EasyChair Preprint

№ 6252

Elastic Deep Learning using Knowledge Distillation with Heterogeneous Computing Resources

Daxiang Dong, Ji Liu, Xi Wang, Weibao Gong, An Qin, Xingjian Li, Dianhai Yu, Patrick Valduriez and Dejing Dou

August 6, 2021

Elastic Deep Learning using Knowledge Distillation with Heterogeneous Computing Resources

Daxiang Dong[1†], Ji Liu[1†*], Xi Wang[1], Weibao Gong[1], An Qin[1], Xingjian Li[1], Dianhai Yu[1], Patrick Valduriez[2], and Dejing Dou[1]

[1] Baidu, Beijing, China
[2] Inria, University of Montpellier, CNRS, LIRMM, France
{dongdaxiang, liuji04, wangxi16, gongweibao, qinan, lixingjian, yudianhai, doudejing}@baidu.com, Patrick.Valduriez@inria.fr

Abstract. In deep neural networks, using more layers and parameters generally improves the accuracy of the models, which get bigger. Such big models have high computational complexity and big memory requirements, which exceed the capacity of small devices for inference. Knowledge distillation is an efficient approach to compress a large deep model (a teacher model) to a compact model (a student model). Existing online knowledge distillation methods typically exploit an extra data storage layer to store the knowledge or deploy the teacher model and the student model at the same computing resource, thus hurting elasticity and fault-tolerance. In this paper, we propose an elastic deep learning framework, EDL-Dist, for large scale knowledge distillation to efficiently train the student model while exploiting elastic computing resources. The advantages of EDL-Dist are three-fold. First, it decouples the inference and the training process to use heterogeneous computing resources. Second, it can exploit dynamically available computing resources. Third, it supports fault-tolerance during the training and inference processes within knowledge distillation. Our experimental validation, based on industrial-strength implementation and real datasets, shows that the throughput of EDL-Dist is up to 181% faster than the baseline method (online knowledge distillation).

Keywords: Knowledge distillation · Distributed computing · Deep neural network.

1 Introduction

In recent years, Deep Neural Networks (DNNs) have achieved major success in various domains, such as computer vision [15] and natural language processing [14]. Bigger models with more layers, neurons and parameters generally improve

[†]Equal contribution
[*]Corresponding author

the accuracy of a model. For instance, BERT [3] and ERNIE [14] exploit large numbers of parameters, e.g., from 110 million to 340 million parameters for BERT. With a large number of parameters, deep neural networks have high computational complexity and big memory requirements, which exceed the capacity of small devices (mobile phones or IoT devices) on which they are deployed for inference.

Knowledge distillation [7] is an efficient approach to distill the knowledge from a big model into a smaller model while retaining its accuracy. During knowledge distillation, a small model (a student model) is trained with the supervision of a large model (a teacher model). The teacher model is a cumbersome model, which could be an ensemble of separately trained models or a single very large model trained with a strong regularizer [7]. Compared with the teacher model, the student model is relatively small and compact.

Different from normal training, which does not rely on a trained teacher model, knowledge distillation requires a pre-trained teacher model. During knowledge distillation, the teacher model is used by the inference process to generate supervision knowledge while the student model is trained. These inference and training processes can be sequentially performed or in parallel. The training can be carried out offline or online. The offline approach exploits an extra data store to cache the knowledge distilled from the teacher model, which is used to train the student model separately [5]. This approach can decouple the inference of a teacher model and the training of a student model. However, this approach requires much storage and distilling the knowledge from the teacher model may take much time when the teacher model or input data are big. The online approach puts the teacher model and the student model into the same server and performs the training of the student model and the inference of the teacher model synchronously. When the teacher model is very big, the training of the student model gets limited by the synchronization of the inference of the teacher model, which takes much computing time. In addition, both the offline and online approaches do not support elastic computing resources or fault-tolerance.

