On Efficient Training of Large-Scale Deep Learning Models

LI SHEN[*], Sun Yat-sen University, Shenzhen, China
YAN SUN[*], Engineering, The University of Sydney Faculty of Engineering and Information Technologies, Sydney, Australia
ZHIYUAN YU[*], University of Science and Technology of China, Hefei, China
LIANG DING, JD.com Inc, Beijing, China
XINMEI TIAN, University of Science and Technology of China, Hefei, China
DACHENG TAO, Nanyang Technological University, Singapore, Singapore

The field of deep learning has witnessed significant progress in recent times, particularly in areas such as computer vision (CV), natural language processing (NLP), and speech. The use of large-scale models trained on vast amounts of data holds immense promise for practical applications, enhancing industrial productivity and facilitating social development. However, it extremely suffers from the unstable training process and stringent requirements of computational resources. With the increasing demands on the adaption of computational capacity, though numerous studies have explored the efficient training field to a certain extent, a comprehensive summarization/guideline on those general acceleration techniques of training large-scale deep learning models is still much anticipated. In this survey, we present a detailed review of the general techniques for training acceleration. We consider the fundamental update formulation and split its basic components into five main perspectives: (1) "data-centric": including dataset regularization, data sampling, and data-centric curriculum learning techniques, which can significantly reduce the computational complexity of the data samples; (2) "model-centric", including acceleration of basic modules, compression training, model initialization and model-centric curriculum learning techniques, which focus on accelerating the training via reducing the calculations on parameters and providing better initialization; (3) "optimization-centric", including the selection of learning rate, the employment of large batchsize, the designs of efficient objectives, and model average techniques, which pay attention to the training policy and improving the generality for the large-scale models; (4) "budgeted training", including some distinctive acceleration methods on source-constrained situations, e.g. for limitation on the total iterations; (5) "system-centric", including some efficient distributed frameworks and open-source libraries which provide adequate hardware support for the implementation of above mentioned acceleration algorithms. By presenting this comprehensive taxonomy, our survey presents a comprehensive review to understand the general mechanisms within each component and their joint interaction. Meanwhile, we further provide a detailed analysis and discussion of future works on the development of general acceleration techniques, which could inspire us to re-think and design novel efficient paradigms. Overall, we hope that this survey will serve as a valuable guideline for general efficient training.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**.

[*]Equal contribution to this research. Correspondence to Li Shen.

Authors' Contact Information: Li Shen, Sun Yat-sen University, Shenzhen, Guangdong, China; e-mail: mathshenli@gmail.com; Yan Sun, Engineering, The University of Sydney Faculty of Engineering and Information Technologies, Sydney, New South Wales, Australia; e-mail: woodenchild95@outlook.com; Zhiyuan Yu, University of Science and Technology of China, Hefei, Anhui, China; e-mail: yuzhiyuan@mail.ustc.edu.cn; Liang Ding, JD.com Inc, Beijing, China; e-mail: liangding.liam@gmail.com; Xinmei Tian, University of Science and Technology of China, Hefei, Anhui, China; e-mail: xinmei@ustc.edu.cn; Dacheng Tao, Nanyang Technological University, Singapore, Singapore; e-mail: dacheng.tao@ntu.edu.sg.

ACM 1557-7341/2024/10-ART
https://doi.org/10.1145/3700439

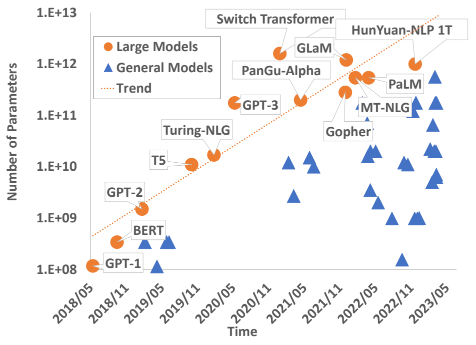

Fig. 1. Milestone of large-scale models.

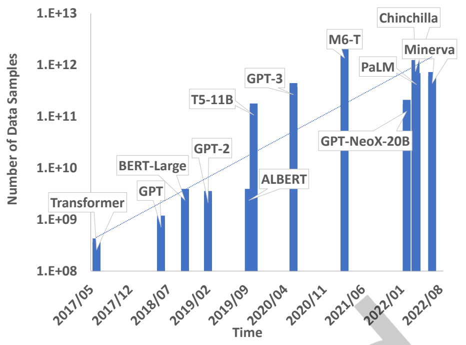

Fig. 2. Milestone of the large datasets.

Additional Key Words and Phrases: efficient training, large-scale models, general acceleration techniques

1 Introduction

With the rapid development of artificial intelligence techniques, parameters of deep models have a remarkable growth spurt with millions and even billions. Kaplan et al. [84] study the relationships of model size, dataset size, and the amount of computation used for training as a power-law and indicate that larger models are inherently data-hungry and significantly more sample-efficient on the learning. The deployment of large models has also become one of the most important research fields. Two years later, GPT-3 [14] holds 175B parameters trained on 45TB data samples and successfully achieves state-of-the-art results on various natural language processing tasks. Turing-NLG employs the generative language model with approximately 17.2B parameters and it takes only one year to quickly iterate to a sizeable model MT-NLG with 530B parameters [170], which is well ahead of the GPT-3 in several tasks. We summarize the milestone of the size development of models and datasets with time in Fig. 1 and 2. Though gains from the rapid growth are stunning, substantial progress in exploring novel techniques and training capabilities is eagerly awaited to sustain high efficiency. The enormous and expensive costs of training such sizeable models are usually unacceptable for now. Training the GPT-3 consumes approximately 355 GPU years at a cost of $4.6M. Under such a huge amount of parameters and data samples, traditional training from scratch obviously can not afford the massive expense, especially when expanding to the downstream tasks [24, 139, 140, 155, 178], which introduce additional architectures and parameters. Therefore, efficient training has increasingly drawn lots of attention and shines in deep learning.

Efficient training of large-scale deep learning models has become a critical research area in machine learning. While there has been significant progress in this field, much of the existing studies focus on specific model architectures or serve particular communities. In contrast, our study offers a comprehensive review of practical acceleration techniques for any large-scale deep learning model, independent of the task or model architecture. From the perspective of practical efficiency, we consider that efficient training mainly focuses on two clear targets:

- *To achieve comparable test accuracy, efficient training requires less training time.*
- *Under similar training costs, efficient training achieves higher performance.*

Our survey provides insightful guidance on the general training acceleration of deep learning models. We analyze the efficacy of training acceleration techniques on various basic backbone architectures that underpin many modern deep learning models. By examining different architectures of deep networks, our review can help achieve efficient training across any type of deep learning model. Additionally, since our survey is task-free and model-free, it provides a broad generalization of training acceleration techniques that can be applied

Table 1. Notations of the terms in Equation (2). The textual structure of this survey is followed by this table. We disassemble the fundamental update formulation in the training to introduce how the current acceleration techniques are designed and implemented.

Components	Definition	Research Field	Text
(X_i, Y_i)	dataset samples	Data-centric	Section 3
w w_0	model parameters weight initialization	Model-centric	Section 4
B γ G	size of the minibatch learning rate gradient estimator	Optimization-centric	Section 5
T	maximum iterations	Budgeted Training	Section 6
$\sum$	summation	System-centric	Section 7

across different domains and model architectures. Our survey aims to be a helpful resource for researchers and practitioners looking to accelerate the training of their large-scale deep learning models. By understanding the general principles behind effective training acceleration techniques, researchers can develop faster and more efficient models without being constrained by specific architectures or tasks. Overall, our study makes a significant contribution to the field of machine learning by providing a comprehensive survey of general training acceleration techniques for large-scale learning models.

In our survey, we are interested in solving the general fundamental minimization problem which could be easily expanded to the training foundation models or pretraining tasks:

$$\min_{w} \left\{ F(w) \triangleq \mathbb{E}_{\xi\sim\mathcal{D}}\left[f(w;\xi) \right] \right\}, \tag{1}$$

where w is the parameters, and ξ represents the data samples subject to $\mathcal{D}$. On the practical scenarios, we consider a large training set (X, Y) and utilize the empirical risk minimization (ERM) loss to approximate the full expectation F. In this case, each single data pair (X_i, Y_i) is sampled uniformly.

Different from the previous works, we deconstruct the general *gradient-based* descent formulation as the architecture of this review. Specifically, we consider all the components in the fomulation (2) which could cover the total training process in deep learning. Without loss of the generality, we use the update vector G instead of the gradients to encompass a wide range of methods. We consider the fundamental update formulation as:

$$w_T = w_0 - \sum_{t=0}^{T-1} \frac{\gamma_t}{B} \sum_{i=1}^{B} G(w_t; (X_i, Y_i)). \tag{2}$$

In Table 1, we summarize the notations and their corresponding research field. Based on Equation (2), by refining and splitting the different roles of components, we distinguish the previous works according to their inherent heuristic insights and theoretical scenes into 5 major categories. Each corresponds to the optimized target of the computation efficiency under the classified group. We fine-grain the above components to categorize current general acceleration techniques for training large-scale models with the feasibility of practical implementation. Specifically, they are:

- **Data-centric efficient training.** Data-centric methods are employed to expand the sample capacity of the training set via effective data augmentation and regularization policies. It requires additional pre-processing calculations to enhance the diversity and maintains the potential for higher stability, leading to better generalization performance in real-world applications. Meanwhile, to achieve efficient accelerations and further improve the generality of models, data-centric methods study the valid sampling techniques to select a key sub-set during stochastic optimization. It effectively reduces the number of samples that are required to be involved in calculating the gradients. Furthermore, it protects the models from over-fitting in training at those unimportant samples or data that have been learned well enough. In summary, the core consideration of data-centric methods is how to reduce data processing requirements without compromising performance.

- **Model-centric efficient training.** The deep model is an elaborate mapping function from the data domain to the ground truth. Past works have explored a number of mature architectures to construct a network for efficient training, e.g. convolution-based neural networks (CNN), multilayer perceptron (MLP), and transformer models. The model-centric methods focus more on the computational complexity of DNNs via efficient architectural approximations, compression, and efficient initialization for better generality. These methods focus on reducing the parameter size of DNNs while maintaining good performance. In summary, model-centric methods provide a promising approach to reducing the computational complexity of deep models for efficient training, which offers strong practicality and can be easily implemented.
- **Optimization-centric efficient training.** Optimization-centric efficiency focuses on the key factors in an optimizer, mainly consisting of the learning rate, batch size, and optimization objectives. Policies on learning rate selections aim to develop efficient and flexible strategies to train models. Batch-size-centric methods usually focus on adopting large minibatch to boost optimization speed. And from the objective perspective, we always strive to implement a global target with high stability. In summary, optimization-centric methods can study efficient iterative calculations in the training process to provide solid guarantees.
- **Budgeted efficient training.** Budgeted training is an approach that takes into account the available resources during practical training. It is mainly concerned with training efficiency in source-constrained scenarios, where computational resources such as training wall-clock time or calculation flops are limited. The primary objective of budgeted training is to ensure efficient and stable training while maximizing the potential of the model within the given constraints. This approach can lead to significant gains in the early stages of training. By adopting budgeted training, researchers and practitioners can make the most of available resources and avoid wasting them on inefficient models or training procedures.
- **System-centric efficient training.** System-centric methods focus on practical implementation under hardware support, which could turn algorithm design into real executable projects. Training large-scale models usually employs multi-node multi-device environments to implement parallel computing. It is primarily concerned with designing the underlying logic to address bottlenecks in communication across devices and coordinate the entire training process efficiently. To efficiently utilize distributed training, the training process is distributed into smaller computational tasks that are executed on different nodes or devices in parallel. This distributed system enables the training of large datasets and complex models.

In summary, our survey reviews the general training accelerations for efficient training. In the sections “data-centric", “model-centric", “optimization-centric", and “budget training", we mainly focus on the comprehensive studies from the perspectives of algorithm design and methodology, while in the section “system-centric", we focus on the practical implementation from the perspectives of paradigm innovation and hardware support. The main contributions of this survey are:

- We review the general acceleration techniques for training large-scale models from the “Data", “Model", “Optimization", “Budgeted training" and “System" perspectives to summarize their technical routes and implementation, which contributes to providing solid guidelines for efficient training with task-free and model-free.

- We compare the strengths and weaknesses of each component in training acceleration and demonstrate their insights and interactions, which could inspire us to rethink the design of the high-efficiency paradigms for the large-scale models.
- We provide a comprehensive analysis of each technical route and its major challenges in the practical scenarios, which could provide guidance on their promising developments.

The main structure of this survey is organized as follows. In section 2, we introduce related work and some domain-specific examples of efficient training processes. From section 3 to section 6, we introduce the details of characteristics and properties from the perspective of “Data-centric", “Model-centric", “Optimization-centric", “Budgeted training", and “System-centric" based on the iteration formulation (2), which contain their insights of different technical routes. We also analyze and assess the strengths and drawbacks of each implementation. This novel taxonomy could provide a clear and comprehensive guideline on the existing approaches for efficient training. In Section 8, we discuss and summarize the techniques in this survey and further suggest some promising research in the future studies.

2 Related Work

Many recent works review and summarize the efficient training techniques, mainly including the introduction of efficient pre-training models, the novel designed components for acceleration, advanced optimization methods, efficient training on the NLP or CV communities, and the bag of tricks in the training process. In early works, He et al. [66] summarize a bag of tricks for training CNN models. They conduct systematic experiments and summarize some effective techniques for data augmentation and the design of the ingenious learning rate scheduler. To review the NLP-based training efficiency, Treviso et al. [186] summarize efficient methods for NLP and discuss their efficiency and shortcomings. Then Qiu et al. [144] present a systematically categorized list of pre-trained models and study the development history of language models. Later, to further provide directions for exploration, Han et al. [62] summarize the research works related to the general pre-training techniques and provide some insights into their future research. Although this survey highlighted many potential areas of knowledge and exploration, the community still lacks efficient components for fundamental models. To fill this gap, Bommasani et al. [12], Zhou et al. [221] introduce the efficient foundation models, mainly from the perspectives of their general concepts, their powerful capabilities, their underlying training techniques, and applications. They also summarize the evolution of the pre-training and the current challenges in the practical scenarios respectively. On this basis, Lin et al. [113] further focus on the novel Transformer models and review the several variants of the Transformer models, which are aligned to consider the efficient architecture modification, pre-training techniques, and training accelerations. Contemporaneous works [181, 197] have also focused on this point. As researchers further explore computational techniques, Zhuang et al. [222] study the overview of the efficient training of Transformers including computation efficiency, memory efficiency, and hardware/algorithm co-design. The open-source community has contributed significantly to educational materials on the foundational infrastructure of AI systems [23, 135], emphasizing engineering aspects such as compilation and AI hardware. In contrast to this perspective, research into optimization theory has also gained increasing attention. To explore the trade-off between generalization and optimization performance, He et al. [67] study the recent advances of large-scale deep learning in terms of generalization guarantee and optimization efficiency, which include the novel optimizers and strategies to address the training overheads and reduce the required memory in the computational devices. They also elaborate on the exploration of large-batch training. Different from above, we focus more on the fundamental acceleration techniques which are not limited to the Transformer models.

