_y^2)}}\sqrt{\frac{2}{\pi}}\{-Erf[\frac{(z-b_x-\mu_y)\sigma_x^2+(-b_x+\mu_x)\sigma_y^2}{\sqrt{2}\sigma_x\sigma_y\sqrt{\sigma_x^2+\sigma_y^2}}] + Erf[\frac{(b_y-\mu_y)\sigma_x^2+(-z+b_y+\mu_x)\sigma_y^2}{\sqrt{2}\sigma_x\sigma_y\sqrt{\sigma_x^2+\sigma_y^2}}]\}}{(Erf[\frac{a_x-\mu_x}{\sqrt{2}\sigma_x}] - Erf[\frac{b_x-\mu_x}{\sqrt{2}\sigma_x}])(Erf[\frac{a_y-\mu_y}{\sqrt{2}\sigma_y}] - Erf[\frac{b_y-\mu_y}{\sqrt{2}\sigma_y}])\sqrt{\sigma_x^2+\sigma_y^2}} \tag{0.10}$$

Erf is the gauss error function which is a non-elementary function. But erf(x) has bound, $erf(x) \in (-1, 1)$. So $\exists\, a\ constant\ C\ s.t.\ PDF(z) \leq e^{\frac{(-z+\mu_x+\mu_y)^2}{2(\sigma_x^2+\sigma_y^2)}} * C$. Now we get:

$$log(L) = \sum_{i=0}^{n} log(P(Z_i)). \tag{0.11}$$

$$log(L) \leq \sum_{i=0}^{n}(log(C) + \frac{(-z+\mu_x+\mu_y)^2}{2(\sigma_x^2+\sigma_y^2)}) \tag{0.12}$$

μ_x is mean of Y_b and μ_y is mean of $-Y_l$, so $\mu_x = -\mu_y$. Thus:

$$log(L) \leq \sum_{i=0}^{n}(log(C) + \frac{(-z)^2}{2(\sigma_x^2+\sigma_y^2)}) \tag{0.13}$$

$$log(L) \leq \sum_{i=0}^{n}(log(C) + \frac{(Y_b-Y_l)^2}{2(\sigma_x^2+\sigma_y^2)}) \tag{0.14}$$

So the MSE of logits: $(Y_b - Y_l)^2$ is an upper bound of the log-likelihood to optimize the error $Z = Y_b - Y_l$.