The training of knowledge distillation typically exploits many computing resources, e.g., CPU cores or GPU cards. However, the availability of computing resources may vary dynamically as there may be concurrent users with same priority. Thus, some computing resources can be granted to a user for a long time while some other computing resources can only be used for a short time and may be dynamically withdrawn. During the long training of knowledge distillation with elastic computing resources, some will become unavailable, while some others will become available. Furthermore, these computing resources are heterogeneous, e.g., GPU cards with diverse computing capabilities.

In this paper, we address the problem of efficient knowledge distillation with heterogeneous computing resources. We assume a distributed environment with two kinds of computing resources: dedicated and elastic. The dedicated computing resources are provided by powerful servers, e.g, V100 GPU cards, for knowledge distillation only. The elastic computing resources are smaller servers, e.g., P4 GPU cards, and can be dynamically allocated to other tasks of higher priority or knowledge distillation. We propose an Elastic Deep Learning framework, i.e.,

EDL-Dist, with a distributed, fault-tolerant architecture. EDL-Dist provides an elastic service that manages multiple GPU cards, for inference of teacher models. It manages the training of knowledge distillation with multiple GPU cards on multiple servers (a server may have one GPU or more GPU cards). Furthermore, the type of GPU cards can be different depending on the process, e.g., training versus inference, which are decoupled in order to exploit elasticity. Our solution to fault-tolerance is based on a fail-over mechanism [13] using check-points and re-execution of tasks in Teacher. In addition, EDL-Dist comes with two main algorithms for scheduling and knowledge distillation. The scheduling algorithm is hybrid, i.e., combines static and dynamic scheduling, and associates computing resources from different processes. The knowledge distillation algorithm, EDL-Dist algorithm, is distributed and enables decentralized training of the student model with the knowledge from the teacher model.

This paper is organized as follows. Section 2 introduces the related work of knowledge distillation. Section 3 describes the EDL-Dist framework. Section 4 presents the experimental results, which show the advantage of EDL-Dist compared with the baseline method (the online approach) and normal training. Finally, Section 5 concludes the paper.

2 Related Work

In this section, we first introduce knowledge distillation. Then, we discuss solutions for supporting knowledge distillation, with distributed or decentralized training, and elastic computing resources.

Knowledge distillation is based on the popular machine learning Softmax function and a temperature [7]. The Softmax output layer converts the logit computed for each class (predefined category associated to the input data) into a probability with a temperature T. The temperature indicates the impact of the output from the teacher model, where a higher value of T corresponds to a weaker impact of the output of the teacher model.

During the training of knowledge distillation, two neural networks are used: teacher model and student model. The student model is trained using the combination of two loss functions. One loss function is based on a soft prediction, which considers the soft labels from the teacher model. The other loss function is based on a hard prediction, which considers the ground truth label from the training data. The soft prediction corresponds to the outputs of the student model, while the hard prediction is the original output of the student model.

Knowledge distillation can be carried out based on two methods: offline and online. With the offline method, a teacher model is trained before distillation. Then, the knowledge of the teacher model can be extracted and stored in a cache [5]. This method requires large extra storage resources. With the online method [16], the inference of the teacher model and the training of the student model are performed in the same GPU card. Thus, when the teacher model is big, the training of the student model gets limited by the synchronization of the inference

of the teacher model, which takes much computing time. Furthermore, these two approaches do not support elastic computing resources or fault-tolerance.

In order to accelerate the training of a deep learning network, multiple GPUs can be exploited using data parallelism. The model is replicated in each GPU while the data is distributed in different GPUs [1]. Ring all-reduce [4] is generally exploited to realize the distributed training process with the data parallelism method. However, the ring all-reduce method is only designed for the training without consideration of knowledge distillation. Furthermore, it cannot support elastic computing resources and do not provide fault-tolerance.

3 EDL-Dist

In this section, we present the EDL-Dist framework, its architecture, algorithms for scheduling and knowledge distillation, and our solution for fault-tolerance.