Table 2. Overview of Data-centric efficient training.

Category	Details
Data Regularization	Data regularization is the pre-processing technique to enhance the diversity of raw data samples through transformations. It improves the representation of training samples in the feature space without requiring additional labeling costs.
	Papers Included: [97], [65], [179], [37], [220], [213], [210], [187], [189], [73], [114], [90], [30], [18], [28], [69], [29], [47], [196], [216], [193], [58], [43], [31]
Data Sampling	Data sampling is an effective method to select a subset of samples from a large batch to perform the updates. It benefits from ignoring those unimportant or bad samples in the current batch to adopt the small batchsize training. Usually, sampling is more important, yielding higher efficiency.
	Papers Included: [87], [86], [81], [22], [214], [88], [133], [63], [202], [174], [15]
Data-centric Curriculum Learning	Curriculum learning studies the progressive training setups at different stages in the training process to reduce the total computation costs. Initially, training with a low-quality dataset is sufficient to learn low-level features. Then, adopting a high-quality dataset (more augmentations and complicated pre-processing methods) gradually helps to learn sophisticated features.
	Papers Included: [44], [6], [199], [85], [74], [185], [194], [94], [141], [39]

3 Data-Centric Efficient Training

Recent advances in large-scale models shine significantly while their requirements on the dataset increase dramatically. Enormous data samples are employed to drive the training process and achieve outstanding performance. Therefore, studies in the Data-centric are crucial for practical acceleration. The essential role of data processing is to efficiently increase the diversity of data samples without additional labeling. Data annotation is often too expensive to afford, which further underscores the importance of research in the data-centric field. In this section, we refer to all these efficient processing of data as “data-centric" approaches that will significantly improve the efficiency of training large-scale models. We review and study techniques from the following perspectives as presented in Table 2.

3.1 Data Regularization

Training on a large, high-quality dataset will result in better performance than on a small, poorly organized dataset. For individual researchers, it is often impractical to collect and process large data manually, and they need to choose some open source datasets and consider combining multi-domain, multi-lingual datasets to improve data diversity and thus improve the generalization ability of the model. In addition, studying efficient dataset regularization helps to improve the training performance and generality of large-scale models. Data regularization aims to reduce overfitting, make the model easier to adapt, and improve model generalization without affecting the training process too much. However, it usually introduces additional computational costs and sometimes becomes a bottleneck in the training process. For efficient training, it is crucial to select and test from a range of techniques and decide which one to use.

In the CV community, data augmentation (DA) is an important regularization strategy for efficient training. By making better use of the original dataset, such as transformed images, data augmentation artificially and effectively increases the training data to improve the model's generalization ability. Even some counter-intuitive methods of data augmentation are instead effective in improving the performance of the model [78]. Previous state-of-the-art models such as AlexNet [97], ResNet [65], and EfficientNetV2 [179] all use image augmentation techniques. There are many basic data augmentation methods, most of which are relatively simple, such as rotation, translation, shearing, flipping, cropping, resizing, adjusting contrast and brightness, adding noise, Gaussian blur, color jittering, format conversion, etc. In recent studies, basic erasing methods are proven to be effective, such as CutOut [37] which randomly cuts a square out of the input during training, and Random Erasing [220] which also erases a rectangle region of the image and fills it with random values or a mean value.

Many data augmentation techniques are based on the idea of Mixup proposed by Zhang et al. [213], which mixes up the pairs of features and their corresponding labels. For instance, Yun et al. [210] propose CutMix which cuts out a small patch of image and replaces it with another one. More recent studies include SaliencyMix [187], Manifold Mixup [189], StyleMix [73], TokenMix [114], Co-mixup [90], Supermix [30], and TransMix [18]. Some recent advances investigate automatic data augmentations [28], which uses a search algorithm implemented as a controller RNN in reinforcement learning to find data-driven augmentation strategies. Random combinations can be efficient. Hendrycks et al. [69] propose a data augmentation technique called *AugMix*, which randomly selects different augmentations and then mixes the augmented images. Cubuk et al. [29] propose RandAugment which randomly selects a certain number of methods from several image augmentation methods, and applies them sequentially for each sample. Further, a series of studies learn the efficiency of the improved label-aware techniques in video action recognition [1, 151, 154]. These brilliant works unanimously highlight that advancements in data-centric technologies.

In the NLP community, data augmentation is usually done before the training process in different ways. According to Feng et al. [47], some of the data augmentation techniques are: Rule-based techniques such as *Easy Data Augmentation* which is summarized by Wei and Zou [196], include randomly inserting, swapping, deleting, and synonym replacement. Among them, the synonym replacement methods are such as Zhang et al. [216] propose to randomly replace a word or a phrase with its synonyms ranked by semantic closeness, Wang and Yang [193] propose to create new instances by replacing a word with the results from searching for the k-nearest-neighbor (KNN) of the word in the embedding vocabulary. Example interpolation techniques that are pioneered by Mixup, for example, SeqMix [58]. Model-based techniques such as Back translation [43], which translate the text to another language. As LLMs show great abilities in language understanding, some recent studies are using state-of-the-art models for data augmentation, for example, ChatAug [31] use ChatGPT as augmentation tools to rephrase input sentence into more additional sentences.

3.2 Data Sampling

During training, updates are performed on batches of samples, and all samples within a batch are treated equally. However, this results in the training process spending more time on normal or less informative examples. To further speed up training, advanced data samplings can be adopted.

Importance sampling is a Monte Carlo method that involves sampling from a different distribution to help evaluate the properties of a certain distribution and reduce variance. In stochastic gradient descent, importance sampling focuses on the samples that have the most significant impact on the model parameters, which reduces the variance of the gradient estimates. This approach is useful to reduce computational costs when it is fixed or when the batch size is increased. It is important to choose an appropriate importance sampling distribution that accurately represents the original data distribution to avoid introducing bias. Katharopoulos and Fleuret [87] derive an upper bound for the gradient norm and propose an estimator of the variance reduction using importance sampling. Katharopoulos and Fleuret [86] propose that the loss value can be used as a metric for importance sampling instead of gradient norm. Jiang et al. [81] propose Selective Backpropagation which speeds up training by skipping the backward pass for training examples with low loss, thus using fewer samples and reducing backpropagation computations. They show that this method converges faster than vanilla standard SGD and sampled batches support a larger dataset for acceleration.

Chen et al. [22] propose to generate a mask on the parameters to freeze the backpropagation of the inputs. It retains the forward graph, and the statistical characteristics of all batches for the layers such as normalization maintain the mapping of the total dataset. Zhang et al. [214] focus on the worse samples during the training. They utilize a selector to highlight the training samples in a large batch that have larger gradients as a valid approximation. Samples with small gradients are considered "good points" in this epoch and the network needs to pay extra attention to the "bad points". Similar ideas are implemented by Kawaguchi and Lu [88], Ni et al. [133]. Kawaguchi and Lu [88] study the *Ordered-SGD* which directly selects the top-q samples in a minibatch to maximize the loss function to perform the update in each iteration. In each minibatch, k samples are selected according to the loss value for training, so that the small minibatch can achieve the competitive training effect of the large minibatch. He and Dube [63] simulate the decentralized framework with edge devices of different computational abilities. They remove the slowest device in the training alternately to directly reduce the delay time costs, which can be thought of as a variant of the dynamic batchsize approach. Xie et al. [202] work in the reduced feature space to make importance weight estimation tractable over the space of text with the *KL* reduction, which efficiently trains the language models with importance sampling. Sujit et al. [174] apply a similar idea to clip the input samples for reinforcement learning. Cai et al. [15] discard the dissimilar patches to achieve the same effect of reducing data in the version transformer models. The acceleration methods based on the sampling focus on directly reducing the total amount of data involved in the calculation. These studies indirectly verify the data redundancy in the training.

3.3 Data-centric Curriculum Learning

Curriculum learning [6, 44] is based on the intuition of starting with learning from simple tasks and gradually increasing the complexity of the task with some general features of data samples. This method has also been applied in various ways to pre-training and has shown potential efficiency in several works. Wu et al. [199] show that curriculum learning can improve performance when the training time budget is limited or when noisy data exists. In applying curriculum learning, one of the more important challenges is finding a suitable criterion to judge the difficulty of the sample.

Progressive Resizing [74, 85] is a technique that trains small down-sampled to large full images with the training process to improve model performance and time to convergence. For example, a set of suggested parameters is that the initial image is scaled to half of the full image, with increments of 4 pixels at a time during half of the training process, and the last 20% process is for fine-tuning. Touvron et al. [185] propose FixRes and show that training a network with a small resolution but inferring it with a higher resolution can result in a better generalization. Training with low-resolution images also helps to save training costs, when training with 224 and testing with 384 is better than training directly with 384. Wang et al. [194] introduce a clipping operation in the Fourier spectrum of the input image, which allows the model to learn from the low-frequency components first, as a form of curriculum learning, and therefore reduces training time. Koçyiğit et al. [94] adopt the progressive resolution training [179], 1-cyclic learning scheduling technique [169], and hard augmentation selection technique for training self-supervised learning tasks and achieve training speedup. For NLP tasks, the short/long sequences correspond to the low/high-resolution images. Press et al. [141] show that training transformers with a short subsequence first and then switching to a longer subsequence can achieve speedup. Ding et al. [39] find that training the machine translation model with a word-phrase-sentence data learning order could efficiently facilitate cross-lingual modeling, thus obtaining better translation. During the training process, we should keep the number of tokens and increase the batch size when using low-resolution images.

3.4 Discussion

In this section, we review efficient training techniques from a data-centric perspective. We focus on the data efficiency of dataset regularization including the augmentation and pre-processing to enhance its diversity, sampling the valid subset from a large batch (or entire dataset) to improve the training efficiency, and curriculum learning methods to alleviate the expensive consumption.

Data regularization is an effective way to expand the diversity of data samples without additional labeling. Because of the complicated and random transformations, the dataset is equivalent to being expanded multiple times, which helps train the large-scale models with better generalization performance. However, transforming data is likely to increase computation and I/O costs. In addition, overuse of data regularization introduces huge biases, resulting in poor quality. As more regularization methods are available, it is important to select effective combinations. In terms of data sampling, it can achieve the same or even better performance with small-batch training. Data sampling via well-designed algorithms can effectively increase the capacity of the training model. Usually, at the beginning of the training process, the model benefits more from the worse-performing samples. Ignoring good samples helps speed up the balance of learning the characteristics of different labeled samples. And, curriculum learning provides a progressive pipeline with general features. It allows feeding the low-resolution data with less augmentation to learn the simple cases and gradually enhances the data quality to capture fine-grained and detailed knowledge. Current studies show its promising performance of accelerations while maintaining efficiency. In the future, we think that promising studies from the data-centric perspective include:

- Efficient regularization implementation. More task-specific approaches to data regularization are worth studying. Designing novel data regularization methods based on prior knowledge can further improve the generalized performance of large networks. Meanwhile, adopting the parallel processing of data regularization can also greatly accelerate the training process.
- Efficient data sampling. Novel sampling methods are expected which jointly consider the training acceleration and generalization guarantees.
- Efficient label-aware techniques. Enhancing the label representation features of traditional data has also become one of the potential technological areas for efficient training [151].

4 Model-Centric Efficient Training

Designing efficient model architectures is always one of the most important research in the field of deep learning. An excellent model is an effective extractor and could be projected into high-level features that are easily separated. Different from other works with an extra focus on those efficient novel structures, our model-centric study pays more attention to the equality alternatives to the general modules, which achieve higher efficiency with comparable performance. Therefore, our review can provide good guidelines for efficiently training large-scale models. In this section, we investigate those efficient "model-centric" techniques which are summarized in Table 3.

4.1 Architecture Efficiency

The technique we review is to take efficient alternatives to accelerate calculations on the existing model modules. As the cornerstone of a large model, each basic module has different properties. In this section, we mainly study the efficiency of the following modules, which are widely used in the construction of large models. Our review clearly points to the efficiency of general acceleration techniques from the model-centric perspective.

4.1.1 Efficient Linear Layer. The linear layer is a basic module in the traditional models, which could be formulated as $H_{out} = HW + b$, where W is the weights and b is the biases. It involves underlying matrix multiplication, where fundamental optimizations can significantly save resources and speed up calculations. Besides, in deep training, plenty of works study its approximation, which can reduce the parameters and calculation complexity in practice. Li et al. [108] introduces a GPU-friendly butterfly factorization to split the multiplication as two recursive block diagonal matrices, which achieves comparable performance with only $1/r$ parameters. Theoretically, it reduces a $N \times N$ matrix into the product of $O(\log N)$ matrices with only $O(N \log N)$ operation and memory required. Meng et al. [127] propose the efficiently-computable and -invertible butterfly layers based on the butterfly factors, which contain a Jacobian determinant with less complexity. Song et al. [172] study a specific GPU-friendly sparsity on the MLP layer to accelerate the training. It adopts an approximate random dropout with a regular

Table 3. Overview of Model-Centric Efficient Training.

Category	Details
Architecture Efficiency	With the rapid increase of parameters in the deep models, it also introduces huge computational consumption. Therefore, implementing an efficient alternative to approximate the vanilla architecture becomes a trending topic. This substitution is not just an approximation of numerical calculations but also includes structural simplification and fusion in the deep models. In this part, we distinguish the existing techniques via architectures and implementations.
	Papers Included: [108], [127], [172], [177], [219], [25], [52], [27], [68], [93], [16], [147], [92], [26], [192], [198], [145], [32], [181], [71], [38], [212], [200]
Compression Training Efficiency	Compression has been a long research of computational acceleration, playing a key role in digital signal processing (multimedia computing/image processing). Traditional compression consists of two main branches: quantization and sparsity. We elaborate on their achievements.
	Papers Included: [191], [156], [36], [115], [40], [75], [54], [42], [61], [11]
Initialization Efficiency	Initialization of model parameters is a very important factor in both existing theoretical analysis and practical scenarios. A bad initialization state will cause the whole training to crash and stall at an early training stage, while a good initial state helps speed up the training within a smooth landscape. In this part, we mainly study the evaluation and algorithm design of the initialization.
	Papers Included: [129], [158], [98], [34], [217], [146], [14], [64]
Model-centric Curriculum Learning	From the model-centric perspective, curriculum learning usually begins to train a small model or partial parameters in the large-scale model and gradually recovers it to the entire architecture. It shows the favorable capability of accelerating the training process without significant negative effects. In this part, we review its implementation and efficiency in the training process.
	Papers Included: [5], [168], [55], [102], [57], [103]

and online row-based or tile-based dropout. This sensitivity-aware dropout helps to reduce the required input features in both forward and backward.

