Efficient Neural Network Training via Subset Pretraining

Jan Spörer[1,*][a], Bernhard Bermeitinger[2,*][b], Tomas Hrycej[1,*], Niklas Limacher[1,**]
and Siegfried Handschuh[1,*][c]

[1] *Institute of Computer Science, University of St.Gallen (HSG), St.Gallen, Switzerland*

[2] *Institute of Computer Science in Vorarlberg, University of St.Gallen (HSG), Dornbirn, Austria*

[*] *{firstname.lastname}@unisg.ch,* [**] *{firstname.lastname}@student.unisg.ch*

Keywords: Deep Neural Network, Convolutional Network, Computer Vision, Efficient Training, Resource Optimization, Training Strategies, Overdetermination Ratio, Stochastic Approximation Theory.

Abstract: In training neural networks, it is common practice to use partial gradients computed over batches, mostly very small subsets of the training set. This approach is motivated by the argument that such a partial gradient is close to the true one, with precision growing only with the square root of the batch size. A theoretical justification is with the help of stochastic approximation theory. However, the conditions for the validity of this theory are not satisfied in the usual learning rate schedules. Batch processing is also difficult to combine with efficient second-order optimization methods. This proposal is based on another hypothesis: the loss minimum of the training set can be expected to be well-approximated by the minima of its subsets. Such subset minima can be computed in a fraction of the time necessary for optimizing over the whole training set. This hypothesis has been tested with the help of the MNIST, CIFAR-10, and CIFAR-100 image classification benchmarks, optionally extended by training data augmentation. The experiments have confirmed that results equivalent to conventional training can be reached. In summary, even small subsets are representative if the overdetermination ratio for the given model parameter set sufficiently exceeds unity. The computing expense can be reduced to a tenth or less.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

Neural networks as forecasting models learn by fitting the model forecast to the desired reference output (e.g., reference class annotations) given in the data collection called the training set. The fitting algorithm changes model parameters in the loss function's descent direction, measuring its forecast deviation. This descent direction is determined using the loss function gradient.

The rapidly growing size of neural networks (such as those used for image or language processing) motivates striving for a maximum computing economy. One widespread approach is determining the loss function gradient from subsets of the training set, called batches (or mini-batches). Different batches are alternately used to cover the whole training set during the training.

There are some arguments supporting this procedure. (Goodfellow et al., 2016, Section 8.1.3) refers to the statistical fact that random standard deviation decreases with the square root of the number of samples. Consequently, the gradient elements computed from a fraction of $1/K$ training samples (with a given positive integer K) have a standard deviation equal to the factor $\sqrt{K}$ multiple of those computed over the whole training set, which seems to be a good deal.

Another frequent justification is with the help of stochastic approximation theory. The stochastic approximation principle applies when drawing training samples from a stationary population generated by a fixed (unknown) model. (Robbins and Monro, 1951) discovered this principle in the context of finding the root (i.e., the function argument for which the function is zero) of a function $g(x)$ that cannot be directly observed. What can be observed are randomly fluctuating values $h(x)$ whose mean value is equal to the value of the unobservable function, that is,

$$E\left[h(x)\right] = g(x) \tag{1}$$

The task is to fit an input/output mapping to data by gradient descent. For the parameter vector w of this mapping, the mean of the gradient $h(w)$ with respect to the loss function computed for a single train-

[a] https://orcid.org/0000-0002-9473-5029
[b] https://orcid.org/0000-0002-2524-1850
[c] https://orcid.org/0000-0002-6195-9034

Spörer, J., Bermeitinger, B., Hrycej, T., Limacher, N. and Handschuh, S.
Efficient Neural Network Training via Subset Pretraining.
DOI: 10.5220/0012893600003838
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 16th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2024) - Volume 1: KDIR*, pages 242-249
ISBN: 978-989-758-716-0; ISSN: 2184-3228
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

ing sample is expected to be equal to the gradient $g(w)$ over the whole data population. The local minimum of the loss function is where the gradient $g(w)$ (i.e., the mean value of $h(w)$) is zero. (Robbins and Monro, 1951) have proven that, under certain conditions, the root is found with probability one (but without a concrete time upper bound).