3.1 Architecture

The architecture of EDL-Dist (see Figure 1) has three modules: Student, Teacher and Coordinator. Student is composed of dedicated computing resources, which are used to train a student model with a distributed or decentralized method. Teacher consists of dynamic computing resources. In the dynamic computing resources, the teacher models are deployed for the inference process. Coordinator coordinates data transfer and training in Student and inference in Teacher.

We exploit a decentralized training algorithm, i.e., ring allReduce [4], to perform parallel training in Student. Since transferring data among different GPU cards is time consuming, the training data is partitioned and cached in the host memory of each server for fast data access. During training in Student, only the gradients are transferred among different servers. Within each iteration of the training process, each computing resource in the student model takes data, including the input training data, the hard labels and soft labels from a data service, DistilReader, to update the student mode.

DistilReader is a service that caches the input training data and the corresponding soft labels, generated from Teacher, in the host memory of each computing resource in Student. It provides an interface between a Student server and the Coordinator server or Teacher servers. The Student server, Coordinator server or Teacher server denotes a server that supports the corresponding module. This service is deployed in each Student server. As shown in Figure 2, DistilReader sends the input data to Teacher and receives the soft labels from Teacher. In order to know which Teacher server to connect to, DistilReader retrieves the server information from Coordinator. In addition, DistilReader regularly queries Coordinator to know if the Teacher server is still alive. When a Teacher server becomes unavailable, DistilReader searches for available Teacher servers from Coordinator to replace the unavailable server.

Teacher is composed of multiple dynamic computing resources, each of which can become unavailable at any moment because of unexpected changes in the

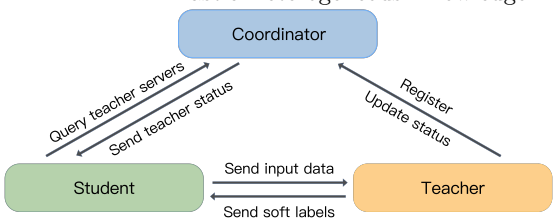


Fig. 1: Functional architecture.

multiuser workload. When a dynamic server is alive and added as a Teacher server, it is registered in Coordinator. Then, a teacher model instance is deployed in the server in order to perform inference, which takes input data and generates corresponding soft labels. When the Teacher server remains available to be connected for knowledge distillation, it sends heartbeat messages to Coordinator in order to maintain its status until the end of the knowledge distillation task.

Coordinator has two components: a service manager and a database, which is an in-memory database for efficient data processing. The service manager can query the database in order to search for available computing resources in Teacher. The service manager answers the queries from DistilReaders in Student. The register information from Teacher is directly stored in the database. The alive status stored in the database has a time limit, i.e., Time to live (TTL). When the heartbeat information is sent from a Teacher server to the database, the corresponding alive status is prolonged, i.e., the corresponding TTL is updated. If the Teacher server does not send heartbeat messages to the database for a long time, when its TTL expired, its status will be considered unavailable.

3.2 Hybrid Scheduling Algorithm

In order to speed up the inference process, it is critical to schedule the workloads, i.e., the inference to generate soft labels for input data, requested from Student to computing resources in Teacher. A resource represents a computing unit that can perform training, e.g., a GPU card or a CPU care. The resource scheduling problem is NP-hard [11]. When a Student resource is scheduled to a smaller number of Teacher resources than an appropriate number, the throughput of the student model is restricted by the inference of its scheduled resources. Otherwise, when a resource in Student is scheduled to a bigger number of resources in Teacher, more and more soft labels and corresponding input data will be stored in the host memory of Student servers to be used. The accumulated stored soft labels and corresponding input data may occupy large amount of memory, which may block the training process. Thus, it is important to schedule the appropriate number of resources to each Student resource.