4.1.2 Efficient Convolution. Convolution is always an effective technology to extract high-level features for downstream tasks, especially in several computer vision tasks. It maintains the translational invariance and rotational invariance on the images which can efficiently capture detailed features. The general convolution in the deep models is: $H_{out}(x,y)=\sum_{i=0}^{h-1}\sum_{j=0}^{b-1}H(x-i,y-j)W^c(i,j)$, where (h,b) are the height and width of the input H (we consider the single feature map), W^c is the convolution kernel. It can be seen that convolutional calculations are quite repetitive, where a large number of matrix multiplications make them time-consuming extremely. Therefore, acceleration on convolution is always an important research topic in deep networks. Tai et al. [177] utilize the low-rank decomposition of tensors to evaluate the redundancy in the convolution filters and develop a new low-rank constraint on the parameters. It achieves a significant speed-up training and better performance than the vanilla convolution. Zhao et al. [219] propose a hybrid method that combines the Winograd minimal filtering and Strassen algorithm to dramatically reduce the computational complexity. It jointly reduces the required flops and the number of convolutions in the network simultaneously. Cheng and Parhi [25] study a novel fast convolution algorithm to remove the redundant multiplication operations at the controlled units on both the 1D and 2D convolutions. It also processes the input features and generates the outputs via a flexible block size which is independent of the kernel size. Ghimire et al. [52] summarize the mainstream techniques of accelerating convolution computing and hardware-efficient implementation.

4.1.3 Efficient Activation. The activation function is a handcraft function to element-wise scale the inputs to help the deep models learn the complex patterns and high-level features. Similar to a neuron-based mechanism in our brain, it ultimately determines what gets fired to the next neuron. It could be represented as $H_{out}=\phi(H)$, and usually $\phi(\cdot)$ is a non-linear function. Activation functions create nonlinear maps for deep networks which effectively improves the potentiality to fit the complicated objective functions in the scenarios. Since *ReLU* [60, 97] is proposed to simulate the bionics, deep neural networks have made great strides in many fields. A series of related optimization activation functions are also applied, such as *LReLU*, *ALReLU*, etc. As huge models are progressively designed to solve large tasks, more efficient activation functions are implemented. Clevert et al. [27] propose the *ELU* activation function which can be considered as a *Tanh* with two-side saturation. It solves the vanishing gradients and dead state problems. Hendrycks and Gimpel [68] study the *GeLU* to weight inputs by their specific values instead of signs. It contributes to improving the flatness of various large models. To further promote the stability of the data flow in the network, Klambauer et al. [93] develop the *SeLU* activation to be able to make the output into a normal distribution across all layers. As regards the benefit of the shifting means around zero value, Carlile et al. [16] design the *ISRLU* to save the high computation costs of the exponent function on the left side of zero. Raffel et al. [147] verify the efficiency of the *SwiGLU* activation in the FFN modules, which is an important component of the transformer. The finetuning coefficient can improve the distribution similarity of outputs across different layers. Appropriate activation functions can provide higher smoothness to the network on the different stages.

4.1.4 Efficient Attention. Attention has successfully developed the transformer models and achieved SOTA results on several tasks. The current general attention module is the multi-head attention (MHA), which employs many self-attention branches to focus on the different traits. We denote the H as the inputs vector and show its inference as: $H_{out} = softmax\left(\frac{Q_i K_i^\top}{\sqrt{d}}\right) \cdot V_i$ where $Q_i, K_i, V_i = HW_i^Q, HW_i^K, HW_i^V$. d is the dimension. In each head, each token will calculate an inner product with the others to generate the relationship value, and then normalize to a ratio within the range of $[0, 1]$ to get the intensity of attention. Therefore, by assuming total n tokens, it requires $O(n^2)$ computing complexity, which indicates it is generally time-consuming.

Before the introduction of the transformer architectures, the attention mechanism is implemented by complicated RNN-based encoder-decoder modules with the $O(n)$ complexity and sequential operation. Transformer models [188] study the alternative scaled-dot product modules to free the recurrence calculations and achieves a similar effect. Computational issues are less important on small tasks [51]. When the dimensionality of the data increases significantly, the self-attention module will play a key role and account for the dominant costs of training the transformer models. Therefore, a series of works are carried out around how to efficiently calculate the attention module. Kitaev et al. [92] adopt locality-sensitive hashing with the reversible residual layers instead of the vanilla dot-product attention, which reduces the complexity from $O(n^2)$ to $O(n \log n)$ in the *Reformer* model. Choromanski et al. [26] estimate the full-rank-attention under the provable *Performer* by the positive orthogonal random features approach to maintain the linear space and time complexity without relying on any priors. Wang et al. [192] propose the *Linformer* to approximate the self-attention by a low-rank matrix with $O(n)$ complexity in both time and space. Wu et al. [198] consider each token representation with global context instead of modeling the pair-wise interaction among the whole tokens and realize the efficient *Fastformer* under the theoretical linear complexity. Rabe and Staats [145] provide a practical and stable implementation for accelerating the self-attention modules into $O(\sqrt{n})$ memory costs. Dao et al. [32] consider co-designed methods with the hardware support and propose the *FlashAttention* implementation to practically accelerate the self-attention calculations on the wall-clock time, which has been assembled in many frameworks. As a survey in high-efficient transformers, Tay et al. [181] learn the linear attention advances.

4.1.5 Efficient Normalization. Normalization is an important technique to attenuate the covariance shifts in the models. When the initial state is bad, the beginning stage of the training will be affected significantly, which causes the instability of the gradients. Therefore, normalization is introduced to regularize the mean and variance of features: $\Phi(H) = \left(\frac{H-\mu_H}{\sigma_H}\right) \odot W + b$, where the W and b are the weights and bias respectively, and μ_H and σ_H is the statistical mean and variance of the inputs H. Their statistical calculations are determined by the normalization methods.

Traditional normalization employs the *BN* (batch normalization) to avoid the gradient diminishing in the backpropagation and it makes great progress in several CV tasks. Due to the batchsize becoming small in the NLP tasks, the *LN* (layer normalization) is adopted. In the current SOTA transformer models, it is still inherited this technique. Besides these, there are *IN* (instance normalization) and *GN* (group normalization). They differ from each other by which dimension to normalize the data samples. [71] propose the *GhostNorm* (ghost batch normalization), a batch normalization technique that divides the batch into smaller batches and normalizes independently them in parallel. [38] propose *SeqNorm* (sequential normalization) to normalize the inputs across two dimensions of the input feature maps. Zhang and Sennrich [212] expand the layer normalization to the *RMSNorm* (root mean square layer normalization) which assumes that re-centering invariance in the LN is dispensable and regularizes the summed inputs via their root mean square. It is computationally simpler and more efficient than vanilla LN. Xia and Ding [200] propose the *CoN* (collaborative normalization) to eliminate the domain discrepancy and accelerate the training.

4.2 Compression Training Efficiency

Compression is also one of the most mainstream acceleration methods at present. From the time consumption of the whole training process, most of the time is spent on inference (about 30%) and backpropagation (about 60%). A large number of studies have shown that large neural networks have redundancy in parameters and gradients in the training process, which means that the update frequency on several parameters can be appropriately reduced. In this part, we mainly focus on those methods that will maintain the total number of parameters in the whole training process.

4.2.1 Quantization Efficiency. The basic and fundamental rationale for quantization to accelerate the training is that the deep networks would consume less memory and computations if the parameters and inputs are converted to low-precision storage. However, there is always a trade-off between quantization and accuracy. A series of studies have investigated the link between quantization and accuracy, and propose how to preserve training accuracy.

At present, the common algorithms use the standard FP32 floating point type for the inference and backpropagation of the parameters by default. This data type can meet the high precision requirements in the current deep training. Correspondingly, the half-precision data type FP16 maintains the 5-bit exponent and 10-bit fraction and becomes a substitute for the FP32 in current training. To balance the efficiency and accuracy, *AMP* (auto mixed precision) is proposed to be widely used in deep training. It utilizes empiricism to use FP16 for some unimportant parameters and FP32 for other parameters to achieve the best performance, which is currently one of the most efficient acceleration techniques. It is challenging to use quantized models for the entire training process directly. Due to the limitation of representations on small values via quantization, direct training quantized models will lead to zero gradients too early and can not update small gradients. To avoid these issues, existing techniques typically adopt a compromise approach. Wang et al. [191] propose the *HAQ* framework to leverage reinforcement learning to automatically determine the quantization policy on the hardware accelerators. Ryu et al. [156] propose an efficient precision scalable accelerating structure with novel bit-wise summation and channel-wise aligning scheme. It significantly reduces the area overhead for the scaling and correctly fetches inputs and parameters from the on-chip SRAM buffers. Dettmers et al. [36] try the block-wise quantization with the 8-bit optimizer which could achieve comparable performance. Block-wise quantization splits inputs into several blocks which are independently quantized in parallel, yielding faster optimization and higher precision quantization. Realizing quantization methods, which can be effectively supported on the computation of hardware devices, is still one of the most important research directions.

4.2.2 Sparse Efficiency. Sparse computing requires the support of the hardware, otherwise, the acceleration cannot be achieved in the practical scenario. Liu and Mozafari [115] propose a novel gradient compression method *JOINTSPAR* for the large batch optimization via the layerwise adaptive learning rates. It drops the coordinates of the vanilla gradients by minimizing the combinations of gradient norms and parameters. Ding et al. [40] propose an effective method for convolutional networks to achieve reparameterization on the optimizer. It combines the convolution parameters between different branches in the parallel residual structure to achieve equivalent training. The reparameterized structure can compress the wider network into a narrow one while retaining the characteristics of the vanilla network. Huo et al. [75] decouple the backpropagation using the delayed gradients, and remove the backward locking via splitting the networks into multiple modules. This decoupled parallel mechanism save the waiting time on the chain of the gradient backpropagation. Goli and Aamodt [54] observe that over 90% of gradients can be reusable during the training process. Leveraging this phenomenon, they propose the *ReSprop* method to specify the gradient vectors by enabling the reduction in the backpropagation computations. To further match the fine-grained calculation of GPU devices, *ReSprop* introduces a generic sparse convolution accelerator. Dun et al. [42] consider the blockwise training methods and propose the *ResIST* to accelerate the ResNet. In each round, it trains a randomly generated sub-network and communicates a small portion of the parameters, and merges the results together at the end stage. Han et al. [61] propose the *Turbo* training for the transformer models on the action classification tasks and video-language representation learning. They allow to drop the tokens in the training to specify the input vectors and enable to split the long-schedule video frames into fragmented groups. Bolya et al. [11] are inspired by the idea of sparse inputs and introduce *ToMe* (token merging) technique to implement the sparsification of the inputs. Different from directly removing the tokens, *ToMe* allows the selection of several tokens and then merges the others into one token as the inputs.

In summary, the main approaches to the sparse training for acceleration focus on the following aspects: 1) *Sparse Gradients.* For parameters with small gradient values in an update, their offsets are also small. Compared with other parameters, pausing their updates will not hurt the training performance; 2) *Sparse Parameters.* Due to the parameter redundancy, pruning, and reconfiguration-based techniques to freeze partial parameters in the training could greatly save time and memory costs without reducing parameters; 3) *Sparse Inputs.* Due to the data redundancy, in a mini-batch of data, ignoring some specific samples could improve efficiency without accuracy loss.

4.3 Initialization Efficiency

Studies on model initialization indicate that arbitrary initialization strategies may lead to a slow convergence or even completely stall at the beginning of the training [129]. Therefore, proper initialization could do more with less for efficient training. Recent advances also indicate that a better initial state could help to understand the general knowledge of task-free. In this part, we mainly focus on the efficient initialization methods and the pretraining techniques.

Train-from-scratch Initialization. Model initialization plays an important role in efficiently training a model from scratch. Saxe et al. [158] propose that the proper scaling for the initial condition of the parameters matrix W between two linear layers corresponds to selecting the initial parameters from the zero-mean and $1/d$-variance Gaussian distribution under an orthonormal matrix, where d is the dimension of the matrix W. Kumar [98] study the empirical evidence on the impacts of activation layers. The functions, like hyperbolic tangent and sigmoid, which are differentiable at zero, will show more sensitive properties than those that are not differentiable at zero on the random initialization. Dauphin and Schoenholz [34] propose the *MetaInit* to search for the good state with an approximately linear landscape. It adopts several gradient descents to tune the norms of the parameters via minimizing the curvature information of second-order effects. Zhao et al. [217] propose the *Zero-Initialization* to initialize the model with only binary neural values. It adopts the Hadamard transformation to remove the whole randomness in the initialization which could be expanded easily to the sparse case. Current studies also shows its potential in large-scale models.

Pretraining-finetuning Initialization. Pretraining-finetuning mode is an efficient paradigm to achieve excellent performance. It comes from transfer learning which allows training the large-scale model on an extensive dataset to learn a good feature extractor. The pretraining phase could be considered as searching for a better-initialized state for the downstream tasks. Recent advances shine a dazzling light on pretraining. Radford et al. [146] greatly increases the model parameters and data samples in the pretraining phase with joint multi-tasks. They indicate that under sufficient data samples, the large-scale model has great potential for efficient training on the downstream tasks. Furthermore, Brown et al. [14] use an extremely large foundation model as the pretraining to test on the other tasks. It could achieve SOTA performance on some few-shot and even zero-shot scenarios. He et al. [64] also use a similar training strategy for CV models. Current trends reveal that pretraining provides an excellent initialization of the model for efficient training.

4.4 Model-centric Curriculum Learning

Model-centric curriculum learning mainly focuses on circumventing the training difficulties in large-scale models. It explores the small capacity for the large-scale models at the early stage of training and suggests reducing the number of parameters involved in the calculation accordingly. With the training process, the capacity of the model increases gradually and all the parameters are recovered for the entire training to learn the fine-grained knowledge.

The layer-wise curriculum learning could be considered as a development of the general greedy approach training with progressive learning [5]. Then, several studies begin to learn its practical implementation. Smith et al. [168] propose the *DropIn* to train the deep models from a shallow sub-net and then gradually add the new layers to maintain its efficiency. They demonstrate that model-centric progressive training works as a regularization term on the parameters. Gong et al. [55] study the unsupervised pre-training techniques and propose the *stacking* algorithm to transfer the fundamental knowledge from the well-trained shallow model to an enlarged model. In the transferring, a progressively increased stacking is adopted to accelerate the training of BERT. A similar idea is proposed by Li et al. [102] on the neural machine translation (NMT) task. Gu et al. [57] observe that it is beneficial to adopt the joint growth on multiple dimensions of the models. They propose to explore the alternative growth in each dimension instead of only the layers along the depth. Li et al. [103] relax the optimization problem of the progressive learning to a sub-network optimization process, which could be solved by an automated one-shot estimation. It also minimizes the searching overhead by recycling the parameters of the sub-nets. Model-centric curriculum learning achieves great success in large-scale model training. Designing sophisticated progressive schedules aligned to data samples could be a promising future study.

