SustainML

Application Aware, Life-Cycle Oriented Model-Hardware Co-Design Framework for Sustainable, Energy Efficient ML Systems

Resource utilization aware ML framework

Deliverable D3.2

WP3 - Energy Consumption Optimized ML Toolkit and Methods

This project has received funding from the European Union's Horizon Europe research and innovation programme (HORIZON-CL4-2021-HUMAN-01) under grant agreement No 101070408

eProsima The Middleware Experts

DFKI

Inria

upmem

Technische Universität Kaiserslautern

IBM

Project

Title:	SustainML: Application Aware, Life-Cycle Oriented Model-Hardware Co-Design Framework for Sustainable, Energy Efficient ML Systems
Acronym:	SustainML
Coordinator:	eProsima
Grant agreement ID:	101070408
Call:	HORIZON-CL4-2021-HUMAN-01
Program:	Horizon Europe
Start:	01 October 2022
Duration:	36 months
Website:	https://sustainml.eu
E-mail:	sustainml@eprosima.com
Consortium:	**eProsima (EPROS)**, Spain **DFKI**, Germany **TU Kaiserslautern (TUK)**, Germany **University of Copenhagen (KU)**, Denmark **National Institute for Research in Digital Science and Technology (INRIA)**, France **IBM Research GmbH**, Switzerland **UPMEM**, France

Deliverable

Number:	**D3.2**
Title:	**Resource utilization aware ML framework**
Month:	18
Work Package:	WP3 - Energy Consumption Optimized ML Toolkit and Methods
Work Package leader:	KU
Deliverable leader:	KU
Deliverable type:	R,DEM
Dissemination level:	PU
Date of submission:	2024-03-31
Version:	v1.0
Status:	Finished

Version history

Version	Date	Responsible	Author/Reviewer	Comments
v1.0	31-03-2024	Raghavendra Selvan	Bob Pepin	

Executive summary

This deliverable, corresponding to *Deliverable D3.2 - Resource utilization aware ML framework* of SustainML project focuses on advancing *Task 3.2* on "Resource Aware Training Cycle Modeling". This report documents the methods developed that address the resource costs associated with the training phase of machine learning (ML) model development. We introduce three novel methodologies that significantly enhance the training efficiency of ML models by optimizing the number of training examples required, minimizing the necessity for labeled data, and reducing memory consumption. These optimizations subsequently impact computational demand, energy usage, and the carbon footprint of training ML models.

Dataset condensation (DC) focuses on generating smaller, synthetic datasets that encapsulate the critical information from larger datasets. This approach enables ML models to achieve performance comparable to those trained on full datasets but with a reduced data budget. In Section 2, we introduce a method leveraging finite coverings for dataset condensation, providing adversarial robustness guarantees and demonstrating a balanced trade-off between model performance, efficiency, and robustness. This method is currently under review at a leading ML conference.

Achieving effective learning with limited labeled data is a pivotal challenge in ML. Recent advancements in contrastive learning and self-supervised learning have shown efficacy in deriving useful data representations for various applications, such as image classification and object detection. In Section 3, we present a novel self-supervised learning technique for image segmentation, which significantly reduces training time by a factor of 144. This efficiency is achieved by integrating a contrastive loss objective with a confidence network that optimally selects positive and negative examples. This work has been presented at the 6th Northern Lights Deep Learning Conference and published in a Level-1 journal in Norway.

The growing memory requirements of modern deep learning (DL) methods necessitate innovative strategies to reduce computational resources. One prevalent approach is quantizing neural networks, which involves training with reduced-precision weights and activation maps. In Section 4, we propose an advanced quantization technique that decreases memory consumption during training by 15%. This improvement is achieved through a novel variance minimization strategy combined with block-wise quantization, validated across multiple graph ML tasks. This work has been accepted and presented at the 2024 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP).

The methodologies detailed in this deliverable contribute substantially to the field of Sustainable ML by addressing critical resource constraints in the training cycle. These approaches not only enhance training efficiency but also reduce the environmental impact of ML model development. Each method has been rigorously evaluated and disseminated through reputable conferences and publications, underscoring their significance and potential for broader application in the ML community.

Contents

Executive Summary **3**

Contents **4**

1 Introduction **5**

2 Dataset Compression with Guarantees on Robustness **6**
2.1 Introduction to dataset compression and robustness 6
2.2 Experiments 6
2.2.1 Dataset compression in large compression budget regime 7
2.2.2 Dataset compression in small compression budget regime 9

3 Efficient Self-Supervision for Computer Vision **9**
3.1 Introduction 10
3.2 Method 11
3.2.1 Notation 11
3.2.2 Confidence Network 11
3.2.3 Entropy-based Patch Sampler 12
3.2.4 Similarity Measures 13
3.2.5 Contrastive Loss 13
3.3 Data & Experiments 13
3.3.1 Data 13
3.3.2 Experimental Set-up 14
3.4 Results 14
3.5 Discussions & Conclusions 16

4 Activation Compression of Graph Neural Networks using Block-wise Quantization with Improved Variance Minimization **17**
4.1 Notations and Background 17
4.2 Methods 19
4.2.1 Block-wise Quantization of Activation maps 19
4.2.2 Improved Variance Minimization 19
4.3 Experiments and Results 21

5 Conclusion **23**

References **24**

1 Introduction

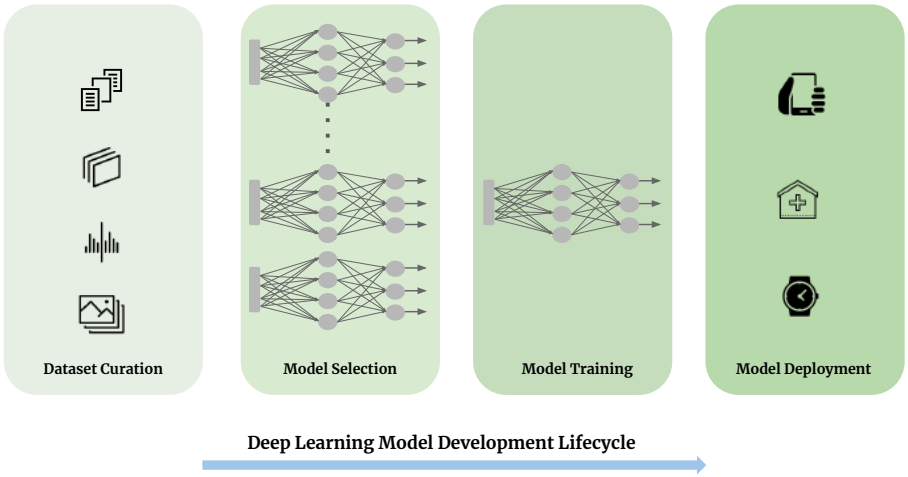


Figure 1: Overview of a typical deep learning model development lifecycle.