However, this approach has some shortcomings. For different batches, the gradient points in different directions. So, the descent for one batch can be an ascent for another. To cope with this, (Robbins and Monro, 1951) formulated the convergence conditions. They require that if the update rule for the parameter vector is

$$w_{t+1} = w_t - c_t h(w_t) \tag{2}$$

which corresponds to the usual gradient descent with step size c_t, the step size sequence c_t has to satisfy the following conditions:

$$\sum_{t=1}^{\infty} c_t = \infty \tag{3}$$

and

$$\sum_{t=1}^{\infty} c_t^2 < \infty \tag{4}$$

Condition Eq. (3) is necessary for the step not to vanish prematurely before reaching the optimum with sufficient precision. Condition Eq. (4) provides for decreasing step size. With a constant step size, the solution would infinitely fluctuate around the optimum. This is because, in the context of error minimization, its random instance $h(w)$ will not diminish for individual samples, although the gradient $g(w) = E[h(x)]$ will gradually vanish as it approaches the minimum. Finally, the gradients of individual samples will not vanish even if their mean over the training set is zero. At the minimum, $g(w) = 0$ will result from a balance between individual nonzero vectors $h(w)$ pointing to various directions.

2 SHORTCOMINGS OF THE BATCH ORIENTED APPROACH

The concept of gradient determination using a subset of the training set is mostly satisfactory. However, several deficiencies from theoretical viewpoints suggest an enhancement potential.

2.1 Violating the Conditions of the Stochastic Approximation

The conditions Eq. (3) and Eq. (4) for convergence of the stochastic approximation procedure to a global (or at least local) minimum result from the stochastic approximation theory. Unfortunately, they are almost always neglected in the neural network training practice. This may lead to a bad convergence (or even divergence). The common *Stochastic Gradient Descent* (SGD) with a fixed learning step size violates the stochastic approximation principles. However, even popular sophisticated algorithms do not satisfy the conditions. The widespread (and successful) *Adam* (Kingma and Ba, 2015) optimizer uses a weight consisting of the quotient of the exponential moving average derivative and the exponential moving average of the square of the derivative

$$\begin{aligned} w_{t+1,i} &= w_{t,i} - \frac{c m_{t,i}}{\sqrt{d_{t,i}}} \frac{\partial E(w_{t,i})}{\partial w_{t,i}} \\ m_{t,i} &= \beta_1 d_{t-1,i} + (1-\beta_1) \frac{\partial E(w_{t-1,i})}{\partial w_{t-1,i}} \\ d_{t,i} &= \beta_2 d_{t-1,i} + (1-\beta_2) \left(\frac{\partial E(w_{t-1,i})}{\partial w_{t-1,i}} \right)^2 \end{aligned} \tag{5}$$

with metaparameters c, β_1, and β_2, network weights $w_{t,i}$, and the loss function E. β_1 is the decay factor of the exponential mean of the error derivative, β_2 is the decay factor of the square of the error derivative, and c is the step length scaling parameter. Their values have been set to the framework's sensible defaults $c = 0.001$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$ in the following computing experiments.

Normalizing the gradient components by the moving average of their square via $\sqrt{d_{t,i}}$ is the opposite of the decreasing step size required in the stochastic approximation theory. If the gradient becomes small (as expected at the proximity of the minimum), the normalization increases them. This may or may not be traded off by the average gradient $m_{t,i}$.

2.2 "Good" Approximation of Gradient Is not Sufficient

The quoted argument of (Goodfellow et al., 2016, Section 8.1.3) that the standard deviations of gradient components are decreasing with the square root $\sqrt{K}$ of number of samples used while the computing expense is increasing with K is, of course, accurate for independent samples drawn from a population.