We propose a hybrid scheduling algorithm (see Algorithm 1), i.e., which combines static and dynamic scheduling methods. We assume historical information on the execution of the training and inference processes. For instance, the throughput of the training in a Student server, e.g., one GPU card in Student, is t_s and the throughput of the inference in a Teacher server, e.g., one GPU card in Teacher, is t_t. The throughput gives the number of images or the amount of input data that can be processed in the same resource per time unit by the student

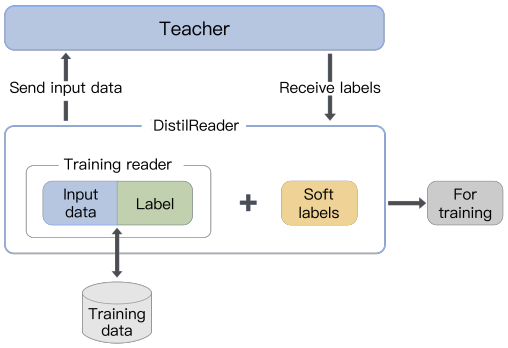


Fig. 2: DistilReader service.

model (or the teacher model) without restriction of another module. We assume that the resources in the same module, e.g., Student or Teacher, are of the same type while the types of GPU cards in different modules can be different. We set the number of Teacher resources as $n = \frac{t_t}{t_s}$ for each Student resource, i.e., we schedule $\lceil n \rceil$ Teacher resources to each Student resource. During the training of knowledge distillation, when a Student resource searches for Teacher resources, it is scheduled $\lceil n \rceil$ Teacher resources (Line 1). As the execution environment may vary during the training of knowledge distillation, we dynamically adjust the scheduling (Lines 3-12). We use a monitoring task in each Student resource to monitor the number of combinations of soft labels and input data (Line 3). The occupied volume is calculated based on the number and average size of a combination of input data and soft labels, which can be measured with an offline method. When the growing volume exceeds a predefined upper threshold value (Line 4), the Student resource stops sending input data to the Teacher resource (Line 5) in order to consume the unused soft labels until the volume decreases to a smaller value than another lower bound threshold value (Lines 10-12). The upper threshold and the lower threshold can be set by the user based on the size of storage in the resource of Student. This mechanism ensures that the number of soft labels remains reasonable in each Student resource, which does not slow down the training or incur memory leaks in the Student resource. Otherwise, if the resources in Student stay idle in order to wait for the soft labels from Teacher, more Teacher resources are required by the Student resource in order to accelerate the inference in the teacher model (Lines 7-9). When there are available Teacher resources, they are scheduled to the Student resource.

3.3 EDL-Dist Algorithm

We now present our EDL-Dist Algorithm 2 for the parallel training in each Student resource during the training of knowledge distillation. The input data and the hard label y are retrieved from the host memory (Line 3), which can be done by DistilReader. Then, the soft labels are prepared by the DistReader service from Teacher in Line 4. Based on the hard label and the soft labels, the student model θ is updated in Line 5. The loss function in each server is a weighted function based on the loss function of the hard labels and the soft

Algorithm 1 Hybrid Scheduling Algorithm

Require: number of Teacher resources n
Require: lower threshold of the volume of soft labels lt
Require: upper threshold of the volume of soft labels ut
1: schedule n Teacher resources to the Student resource
2: **while** knowledge distillation is not terminated **do**
3: volume = get_volume(unused soft labels)
4: **if** volume $> ut$ **then**
5: stop sending input data to Teacher servers
6: **end if**
7: **if** volume == 0 **then**
8: schedule an additional available Teacher resources to the Student resource
9: **end if**
10: **if** volume $< lt$ **then**
11: continue sending input data to Teacher resources
12: **end if**
13: **end while**

labels. λ is the learning rate, which can be set corresponding to the student model. Then, an average student model is calculated in Line 7.