In this deliverable, which is related to Task 3.2 about *Resource aware training cycle modeling* we focus on the resource costs incurred during the training part of the machine learning (ML) model development cycle. We present three methods that achieve a significant improvement in training efficiency of training ML models in terms of number of training examples required, number of labels required and memory consumption. Each of these factors further have an impact on the compute, energy consumption, and the carbon footprint of training these models.

Data as resource: Dataset Condensation (DC) is a method that generates a smaller, synthetic dataset from a larger one. The synthetic dataset retains the essential information of the original dataset, enabling AI models to achieve performance levels comparable to those trained on the full dataset but with a limited data budget. Our first contribution, presented in section 2, uses finite coverings to condense datasets. It provides adversarial robustness guarantees and shows how to achieve a desired trade-off between performance, efficiency and robustness of machine learning models.

Labels as resource: Learning task specific representations with limited labelled data is an important but challenging goal of machine learning. Recent advancements in contrastive learning and self-supervised learning have shown promising results in obtaining discriminative representations of the data which can be useful for downstream applications such as image classification, object detection and speech recognition. In section 3 we present a new method for self-supervised learning for image segmentation tasks that reduces training time by a factor of 144. This drastic reduction in training time is due to the combination of a contrastive loss objective with a confidence network that generates optimal sets of positive and negative examples.

Memory as resource: The increase in memory consumption in recent classes of deep learning methods necessitates the use of more computational resources. A common approach to reducing resource consumption of DL methods is to explore different efficiency strategies such as training neural networks with quantized weights or quantized activation maps. In section 4 we propose an improved quantization method that is able to reduce memory consumption during training by 15%. This is made possible by the introduction of a new variance minimization technique combined with block-wise quantization. We evaluated our method on a number of graph machine learning tasks.

The method from section 2 is currently under review at a top ML conference, the method from section 3 has been accepted and presented at the 6th Northern Lights Deep Learning Conference (with proceedings

in a Level-1 publication in Norway) and the work from section 4 has been accepted and presented at the 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

2 Dataset Compression with Guarantees on Robustness

2.1 Introduction to dataset compression and robustness

This part of the project report is centered around the study that evaluates the effectiveness of adversarial training with compressed datasets. Dataset Condensation (DC) is a method that generates a smaller, synthetic dataset from a larger one. The synthetic dataset retains the essential information of the original dataset, enabling AI models to achieve performance levels comparable to those trained on the full dataset but with a limited data budget.

While DC methods have been successful in achieving high test performance with a limited data budget, they have not directly addressed the question of adversarial robustness. Adversarial robustness refers to the ability of an AI model to withstand adversarial attacks, where inputs to the model are intentionally modified to cause the model to make mistakes. This study investigates the impact of adversarial training with compressed datasets on adversarial robustness.

We propose a novel method based on Minimum Finite Covering (MFC) to improve both dataset compression efficiency and adversarial robustness. Unlike DC methods, which focus solely on retaining essential information for high test performance, MFC aims to cover the robust frontier of the dataset. This means it seeks to find a balance between minimizing the maximum possible loss (robustness) and minimizing the size of the dataset (efficiency).

Empirical evaluation on multiple datasets and neural networks shows that the proposed MFC method can achieve a better robustness and performance trade-off compared to relevant baseline DC methods. This work could contribute significantly towards reducing data usage in developing AI models while ensuring guarantees on their robustness, addressing both efficiency and security concerns inherent in AI model training.

In conclusion, this study reveals an unaddressed aspect of adversarial robustness in models trained with compressed datasets. It proposes a novel method that improves both dataset compression efficiency and adversarial robustness, providing a new direction for future research in this area. The results demonstrate the potential of this method in developing robust AI models with limited data, contributing to the broader goal of efficient and secure AI model training.

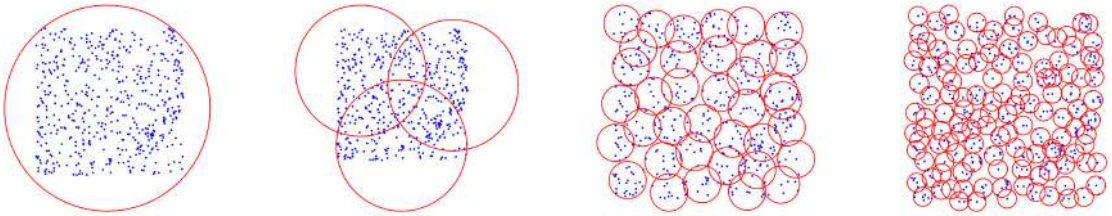
By reducing the amount of data required to train robust AI models, it can decrease the computational resources needed, leading to lower energy consumption. This is particularly important given the growing concerns about the environmental impact of large-scale ML. Furthermore, by ensuring adversarial robustness, it can also contribute to the long-term reliability and trustworthiness of AI systems, which are crucial for sustainable ML practices.

We envision the results from this deliverable will be input to the SustainML framework, where the users can indicate the *data budget* they can afford which will then be translated into picking the most representative data points. Further, the proposed method provides intuitive visualizations of the chosen data points (see Figure 2) which can then be useful in the design of WP5.

For details of the method we refer to the original manuscript [1].

2.2 Experiments

In this section, we provide empirical evidence supporting the trade-off between robustness and accuracy for compressed dataset, as previously discussed. Our experiments are conducted on the following



(a) $\eta = 0.7, k = 1$ (b) $\eta = 0.5, k = 3$ (c) $\eta = 0.1, k = 37$ (d) $\eta = 0.05, k = 127$

Figure 2: Visualization of the minimal finite covering with fixed radius η and the corresponding minimal k.

datasets: MNIST, CIFAR10, CIFAR100, SVHN, TinyImageNet. We explore three DC methods: distribution matching (**DM**) [2], gradient matching (**GM**) [3], trajectory matching (**TM**) [4]; various coreset methods: minimal coreset (**MCS**, ours) of fixed size, gradient-based methods **Craig** [5] and **GradM** [6], submodularity based methods with graph cut and facility location functions (**SubMod**) [7]; and baselines: original full dataset (**Raw**), random coreset selection (**Rand**).