However, this relationship is only valid if the whole statistical population (from which the training set is also drawn) is considered. This does not account for the nature of numerical optimization algorithms. The more sophisticated among them follow the descent direction. The gradient's statistical deviation is relatively small with respect to the statistical population, but this does not guarantee that a so-determined estimated descent direction is not, in fact, an ascent direction. The difference between descent and ascent may easily be within the gradient estimation error — the batch-based gradient is always a sample estimate, with standard deviation depending on the unknown variance of the individual derivatives within the training set. By contrast, optimizing over the training set itself, the training set gradient is computed deterministically, with zero deviation. Then, the descent direction is certain to lead to a decrease in loss.

The explicit task of the optimization algorithm is to minimize the loss over the training set. If the goal of optimizing over the whole (explicitly unknown) population is adopted, the appropriate means would be biased estimates that can have lower errors over the population, such as ridge regression for linear problems (van Wieringen, 2023). The biased estimate theory provides substantial results concerning this goal but also shows that it is difficult to reach because of unknown regularization parameters, which can only be determined with computationally expensive experiments using validation data sets.

Even if the loss is extended with regularization terms to enhance the model's performance on the whole population (represented by a validation set), the optimized regularized fit is reached at the minimum of the extended loss function once more over *the given training set*. Thus, as mentioned above, it is incorrect from the optimization algorithm's viewpoint to compare the precision of the training set gradient with that of the batches, which are subsamples drawn from the training set. The former is precise, while the latter are approximations.

The related additional argument frequently cited is that what is genuinely sought is the minimum for the population and not for the training set. However, this argument is somewhat misleading. There is no method for finding the true, exact minimum for the population only based on a subsample such as the training set — the training set is the best and only information available. Also, the loss function values used in the algorithm to decide whether to accept or reject a solution are values for the given training set. Examples in (Hrycej et al., 2023) show that no law guarantees computing time savings through incremental learning for the same performance.

2.3 Convexity Around the Minimum Is not Exploited

Another problem is that in a specific environment of the local minimum, every smooth function is convex — this directly results from the minimum definition. Then, the location of the minimum is not determined solely by the gradient; the Hessian matrix also captures the second derivatives. Although using an explicit estimate of the Hessian is infeasible for large problems with millions to billions of parameters, there are second-order algorithms that exploit the curvature information implicitly. One of them is the well-known conjugate gradient algorithm (Hestenes and Stiefel, 1952; Fletcher and Reeves, 1964), thoroughly described in (Press et al., 1992), which requires only the storage of an additional vector with a dimension equal to the length of the plain gradient. However, batch sampling substantially distorts the second-order information more than the gradient (Goodfellow et al., 2016). This leads to a considerable loss of efficiency and convergence guarantee of second-order algorithms, which is why they are scarcely used in the neural network community, possibly sacrificing the benefits of their computing efficiency.

Second-order algorithms cannot be used with the batch scheme for another reason. They are usually designed for continuous descent of loss values. Reaching a specific loss value with one batch cannot guarantee that this value will not become worse with another batch. This violates some assumptions for which the second-order algorithms have been developed. Mediocre computing results with these algorithms in the batch scheme seem to confirm this hypothesis.

3 SUBSTITUTING THE TRAINING SET BY A SUBSET

To summarize the arguments in favor of batch-oriented training, the batch-based procedure is justified by the assumption that the gradients for individual batches are roughly consistent with the gradient over the whole training set (epoch). So, a batch-based improvement is frequently enough (but not always, depending on the choice of the metaparameters) also an improvement for the epoch. This is also consistent with the computing experience record. On the other hand, one implicitly insists on optimizing over the whole training set to find an optimum, as one batch is not expected to represent the training set fully.

Batch-oriented gradient optimization hypothesizes that the batch-loss gradient approximates the training set gradient and the statistic population gradient well enough.

By contrast, the hypothesis followed here is related but essentially different. It is assumed that *the optimum of the loss subset is close to the optimum of the training set*.