3.4 Fault-tolerance

We consider the fault-tolerance in Student and Teacher, assuming that Coordinator is always available. If the Coordinator server is not stable, fault-tolerance can be simply achieved by having multiple instances of the in-memory database deployed in multiple servers using existing frameworks, e.g., Zookeeper [8]. If a Teacher resource is not available, its status will become unavailable when its TTL expires in the database. The Teacher resource can become unavailable in three cases. The first case is before the resource is scheduled to a Student resource. In this case, EDL-Dist simply ignores this Teacher resource. The second case is when the Teacher resource is scheduled to a Student resource that does not send input data to it or does not wait for soft labels from it. In this case, the Student resource will search for another available Teacher resource that is not scheduled to any Student resource. The third case is when the Teacher resource is scheduled to a Student resource that sends input data to it and is waiting for soft labels from it. In this case, as presented in Section 3.1, the Student resource will search for another available Teacher resource. Once a Teacher resource is re-scheduled to it, the Student resource sends the input data to the Teacher resource again. When a new Teacher resource is available in Teacher, it is scheduled to a Student resource that is searching for Teacher resources. If there is no such Student resource, the Teacher resource will wait for such a Student resource.

To address fault-tolerance in Student, we exploit a fail-over mechanism [13] that uses check-points during the training of knowledge distillation. A checkpoint is a copy of the student model. Before the training process, a server is selected as a master node and saves the checkpoint at every certain iterations. The checkpoint

Algorithm 2 EDL-Dist Algorithm

Require: hard loss function ϕ(hard label, hard prediction)
Require: soft loss function ψ(soft labels, soft predictions)
Require: hard prediction function $F(\theta, input)$
Require: soft prediction function $F'(\theta, input)$
Require: learning rate η
Require: weight for hard loss function α
Require: weight for soft loss function β
Require: number of Student resources N
while not converged **do**
  for θ_i in resource i **do**
    y, input = get_training_sample()
    soft_labels = get_soft_labels(input)
    $\theta_i = \theta_i - \eta \bigtriangledown_{\theta_i} \{\alpha\phi(y, F(\theta_i, input)) + \beta\psi(soft\_labels, F'(\theta_i, input)\}$
  end for
  $\theta = \frac{\sum_{j=1}^{N} \theta_j}{N}$
end while

is saved in a distributed file system, which is accessible to all the Student servers. Each Student server updates the student model in each iteration. Then, when a Student server becomes unavailable or a new Student server is added to Student, the training in all the Student servers stops. Afterward, each Student server loads the student model from the checkpoint and continues the training process. Thus, the consistency of the student model is ensured while addressing fault-tolerance.

4 Experimental Validation

In this section, we present our experimental validation of EDL-Dist in comparison with online knowledge distillation (Online) (baseline) and normal training (N-training). We present the experimental setup and then give the results.

4.1 Experimental Setup

EDL-Dist is implemented based on the PaddlePaddle framework [12] and publicly available at Github[1]. Student is based on Paddle FleetX[2], which implements the ring allReduce algorithm using NCCL[3] for decentralized training. We use Redis[4] as the in-memory database in Coordinator [10].

We carry out three experiments to show the advantages of EDL-Dist compared with Online and N-training. Online deploys the teacher and student models in the same GPU server. N-training represents the training with GPU cards without

[1] `https://github.com/elasticdeeplearning/edl`
[2] Paddle Fleet: `https://github.com/PaddlePaddle/FleetX`
[3] NCCL: `https://developer.nvidia.com/nccl`
[4] Redis: `https://redis.io/`

Table 1: Throughput for different approaches.

CPUcores	N-training	Online	EDL-Dist	Advantage
1	14.16	5.92	**14.34**	**142.23%**
2	28.44	11.51	**28.07**	**143.87%**
4	55.17	21.76	**54.92**	**152.39%**
8	101.59	37.87	**102.40**	**170.40%**
16	168.42	59.94	**168.42**	**180.98%**

Table 2: Throughput for different approaches.