For all datasets, we focus on MCS w.r.t. ℓ_2-norm, and perform adversarial training over them for both ℓ_∞- and ℓ_2- norms. This is because real image data are distributed very closely to the boundary of ℓ_∞-balls, resulting in similar ℓ_∞-distances between data points within the same class. All the MCS are obtained using the existing MILP solver in GUROBI[1]. We use PGD-ℓ_∞ and PGD-ℓ_2 attack to perform adversarial training and compute the robust accuracy in Section 2.2.1, and use AutoAttack [8] instead for robustness evaluation in Section 2.2.2. Following the common setting in the robustness literature [9], we set the adversarial perturbation to $\varepsilon_\infty = 0.1$ for MNIST dataset, $\varepsilon_\infty = 8/255$ for CIFAR10, CIFAR100, SVHN datasets, and $\varepsilon_\infty = 4/255$ for TinyImageNet dataset, for ℓ_∞-norm. For ℓ_2-norm, in order to keep the volume of ℓ_∞-ball and ℓ_2-ball be similar, we set $\varepsilon_2 = \sqrt{\frac{2n}{\pi e}}\varepsilon_\infty$, where n is the (flattened) dimension of input images. We use SGD optimizer in Pytorch with momentum 0.9 and weight decay 5×10^{-4}, and all the experiments are run with 3 repeats using an NVIDIA A100 40GB GPU.

2.2.1 Dataset compression in large compression budget regime

In this section, we mainly consider DM from [2] as the baseline DC method due to the considerable computational costs associated with obtaining the compressed datasets from other DC methods involving bi-level optimizations; particularly when scaling up the size, such as with $k = 500$. We choose the computationally cheaper DM method, as the baseline which also shows competitive performance compared to other DC methods [10]. For each compressed dataset, we consider a multilayer perceptron (MLP) for MNIST dataset, and convolutional neural networks (ConvNet) for CIFAR10 dataset. Specifically, the MLP architecture consists of two hidden layers, each comprising 128 neurons. The ConvNet architecture includes 3 blocks, each containing 128 filters of size 3×3, followed by instance norm [11], ReLU activation and average pooling layers. We perform generalized adversarial training over the MCS, and classical adversarial training over other compressed dataset.For convenience of comparison, we consider k-MCS for both MNIST and CIFAR10 dataset with $k = [50, 100, 200, 300, 400, 500]$.

Figure 3 shows the test, ℓ_∞- and ℓ_2- robust accuracy of models obtained by standard, ℓ_∞- and ℓ_2-adversarial training over the original MNIST and compressed dataset. Notice that for **Rand** and **MCS**,

[1] `https://www.gurobi.com/`

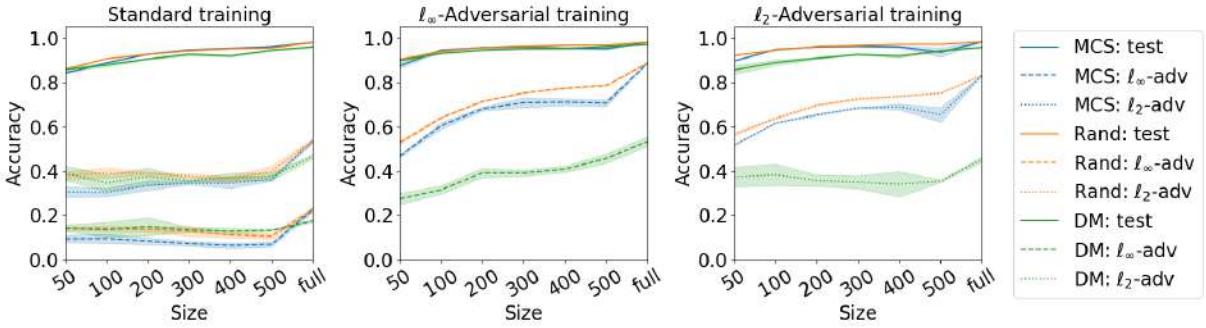


Figure 3: Performance of standard and robust models trained with different compressed datasets from MNIST. The figures from left to right are standard training, ℓ_∞-adversarial training and ℓ_2-adversarial training. The blue, orange and green lines stand for **MCS**, **Rand**, and **DM** respectively. The solid, dashed and dotted lines stand for test, ℓ_∞ and ℓ_2 robust accuracy respectively. In the horizontal axis, "full" means the original dataset for **MCS** and **Rand**, and size 5000 for **DM**.

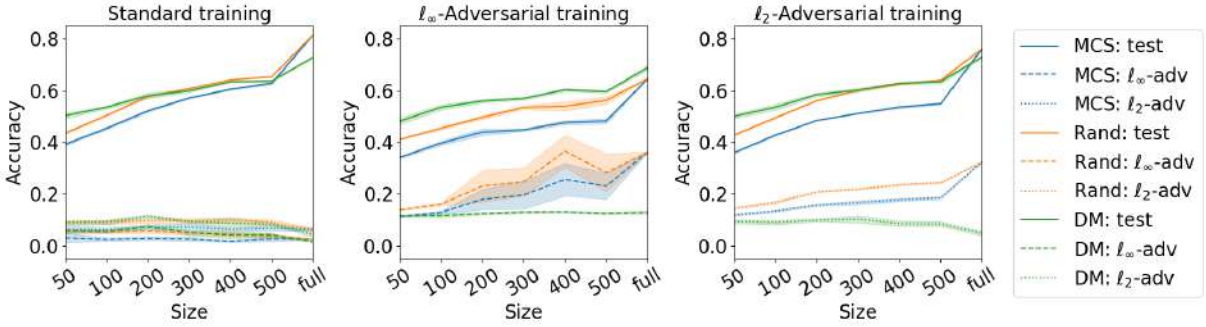


Figure 4: Performance of standard and robust models trained over different compressed dataset from CIFAR10. The figures from left to right are standard training, ℓ_∞-adversarial training and ℓ_2-adversarial training. The blue, orange and green lines stand for **MCS**, **Rand**, and **DM** respectively. The solid, dashed and dotted lines stand for test, ℓ_∞ and ℓ_2 robust accuracy respectively. In the horizontal axis, "full" means the original dataset for **MCS** and **Rand**, and size 4000 for **DM**.

the coresets becomes exactly the original dataset if the size equals N. However, even for size N, method **DM** is still different from the original dataset. We compute **DM** with size 5000 for MNIST dataset. We see that, when applying standard training, all models show similar performance in terms of test accuracy and robust accuracy. However, adversarial training over **DM** does not seem to be more effective, while **Rand** and **MCS** have large improvement in robustness. Even if the size of **DM** increases to 5000, the robust accuracy after adversarial training is still *much* lower than when using the original dataset. DC methods violate the underlying distribution of the original data.