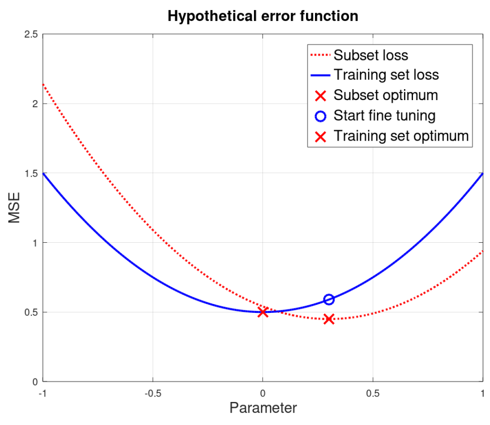

Figure 1: Optima for a subset and the whole training set.

Even if the minimum approximation is imperfect, it can be expected to be a very good initial point for fine-tuning the whole training set so that a few iterations may suffice to reach the true minimum. This principle is illustrated in Fig. 1. The subset loss function (red, dotted) is not identical to the training set loss function (blue, solid). Reaching the minimum of the subset loss function (red cross) delivers an initial point for fine-tuning on the training set (blue circle). This initial point is close to the training set loss minimum (blue cross) and is very probably within a convex region around the minimum. This motivates using fast second-order optimization methods such as the conjugate gradient method (Press et al., 1992).

Another benefit of such a procedure is that for a fixed subset, both the gradient and the intermediary loss values are exact. This further justifies the use of second-order optimization methods.

Of course, the question is how large the subset has to be for the approximation to be sufficiently good. As previously noted, a smooth function is always locally convex around a minimum. If the approximate minimum over the training subset is in this environment, conditions for efficient minimization with the help of second-order algorithms are satisfied. Then, a fast convergence to the minimum over the whole training set can be expected.

Consequently, it would be desirable for the minimum of the subset loss to be within the convex region of the training set loss.

4 SETUP OF COMPUTING EXPERIMENTS

The following computing experiments investigate the support of these hypotheses. The experimental method substitutes the training set with representative subsamples of various sizes. Subsequently, a short fine-tuning on the whole training set has been performed to finalize the optimum solution. The model is trained on the subsamples for exactly 1,000 epochs, while the fine-tuning on the whole training set is limited to 100 epochs.

4.1 Benchmark Datasets Used

The benchmarks for the evaluation have been chosen from the domain of computer vision. They are medium-sized problems that can be run for a sufficient number of experiments. This would not be possible with large models such as those used in language modeling.

For the experiments, three well-known image classification datasets MNIST (LeCun et al., 1998), CIFAR-10, and CIFAR-100 (Krizhevsky, 2009) were used. MNIST contains grayscale images of handwritten digits (0–9) while CIFAR-10 contains color images of exclusively ten different mundane objects like “horse”, “ship”, or “dog”. CIFAR-100 contains the same images; however, they are classified into 100 fine-grained classes. They contain 60,000 (MNIST) and 50,000 (CIFAR-10 and CIFAR-100) training examples. Their respective preconfigured test split of each 10,000 examples are used as validation sets. While both CIFAR-10 and CIFAR-100 are evenly distributed among the classes, MNIST is roughly evenly distributed. We opted to proceed without mitigating the slight class imbalance.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

4.2 The Model Architecture

The model is a convolutional network inspired by the *VGG* architecture (Simonyan and Zisserman, 2015). It uses three consecutive convolutional layers with the same kernel size of 3×3, 32/64/64 filters, and the ReLU activation function. Each is followed by a maximum pooling layer of size 2×2. The last feature map is flattened, and after a classification block with one layer of 64 units and the ReLU activation function, a linear dense layer classifies it into a softmax vector. All trainable layers’ parameters are initialized randomly from the Glorot uniform distribution (Glorot and Bengio, 2010) with a unique seed per layer such that all trained models throughout the experiments have an identical starting point. The biases of each layer are initialized by zeros. The number of parameters for the models differs only because MNIST has one input channel, while CIFAR-10 and CIFAR-100 have three, and CIFAR-100 has 100 class output units instead of 10.