4.5 Discussion

In this section, We focus on how to achieve training acceleration from a general model perspective. We consider general approaches to accelerating from the perspective of the essential training process, including adopting alternative efficient architectures to reduce the computational complexity, compressing the tensors participating in the training procedures to consume less memory and computational costs, and efficient model initialization methods for fast convergence.

In terms of the model architecture, our focus is to review the efficient alternatives of the fundamental modules instead of explorations on designing efficient models, including the basic components of linear, convolution, attention, activation, and normalization. Utilizing efficient alternative calculations performs better on the generality and stability than reducing parameters. However, designing the equivalent alternatives is a very difficult task, which mostly focuses on the approximation of their calculations. Another mainstream solution is compression training. We focus on quantization and sparse methods that have proven effective in practice and are now widely used in large model training or are fundamental designs. Sparsity is a good implementation of accelerating training, while it requires rigorous hardware support to improve its practical efficiency. The parallel units of the GPU cannot directly apply the sparse calculations, which is the major flaw in the training process. To effectively integrate sparse training techniques, practitioners should carefully adjust the algorithm and sparsity settings, taking into account the performance capabilities of their training device and the hardware's support for sparse operations. Failing to do so could lead to suboptimal sparsity configurations that increase computation and I/O costs, or even result in poor final model accuracy. In terms of model initialization, a good initial state ensures the stability and efficiency of training large-scale models. To improve training performance, initialization techniques could be applied whenever training begins with an uninitialized set of model parameters. Recent advances also suggest that pretraining has great potential to achieve better performance. To alleviate the huge consumption in the pretraining, curriculum learning on the model level plays a key role. It allows the training process to begin with shallow models or partial parameters and then on full parameters. In the future, we speculate that important scenarios of accelerating training include:

- Efficient alternatives to the sophisticated modules. Large biases are easily caused by the accumulated errors of the accelerating alternatives with several individual layers. Considering some sophisticated modules as a whole to design their alternatives could shrink the errors.
- Efficient operator fusion. The combination of operators in adjacent steps can greatly improve computational efficiency and precision, e.g. for *FlashAttention* [32]. These designs make efficient use of the characteristics of the hardware and improve the throughput rate.
- Efficient model initialization. Trainable efficient initialization like pretraining, is a promising research direction in the future. Exploring the interaction of model initialization on different modules is also a worthy direction to study, which will provide valid guidelines to understand the essence of training large-scale models.
- Efficient progressive schedule. Designing the novel progressive schedule to guide the training process of large models along all dimensions, could be a promising study in the future. A key issue is exploring the relationship between model size and data size required for training, which may provide solid insights for the curriculum learning.

5 Optimization-Centric Efficient Training

Acceleration schemes for optimization methods have always been a key research direction in the field of machine learning. Reducing the complexity while achieving optimal conditions has always been the target pursued by the academic community. As the basic optimizer widely used in machine learning, *SGD*-type optimizers successfully help the deep models to achieve various practical applications. However, as the problem grows more complex, it is always more likely to fall into local minima and fail to generalize stably. In addition to the own performance of the designed optimizer, it is also important for the combination of accelerated training techniques. We summarize the current text on efficient training from the perspective of optimization in Table 4.

5.1 Learning Rate

The learning rate is one of the most important hyperparameters which controls the updates in training. It should be elaborately selected by jointly considering the optimizer and batchsize. From the perspective of the implementation of the adaptation policy, we distinguish them into three types, scaled coefficient, element-wised adaptivity, and matrix-based preconditioner. They adopt a single *value*, a *vector*, and a *matrix* to adjust the learning rate respectively.

Table 4. Overview of Optimizer-Centric Efficient Training.

Category	Details
Learning rate	Learning rate is an important hyperparameter in non-convex optimization and critical in deep network training. Adaptive methods like *Adam* and its variants, have made excellent progress on deep models and strategies for adjusting learning rates based on higher-order gradients.
	Papers included: [117], [167], [137], [94], [41], [211], [184], [91], [118], [208], [207], [203], [21], [35], [7], [59], [125], [50]
Large batchsize	Adopting large batchsize improves training efficiency. It directly reduces iterations required in each epoch. At the same time, processing a large batch also costs less than processing several small batches under the same total size.
	Papers Included: [171], [71], [126], [222], [160], [206], [204], [100], [143], [190]
Efficient objective	Vanilla ERM plays a key role in minimization problems enabling many practical tasks. Research on large networks pays attention to the gaps between optimization and generalization and proposes similar objectives. They interpret the importance of generalization from different perspectives.
	Papers Included: [162], [89], [20], [56], [48], [3], [128]
Averaged weights	Weighted average enhances models' generality, considering the weighted average of historical states with a set of frozen or learnable coefficients.
	Papers Included: [215], [161], [76], [83], [106], [82], [107], [150]

5.1.1 Learning Rate Scheduler. The Learning rate scheduler usually adopts a single decaying coefficient, which gradually reduces it to learn the fine-grained knowledge of the dataset. Usually, we use epoch as a unit to adjust the change in the learning rate and we can write it as $\gamma = \gamma_0 \cdot \rho(t)$, where γ_0 is the initial learning rate, and $\rho(t)$ is a scalar function. The common scheme is staged-wise decaying with a constant bound, where $\rho(t) = \rho_0^{\lfloor t/t_s \rfloor}$. t_s is the pre-defined decaying stage, which works well in many tasks. However, it fails to perform well on the large models. Loshchilov and Hutter [117] demonstrates the negative impact of the simple exponential stage-decayed learning rate. The rapid decrease in the learning rate at the mid-stage of training results in an eventual inability to escape from the local minimum. To further improve its performance, a *CLR* (Cyclical Learning Rates) [167] policy is proposed to alternately update within the maximum bound and minimum bound. It greatly enhances the generalization performance through periodic adjustments. Pan et al. [137] propose a modified *ESD* (Elastic Step Decay) learning rate scheduler to improve the training for Berts. It adopts a relaxed selection of the learning rate and drives centralized training. Koçyiğit et al. [94] adopt a scaled 1-cycle learning rate to match the training process with a refined momentum coefficient $\beta_t = \beta_t + 0.5\,(\beta_h - \beta_t)\,(\cos(t\pi/L) + 1)$, where β_h is the maximum β.

5.1.2 Adaptivity. Adaptive learning rate focuses on the element-wised adaptation for the entire gradient, which considers that every single element of the gradient is related to the objective with different importance. It contributes to a more accurate and precise controller of the learning rate and works in wide domains. A variety of adaptivity-based optimizers are designed to improve training efficiency. The general formulation could be written as $\gamma = \gamma_0 \odot v(t)$, where $v(t)$ is a vector with the same dimension as gradients. Traditional adaptive methods includes *AdaGrad* [41], *Adadelta* [211], *RMSprop* [184], and *Adam* [91], etc. They utilize the second-order momentum to refine the learning rate. The $v(t)$ function considers that if one parameter has been updated largely in the past, it should be updated less than other parameters in the future, which contributes to balancing the local gradients not to dramatically change.

In recent advances, Loshchilov and Hutter [118] propose the *AdamW* which considers both the gradient and the weight decay regularization as the momentum term. It has been a great success in transformer training. Due to the chain rules of the backpropagation, gradients have obvious stratification characteristics, which causes the instabilities from the imbalance between the gradient norm and the weight norm of certain layers. Thus, You et al. [208] uses the layer-wise regularization *LARS* to facilitate a more accurate scalar on every single layer.

Similarly, You et al. [207] use this technique on the *Adam* as the base optimizer and achieve higher performance. To further normalize the l_2 regularization term, *AdamW* is proposed to reduce the impact of differences across the parameters of each layer. Xie et al. [203] propose the *Adan*, the nesterov momentum to accelerate the vanilla *Adam* optimizer, and implement a faster training. It transfers the vanilla gradients into the inertial-based gradients to approximate the direction of the next state. Recently, Chen et al. [21] propose the *Lion* (evolved sign momentum) optimizer, which highly accelerates the training speed. It comes from a policy search within several pre-defined mathematical function pools.

5.1.3 Precondition. Preconditioner finetunes the learning rate by a matrix. Beyond the dominant of the first-order gradient methods, the preconditioner involves the high-order derivatives, mainly including the Hessian and Fisher Matrix (FIM). In this survey, we generally consider the following adaptation on the learning rate: $w = w - \gamma P(w)\nabla F(w)$, where $P(w)$ is the preconditioner matrix w.r.t. w. Classical Newton's method adopts the inverse of the Hessian as $P(w) = H^{-1}$. However, due to the enormous computational complexity, it is hard to directly benefit from the curvatures. Therefore, the quasi-Hessian methods are proposed [7, 35]. They construct the update of the hessian with a Newton function to iterate the quasi-hessian matrix, which reduces the most complexity. To further implement the efficient preconditioner, researchers learn the FIM, which could be proved as the expectation of the Hessian on the log-likelihood. It maintains high efficiency with comparable calculations with the first-order gradient. Gupta et al. [59] propose the *Shampoo* method to approximate the FIM tensor as the multiplication of the 3-dimensional matrix. It uses the Kronecker products of two FIM information, which could be calculated in block-wise structures. Similar work like *K-FAC* [125] proposes to employ the multiplication of the expectation of the inputs and gradients respectively instead of the expectation of their multiplication. Frantar et al. [50] study the approximation by its definition which records the adjacent difference of the gradients and calculates their product. The preconditioner matrix is estimated as a sum of several rank-one matrices via a pre-computation process and inverse alignment.

5.2 Large Batchsize

Appropriately adopting large-batch training could reduce time consumption, improve the utilization of storage, generate smooth gradients, and maintain training stability. However, training with excessive batchsize is counterproductive which may dramatically damage its generalization and even lead to divergence, especially on large-scale models [100, 143, 190]. Therefore, it is important to learn how to train with large batchsize while ensuring performance [160, 204, 206, 222]. To discuss the batch size term in our equation, we will focus on mainly how to enable large batch-size training. We consider the iterative step of minibatch gradient descent with batch B. $B_t \subset B$ is a minibatch sampled (generally uniform sampling) at the t-th step from an entire training dataset B. To achieve training with large batches and avoid poor generalization performance, some existing solutions focus on jointly finetuning the learning rate and batchsize, including linear scaling of the learning rate as the batch size expands, adopting the warm-up of the learning rate at the beginning stage, and gradually increasing the training batchsize. Smith et al. [171] show that increasing the batch size naturally leads to similar results as the expeditiously decayed learning rate, which could reduce the unexpected errors in the parameter updates, facilitate its parallelism, and achieve higher training efficiency. Hoffer et al. [71] show using specific normalization to control the covariance shift in training could help to stabilize the training and reduce the gap between generalization and optimization. McCandlish et al. [126] propose an empirical statistic called the gradient noise scale to predict the largest batch size for the model. Consider a model parameterized by $w \in \mathbb{R}^d$, let G denote the true gradient, H denote the true Hessian at w, and Σ denote the per-example covariance matrix, the gradient noise scale $\mathcal{B}_{\text{noise}} = \frac{\text{tr}(H\Sigma)}{G^\top HG}$, simplified as $\mathcal{B}_{\text{simple}} = \frac{\text{tr}(\Sigma)}{|G|^2}$ if under the assumption that the optimization is perfectly well-conditioned (Hessian is a multiple of the identity matrix), and it predicts the critical batch size $\mathcal{B}_{\text{crit}}$. The gradient noise scale is defined as scalar to balance the trade-off. The current theory proves that batchsize negatively impacts the stability and the generalization error bound in the training [100, 101, 112], which means that larger minibatch results in poorer model generalization performance. And its selection is a balance between convergence and performance. Finding ways to achieve smaller generalization errors under large batchsize will become an important area in the field of large batch training.

5.3 Efficient Objective

The objective is always one of the most important factors in the optimization, which is a principle we want to optimize for the tasks. Optimization objectives directly determine the ultimate targets and training efficiency. In this part, we review the optimization-centric acceleration techniques which benefit from the considerations on designing novel objectives. Instead of the narrowly defined objectives, we summarize the methods which aim to construct the vector G in equation (2).

Empirical risk minimization (ERM) is a principle in statistical learning theory that defines a family of learning algorithms. However, it does not prevail in balancing the optimization and generalization errors. Because the optimization objective and the expected target in practice are biased, the ERM is with low tension and lacks precision on the representation of generality. Therefore, a series of amended objectives are proposed as the global target jointly considering both the efficiency and stability [162]. Khosla et al. [89] adopt a *LARS* selection on the supervised contrastive learning on different modules to stabilize the training process and it achieves a better performance on the modern batch contrastive approaches compared to the traditional contrastive losses such as triplet, max-margin, and the NN-pairs loss. A similar idea can be found in [20]. Goyal et al. [56] try a self-supervised pretraining mechanism on the uncurated ImageNet dataset. The *LARS* technique makes the huge models surpass the best self-supervised pre-trained models. Foret et al. [48] propose *SAM* (Sharpness-Aware Minimization), which simultaneously minimizes loss value and loss sharpness to improve generalization. It reveals that vanilla ERM leads to a sharp minimal with low generalization performance. Therefore, *SAM* considers optimizing the worst point within the ball at the current state in each iteration. Jointly considering this with its sharpness, it searches for the flat landscape of the optimized state. Andriushchenko and Flammarion [3] propose an understanding of SAM and propose a variant that updates with both the SGD and SAM gradient. Mindermann et al. [128] introduce the *RHO-LOSS* (reducible holdout loss selection) to accelerate the training on the web-scale data, which focuses on the optimization of the holdout dataset. It estimates the optimization objective of generalization awareness through likelihood estimation and further accelerates the training process with simple additional forward computations. Furthermore, the forward phase also can be supported by a simple and small networks to save costs.

5.4 Averaged Weights

Model averaging is an important technique to generate a generalization-efficient model and share knowledge via several historical states. The generated models usually show higher generalizations than the trained ones. The averaged model does not require training and is tested only, which introduces no extra costs on the calculations and memory utilization.