Figure 4 shows exactly the same behavior for CIFAR10 dataset. Due to memory issue, we only have access to **DM** with size 4000. As the size of compressed dataset increases, the robust accuracy of both **MCS** and **Rand** are both increasing. However, the robustness of **DM** does not seem to improve. Especially if we increase the size k of **DM** to 4000, the test accuracy of ℓ_∞-robustly trained model slightly outperforms the original CIFAR10 data (68.86% v.s. 64.59%), whereas the robust score of the former is *much* smaller than the latter (12.98% v.s. 36.21%), which is actually at the same level of compression size 50.

2.2.2 Dataset compression in small compression budget regime

In this section, we fix the compression budget of all methods to 50 and extend the experiments to include additional DC methods (**DM**, **GM**, and **TM**) and larger datasets (MNIST, CIFAR10, CIFAR100, SVHN, and TinyImageNet). For all the cases, we use ResNet18 [12] for both standard and adversarial training, applying AutoAttack with APGD-CE and APGD-DLR to test the l_∞-robustness of each model. The learning rate and number of epochs are fixed at 0.01 and 20, respectively, and experiments are repeated for three runs.

Table 3 summarizes the standard and robust scores of models trained on different compressed datasets. We also compare three coreset methods (**Craig**, **GradM**, and **SubMod**). We observe similar trends in test and robust performance as noted in the previous section. The three accuracy-oriented compression methods (**DM**, **GM**, **TM**) demonstrate relatively good test performance but are less effective for adversarial training. For example, on the MNIST dataset, while the **DM**, **GM**, and **TM** methods exhibit excellent test performance (standard scores of 96.70%, 96.54%, and 95.93%, respectively, versus 96.42% for **MCS**), their robust performance is inferior to that of **MCS** (91.20%, 83.68%, and 82.84%, respectively, versus 92.83%). Similarly, for the SVHN dataset, which contains digit images from 0 to 9 but with a much larger size, the robust performance of these DC methods is significantly worse than that of **MCS** (4.21%, 0.35%, and 1.19%, respectively, versus 13.74%). These experiments provide extensive evidence that DC methods tend to overfit to test performance, while our **MCS** method and other coreset methods, such as random selection, show more balanced behavior in terms of both accuracy and robustness.

Table 1: Downstream performance of models trained over compressed dataset of MNIST, CIFAR10, CIFAR100, SVHN, and TinyImageNet. We consider coreset methods **Rand**, **MCS**, **Craig**, **GradM**, **SubMod**, and compression methods **DM**, **GM**, **TM** for fixed size 50. All robust scores are computed by AutoAttack with APGD-CE and APGD-DLR.

DATASET	SCORE	RAND	MCS (OURS)	CRAIG	GRADM	SUBMOD	DM	GM	TM
MNIST	STD	95.15±0.14	96.42±0.21	87.02±0.81	91.24±0.44	82.22±0.02	**96.70±0.09**	96.54±0.15	95.93±0.36
	ROB	92.50±0.37	**92.83±0.07**	72.45±1.11	86.46±0.37	75.40±0.21	91.20±0.27	83.68±0.74	82.84±1.73
CIFAR10	STD	32.29±0.76	24.88±3.12	29.24±2.79	27.53±0.47	**38.21±0.91**	32.59±2.49	30.67±2.10	30.62±3.03
	ROB	8.79±1.05	5.26±0.16	9.01±1.98	5.83±1.70	**9.14±0.26**	2.68±0.30	0.73±0.21	0.87±0.19
CIFAR100	STD	20.24±0.54	13.71±1.05	17.46±2.12	19.78±0.35	22.89±0.20	16.00±0.95	8.61±0.88	**24.45±0.14**
	ROB	**6.59±0.23**	4.84±0.58	4.70±3.22	5.24±0.53	**7.35±0.63**	3.52± 0.65	0.99±0.30	1.09±0.10
SVHN	STD	65.05±5.61	**66.61±5.73**	57.37±1.04	36.00±1.18	53.48±3.86	62.23±13.59	51.68± 13.86	47.08±10.63
	ROB	9.36±0.58	13.74±2.20	**19.59±0.00**	6.69±1.23	18.62±1.66	4.21±0.43	0.35±0.13	1.19±0.31
TINY	STD	**18.23±0.41**	14.34±0.20	–	–	–	12.01±0.96	2.15±0.24	14.54±1.61
	ROB	0.17±0.04	**4.48±1.57**				1.16±0.24	0.03±0.00	3.22±0.66

3 Efficient Self-Supervision for Computer Vision

Learning discriminative representations of unlabelled data is a challenging task. Contrastive self-supervised learning provides a framework to learn meaningful representations using learned notions of similarity measures from simple pretext tasks. In this work, we propose a simple and efficient framework for self-supervised image segmentation using contrastive learning on image patches, without using explicit pretext tasks or any further labeled fine-tuning. A fully convolutional neural network (FCNN) is trained in a self-supervised manner to discern features in the input images and obtain confidence maps which capture the network's belief about the objects belonging to the same class. Positive- and negative- patches are sampled based on the average entropy in the confidence maps for contrastive learning. Convergence is assumed when the information separation between the positive patches is small, and the positive-negative pairs is large. We evaluate this method for the task of segmenting nuclei from multiple histopathology datasets, and show comparable performance with relevant self-supervised and supervised methods. The proposed model only consists of a simple FCNN with 10.8k parameters and requires about 5 minutes

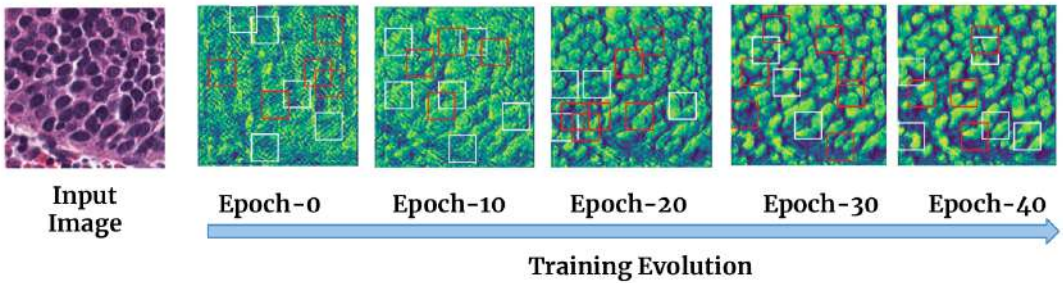


Figure 5: Overview of the training evolution of the proposed contrastive self-supervised learning model. At each of the five illustrated epochs, predicted confidence map (from which segmentation masks are derived by thresholding) for a single validation set image is shown, along with the patches sampled as positive (white squares) and negative (red squares).

to converge on the high resolution microscopy datasets, which is orders of magnitude smaller than the relevant self-supervised methods to attain similar performance.[2]

3.1 Introduction

Learning task specific representations without any – or with limited – labelled data continues to be an elusive goal of machine learning. Recent advancements in contrastive learning and self-supervised learning have shown promising results in obtaining discriminative representations of the data which can be useful for downstream applications such as image classification [13], object detection [14] and speech recognition [15]. Contrastive self-supervised learning (CSL) has been successfully used as a form of pre-training to reduce the dependence on labeled data for more complex tasks such as image segmentation [16]. Most self-supervised methods rely on pretext tasks for training them [17, 18]. Designing relevant pretext tasks can be challenging and even if a useful pretext task is obtained they may not easily generalise across datasets [19].