4.3 Preventing Underdetermination of Model Parameters

An important criterion is that the training set size is sufficient for this procedure. The size of the training subsets (as related to the number of model parameters) must be large enough for the model not to be underdetermined. This should be true for most of the subsets tested so that we can fairly compare subsets that are a relatively small fraction of the training set. As a criterion for this, the overdetermination ratio of each benchmark candidate has been evaluated (Hrycej et al., 2023):

$$Q = \frac{KM}{P} \tag{6}$$

with K being the number of training examples, M being the output vector length (usually equal to the number of classes), and P being the number of trainable model parameters.

This formula justifies itself by ensuring that the numerator KM equals the number of constraints to be satisfied (the reference values for all training examples). This product must be larger than the number of trainable parameters for the system to be sufficiently determined. (Otherwise, there are infinite solutions, most of which do not generalize.) This is equivalent to the requirement for the overdetermination ratio Q to be larger than unity. It is advisable that this is satisfied for the training set subsets considered, although subsequent fine-tuning on the whole training set can "repair" a moderate underdetermination.

The two datasets MNIST and CIFAR-10 have ten classes. This makes the number of constraints KM in Eq. (6) too small for subsets with $b > 4$. This is why these two datasets have been optionally expanded by image augmentation. This procedure implies slight random rotations, shifts, and contrast variations. So, the number of training examples has been increased tenfold by augmenting the training data. With CIFAR-100 containing 100 classes, this problem does not occur, and it was not augmented.

4.4 Processing Steps

The processing steps for every given benchmark task and a tested algorithm have been the following:

- The number of subsets b such that a subset is the fraction $1/b$ of the training set has been defined. These numbers have been: $b \in B$ with $B = \{2, 4, 8, 16, 32, 64, 128\}$. With a training set size K, a subset contains K/b samples. For example, a value of $b = 2$ results in a subset with half of the samples from the original training set.
- All b subsets of size K/b have been built to support the results statistically. Each subset $B_{bi}, i = 1, \ldots, b$, consists of training samples with index i selected so that the subsets partition the entire training set. The number of experiments is excessive for larger values of b, so only five random subsets are selected. All randomness is seeded such that each experiment receives the same subset.
- For every $b \in B$ and every $i = 1, \ldots, b$, the subset loss E_{bi} has been minimized using the selected training algorithm. The number of epochs has been set to 1,000. Additionally, the losses for the whole training set (E_{BTbi}) and validation set (E_{BVbi}) have been evaluated. Subsequently, a fine-tuning on the whole training set for 100 epochs has been performed, and the metrics for the training set (E_{Tbi}) and validation set (E_{Vbi}) have been evaluated. In summary, the set of loss characteristics E_{bi}, E_{BTbi}, E_{BVbi}, E_{Tbi}, and E_{Vbi} have represented the final results.
- For comparison, the typical training on the original training set is given by choosing $b = 1$.

The conjugate gradient algorithm would be the favorite for optimizing the subset (because of its relatively small size) and fine-tuning (because of its expected convexity in the region containing the initial point delivered by the subset training). Unfortunately, this algorithm is unavailable in deep learning frameworks like *Keras*. This is why the popular Adam algorithm has been used. For reproducibility and removing additional hyperparameters, a fixed learning rate of 0.001 was employed for all training steps.

5 RESULTS

5.1 Dataset MNIST

The results for the non-augmented MNIST dataset are depicted in Figs. 2 and 3. The training has two phases:

1. the *subset training* phase, in which only a fraction of the training set is used; and
2. the *fine-tuning* phase, in which the optimized parameters from the subset training phase are fine-tuned by a (relatively short) training over the whole training set.