CPUcores	N-training	Online	EDL-Dist	Advantage
8	57.14	46.04	35.68	-22.50%
12	57.14	46.04	**52.46**	**13.94%**
16	55.17	46.04	**57.65**	**25.22%**

knowledge distillation. In all experiments, we use real datasets, i.e., ImageNet data set [2]. In the first experiment, we combine CPUs and GPU cards, in order to show that EDL-Dist can efficiently exploit heterogeneous computing resources.

In the next two experiments, we use ResNet101 [6] as the teacher model, ResNet50 [6] as the student model, and set the batch size as 32 in each Student GPU card. The second experiment (Section 4.3) figures out the fine-tuned number of Teacher GPU cards (NVIDIA Tesla P4 GPU card) for each Student GPU card (NVIDIA Tesla V100 GPU card). The single-precision performance, which represents the speed to perform calculation, of P4 is 5.5 Teraflops while that of V100 is 14 Teraflops. The third experiment (Section 4.4) is performed with 8 v100 GPU cards in Student and various numbers of P4 GPU cards in Teacher for EDL-Dist. We compare the throughput and the training time to that of Online and N-training.

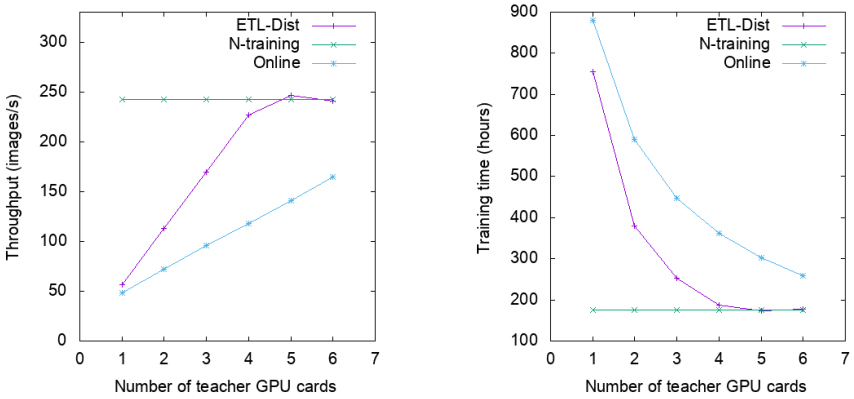
4.2 Comparison with Heterogeneous Resources

To validate that our solution is efficient with heterogeneous computing resources, we experiment with the combination of CPU and GPU cards for knowledge distillation. We take MobileNetV3_small [9] as the student model and Resnet50 [6] as the teacher model. We use Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz CPU cores and a P4 GPU card. We set the batch size as 64 in Student.

First, we take the P4 GPU card in Teacher and different numbers of CPU cores in Student. The results are shown in Table 1. The throughput of our proposed approach, i.e., EDL-Dist, is similar to that of N-training and significantly outperforms Online (up to 181%). Then, we take the P4 GPU card as the Student GPU card and different numbers of CPU cores as the Student resources. As shown in Table 2, the throughput of EDL-Dist is smaller than that of N-training and Online when Teacher resources are not enough (8). The throughput of EDL-Dist is similar to that of N-training and significantly outperforms Online (up to 25.22%) when the Teacher resources are enough (12 and 16).

4.3 Fine-tuning of EDL-Dist

The throughput of EDL-Dist increases with the number of Teacher GPU cards. With enough Teacher GPU cards, the throughput of EDL-Dist can be similar



(a) Throughput (b) Training time

Fig. 3: Fine-tuning with various numbers of P40 Teacher GPU cards.