Scieur et al. [161] propose the *RNA* (Regularized Nonlinear Acceleration) to aggregate the historical states via a set of optimized weights which are solved by a minimization on a regularization term. It searches for the scaled coefficient for every single state and merges them as the next initial point. *EMA* (Exponential Moving Average) is a simple model merging technique that uses an exponentially decaying weight to calculate the weighted average of the parameters at each previous training step. It significantly improves the test accuracy in the middle stage of the training. Izmailov et al. [76] propose the *SWA* (Stochastic Weight Averaging), a model averaging technique that captures model weights according to a constant or cyclical learning rate and maintains an average of the weights to reduce the generalization error. Compared with the *EMA*, *SWA* remembers all the historical states and averages them, which provides a flat minimal. They show that for Top-1 Accuracy with ResNet-50, ResNet-152, and DenseNet-161 on ImageNet, it provides about 0.6-0.9% improvement over the pre-trained models. Li et al. [106] propose the *TWA* (Trainable Weight Averaging) to further improve the generalization performance, which is based on the projection of a set of orthogonal bases. It works in a reduced subspace spanned by several historical states which are usually frozen at the beginning of the training process. It largely reduces the approximation error in *SWA*. It is easy to implement and could be flexibly plugged into different parts of training. Kaddour [82] study a truncated moving window with a fixed size to average the historical states, which is named *LAWA* (Latest Weight Averaging). It indicates that only the last few states help to generalize well. The selection of the states could not be the adjacent points one by one, and it selects each state after a total epoch training. When the model stabilizes, the improvement of *LAWA* is excellent with usually higher acceleration. Model average helps to approach a set of better-generalized states during the training process [83, 150, 215]. Although such methods are not directly involved in optimizing updates, their significant improvements in the generality help to early stop the training with the comparable parameters [107]. It dramatically reduces the required training epochs. In the field of efficient training, it is a promising study in practical applications.

5.5 Discussion

In this section, we review the general acceleration techniques from the optimization-centric perspective. We mainly focus on the practical applications of selecting an efficient learning rate to accelerate training, adopting large batch training, designing efficient objectives, and utilizing the model average techniques. These techniques imply two mainstream scenarios of accelerating the optimization process and increasing the generalization performance.

Recent advances in the selection of the learning rate indicate that adaptivity is the right hand to maintain high efficiency with acceptable extra computational costs, which benefits from the element-wise scalars on the different dimensions of gradients. One major problem to note is that scalars of squared momenta may cause a sharp diminishing of the learning rate, yielding much slower convergence. The application of a preconditioner is a promising approach to introducing the correction of the curvature which may lead to fast convergence. However, it suffers from the expensive costs to update the preconditioner in practical training, especially on large-scale models. Meanwhile, inaccurate curvature direction may mislead the optimization into the local minimum. Large batch training shows a strong dependence on the selection of hyperparameters. The solution to efficient large batch training usually adopts adjusting other hyperparameters or designing novel objectives. These two routes focus on the acceleration of the optimization process, which encourages achieving a better convergence. The other scenario pays more attention to improving the generalization efficiency. Many previous works have studied the gaps between optimization and generalization, and propose to search for the appropriate state with a flat loss surface. Several objectives based on the valid measurement strategies for flatness are also proposed to enhance the generality of the large-scale models and achieve great success. Compared with objective design, model averaging is a straightforward and effective method to share and transfer knowledge. The average can achieve effective acceleration without introducing any additional computational costs and shows great potential in the middle stage of training. And its advances eventually disappear. In summary, we perceive that future developments from the optimization-centric perspective include:

- Refined adaptive learning rate. Further balancing the biases caused by erratic gradients when training large-scale models help to achieve higher efficiency. Adopting curvature information to construct low-cost preconditioner with fine-grained training is promising.
- Valid measurements on the generality of the training models. Several measurements are adopted to evaluate the local stability and flatness, e.g. top eigenvalue of hessian, gradient norm, the worst state in the

neighborhood, and achieve great efficiency in the training accelerations. Exploring low-cost representations helps to refine the training process.

- Training involvement of averaged models. The averaged model with good generalization performance can be considered a better initial state for the next iteration. Designing better optimization algorithms to make full use of this aggregation state shows potentials.

6 Budgeted Efficient Training

Recently, several works focus on training deep learning models with fewer resources but achieving higher accuracy as much as possible. This type of problem is defined as budgeted training, i.e., training under the given budget to achieve the highest model performance [104, 137]. To systematically consider the hardware support to get closer to the real situation, we define budgeted training as training with given devices and limited time to obtain the model with the best performance. In this section, we mainly review the advances in the budgeted training, summarized in Table 5.

6.1 Techniques for Budgeted Training

In this part, we primarily summarize the approach to budgeted efficient training from the perspectives of data, model, and optimization. Then, we introduce the summary of efficient techniques adapted for budgeted training for the large-scale training.

Data. From the data perspective, budgeted training focus on the budget allocation regarding the dataset. Kaplan et al. [84] propose that there exists an optimal budget allocation between the model size and the amount of data processed in training. They demonstrate that within a fixed budget, the optimal model is obtained by stopping early before convergence. Hoffmann et al. [72] show different results and propose that given the compute budget, the number of data tokens processed in training should be scaled equally. For limited data scenarios, Chen et al. [19] propose a data-efficient GAN training method that identifies trainable sparse subnetworks ("winning tickets" [49]) from the original GAN using the small training set of real images. This subnetwork is then further trained using the same limited dataset with data- and feature-level augmentation techniques. In addition, Zhao et al. [218] propose Differentiable Augmentation (DiffAugment) for GAN training, which applies differentiable augmentation techniques to both real and generated samples.

Model. As we mentioned, given a compute budget, the optimal number of model parameters and the amount of data can be derived [84]. Geiping and Goldstein [51], Izsak et al. [77] also demonstrated that training large models rather than small ones is suitable for low-budget situations. Similar findings are supported by Li et al. [109], who propose a training strategy that trains a larger model (wider and deeper) and stops early, then compresses it heavily with quantization and pruning. Moreover, Wei et al. [195] show that scaling up can bring additional capabilities for large language models. When the model is fixed, it is still possible to find and optimize effective architectures to accelerate budgeted training. Izsak et al. [77] investigate the pretraining of a BERT-like masked language model in 24 hours using a single low-end deep learning server by a training recipe of sparse token prediction [116]. With the training budget on an RTX 2080Ti for 24 hours, which is similar to the budget set by Izsak et al. [77], Geiping and Goldstein [51] demonstrate that removing biases from the QKV projections of the attention block can improve the computation efficiency.

Optimization. Izsak et al. [77] show several important hyperparameters to optimize for higher performance in budgeted training, including batch size, peak learning rate, warmup proportion of the learning rate, and the number of days for training. The hyperparameters in budgeted training are configured differently than in the large-scale framework and need to be adjusted to fit a fixed budget. To find the best hyperparameter configuration, it may be necessary to perform several tests with different hyperparameters within the estimated hyperparameter

Table 5. Techniques to improve speed and availability for budgeted training.

Centrics	Aspects	Methods	Benefits
Data	dataset	short sequence	budgeted computation and memory
	dataloader	disk serialization	budgeted memory, save RAM
		sharding offline and loading into RAM [77]	avoid disk I/O
Model	model architecture	diabling QKV and linear biases [51]	computation efficiency
		PreNorm [77, 166]	stability, large learning rate
		Gated Linear Units [51, 165]	improving performance
		removing nonlinear attention head [51]	computation and memory efficiency
		complexity schedule [138]	efficiency
	model size	training large models	efficiency
Optimizer	learning rate schedule	linearly decaying to zero [104]	fast budgeted convergence
		one-cycle learning rate [169]	fast convergence
		Elastic Step Decay [137]	pretraining acceleration
	large batch size	gradient accumulation	computation efficiency
		gradient checkpointing	budgeted memory
		aggresive batch size schedule [51]	progress in early training
		importance sampling	computation efficiency
	weight averaging	latest weight averaging [82]	speedup
	others	sparse token prediction [51, 77, 116]	memory efficiency
		gradient clipping	stability
		disabling dropout during pretraining [51]	computation efficiency
		early stopping	efficiency

range. In addition, several studies focus on efficient training schedules. Li et al. [104] propose the budgeted training setting and show that a new learning scheduler (vanilla linear scheduler) could be adopted to improve the performance, particularly for small budgets. Chen et al. [17] propose the combination of a new function of the learning rate schedule and a sampling rate to update the learning rate by sampling from the given function. They show that their schedule outperforms the linear.

6.2 Discussion

In this section, we have reviewed efficient training techniques from the perspective of budgeted training. We summarize techniques for efficient budgeted training in terms of data, model, and optimization, respectively. Several systematic studies that examine various training configurations provide empirical recommendations for deciding on the trade-offs in practice and have suggested effective budget allocation schemes [72, 84]. These laws may still require future work to train more models to validate them in new settings, which will incur more costs than following the configuration of existing model training to get good results. It starts with small-scale, low-budgeted training setups to decide on an efficient budgeted training recipe and then scales up to larger scales. Another valuable idea is to design more budget-aware training components, i.e., adjusting training components according to the remaining budget. In summary, we recommend that future research from the budgeted training perspective could include:

- Practical benchmarks. It is possible to study an identical and practical budget benchmark, e.g., a day of training on a low-end GPU server [51, 77]. It is too expensive for researchers to conduct a systemic study

under different configurations (such as [72, 84, 173]). Instead, for the same practical budget, future work can also follow the existing training recipe.
- Budget-aware schedulers. Algorithms such as efficient learning rate schedules [17] can be tuned to the remaining budget, which facilitates fast convergence and improves performance. More techniques with dynamically supports to the remaining budget are still to be developed.

7 System-Centric Efficient Training

Researches from the system-centric perspective provide the specific implementation methods for the designed algorithms. It studies the valid and practical executions of the hardware which can truly achieve efficient training. In our survey, we focus on the implementation of general computational devices. Solving potential conflicts in the designed algorithms from the perspective of hardware is the core concern. In this section, we mainly review the hardware implementation techniques in existing frameworks and third-party libraries, which efficiently support the processing of data, models, and optimization. Then we introduce some of the existing open-source platforms.

- **System-centric Data Efficiency.** Efficient data processing and data parallelism are two important concerns in systematic implementation. Designing more hardware-friendly computing methods and parallelization can reduce time wasting in the data loader.
- **System-centric Model Efficiency.** Storage and computational efficiency of large-scale models bring great challenges to hardware implementations. In this part, we mainly review how to achieve efficient I/O of deployment and streamlined implementation of model parallelism.
- **System-centric Optimization Efficiency.** The optimization process represents the backpropagation and update in each iteration, which is the most time-consuming calculation in the training. Therefore, the implementation of system-centric optimization directly determines the efficiency of the training. We mainly focus on the efficiency at different computing stages.
- **Open Source Frameworks.** We investigate and analyze a large range of open-source frameworks which facilitate training as a bridge to grafting software and hardware.

7.1 System-centric Data Efficiency

Although data processing does not have a high proportion of time costs in training, inefficient handling still leads to drastic increases in training time. Especially when the large batch is trained on the large-scale model, rational use of non-conflict processes to accelerate data processing can effectively reduce additional consumption. A lot of work has been proposed at the system levels.

Efficient data processing can bring significant benefits to accelerated training, which achieves high-efficiency training and saves redundant wait time in the system. Bai et al. [4] propose the *PaGraph* to support general and efficient sampling-based graph neural network training on multi-GPU devices. It exploits available GPU devices to maintain the frequently-accessed graph data with cache storage. Qi et al. [142] introduces *SpeeChain*, an open-source PyTorch toolkit to develop the large-scale machine speech chain application. It supports both the multi-dataloader batch generation and on-the-fly data selection techniques. Jia et al. [80] design the *A-Dloader* to dynamically allocate the CPU resources to concurrently run multiple programs and adopt the dataloader allocation as the local valve to reduce the visiting intervals of the GPU devices, which improves the overall efficiency. Svogor et al. [176] evaluate several benchmarks to outline the performance issues within certain steps of the dataloader pipeline and modify the *ConcurrentDataloader* to further improve the GPU utilization. It supports the efficient training of cloud-based datasets. Xie et al. [201] focus on the overlap datasets executed by multiple training tasks in parallel. They propose a dependent sampling method *JOADER* supported by the domain-specific cache policy and a novel tree structure for the loading data. Dash et al. [33] develop a solution for supercomputers to alleviate the low loading efficiency of the small patches in the very high-resolution images, extreme sparsity of the notations, and I/O latency on the disk storage. Leclerc et al. [99] design the novel structure to store the image sample in the *FFCV* as the *.beton* file with four major sections of the header, data table, heap storage, and allocation table. They recombine and compile each section in the whole process to make full use of the cache structure. To further accelerate basic matrix calculations, NVIDIA proposes the Data Loading Library (*DALI*) to support fast, portable, and flexible data loading.

7.2 System-centric Model Efficiency

Hardware design and algorithm design are complementary processes. As for the researchers, based on their understanding of the model, they can optimize the algorithms for computation and memory based on the memory, computation, and communication features of the device, e.g. CPUs and GPUs.

Channel Last is an efficient storage format implemented as an alternative to 4D *NCHW* tensors in the training. Under mixed precision training, models trained in channel-last format on a tensor core can increase training throughput [46]. Another efficient method is adopting quantized low-precision to store and compute the tensors. Recent advanced hardware designs have introduced more support for sparsity, such as NVIDIA's Ampere GPU architecture that introduces support for sparsity in its matrix math unit, i.e., the tensor core, which utilizes a 2:4 (50%) sparsity pattern [130], enabling twice the mathematical throughput of dense math. Structured pruning-based approaches leverage sparsity in hardware in CPU and GPU architecture, and can benefit the up-scaling of the models [79]. To achieve structured sparsity to fit the hardware, Ma et al. [121] propose Hardware-friendly Regrouping towards Block-based Pruning (HRBP), which preserves the extracted dense blocks in backpropagation. The column-parallel linear layer requires the application of an all-reduce across processors in the backward pass to aggregate the split gradient of the input tensor. For example, for $Y = XA$, all-reduce is applied to obtain $\dot{X} = \dot{Y}A^\top = \dot{Y}_1A_1^\top + \dot{Y}_2A_2^\top$. The hardware support from the model-centric perspective is mainly aimed at storage and calculations. Efficient storage allows the model to save a lot of time and achieve acceleration during transferring and allocating of devices. Some of the underlying approaches based on module-specific or layer-specific optimization also contribute significant benefits, which can provide a solid foundation.

7.3 System-centric Optimization Efficiency

The optimization corresponds to the backpropagation and parameter updating process, which is the largest time-consuming calculation in the training. It is critical for the accelerations. The traditional paradigm has relatively limited operational space in the optimization process, influenced by the backpropagation mechanism. Therefore, one of the main determinants of training efficiency is the ability to achieve an efficient underlying optimization process.

Pipeline parallelism is an efficient parallel technique that decomposes incoming batches into mini-batches (which reduce the pipeline bubble), and divides the layers of the model across multiple GPUs, thus creating a pipeline where each stage, i.e. a set of contiguous layers takes the results of the previous stage as input and passes the results downstream, allowing different GPUs to participate in the computational process in parallel. It has the lowest communications and can be scheduled across nodes. State-of-the-art system [170] use a combination of reduced communication and increased locality and parallelism to improve training efficiency. Narayanan et al. [131] propose a scheduling mechanism one-forward-one-backward (1F1B). Fused implementations adopt the high efficiency of reusing the computational units which improve efficiency by combining operations.