On the x-axis, fractions of the complete training set are shown as used for the subset training. The axis is logarithmic, so the variants are equally spaced. These are $1/2$, $1/4$, $1/8$, $1/16$, $1/32$, $1/64$, and $1/128$, as well as the baseline (fraction equal to unity). This baseline corresponds to the conventional training procedure over the full training set.

The plotted magnitudes in Figs. 2 and 3 refer to

- the loss or accuracy reached for the given subset (*Subset pre-training*);
- the loss or accuracy over the whole training set in the subset training optimum (*Tr.set pre-training*);
- the loss or accuracy over the validation set in the subset training optimum (*Valid.set pre-training*);
- the loss or accuracy over the whole training set attained through fine-tuning (*Tr.set fine-tuning*); and
- the loss or accuracy over the validation set in the fine-tuning optimum (*Valid.set fine-tuning*).

All of them are average values over the individual runs with disjoint subsets.

The dotted vertical line marks the subset fraction with overdetermination ratio Eq. (6) equal to unity. To the left of this line, the subsets are underdetermined; to the right, they are overdetermined.

Both loss (Fig. 2) and accuracy (Fig. 3) suggest similar conjectures:

- The subset training with small subsets leads to poor training set and validation set losses. This gap diminishes with the growing subset fraction.
- Fine-tuning largely closes the gap between the training and validation sets. The optimum value for the training set tends to be lower for large fractions (as they have an "advance" from the subset training, but this does not lead to a better validation set performance. The baseline loss (the rightmost point) exhibits the highest validation set loss.

The overdetermination ratio delivers, together with the mentioned vertical lines in Fig. 2 and Fig. 3, an additional finding: The gap between the performance on the subset and on the whole training set after the subset training is very large for $Q < 1$ (the left side of the plot) and shrinks for $Q > 1$ (the right side).

The results for the augmented data are depicted in the plots Fig. 4 and Fig. 5. As the augmented data are more challenging to fit, their performance characteristics are generally worse than those of the non-augmented dataset. However, an important point can be observed: the performance after the pre-training (particularly for the validation set) does not differ to the same extent as it did with non-augmented data. As with the non-augmented dataset, the baseline loss (the rightmost point) exhibits the highest validation set loss. There, the difference between the lowest and the highest subset losses has been tenfold, while it is roughly the same for all subset fractions with the augmented data.

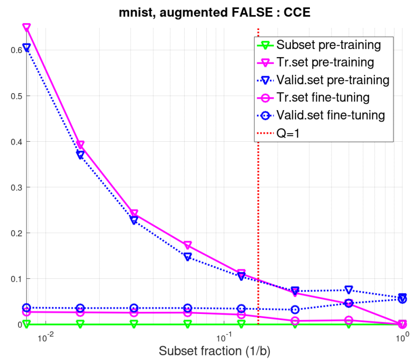

Figure 2: Dataset MNIST (not augmented), loss optima for a pre-trained subset and the whole training set in dependence from the subset size (as a fraction of the training set).

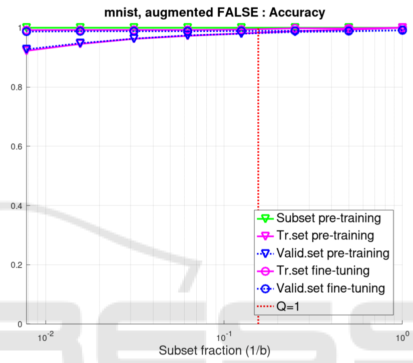

Figure 3: Dataset MNIST (not augmented), accuracy optima for a pre-trained subset and the whole training set in dependence from the subset size (as a fraction of the training set).

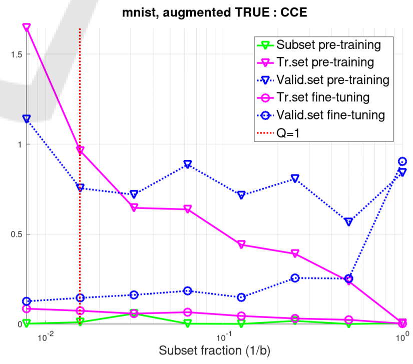

Figure 4: Dataset MNIST (augmented), loss optima for a pre-trained subset and the whole training set in dependence from the subset size (as a fraction of the training set).