In this work, we present a self-supervised learning framework that contrastively learns an object detection model for segmenting nuclei in histopathology images. The model comprises a fully convolutional neural network (FCNN) that predicts one confidence map per output channel, which captures the confidence of each pixel belonging to a particular object class. The FCNN is contrastively trained using smaller positive- and negative patches stochastically sampled from the images. Patches within a training batch are sampled from an entropy based distribution, where the entropy is based on the patch-level confidence scores. The intuition behind the entropy-based sampling is to obtain positive patches that contain similar information and negative patches that contain contrasting information, with respect to features that can discriminate between objects of different classes. Through iterative training we are able to improve the information separation between the positive- and negative patches, resulting in an object detection model which can be used for segmentation.

We use a simple FCNN with $10.8k$ tunable parameters which converges in about 5 minutes on a stand-alone GPU workstation. Experimental evaluation on two histopathology datasets [20, 21] show that our efficient, contrastive self-supervised learning method obtains performance comparable to relevant supervised and self-supervised baseline methods, using only a fraction of the compute time and resources. An illustration of how the confidence map evolves during the training process is illustrated in Figure 5.

[2]Source code: `https://github.com/nickeopti/bach-contrastive-segmentation`

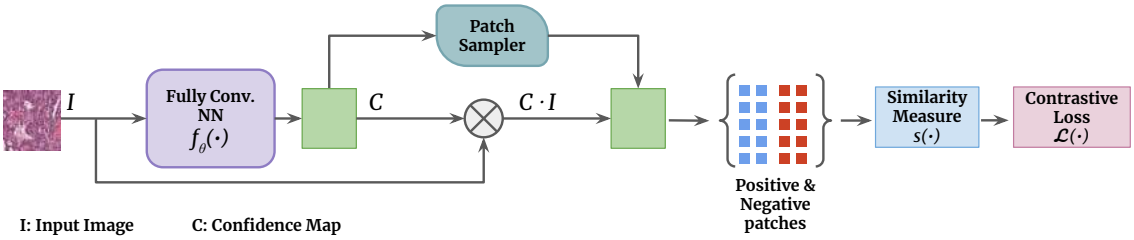


Figure 6: High level overview of the proposed patch-based contrastive self-supervision method. The input color image, $\mathbf{I}$, is input to the fully convolutional neural network, $f_\theta(\cdot)$, to obtain a pixel level foreground confidence map, $\mathbf{C}$. The confidence map is used by the *patch sampler* which returns locations of positive and negative patches. The patches are then obtained from the attended image, $\mathbf{C} \cdot \mathbf{I}$, to obtain the inter- and intra- class similarity measures, $s(\cdot)$. A contrastive loss, $\mathcal{L}(\cdot)$, is computed based on these similarities which is then back-propagated through the entire pipeline. The trainable weights, θ, are tuned until the confidence map, $\mathbf{C}$, corresponds to segmentation masks of interest.

3.2 Method

Segmentation is fundamentally the task of partitioning an image into areas of interest and a background class. Assuming that the areas of interest within the same class are similar in some feature space according to a *similarity measure*, we present our contrastive learning framework that results in an unsupervised segmentation model. A high level overview of the proposed framework is illustrated in Figure 6.

3.2.1 Notation

Consider a batch of M images, with the i'th image represented as the pair (U_i, I_i), where $U_i \subset \mathbb{N}^d$ is a finite set representing the d-dimensional pixel coordinates and I_i is a function $I_i \colon U_i \to [0,1]^\lambda$, with $\lambda \in \mathbb{N}$ denoting the number of colours/channels, mapping such pixel coordinates into their respective values.

Next, we introduce the notation for denoting the patch locations sampled from the image i as the tuple (R, i), where $R \subset U_i$ is the set of pixel coordinates of the image patch. Note that the power set $\mathcal{P}(U_i)$ denotes the set of all possible smaller patches that can be sampled from image i. Denoting the set of all such patch tuples across images $\mathcal{X}$, we consider the subset $S \subset \mathcal{X}$ of regular (square) and spatially connected patches in this work.

The proposed framework uses a *confidence network* (Section 3.2.2), f_θ, parameterised by θ which for a given input image (U_i, I_i) computes the *confidence map*, C_i, i.e., $f_\theta : (U_i, I_i) \in [0,1]^\lambda \mapsto C_i \in [0,1]^K$. The k'th coordinate in $C_i^k(u)$ indicates the belief, or confidence, that the pixel $u \in U_i$ in image i belongs to class k, for $k = 1, \ldots, K$, with $K \geq 1$.

The contrasting of patches depends on a similarity measure (Section 3.2.4), $s \colon \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, which compares the similarity of two patches. This similarity computation relies on (I_i, I_j) and (C_i, C_j), where i and j are the images from which the two compared patches originate from.

3.2.2 Confidence Network

The aim of the confidence network, f_θ, is to learn to detect objects in images without any pixel-level supervision, such that areas with high confidence of the k'th class actually belong to a specific type of object in the images. Conversely, objects detected with high confidence of the j'th class, when $k \neq j$, should belong to a different class. The confidence network, f_θ, can be any trainable model which given an input image produces a set of *confidence maps*, describing the confidence that each pixel of the image

belongs to a corresponding class. This, for instance, can be implemented as a fully convolutional neural network (FCNN), with multi-class support i.e., with multiple output channels such that each channel would correspond to a different output class.

The confidence network is trained to discriminate between objects that could belong to different classes by contrasting *patches* of images against each other. Obtaining meaningful positive- and negative patches for the contrastive learning is the foundational problem in this framework. We next describe a novel approach to mine for such contrastive patches for self-supervision of the confidence network.