to that of N-training. As we add more Teacher GPU cards, the throughput of EDL-Dist becomes a little bit lower as it takes some time to manage unused intermediate soft labels from Teacher. In order to validate this property of EDL-Dist, we also experiment using a v100 GPU card in Student and various numbers of P4 GPU cards as Teacher resources. The throughput of EDL-Dist is shown in Figure 3a when using different numbers of P4 GPU cards. The training time is shown in Figure 3b. Figures 3a and 3b indicate that the fine-tuned number of Teacher resources (P4 GPU card) is 5 when we use a single v100 GPU card as the Student resource. When the number of Teacher GPU cards is smaller than 5, the throughput increases linearly as number of P4 GPU cards increases, which shows the good scalability of EDL-Dist. When the number of Teacher GPU cards is greater than 5, the throughput slightly decreases as it takes time to manage unused soft labels in the Student server. Furthermore, we find that the throughput of Online is much smaller (up to 93.0%) than that of EDL-Dist and the training time of the Online is much longer (up to 92.9%) than that of EDL-Dist when the number of Teacher GPU cards is smaller than 8.

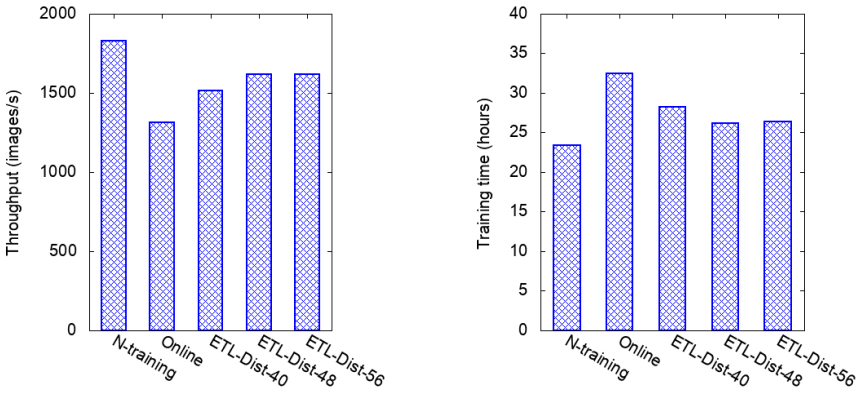
4.4 Comparison with multiple Student GPU Cards

In this experiment, we take 8 V100 GPU cards and 40 - 56 P4 GPU cards for different approaches. We compare the throughput between EDL-Dist, Online and N-training. We take 8 NVIDIA Tesla v100 GPU cards as dedicated Student GPU cards while using 48 P4 NVIDIA Tesla GPU cards as Teacher GPU cards as we find 48 is the appropriate number of Teacher GPU cards as shown in Table 3.

Table 3 shows that the accuracy (1 and 5) of EDL-Dist is similar to that of N-training and Online. Accuracy 1 represents the accuracy of the predicted class with the highest probability. Accuracy 5 represents the accuracy of the top 5 ranked classes based on the probability. The accuracy of EDL-Dist can be slightly higher than that of N-training (Accuracy 5). While the student model is trained with the training data and the soft labels with knowledge distillation, the trained student model from knowledge distillation can get more generalization

Table 3: Experimental Results (accuracy). Accuracy 1 is the accuracy of the predicted class with the highest probability. Accuracy 5 is the accuracy of the top 5 ranked classes based on the probability.

	N-training	Online	EDL-Dist (40)	EDL-Dist (48)	EDL-Dist (56)
Accuracy 1	77.1	79.0	**79.0**	**79.0**	**79.0**
Accuracy 5	93.5	94.3	**94.5**	**94.5**	**94.5**



(a) Throughput (b) Training time

Fig. 4: Experimental results with 8 Student v100 GPU cards and 40(EDL-Dist-40)/48(EDL-Dist-48)/56(EDL-Dist-56) P40 GPU cards.

information from the teacher model [7]. Thus, we can efficiently train a student model with higher accuracy (compared with N-training) using EDL-Dist.