The ten times larger size of the augmented dataset leads to overdetermination ratios Q mostly (except for the fraction $1/128$) over unity. Then, even the small-fraction subsets generalize acceptably (which is the goal of sufficient overdetermination).

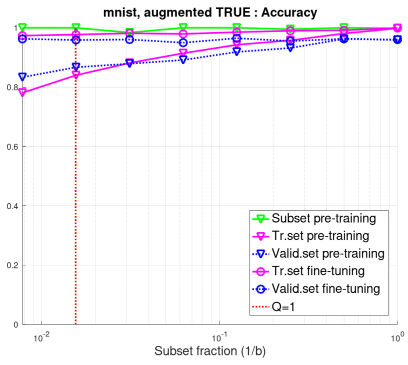

Figure 5: Dataset MNIST (augmented), accuracy optima for a pre-trained subset and the whole training set in dependence from the subset size (as a fraction of the training set).

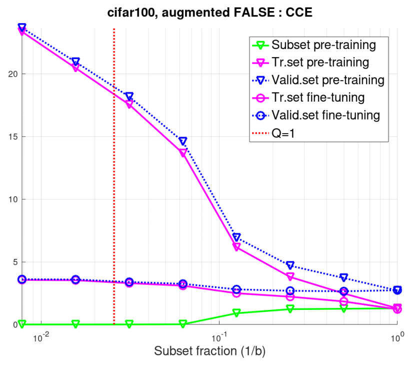

Figure 8: Dataset CIFAR-100 (non-augmented), loss optima for a pre-trained subset and the whole training set in dependence from the subset size (as a fraction of the training set).

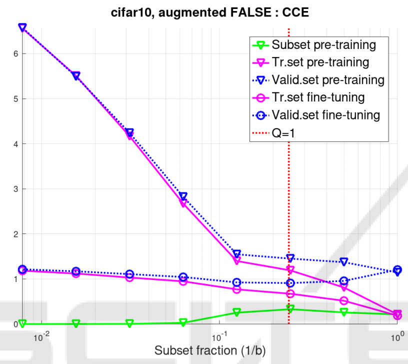

Figure 6: Dataset CIFAR-10 (non-augmented), loss optima for a pre-trained subset and the whole training set in dependence from the subset size (as a fraction of the training set).

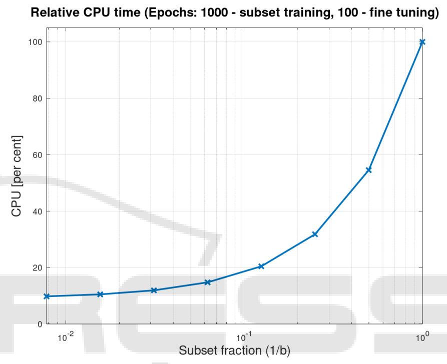

Figure 9: Training time relative to the conventional training in dependence from the subset size (in percent).

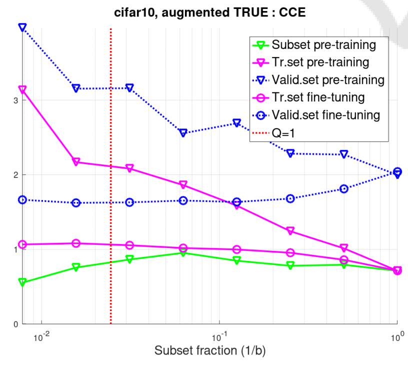

Figure 7: Dataset CIFAR-10 (augmented), loss optima for a pre-trained subset and the whole training set dependent on the subset size (as a fraction of the training set).