3.2.3 Entropy-based Patch Sampler

For each output class $k = 1, \ldots, K$, patches *believed* to contain (a part of) an object belonging to a specific class are treated as *positive* patches, whereas patches *believed* to not contain (a part of) an object of that class are treated as *negative* patches. Appropriate sampling of such positive- and negative patches is of vital importance, as the optimisation of the confidence network, f_θ, is performed according to the contrastive loss computed based on those patches. Improved object detection shall ideally correlate with positive and negative sampled patches being more distinct. Effectively, the appropriate sampling of patches becomes a form of pretext/auxiliary task for training the confidence network, f_θ.

Assume a set of candidate patches, $S \subset \mathcal{X}$. For each patch $(R, i) \in S$ with $|R|$ pixels and for each class k, the average patch confidence is computed as:

$$A_k(R, i) := \frac{1}{|R|} \sum_{u \in R} C_i^k(u) \in [0, 1]. \tag{1}$$

Notice that the higher the average patch confidence, the stronger the belief that the patch contains the type of object belonging to class k.

Recall that $C_i \mapsto [0, 1]^K$, that is, for each class, $k = 1, \ldots, K$, a $C_i^k(u)$ value closer to 1 indicates a higher probability of u belonging to the k'th class. Likewise, a value closer to 0 indicates that u likely does not belong the k'th class, whereas values in between indicate varying degrees of (un)certainty in either direction. Using this intuition about the confidence maps, we model a Bernoulli distributed random variable, $X(u)$, with confidence value as its parameter, denoted as $X(u) \sim \text{Bern}(C_i^k(u))$. Using this, we can now define the average patch entropy as

$$B_k(R, i) := \frac{1}{|R|} \sum_{u \in R} H(X(u)), \text{ where} \tag{2}$$

$$H(X) = -\sum_{x \in \{0,1\}} \mathbb{P}(X(u) = x) \log_2 \mathbb{P}(X(u) = x) \tag{3}$$

$$= -C_i^k(u) \log_2 C_i^k(u) - (1 - C_i^k(u)) \log_2 (1 - C_i^k(u)), \tag{4}$$

which is the entropy of the Bernoulli distributed random variable $X(u)$. The motivation for this choice of Bernoulli random variable is so that good choices of positive- and negative patches, according to the contrastive loss, would correlate with higher certainty from the confidence network. Therefore sampling a set $W_k \subset S$ of n patches according to the unnormalised distribution

$$1 - B_k(R, i)$$

for each class k ensures a stochastic correlation between the confidence map and the appropriateness of patch sampling. Finally, partitioning of these W_k into sets of positive samples, P_k and negative samples N_k is performed according to

$$[(R, i) \in P_k] \sim \text{Bern}(A_k(R, i))$$

for each patch $(R, i) \in W_k$, such that patches with high confidence of belonging to class k are likely to be selected as positive for class k.

3.2.4 Similarity Measures

Ideally, the similarity measure, $s\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, should yield higher values when two input patches are more similar. In this work, for any two patches $(R_1, i), (R_2, j) \in S$, the similarity measure performs the comparison – not on the input image but – on the image scaled by the corresponding confidence map, given by the following pixel-wise product: $F_i(u) := I_i(u)C_i(u), u \in R_1$, and $F_j(v) := I_j(v)C_j(v), v \in R_2$. We employ two pixel-level similarity measures: mean squared error (MSE) and cross-entropy (CE), and treat them as a model hyperparameter and report the best performing measures for each of the datasets.
Backpropagation: The scaling of the input image with the confidence map also serves the purpose of connecting the gradients between the sampling step and the confidence network for backpropagating the contrastive loss to optimise the confidence network.

3.2.5 Contrastive Loss

Within a channel k of the confidence map, we seek to maximise similarity between positive patches while minimising the similarity between positive- and negative- patches. This is facilitated using the intra-channel contrastive loss

$$\mathcal{L}_{\text{intra}} = -\sum_{k=1}^{K} \frac{1}{|P_k|^2} \sum_{r,p \in P} \log \frac{\exp(s(r,p))}{\sum_{n \in N} \exp(s(r,n))}. \tag{5}$$

This contrastive loss, inspired by SimCLR [13], rewards f_θ for learning to detect a single feature in the images for each class in the confidence map, when patches are sampled appropriately. However, this lacks any mechanism for enforcing the individual classes to learn distinct features. In a multi-class scenario, each of the classes shall ideally detect different features. This is achieved by introducing an inter-channel contrastive loss

$$\mathcal{L}_{\text{inter}} = -\sum_{k=1}^{K} \frac{1}{|P_k|^2} \sum_{r,p \in P} \frac{\exp(s(r,p_k))}{\sum_{i=1,[i \neq k]}^{K} \sum_{p \in P} \exp(s(r,p))}. \tag{6}$$

Finally, the combined loss is defined as

$$\mathcal{L} = \mathcal{L}_{\text{intra}} + \mathcal{L}_{\text{inter}}. \tag{7}$$

3.3 Data & Experiments

3.3.1 Data

As the proposed CSL framework is trained exclusively using unlabelled images, without exposure to the labels during training, we can also use the labels from the training set for validation of the segmentation performance. That is, we train the weights of the FCNN without using any labels but can employ the labels for model selection.
MoNuSeg: This dataset [20][3] from MICCAI 2018 contains 37 training images at 1000 × 1000 pixels resolution, obtained at 40× magnification, as well as 14 testing images at the same resolution. Dense annotations of all nuclei in all images are provided.
CoNSeP: This dataset [21][4] contains 27 training images at 1000 × 1000 pixels resolution, obtained at 40× magnification, as well as 14 similar testing images. There are dense annotations of all nuclei in all images.

[3] MoNuSeg data is available at `https://monuseg.grand-challenge.org/Data/`

[4] CoNSeP data is available at `https://warwick.ac.uk/fac/cross_fac/tia/data/hovernet/`

3.3.2 Experimental Set-up

Baseline Methods: The simplest supervised baseline method is to obtain the optimal intensity threshold using the training images. The images are converted to grey-scale and an optimal threshold to obtain binary segmentation is obtained. Additionally, a CNN of the same architecture as the confidence network, f_θ, is used as a supervised baseline. And finally, these results are also compared to the self-supervised method that uses scale prediction as a pretext [18], which is referred to as the Scale Pretext method. All baselines are evaluated on both mentioned datasets.

# Patches	5	**10**	15	20	25
Dice	0.6643	0.7071	0.6682	0.6986	0.6059
Patch Size	20	30	**50**	80	120
Dice	0.5063	0.6562	0.6980	0.6606	0.6871

Table 2: The median validation Dice score over three runs on MoNuSeg is used to select number of patches and patch sizes (shown in boldface).