Efficient communication is extremely important in training accelerations, especially in large-scale training. Communication bus efficiency across multi-nodes becomes the main bottleneck. Even in a single node, extensive communications between the CPU and GPU still lead to a serious obstruction. Existing algorithms and systems can be optimized to improve communication efficiency. *1-bit compression* [163] reduces the communication by quantizing gradients to as low as 1 bit per value. Tang et al. [180] propose *1-bit Adam* which works with the non-linear gradient-based optimizer *Adam*. Lu et al. [120] propose *0/1 Adam* to address the non-linearity challenges in *Adam* when applying aggressive 1-bit quantization and local steps, achieving higher training throughput and end-to-end training time reduction compared to the *1-bit Adam* baseline. Markov et al. [124] propose a communication framework named CGX, which supports compressed communication for data-parallel training. In addition, better parallel strategies for efficient communication can be explored, including data parallelism, tensor parallelism, and pipeline parallelism. Narayanan et al. [132] show how to compose data, tensor, and pipeline parallelism to scale to thousands of GPUs.

Rajbhandari et al. [148] propose the Zero Redundancy Optimizer (ZeRO), which is currently important technique for training large-scale models. ZeRO optimizes redundant model states (i.e. optimizer states, gradients, and parameters) in memory by partitioning them in three corresponding stages across processors and optimizing the communication, with the final model states evenly split on each node. On the second stage (ZeRO-2) which partitions both the optimizer states and the gradients, ZeRO-Offload is built, which offloads the data and computations of both states to the CPU, thus leveraging the CPU to save the GPU memory. Further, Rajbhandari et al. [149] show ZeRO-Infinity, a heterogeneous system technology that leverages CPU and NVMe memory in parallel across multiple devices to aggregate bandwidths.

7.4 Open Source Frameworks

For researchers who want to apply parallel techniques to train large models simply, various existing open-source frameworks provide convenient implementations. Here we introduce some other platforms, open-source frameworks, and libraries for the large scale training.

JAX [13] is a library for machine learning which combines Autograd (an automatic differentiation library) and XLA (Accelerated Linear Algebra, a compiler for linear algebra), and provides the same API as NumPy that is compatible with GPU and TPU. DeepSpeed [152] is an open-source PyTorch library for large model training, which provides Zero Redundancy Optimizer (ZeRO) [148, 149, 153], 3D-parallelism, etc. and is used in BLOOM (176B) [159] and MT-NLG (530B) [170] training. Meta Research propose FairScale [45] is a PyTorch extension library that provides distributed training techniques to improve memory usage and speed up the training process. FairScale also includes some optimizers and schedulers to improve stability and training efficiency. The PaddlePaddle platform [122]is a framework designed for parallel distributed deep learning, supporting various deep learning hardware. Related development Kits and the series of work followed by ERNIE [175] cover a lot of training techniques. Megatron-LM is the large transformer lib developed by NVIDIA, with a efficient training framework. Also, NVIDIA maintains the Apex [134] library of tools for mixed precision and distributed training. MosaicML's Composer [183], a PyTorch library for efficient training which creates an interface for each part of the training process, allowing composing various speedup methods both built-in or customized into the training pipeline to develop efficient training recipes. Colossal-AI [8], which provides a series of parallel components to build and train distributed AI models. In addition to the 1D tensor parallelism mentioned above, the Colossal-AI [8] platform also integrates 2D, 2.5D, 3D tensor parallelism strategies [9, 205] and sequence parallelism [105] that distribute computational and memory load and optimize communication costs further. EleutherAI GPT-NeoX [2], a library for training large-scale language models on which a 20 billion parameter model GPT-NeoX-20B [10] is built. OneFlow-LiBai [110], which is a large-scale open-source training toolbox that provides 3D-parallelism distributed training and other training techniques. Horovod [164] hosted by LF AI & Data Foundation, a distributed deep learning framework. In summary, we organize Table 6 to summarize the benefits that each framework library and clearly show their main contributions to efficient training.

Table 6. Open-source frameworks for large-scale efficient training.

Hosters	Libraries	Benefits
PyTorch	PyTorch 2.0	*torch.compile*, mixed precision, DDP, and FSDP
Baidu	PaddlePaddle	3D Parallelism, Mixed Precision, GroupSharded (ZeRO-like)
ByteDance	LightSeq	mixed precision, fused operators, GEMM optimization
Google	JAX	Autograd, XLA
MicroSoft	DeepSpeed	3D parallelism, ZeRO, mixed precision, data efficiency library
	TorchScale	scaling up Transformers
MetaResearch	FairScale	FSDP, pipeline parallel, general techniques, Adascale
	Fairseq	FSDP, general techniques, model implementations
	xFormers	Transformer model implementations
MosaicML	Composer	multi-GPUs training, implementations of speedup methods
Huggingface	Transformers	model implementations
	Accelerate	distributed training, Mixed Precision, support of FSDP, DeepSpeed
HPC-AI Tech	Colossal-AI	3D(especially for tensor) parallelism, sequence parallelism, ZeRO
OneFlow	LiBai	3D parallelism, mixed precision, ZeRO, general techniques
NVIDIA	Megatron-LM	distributed pretraining, model implementations
	Apex	AMP, distributed training, fused optimizers, fused layers, and layer norm

7.5 Discussion

In this section, we review the training accelerations from the system-centric perspective. The system implementation carries the algorithm design which determines the training efficiency in the practical scenarios. The efficiency of the designed algorithms and techniques also relies heavily on sound hardware support for maximum performance.

In terms of system-centric data processing, the key routes include high-performance data processing and data parallelism. As the volume of the dataset grows rapidly, the pre-processing of the training data samples becomes complicated and onerous. This is very detrimental to the development of large batch training. When data processing time does not match training consumption, the training process goes into a waiting state, resulting in very low efficiency. More and more research tends to reduce communication and maintain the localization of data processing, yielding highly efficient processing and parallelism. The same idea has been applied at the model level. Different from data processing, models are usually stored on computing devices. Therefore, we have to face the challenge of the deployment of large-scale models. A range of quantitative compression-based techniques and mixed precision models effectively guarantee the training accelerations. In addition, in large-scale training, the optimization of distributed architecture also helps to improve the computational efficiency of the models. In addition to data processing and model deployment, hardware support for the optimization process is the most important part of practical training. Current mainstream technology routes revolve around the efficient calculations on the reconstructed computational graph, pipeline parallelism for stage-wise training, fused implementation of the efficient optimizers, and efficient communication across the multi-nodes and multi-devices.

Despite the widespread success of gradient backpropagation methods in various application fields, this monotonous simulation does not accurately correspond to the workings of the human brain. Recently, a lot of attempts have been made to get rid of backpropagation and have shown more promising high efficiency. Inspired by the biological neural network, Löwe et al. [119] split a large network into a stack of gradient-isolated modules which achieve comparable performance. Similarly, Kohan et al. [96] propose the error forward-propagation technique based on a biologically plausible mechanism instead of backpropagation for the implementation of the forward pass [95]. To integrate the mathematical calculations of backpropagation, Glorot and Bengio [53] introduces the most difficulties of its implementation and Hinton [70] proposes the forward-forward mechanism. It successfully surpasses the efficiency of backpropagation. In systems biology research, Lillicrap et al. [111] indicates that modifies synapses to improve behavior which may largely enhance learning efficiency. Recent studies have further revealed its immense potential application value [123, 136, 157, 182]. In the future, this technology holds promise to further drive the development of artificial intelligence communities. In the future, we see the light in the following research directions:

- Localized data processing. We can make full use of the efficient parallel computation in the GPU devices to achieve pre-processing such as data regularization in large batch training, which can benefit from the maximum computation efficiency.
- Vertical Model Deployment. As the model size grows larger, the limitations of computing devices will become more apparent. Meanwhile, the calculation of individual modules in the large-scale model cannot be fully parallelized due to the chain rules. Research on vertical training on multi-devices is a promising direction.
- Novel efficient paradigm. Developing novel and efficient paradigms can further accelerate training for large models, which should jointly consider both the acceleration techniques of each individual module under conflicts. Especially for edge devices, collaborative training models in decentralized setups is also a promising study in the future [209].
- Biologically plausible mechanism. Recent studies have noted that backpropagation cannot truly simulate the functioning of the human brain. How to achieve natural bionic mechanisms for processing knowledge and connections has become a positive and effective direction.

8 Conclusion

In this survey, we review the general training acceleration techniques for efficiently training large-scale deep learning models. We consider all the components in the *gradient-based* update formula (2) which cover the total training process in the field of deep learning. We present a novel taxonomy to summarize and categorize these techniques into five major directions: “Data-centric", “Model-centric", “Optimization-centric", “Budgeted training", and “System-centric". In the first four parts, we mainly focus on comprehensive studies from the perspectives of algorithm design and methodology. In the “system-centric efficient training” part, we summarize the practical implementation from the perspectives of paradigm innovation and hardware support. We review and summarize the commonly used or latest developed techniques corresponding to each component, their benefits and trade-offs of proposed techniques, and discuss their limitations and promising future studies.

Acknowledgement. Li Shen is partially supported by STI 2030 –“Brain Science and Brain-like Research” Major Project (No. 2021ZD0201405). Dacheng Tao is partially supported by NTU RSR and Start Up Grants.

References

[1] Shahzad Ahmad, Sukalpa Chanda, and Yogesh S Rawat. 2023. EZ-CLIP: Efficient Zeroshot Video Action Recognition. *arXiv preprint arXiv:2312.08010* (2023).

[2] Alex Andonian, Quentin Anthony, Stella Biderman, Sid Black, Preetham Gali, Leo Gao, Eric Hallahan, Josh Levy-Kramer, Connor Leahy, Lucas Nestler, Kip Parker, Michael Pieler, Shivanshu Purohit, Tri Songz, Wang Phil, and Samuel Weinbach. 2021. *GPT-NeoX: Large Scale Autoregressive Language Modeling in PyTorch.* https://doi.org/10.5281/zenodo.5879544

[3] Maksym Andriushchenko and Nicolas Flammarion. 2022. Towards understanding sharpness-aware minimization. In *International Conference on Machine Learning*. PMLR, 639–668.

[4] Youhui Bai, Cheng Li, Zhiqi Lin, Yufei Wu, Youshan Miao, Yunxin Liu, and Yinlong Xu. 2021. Efficient data loader for fast sampling-based gnn training on large graphs. *IEEE Transactions on Parallel and Distributed Systems* 32, 10 (2021), 2541–2556.

[5] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2006. Greedy layer-wise training of deep networks. *Advances in neural information processing systems* 19 (2006).

[6] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. 41–48.

[7] Albert S Berahas, Majid Jahani, Peter Richtárik, and Martin Takáč. 2022. Quasi-Newton methods for machine learning: forget the past, just sample. *Optimization Methods and Software* 37, 5 (2022), 1668–1704.

[8] Zhengda Bian, Hongxin Liu, Boxiang Wang, Haichen Huang, Yongbin Li, Chuanrui Wang, Fan Cui, and Yang You. 2021. Colossal-AI: A unified deep learning system for large-scale parallel training. *arXiv preprint arXiv:2110.14883* (2021).

[9] Zhengda Bian, Qifan Xu, Boxiang Wang, and Yang You. 2021. Maximizing parallelism in distributed training for huge neural networks. *arXiv preprint arXiv:2105.14450* (2021).

[10] Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An Open-Source Autoregressive Language Model. In *Proceedings of the ACL Workshop on Challenges & Perspectives in Creating Large Language Models.* https://arxiv.org/abs/2204.06745

[11] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2022. Token Merging: Your ViT But Faster. *arXiv preprint arXiv:2210.09461* (2022).

[12] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258* (2021).

[13] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. *JAX: composable transformations of Python+NumPy programs.* http://github.com/google/jax

[14] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[15] Han Cai, Chuang Gan, and Song Han. 2022. Efficientvit: Enhanced linear attention for high-resolution low-computation visual recognition. *arXiv preprint arXiv:2205.14756* (2022).

[16] Brad Carlile, Guy Delamarter, Paul Kinney, Akiko Marti, and Brian Whitney. 2017. Improving deep learning by inverse square root linear units (ISRLUs). *arXiv preprint arXiv:1710.09967* (2017).

[17] John Chen, Cameron Wolfe, and Tasos Kyrillidis. 2022. REX: Revisiting Budgeted Training with an Improved Schedule. *Proceedings of Machine Learning and Systems* 4 (2022), 64–76.

[18] Jie-Neng Chen, Shuyang Sun, Ju He, Philip HS Torr, Alan Yuille, and Song Bai. 2022. Transmix: Attend to mix for vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 12135–12144.

[19] Tianlong Chen, Yu Cheng, Zhe Gan, Jingjing Liu, and Zhangyang Wang. 2021. Data-efficient gan training beyond (just) augmentations: A lottery ticket perspective. *Advances in Neural Information Processing Systems* 34 (2021), 20941–20955.

[20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning.* PMLR, 1597–1607.

[21] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, et al. 2023. Symbolic Discovery of Optimization Algorithms. *arXiv preprint arXiv:2302.06675* (2023).

[22] Yuzhong Chen, Zhenxiang Xiao, Lin Zhao, Lu Zhang, Haixing Dai, David Weizhong Liu, Zihao Wu, Changhe Li, Tuo Zhang, Changying Li, et al. 2022. Mask-guided Vision Transformer (MG-ViT) for Few-Shot Learning. *arXiv preprint arXiv:2205.09995* (2022).

[23] ZOMI Chen. 2024. AISys. https://github.com/chenzomi12/AISystem.

[24] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. 2022. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 1290–1299.

[25] Chao Cheng and Keshab K. Parhi. 2020. Fast 2D Convolution Algorithms for Convolutional Neural Networks. *IEEE Transactions on Circuits and Systems I: Regular Papers* 67, 5 (2020), 1678–1691. https://doi.org/10.1109/TCSI.2020.2964748

[26] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2020. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794* (2020).

[27] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* (2015).

[28] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. 2018. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501* (2018).

[29] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. 2020. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops.* 702–703.

[30] Ali Dabouei, Sobhan Soleymani, Fariborz Taherkhani, and Nasser M Nasrabadi. 2021. Supermix: Supervising the mixing data augmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 13794–13803.

[31] Haixing Dai, Zhengliang Liu, Wenxiong Liao, Xiaoke Huang, Zihao Wu, Lin Zhao, Wei Liu, Ninghao Liu, Sheng Li, Dajiang Zhu, et al. 2023. ChatAug: Leveraging ChatGPT for Text Data Augmentation. *arXiv preprint arXiv:2302.13007* (2023).