5.2 Dataset CIFAR-10

The results for non-augmented CIFAR-10 data are depicted in Fig. 6, those for augmented data in Fig. 7. Due to the size of CIFAR-10 being close to MNIST, the overdetermination ratios are also very similar. Since CIFAR-10 is substantially harder to classify, losses and accuracies are worse.

The accuracy is a secondary characteristic (since the categorical cross-entropy is minimized), and its explanatory power is limited. For this and the space reasons, accuracies will not be presented for CIFAR-10 and CIFAR-100.

Nevertheless, the conclusions are similar to those from MNIST. The gap between the subset's and the entire training set's fine-tuning performance diminishes as the subset grows. This gap is large with non-augmented data in Fig. 6 because of low to overdetermination ratios Q but substantially smaller for augmented data in Fig. 7 where overdetermination ratios are sufficient. The verification set performance after both training phases is typically better with subsets of most sizes than with the whole training set.

5.3 Dataset CIFAR-100

The results for (non-augmented) CIFAR-100 data are depicted in Fig. 8. This classification task differen-

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

tiates 100 classes so that there are only 500 examples per class. Optimum losses for this benchmark are higher than for the previous ones.

For small subset fractions, the representation of the classes is probably insufficient. This may explain the large gap between the subset loss and the training set loss after the subset training with small subset fractions. These may contain, on average, even as few examples per class as four. Nevertheless, the loss for the validation set with various subset sizes is close to the baseline loss for the conventional full-size training.

6 CONCLUSION

The experiments presented support the concept of subset training. We demonstrated the following elements.

- The subset training leads to results comparable with conventional training over the whole training set.
- The overdetermination ratio Q (preferably above unity) should determine the subset size. Nevertheless, even underdetermined subsets may lead to a good fine-tuning result, although they put more workload on the fine-tuning (to close the large generalization gap).
- To summarize, even small subsets can be representative enough to approximate the training set loss minimum well whenever the overdetermination ratio sufficiently exceeds unity.

The most important achievement is the reduction of computing expenses. Most optimizing iterations are done on the subset, where the computational time per epoch is a fraction of that for the whole training set. In our experiments with ten times more subset training epochs than fine-tuning epochs, the relative computing time in percent of the baseline is shown in Fig. 9. Computational resource savings of 90 % and more are possible.

This empirical evaluation using five benchmarks from the CV domain is insufficient for making general conclusions. Large datasets such as ImageNet are to be tested in the future. They have been omitted because many experiments are necessary to produce sufficient statistics. Furthermore, these experiments can be extended to language benchmarks and language models.

It is also important to investigate the behavior of the second-order optimization algorithms such as conjugate gradient (Hestenes and Stiefel, 1952; Fletcher and Reeves, 1964). Their strength can develop only with a sufficient number of iterations. This is an obstacle if very large training sets are a part of the task. Appropriately chosen subsets can make such training feasible and help to reach good performance even with models of moderate size.

REFERENCES

Fletcher, R. and Reeves, C. M. (1964). Function minimization by conjugate gradients. *The Computer Journal*, 7(2):149–154.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, volume 9, pages 249–256, Sardinia, Italy. PMLR.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts.

Hestenes, M. R. and Stiefel, E. (1952). *Methods of conjugate gradients for solving linear systems*, volume 49. NBS Washington, DC.

Hrycej, T., Bermeitinger, B., Cetto, M., and Handschuh, S. (2023). *Mathematical Foundations of Data Science*. Texts in Computer Science. Springer International Publishing, Cham.

Kingma, D. P. and Ba, J. (2015). Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations*.

Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images. Dataset, University of Toronto.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, USA.

Robbins, H. and Monro, S. (1951). A Stochastic Approximation Method. *Annals of Mathematical Statistics*, 22(3):400–407. 3515 citations (Crossref) [2021-08-17].

Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *ICLR*.

van Wieringen, W. N. (2023). Lecture notes on ridge regression. arXiv:1509.09169 [stat].