Model Hyperparameters: The default configuration uses a confidence network, f_θ, closely inspired by [18]. Entropy based stochastic sampling described in Section 3.2.3 and $K = 4$ classes are used. The patch size of 50×50 pixels and 10 patches from each image in a batch are chosen based on experiments on the MoNuSeg dataset, as reported in Table 2.

The proposed framework was trained for a maximum of 300 epochs. To limit RAM usage, the input images are cropped into 300×300 pixels, where the location is selected uniformly at random as to add some data variance. No other pre-processing nor data augmentation is applied. Batch size of 10 is used for all experiments.
Implementation: The proposed framework is implemented in PyTorch [22] using PyTorch Lightning [23], with support to be trained on GPUs; all training has been performed on a system with an NVIDIA GeForce RTX 3060 and intel-i7 processor with 32GB memory.
Experiments: With the objective of comparing the segmentation performance of the proposed CSL framework with the baselines, we perform experiments on each of the datasets described in Section 3.3.1.

Our CSL framework is initialised with the hyperparameters described in Section 3.3.2 on each of the datasets and the weights of the confidence network are randomly initialised. A single training epoch consists of contrasting 10 patches of size 50×50 (see Table 2) sampled from one random crop from each training image. At the end of each epoch, the confidence maps obtained from the confidence network are thresholded at $p = 0.5$ and the segmentation performance is evaluated using the labels for the training images by computing the Dice score. After training for 300 epochs, the model with highest validation Dice score is selected for evaluation on the test set. These Dice scores are reported and discussed further in Section 3.4. Simple post-processing comprising two iterations of morphological opening and closing operations with radius 3 and 1, respectively, are performed on the thresholded segmentation masks.

The proposed CSL framework is trained 10 times with different initialisations on each dataset, in order to illustrate and quantify its inherent variance in segmentation performance.

3.4 Results

Segmentation results: Segmentation performance of our method and the baselines are reported in Table 3 on both the datasets. The first two rows show the performance of two supervised methods and

[5]Most likely this is a large under-estimation of the actual convergence time based on in-house implementation as the actual run-time was not reported in [18].

Methods	MoNuSeg	CoNSeP	# Par.	Time
Int. Thresh.	0.5823	0.5700	1	<1m
CNN	0.7666	0.7375	10.8k	≈ 10m
Scale Pretext	0.6209	0.5139	21.7M	> 12h
Ours: Max.	0.6797	0.6266		
Mean ± s.d	0.62±0.05	0.52±0.06	10.8k	≈ 5m
80 perc.	0.6469	0.5448		

Table 3: Dice scores on test sets of MoNuSeg and CoNSeP datasets for the different methods along with maximum, mean+sd and 80'th percentile scores for our method. Optimal intensity threshold method (Int. Thresh.), supervised CNN baseline, self-supervised method using scale prediction pretext task (Scale Pretext) and the proposed CSL framework. The number of trainable parameters and convergence time per method are also reported.

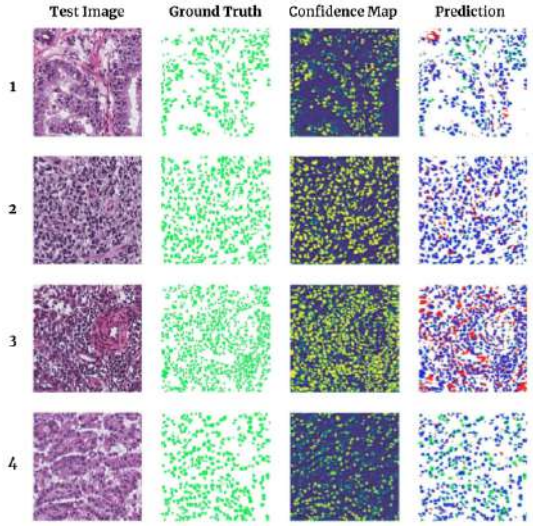


Figure 7: Four test set images from the MoNuSeg dataset. The rightmost column is colour-coded, where blue indicates correct segmentation, green indicates false negatives, and red indicates false positives. The predictions are obtained from the confidence map by a 0.5 threshold.

the third row shows the self-supervised Scale Pretext baseline method [18]. We ran ten random repeats of our model for each dataset configuration and report the mean, maximum and the 80'th percentile Dice score over these runs. Dice score for the Scale Pretext method is obtained from the paper [18]. In all cases, the supervised CNN method performs better than the self-supervised class of methods, which is to be expected, as no further fine-tuning of these methods with labels is performed.

MoNuSeg: Our method outperforms the supervised optimal threshold method when trained on the MoNuSeg dataset using MSE-similarity. Qualitative results on four randomly selected test set images from the MoNuSeg dataset for the model trained on WSI data are shown in Figure 7.

CoNSeP: A similar performance trend is observed on the CoNSeP dataset, where we notice the mean Dice score of our method with the CE similarity measure performs better than the Scale Pretext method. However, the intensity thresholding method performs better than all self-supervised methods.

The number of parameters of our method (10.8k) is several orders of magnitude fewer than Scale Pretext method (21M), and our method takes only a fraction of time until convergence (5m) compared to theirs (>12h) as shown in Table 3. This fast convergence time has implications in overcoming the strong

influence of the confidence maps obtained at initialisation (epoch-0) which can result in convergence to poor solutions. While some runs result in poor performance – dragging the mean down – most runs yield good performance. We capture this as the 80'th percentile performance over multiple runs, which is better than the mean performance. Thus running the model several times (which is still fast) will likely result in good performing solutions. As a standard practice, we therefore recommend running multiple repeats of our model on any dataset.

3.5 Discussions & Conclusions

Confidence maps to Segmentation masks: The efficient CSL framework presented in this work outputs multiple confidence maps corresponding to objects believed to belong to the same class. Going from these confidence maps to the segmentation masks by definition requires expert input. In this work, we alleviate this by using the labels for model selection/validation. For other unlabelled datasets, some other form of validation would be required to align the concept of a biomedical image foreground to that of a confidence map of a self-supervised framework.

Self-supervised segmentation performance: The main baseline we compared our method to is the self-supervised Scale Pretext method [18]. The results of this method is not significantly different from ours. This is even with specific preprocessing (stain-normalisation) and elaborate post-processing followed in [18], which are altogether left out in our work. Further, compared to the Scale Pretext method ours is significantly simpler, both in terms of the model complexity, the elaborate regularisation of their objective function and training time (5m versus 12h).