In Figure 4a, the throughput of EDL-Dist is much higher (23.5% faster) than that of Online. This shows that EDL-Dist significantly speeds up training compared with the Online while not requiring extra storage resources. The throughput of EDL-Dist is slightly lower than that of N-learning because of some overhead when there are multiple Student GPU cards. The training time of N-training, Online and EDL-Dist is shown in Figure 4b. The training time of EDL-Dist (48) is almost the same as that of EDL-Dist (56), which indicates that the bottleneck of the number of Teacher GPU cards is 48. With 48 Teacher cards, the training time of EDL-Dist is 19.4% shorter than that of Online. Compared with N-training, the training time of EDL-Dist is slightly longer (12.8%). As it takes time to transfer the data from Student servers to multiple Teacher servers, the training time of EDL-Dist is slightly longer than that of N-training.

5 Conclusion

In this paper, we proposed EDL-Dist, an elastic deep learning framework for large scale knowledge distillation. EDL-Dist has a distributed, fault-tolerant architecture that leverages heterogeneous computing resources. We did a thorough validation of our solution by implementing an industrial-strenght prototype of EDL-Dist (available at github) and experimenting with real datasets. The experimental results show that EDL-Dist can be 181% faster than online training while its accuracy is a little higher than that of normal training.

References

1. Anil, R., Pereyra, G., Passos, A., Ormándi, R., Dahl, G. E., and Hinton, G. E. Large scale distributed neural network training through online distillation. In *Int. Conf. on Learning Representations (ICLR)* (2018).
2. Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Li, F. Imagenet: A large-scale hierarchical image database. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (2009), pp. 248–255.
3. Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In *Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (NAACL-HLT)* (2019), pp. 4171–4186.
4. Gibiansky, A. Bringing hpc techniques to deep learning. `https://andrew.gibiansky.com/blog/machine-learning/baidu-allreduce/`, 2017. Accessed: 2020-08-12.
5. Gou, J., Yu, B., Maybank, S. J., and Tao, D. Knowledge distillation: A survey. *CoRR abs/2006.05525* (2020).
6. He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778.
7. Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop* (2015).
8. Hunt, P., Konar, M., Junqueira, F. P., and Reed, B. Zookeeper: Wait-free coordination for internet-scale systems. In *USENIX annual technical conference* (2010), p. 11.
9. Koonce, B. *MobileNetV3*. Apress, 2021, pp. 125–144.
10. Liu, J., Pineda-Morales, L., Pacitti, E., Costan, A., Valduriez, P., Antoniu, G., and Mattoso, M. Efficient scheduling of scientific workflows using hot metadata in a multisite cloud. *IEEE Trans. Knowl. Data Eng. 31*, 10 (2019), 1940–1953.
11. Liu, L., Yu, H., Sun, G., Luo, L., Jin, Q., and Luo, S. Job scheduling for distributed machine learning in optical wan. *Future Generation Computer Systems 112* (2020), 549 – 560.
12. Ma, Y., adn Tian Wu, D. Y., and Wang, H. Paddlepaddle: An open-source deep learning platform from industrial practice. *Frontiers of Data and Computing 1*, 1 (2019), 105.
13. Özsu, M. T., and Valduriez, P. *Principles of distributed database systems*, 4 ed. Springer, 2020.
14. Sun, Y., Wang, S., Li, Y., Feng, S., Tian, H., Wu, H., and Wang, H. ERNIE 2.0: A continual pre-training framework for language understanding. In *AAAI Conf. on Artificial Intelligence* (2020), pp. 8968–8975.
15. Villegas, R., Yang, J., Zou, Y., Sohn, S., Lin, X., and Lee, H. Learning to generate long-term future via hierarchical prediction. In *Int. Conf. on Machine Learning (ICML)* (2017), vol. 70, pp. 3560–3569.
16. Zmora, N., Jacob, G., Zlotnik, L., Elharar, B., and Novik, G. Neural network distiller: A python package for DNN compression research. *CoRR abs/1910.12232* (2019).