[32] Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *arXiv preprint arXiv:2205.14135* (2022).

[33] Sajal Dash, Benjamín Hernández, Aristeidis Tsaris, Folami T Alamudun, Hong-Jun Yoon, and Feivi Wang. 2022. A Scalable Pipeline for Gigapixel Whole Slide Imaging Analysis on Leadership Class HPC Systems. In *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 1266–1274.

[34] Yann N Dauphin and Samuel Schoenholz. 2019. Metainit: Initializing learning by learning to initialize. *Advances in Neural Information Processing Systems* 32 (2019).

[35] John E Dennis, Jr and Jorge J Moré. 1977. Quasi-Newton methods, motivation and theory. *SIAM review* 19, 1 (1977), 46–89.

[36] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2021. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861* (2021).

[37] Terrance DeVries and Graham W Taylor. 2017. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552* (2017).

[38] Neofytos Dimitriou and Ognjen Arandjelović. 2022. Sequential Normalization: Embracing Smaller Sample Sizes for Normalization. *Information* 13, 7 (2022), 337.

[39] Liang Ding, Longyue Wang, Xuebo Liu, Derek F Wong, Dacheng Tao, and Zhaopeng Tu. 2021. Progressive Multi-Granularity Training for Non-Autoregressive Translation. In *Findings of the Association for Computational Linguistics*. 2797–2803.

[40] Xiaohan Ding, Honghao Chen, Xiangyu Zhang, Kaiqi Huang, Jungong Han, and Guiguang Ding. 2022. Re-parameterizing Your Optimizers rather than Architectures. *arXiv preprint arXiv:2205.15242* (2022).

[41] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research* 12, 7 (2011).

[42] Chen Dun, Cameron R Wolfe, Christopher M Jermaine, and Anastasios Kyrillidis. 2022. ResIST: Layer-wise decomposition of ResNets for distributed training. In *Uncertainty in Artificial Intelligence*. PMLR, 610–620.

[43] Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381* (2018).

[44] Jeffrey L Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition* 48, 1 (1993), 71–99.

[45] FairScale authors. 2021. FairScale: A general purpose modular PyTorch library for high performance and large scale training. https://github.com/facebookresearch/fairscale.

[46] Vitaly Fedyunin. 2022. (BETA) CHANNELS LAST MEMORY FORMAT IN PYTORCH. https://pytorch.org/tutorials/intermediate/memory_format_tutorial.html#beta-channels-last-memory-format-in-pytorch

[47] Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for NLP. *arXiv preprint arXiv:2105.03075* (2021).

[48] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2020. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412* (2020).

[49] Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635* (2018).

[50] Elias Frantar, Eldar Kurtic, and Dan Alistarh. 2021. M-fac: Efficient matrix-free approximations of second-order information. *Advances in Neural Information Processing Systems* 34 (2021), 14873–14886.

[51] Jonas Geiping and Tom Goldstein. 2022. Cramming: Training a Language Model on a Single GPU in One Day. *arXiv preprint arXiv:2212.14034* (2022).

[52] Deepak Ghimire, Dayoung Kil, and Seong-heum Kim. 2022. A survey on efficient convolutional neural networks and hardware acceleration. *Electronics* 11, 6 (2022), 945.

[53] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 249–256.

[54] Negar Goli and Tor M Aamodt. 2020. Resprop: Reuse sparsified backpropagation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1548–1558.

[55] Linyuan Gong, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tieyan Liu. 2019. Efficient training of bert by progressively stacking. In *International conference on machine learning*. PMLR, 2337–2346.

[56] Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, et al. 2021. Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988* (2021).

[57] Xiaotao Gu, Liyuan Liu, Hongkun Yu, Jing Li, Chen Chen, and Jiawei Han. 2020. On the transformer growth for progressive bert training. *arXiv preprint arXiv:2010.12562* (2020).

[58] Demi Guo, Yoon Kim, and Alexander M Rush. 2020. Sequence-level mixed sample data augmentation. *arXiv preprint arXiv:2011.09039* (2020).

[59] Vineet Gupta, Tomer Koren, and Yoram Singer. 2018. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*. PMLR, 1842–1850.

[60] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. 2000. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *nature* 405, 6789 (2000), 947–951.

[61] Tengda Han, Weidi Xie, and Andrew Zisserman. 2022. Turbo Training with Token Dropout. *arXiv preprint arXiv:2210.04889* (2022).

[62] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. 2021. Pre-trained models: Past, present and future. *AI Open* 2 (2021), 225–250.

[63] Haoze He and Parijat Dube. 2022. Accelerating Parallel Stochastic Gradient Descent via Non-blocking Mini-batches. *arXiv preprint arXiv:2211.00889* (2022).

[64] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16000–16009.

[65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[66] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. 2019. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 558–567.

[67] Xiaoxin He, Fuzhao Xue, Xiaozhe Ren, and Yang You. 2021. Large-scale deep learning optimizations: A comprehensive survey. *arXiv preprint arXiv:2111.00856* (2021).

[68] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* (2016).

[69] Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. 2019. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781* (2019).

[70] Geoffrey Hinton. 2022. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345* (2022).

[71] Elad Hoffer, Itay Hubara, and Daniel Soudry. 2017. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *Advances in neural information processing systems* 30 (2017).

[72] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556* (2022).

[73] Minui Hong, Jinwoo Choi, and Gunhee Kim. 2021. Stylemix: Separating content and style for enhanced data augmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 14862–14870.

[74] Jeremy Howard. 2018. Fastai - progressive resizing. https://www.fast.ai/2018/04/30/dawnbench-fastai/.

[75] Zhouyuan Huo, Bin Gu, Heng Huang, et al. 2018. Decoupled parallel backpropagation with convergence guarantee. In *International Conference on Machine Learning*. PMLR, 2098–2106.

[76] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407* (2018).

[77] Peter Izsak, Moshe Berchansky, and Omer Levy. 2021. How to Train BERT with an Academic Budget. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 10644–10652.

[78] Nishant Jain, Harkirat Behl, Yogesh Singh Rawat, and Vibhav Vineet. 2023. Efficiently robustify pre-trained models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5505–5515.

[79] Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, Lukasz Kaiser, Wojciech Gajewski, Henryk Michalewski, and Jonni Kanerva. 2021. Sparse is enough in scaling transformers. *Advances in Neural Information Processing Systems* 34 (2021), 9895–9907.

[80] Danlin Jia, Geng Yuan, Xue Lin, and Ningfang Mi. 2022. A Data-Loader Tunable Knob to Shorten GPU Idleness for Distributed Deep Learning. In *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*. IEEE, 449–458.

[81] Angela H Jiang, Daniel L-K Wong, Giulio Zhou, David G Andersen, Jeffrey Dean, Gregory R Ganger, Gauri Joshi, Michael Kaminksy, Michael Kozuch, Zachary C Lipton, et al. 2019. Accelerating deep learning by focusing on the biggest losers. *arXiv preprint arXiv:1910.00762* (2019).

[82] Jean Kaddour. 2022. Stop Wasting My Time! Saving Days of ImageNet and BERT Training with Latest Weight Averaging. In *Has it Trained Yet? NeurIPS 2022 Workshop*. https://openreview.net/forum?id=0OrABUHZuz

[83] Michael Kamp, Linara Adilova, Joachim Sicking, Fabian Hüger, Peter Schlicht, Tim Wirtz, and Stefan Wrobel. 2019. Efficient decentralized deep learning by dynamic model averaging. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part I 18*. Springer, 393–409.

[84] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361* (2020).

[85] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2017. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196* (2017).

[86] Angelos Katharopoulos and François Fleuret. 2017. Biased importance sampling for deep neural network training. *arXiv preprint arXiv:1706.00043* (2017).

[87] Angelos Katharopoulos and François Fleuret. 2018. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*. PMLR, 2525–2534.

[88] Kenji Kawaguchi and Haihao Lu. 2020. Ordered sgd: A new stochastic optimization framework for empirical risk minimization. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 669–679.
[89] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *Advances in neural information processing systems* 33 (2020), 18661–18673.
[90] Jang-Hyun Kim, Wonho Choo, Hosan Jeong, and Hyun Oh Song. 2021. Co-mixup: Saliency guided joint mixup with supermodular diversity. *arXiv preprint arXiv:2102.03065* (2021).
[91] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[92] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451* (2020).
[93] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. *Advances in neural information processing systems* 30 (2017).
[94] Mustafa Taha Koçyiğit, Timothy M Hospedales, and Hakan Bilen. 2023. Accelerating Self-Supervised Learning via Efficient Training Strategies. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 5654–5664.
[95] Adam Kohan, Edward A Rietman, and Hava T Siegelmann. 2023. Signal propagation: The framework for learning and inference in a forward pass. *IEEE Transactions on Neural Networks and Learning Systems* (2023).
[96] Adam A Kohan, Edward A Rietman, and Hava T Siegelmann. 2018. Error forward-propagation: Reusing feedforward connections to propagate errors in deep learning. *arXiv preprint arXiv:1808.03357* (2018).
[97] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2017. Imagenet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (2017), 84–90.
[98] Siddharth Krishna Kumar. 2017. On weight initialization in deep neural networks. *arXiv preprint arXiv:1704.08863* (2017).
[99] Guillaume Leclerc, Andrew Ilyas, Logan Engstrom, Sung Min Park, Hadi Salman, and Aleksander Madry. 2022. FFCV: Accelerating Training by Removing Data Bottlenecks. (2022).
[100] Yunwen Lei, Tao Sun, and Mingrui Liu. 2023. Stability and Generalization for Minibatch SGD and Local SGD. *arXiv preprint arXiv:2310.01139* (2023).
[101] Yunwen Lei and Yiming Ying. 2020. Fine-grained analysis of stability and generalization for stochastic gradient descent. In *International Conference on Machine Learning*. PMLR, 5809–5819.
[102] Bei Li, Ziyang Wang, Hui Liu, Yufan Jiang, Quan Du, Tong Xiao, Huizhen Wang, and Jingbo Zhu. 2020. Shallow-to-deep training for neural machine translation. *arXiv preprint arXiv:2010.03737* (2020).
[103] Changlin Li, Bohan Zhuang, Guangrun Wang, Xiaodan Liang, Xiaojun Chang, and Yi Yang. 2022. Automated progressive learning for efficient training of vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12486–12496.
[104] Mengtian Li, Ersin Yumer, and Deva Ramanan. 2019. Budgeted Training: Rethinking Deep Neural Network Training Under Resource Constraints. In *International Conference on Learning Representations*.
[105] Shenggui Li, Fuzhao Xue, Yongbin Li, and Yang You. 2021. Sequence parallelism: Making 4d parallelism possible. *arXiv preprint arXiv:2105.13120* (2021).
[106] Tao Li, Zhehao Huang, Qinghua Tao, Yingwen Wu, and Xiaolin Huang. 2022. Trainable weight averaging for fast convergence and better generalization. *arXiv preprint arXiv:2205.13104* (2022).
[107] Weishi Li, Yong Peng, Miao Zhang, Liang Ding, Han Hu, and Li Shen. 2023. Deep model fusion: A survey. *arXiv preprint arXiv:2309.15698* (2023).
[108] Yingzhou Li, Haizhao Yang, Eileen R Martin, Kenneth L Ho, and Lexing Ying. 2015. Butterfly factorization. *Multiscale Modeling & Simulation* 13, 2 (2015), 714–732.
[109] Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joey Gonzalez. 2020. Train big, then compress: Rethinking model size for efficient training and inference of transformers. In *International Conference on Machine Learning*. PMLR, 5958–5968.
[110] Xingyu Liao, Peng Cheng, Tianhe Ren, Depeng Liang, Kai Dang, Yi Wang, and Xiaoyu Xu. 2021. LiBai. https://github.com/Oneflow-Inc/libai.
[111] Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. 2020. Backpropagation and the brain. *Nature Reviews Neuroscience* 21, 6 (2020), 335–346.
[112] Tao Lin, Sebastian U Stich, Kumar Kshitij Patel, and Martin Jaggi. 2018. Don't use large mini-batches, use local sgd. *arXiv preprint arXiv:1808.07217* (2018).
[113] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2022. A survey of transformers. *AI Open* (2022).
[114] Jihao Liu, Boxiao Liu, Hang Zhou, Hongsheng Li, and Yu Liu. 2022. Tokenmix: Rethinking image mixing for data augmentation in vision transformers. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVI*. Springer, 455–471.
[115] Rui Liu and Barzan Mozafari. 2022. Communication-efficient Distributed Learning for Large Batch Optimization. In *International Conference on Machine Learning*. PMLR, 13925–13946.

[116] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).
[117] Ilya Loshchilov and Frank Hutter. 2016. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983* (2016).
[118] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).
[119] Sindy Löwe, Peter O'Connor, and Bastiaan Veeling. 2019. Putting an end to end-to-end: Gradient-isolated learning of representations. *Advances in neural information processing systems* 32 (2019).
[120] Yucheng Lu, Conglong Li, Minjia Zhang, Christopher De Sa, and Yuxiong He. 2022. Maximizing communication efficiency for large-scale training via 0/1 Adam. *arXiv preprint arXiv:2202.06009* (2022).
[121] Haoyu Ma, Chengming Zhang, lizhi xiang, Xiaolong Ma, Geng Yuan, Wenkai Zhang, Shiwei Liu, Tianlong Chen, Dingwen Tao, Yanzhi Wang, Zhangyang Wang, and Xiaohui Xie. 2023. HRBP: Hardware-friendly Regrouping towards Block-wise Pruning for Sparse Training. https://openreview.net/forum?id=OSS-yWzE9Yu
[122] Yanjun Ma, Dianhai Yu, Tian Wu, and Haifeng Wang. 2019. PaddlePaddle: An open-source deep learning platform from industrial practice. *Frontiers of Data and Domputing* 1, 1 (2019), 105–115.
[123] Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. 2023. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems* 36 (2023), 53038–53075.
[124] Ilia Markov, Hamidreza Ramezani, and Dan Alistarh. 2021. Project CGX: Scalable Deep Learning on Commodity GPUs. *arXiv preprint arXiv:2111.08617* (2021).
[125] James Martens and Roger Grosse. 2015. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*. PMLR, 2408–2417.
[126] Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. 2018. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162* (2018).
[127] Chenlin Meng, Linqi Zhou, Kristy Choi, Tri Dao, and Stefano Ermon. 2022. ButterflyFlow: Building Invertible Layers with Butterfly Matrices. In *International Conference on Machine Learning*. PMLR, 15360–15375.
[128] Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, et al. 2022. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *International Conference on Machine Learning*. PMLR, 15630–15649.
[129] Dmytro Mishkin and Jiri Matas. 2015. All you need is a good init. *arXiv preprint arXiv:1511.06422* (2015).
[130] Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. 2021. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378* (2021).
[131] Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R Devanur, Gregory R Ganger, Phillip B Gibbons, and Matei Zaharia. 2019. PipeDream: Generalized pipeline parallelism for DNN training. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*. 1–15.
[132] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. 2021. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–15.
[133] Renkun Ni, Ping-yeh Chiang, Jonas Geiping, Micah Goldblum, Andrew Gordon Wilson, and Tom Goldstein. 2022. K-SAM: Sharpness-Aware Minimization at the Speed of SGD. *arXiv preprint arXiv:2210.12864* (2022).
[134] NVIDIA. 2022. Apex. https://github.com/NVIDIA/apex.
[135] Open Machine Learning Systems Community. 2022. *Machine Learning Systems: Design and Implementation*. GitHub. https://github.com/openmlsys/openmlsys-zh
[136] Alexander Ororbia, Ankur Mali, Adam Kohan, Beren Millidge, and Tommaso Salvatori. 2024. A review of neuroscience-inspired machine learning. *arXiv preprint arXiv:2403.18929* (2024).
[137] Rui Pan, Haishan Ye, and Tong Zhang. 2021. Eigencurve: Optimal Learning Rate Schedule for SGD on Quadratic Objectives with Skewed Hessian Spectrums. In *International Conference on Learning Representations*.
[138] Xuran Pan, Xuan Jin, Yuan He, Shiji Song, Gao Huang, et al. [n. d.]. Budgeted Training for Vision Transformer. In *The Eleventh International Conference on Learning Representations*.
[139] Ziqi Pang, Zhichao Li, and Naiyan Wang. 2023. Simpletrack: Understanding and rethinking 3d multi-object tracking. In *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part I*. Springer, 680–696.
[140] Aarthi Paramasivam and S Jaya Nirmala. 2022. A survey on textual entailment based question answering. *Journal of King Saud University-Computer and Information Sciences* 34, 10 (2022), 9644–9653.
[141] Ofir Press, Noah A Smith, and Mike Lewis. 2021. Shortformer: Better Language Modeling using Shorter Inputs. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 5493–5505.
[142] Heli Qi, Sashi Novitasari, Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. 2023. SpeeChain: A Speech Toolkit for Large-Scale Machine Speech Chain. *arXiv preprint arXiv:2301.02966* (2023).

[143] Haobo Qi, Feifei Wang, and Hansheng Wang. 2023. Statistical analysis of fixed mini-batch gradient descent estimator. *Journal of Computational and Graphical Statistics* 32, 4 (2023), 1348–1360.

[144] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences* 63, 10 (2020), 1872–1897.

[145] Markus N Rabe and Charles Staats. 2021. Self-attention Does Not Need $O(n^2)$ Memory. *arXiv preprint arXiv:2112.05682* (2021).

[146] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[147] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* 21, 1 (2020), 5485–5551.

[148] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–16.

[149] Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. 2021. Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–14.

[150] Alexandre Ramé, Nino Vieillard, Léonard Hussenot, Robert Dadashi, Geoffrey Cideron, Olivier Bachem, and Johan Ferret. 2024. Warm: On the benefits of weight averaged reward models. *arXiv preprint arXiv:2401.12187* (2024).

[151] Aayush Rana and Yogesh Rawat. 2022. Are all frames equal? active sparse labeling for video action detection. *Advances in Neural Information Processing Systems* 35 (2022), 14358–14373.

[152] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3505–3506.

[153] Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. 2021. ZeRO-Offload: Democratizing Billion-Scale Model Training.. In *USENIX Annual Technical Conference*. 551–564.

[154] Mamshad Nayeem Rizve, Kevin Duarte, Yogesh S Rawat, and Mubarak Shah. 2021. In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning. *arXiv preprint arXiv:2101.06329* (2021).

[155] Anna Rogers, Matt Gardner, and Isabelle Augenstein. 2023. Qa dataset explosion: A taxonomy of nlp resources for question answering and reading comprehension. *Comput. Surveys* 55, 10 (2023), 1–45.

[156] Sungju Ryu, Hyungjun Kim, Wooseok Yi, Eunhwan Kim, Yulhwa Kim, Taesu Kim, and Jae-Joon Kim. 2022. BitBlade: Energy-Efficient Variable Bit-Precision Hardware Accelerator for Quantized Neural Networks. *IEEE Journal of Solid-State Circuits* 57, 6 (2022), 1924–1935.

[157] Tommaso Salvatori, Ankur Mali, Christopher L Buckley, Thomas Lukasiewicz, Rajesh PN Rao, Karl Friston, and Alexander Ororbia. 2023. Brain-inspired computational intelligence via predictive coding. *arXiv preprint arXiv:2308.07870* (2023).

[158] Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120* (2013).

[159] Teven Le Scao, Thomas Wang, Daniel Hesslow, Lucile Saulnier, Stas Bekman, M Saiful Bari, Stella Bideman, Hady Elsahar, Niklas Muennighoff, Jason Phang, et al. 2022. What Language Model to Train if You Have One Million GPU Hours? *arXiv preprint arXiv:2210.15424* (2022).

[160] Max Schwarzer, Johan Samir Obando Ceron, Aaron Courville, Marc G Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. 2023. Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*. PMLR, 30365–30380.

[161] Damien Scieur, Francis Bach, and Alexandre d'Aspremont. 2017. Nonlinear acceleration of stochastic algorithms. *Advances in Neural Information Processing Systems* 30 (2017).

[162] Damien Scieur, Alexandre d'Aspremont, and Francis Bach. 2016. Regularized nonlinear acceleration. *Advances In Neural Information Processing Systems* 29 (2016).

[163] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 2014. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth annual conference of the international speech communication association*.

[164] Alexander Sergeev and Mike Del Balso. 2018. Horovod: fast and easy distributed deep learning in TensorFlow. *arXiv preprint arXiv:1802.05799* (2018).

[165] Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202* (2020).

[166] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053* (2019).

[167] Leslie N Smith. 2017. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*. IEEE, 464–472.

[168] Leslie N Smith, Emily M Hand, and Timothy Doster. 2016. Gradual dropin of layers to train very deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4763–4771.

[169] Leslie N Smith and Nicholay Topin. 2019. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, Vol. 11006. SPIE, 369–386.

[170] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. 2022. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990* (2022).

[171] Samuel L Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V Le. 2017. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489* (2017).

[172] Zhuoran Song, Yihong Xu, Han Li, Naifeng Jing, Xiaoyao Liang, and Li Jiang. 2022. DNN Training Acceleration via Exploring GPGPU Friendly Sparsity. *arXiv preprint arXiv:2203.05705* (2022).

[173] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. 2021. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270* (2021).

[174] Shivakanth Sujit, Somjit Nath, Pedro HM Braga, and Samira Ebrahimi Kahou. 2022. Prioritizing Samples in Reinforcement Learning with Reducible Loss. *arXiv preprint arXiv:2208.10483* (2022).

[175] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223* (2019).

[176] Ivan Svogor, Christian Eichenberger, Markus Spanring, Moritz Neun, and Michael Kopp. 2022. Profiling and Improving the PyTorch Dataloader for high-latency Storage: A Technical Report. *arXiv preprint arXiv:2211.04908* (2022).

[177] Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, et al. 2015. Convolutional neural networks with low-rank regularization. *arXiv preprint arXiv:1511.06067* (2015).

[178] Hao Tan, Le Wang, Huan Zhang, Junjian Zhang, Muhammad Shafiq, and Zhaoquan Gu. 2022. Adversarial attack and defense strategies of speaker recognition systems: A survey. *Electronics* 11, 14 (2022), 2183.

[179] Mingxing Tan and Quoc Le. 2021. Efficientnetv2: Smaller models and faster training. In *International Conference on Machine Learning*. PMLR, 10096–10106.

[180] Hanlin Tang, Shaoduo Gan, Ammar Ahmad Awan, Samyam Rajbhandari, Conglong Li, Xiangru Lian, Ji Liu, Ce Zhang, and Yuxiong He. 2021. 1-bit adam: Communication efficient large-scale training with adam's convergence speed. In *International Conference on Machine Learning*. PMLR, 10118–10129.

[181] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *Comput. Surveys* 55, 6 (2022), 1–28.

[182] Ayush Tewari, Tianwei Yin, George Cazenavette, Semon Rezchikov, Josh Tenenbaum, Frédo Durand, Bill Freeman, and Vincent Sitzmann. 2024. Diffusion with forward models: Solving stochastic inverse problems without direct supervision. *Advances in Neural Information Processing Systems* 36 (2024).

[183] The Mosaic ML Team. 2021. composer. https://github.com/mosaicml/composer/.

[184] Tijmen Tieleman, Geoffrey Hinton, et al. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4, 2 (2012), 26–31.

[185] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. 2019. Fixing the train-test resolution discrepancy. *Advances in neural information processing systems* 32 (2019).

[186] Marcos Treviso, Tianchu Ji, Ji-Ung Lee, Betty van Aken, Qingqing Cao, Manuel R Ciosici, Michael Hassid, Kenneth Heafield, Sara Hooker, Pedro H Martins, et al. 2022. Efficient methods for natural language processing: a survey. *arXiv preprint arXiv:2209.00099* (2022).

[187] AFM Uddin, Mst Monira, Wheemyung Shin, TaeChoong Chung, Sung-Ho Bae, et al. 2020. Saliencymix: A saliency guided data augmentation strategy for better regularization. *arXiv preprint arXiv:2006.01791* (2020).

[188] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[189] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold mixup: Better representations by interpolating hidden states. In *International conference on machine learning*. PMLR, 6438–6447.

[190] Jiahuan Wang and Hong Chen. 2024. Towards Stability and Generalization Bounds in Decentralized Minibatch Stochastic Gradient Descent. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 15511–15519.

[191] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. 2019. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8612–8620.

[192] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768* (2020).

[193] William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using# petpeeve tweets. In *Proceedings of the 2015 conference on empirical methods in natural language processing*. 2557–2563.

[194] Yulin Wang, Yang Yue, Rui Lu, Tianjiao Liu, Zhao Zhong, Shiji Song, and Gao Huang. 2022. EfficientTrain: Exploring Generalized Curriculum Learning for Training Visual Backbones. *arXiv preprint arXiv:2211.09703* (2022).

[195] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682* (2022).

[196] Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196* (2019).

[197] Lilian Weng. 2023. The Transformer Family Version 2.0. *lilianweng.github.io* (Jan 2023). https://lilianweng.github.io/posts/2023-01-27-the-transformer-family-v2/

[198] Chuhan Wu, Fangzhao Wu, Tao Qi, Yongfeng Huang, and Xing Xie. 2021. Fastformer: Additive attention can be all you need. *arXiv preprint arXiv:2108.09084* (2021).

[199] Xiaoxia Wu, Ethan Dyer, and Behnam Neyshabur. 2021. When Do Curricula Work?. In *International Conference on Learning Representations*. https://openreview.net/forum?id=tW4QEInpni

[200] Haifeng Xia and Zhengming Ding. 2022. Cross-domain collaborative normalization via structural knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 2777–2785.

[201] Jian Xie, Jingwei Xu, Guochang Wang, Yuan Yao, Zenan Li, Chun Cao, and Hanghang Tong. 2022. A Deep Learning Dataloader with Shared Data Preparation. In *Advances in Neural Information Processing Systems*.

[202] Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. 2023. Data Selection for Language Models via Importance Resampling. *arXiv preprint arXiv:2302.03169* (2023).

[203] Xingyu Xie, Pan Zhou, Huan Li, Zhouchen Lin, and Shuicheng Yan. 2022. Adan: Adaptive Nesterov Momentum Algorithm for Faster Optimizing Deep Models. https://doi.org/10.48550/ARXIV.2208.06677

[204] Hang Xu, Wenxuan Zhang, Jiawei Fei, Yuzhe Wu, TingWen Xie, Jun Huang, Yuchen Xie, Mohamed Elhoseiny, and Panos Kalnis. 2023. Slamb: accelerated large batch training with sparse communication. In *International Conference on Machine Learning*. PMLR, 38801–38825.

[205] Qifan Xu, Shenggui Li, Chaoyu Gong, and Yang You. 2021. An efficient 2d method for training super-large deep learning models. *arXiv preprint arXiv:2104.05343* (2021).

[206] Zhewei Yao, Reza Yazdani Aminabadi, Olatunji Ruwase, Samyam Rajbhandari, Xiaoxia Wu, Ammar Ahmad Awan, Jeff Rasley, Minjia Zhang, Conglong Li, Connor Holmes, et al. 2023. Deepspeed-chat: Easy, fast and affordable rlhf training of chatgpt-like models at all scales. *arXiv preprint arXiv:2308.01320* (2023).

[207] Yang You, Igor Gitman, and Boris Ginsburg. 2017. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888* (2017).

[208] Yang You, Zhao Zhang, Cho-Jui Hsieh, James Demmel, and Kurt Keutzer. 2018. Imagenet training in minutes. In *Proceedings of the 47th International Conference on Parallel Processing*. 1–10.

[209] Binhang Yuan, Yongjun He, Jared Quincy Davis, Tianyi Zhang, Tri Dao, Beidi Chen, Percy Liang, Christopher Re, and Ce Zhang. 2022. Decentralized training of foundation models in heterogeneous environments. *arXiv preprint arXiv:2206.01288* (2022).

[210] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*. 6023–6032.

[211] Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* (2012).

[212] Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Advances in Neural Information Processing Systems* 32 (2019).

[213] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* (2017).

[214] Jiong Zhang, Hsiang-Fu Yu, and Inderjit S Dhillon. 2019. Autoassist: A framework to accelerate training of deep neural networks. *Advances in Neural Information Processing Systems* 32 (2019).

[215] Sixin Zhang, Anna E Choromanska, and Yann LeCun. 2015. Deep learning with elastic averaging SGD. *Advances in neural information processing systems* 28 (2015).

[216] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems* 28 (2015).

[217] Jiawei Zhao, Florian Schäfer, and Anima Anandkumar. 2021. ZerO initialization: Initializing neural networks with only zeros and ones. *arXiv preprint arXiv:2110.12661* (2021).

[218] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. 2020. Differentiable augmentation for data-efficient gan training. *Advances in Neural Information Processing Systems* 33 (2020), 7559–7570.

[219] Yulin Zhao, Donghui Wang, Leiou Wang, and Peng Liu. 2018. A faster algorithm for reducing the computational complexity of convolutional neural networks. *Algorithms* 11, 10 (2018), 159.

[220] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. 2020. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 13001–13008.

[221] Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, et al. 2023. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *arXiv preprint arXiv:2302.09419* (2023).
[222] Bohan Zhuang, Jing Liu, Zizheng Pan, Haoyu He, Yuetian Weng, and Chunhua Shen. 2023. A Survey on Efficient Training of Transformers. *arXiv preprint arXiv:2302.01107* (2023).

Received 12 June 2023; revised 19 July 2024; accepted 4 October 2024