Predictions in different channels: Having multiple output classes increases the likelihood of the desired class to actually be segmented, at the cost of some computational resources. However, even with $K = 4$ classes, there are runs where none of the classes correspond particularly well with the desired nuclei segmentation. Hence, a few of the runs will turn out useless, requiring training the framework multiple times, and choosing a good instance based on validation performance. Training our method multiple times is recommended, as it could benefit from the variance in segmentation performance. This is captured in Table 3, where the best score is often significantly better than the mean score.

Limitations: The main limitation in the proposed framework coincides with its simplicity — there are no constraints encouraging the model to actually segment the desired objects of interest, resulting only in small performance improvements. Further, obtaining segmentation masks of interest in one of the confidence maps is not guaranteed; so as a standard practice we suggest running the model multiple times. This might not end up being a major limitation, after all, as many model runs do converge to reasonable solutions as captured by the 80-percentile scores in Table 3.

Finally, we assume that the similarity measure plays an important role in model performance. We experimented with a few of these measures. However, thorough investigations of more refined similarity measures and their influence on multi-class segmentation remains as future work.

Conclusions: We presented a self-supervised framework for segmenting nuclei from histopathology data, which uses patch-based contrastive learning. We introduced a novel technique to mine for positive- and negative patches for contrasting based on the average entropy of the confidence maps. This approach encourages the trainable confidence network to discern objects of different classes. The resulting method with only 10.8k trainable parameters takes under 5 min. to converge, yielding useful segmentation masks. We foresee interesting research directions for this work that can make it better suited for diverse image data. This method can be extended to other domains of computer vision. We chose histopathology data as it is an extremely challenging task, and multiple open datasets are available.

4 Activation Compression of Graph Neural Networks using Block-wise Quantization with Improved Variance Minimization

Graph neural networks (GNNs) are a class of deep learning (DL) models most useful when dealing with graph structured data [24, 25]. They have shown widespread relevance in a range of diverse applications [26, 27, 28, 29]. GNNs are known to scale poorly with the number of nodes in the graph data primarily due to the memory requirements for storing the adjacency matrices and intermediate activation maps [30]. The increase in memory consumption necessitates use of more computational resources. This is, unfortunately, in line with the growing resource consumption of recent classes of deep learning methods [31, 32]. A common approach to reducing resource consumption of DL methods is to explore different efficiency strategies [33, 34] such as training neural networks with quantized weights [35] or quantized activation maps [36].

The main focus of efficiency improvements in GNNs has been either by operating on subgraphs to use smaller adjacency matrices [37] or to store compressed node embeddings or activation maps for computing gradients [38]. In this work, we are interested in the latter, specifically following the method introduced in [38] that proposed extreme activation compression (EXACT) using a combination of stochastic rounding-based quantization and random projections.

In this work we make two contributions, starting from EXACT, that further improve the memory consumption and yield training runtime speedup. Firstly, we introduce block-wise quantization [39] of the activation maps which quantizes large groups of tensors instead of individual tensors with support down to `INT2` precision. Secondly, the quantization variance estimation in EXACT is performed using the assumption that the activation maps are uniformly distributed. We show that the activation maps do not follow a uniform distribution but instead follow a type of clipped normal distribution with empirical evidence. Using this insight, we present an improvement to the variance minimization strategy when performing the quantization of activation maps. Experimental evaluation on multiple graph datasets shows a consistent reduction in memory consumption and speedup in training runtime compared to EXACT.

4.1 Notations and Background

We describe a graph with N nodes as $\mathcal{G} = (\mathbf{X}, \mathbf{A})$, with dense node feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$ containing F-dimensional features for each of the N nodes, and sparse adjacency matrix $\mathbf{A} \in \{0, 1\}^{N \times N}$ with the relations between each of the nodes. Specifically $\mathbf{A}_{i,j} = 1$ if there is an edge between node i and j and $\mathbf{A}_{i,j} = 0$ otherwise.

The GNN from [25] with L layers can be compactly written as the recursion:

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(\ell)}\mathbf{\Theta}^{(\ell)}\right) \tag{8}$$

where the symmetric normalized adjacency matrix is $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\mathbf{A}\tilde{\mathbf{D}}^{-1/2}$ with $\tilde{\mathbf{D}}$ as the degree matrix of $\mathbf{A} + \mathbf{I}$, $\mathbf{H}^{(0)} := \mathbf{X}$, the trainable parameters at layer-ℓ are $\mathbf{\Theta}^{(\ell)}$ and a suitable non-linearity $\sigma(\cdot)$.

Since the activation maps, specifically the intermediate results $\left(\mathbf{H}^{(\ell)}\mathbf{\Theta}^{(\ell)}\right)$ and the node embedding matrix $\mathbf{H}^{(\ell)}$, are the biggest users of memory, EXACT [38] focused on reducing the size of the activation maps from `FLOAT32` to lower precision using two methods:

Stochastic Rounding: For a given node i its embedding vector $\mathbf{h}_i^{(\ell)}$ is quantized and stored using b-bit integers as:

$$\mathbf{h}_{i_{\texttt{INT}}}^{(\ell)} = \text{Quant}\left(\mathbf{h}_i^{(\ell)}\right) = \left\lfloor \frac{\mathbf{h}_i^{(\ell)} - Z_i^{(\ell)}}{r_i^{(\ell)}} B \right\rfloor = \lfloor \bar{\mathbf{h}} \rceil \tag{9}$$

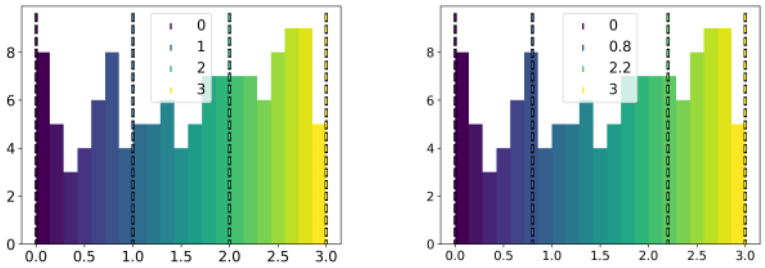


Figure 8: *Demonstration of stochastic rounding for $b = 2$ i.e., $2^b = 4$ quantization bins for 128 points uniformly sampled datapoints. Here the sampled points can be quantized to any of the four levels. The closer the color of the sample is to the color of the vertical bar, the larger the probability that it quantizes to the said vertical bar. Quantization bins when using uniform bin widths (left) and when using non-linear bin widths when performing variance optimization (right) introduced in Sec 4.2.2 is visualized..*

where $B = 2^b - 1$, $Z_i^{(\ell)} = \min(\mathbf{h}_i^{(\ell)})$ is the zero-point, $r_i^{(\ell)} = \max(\mathbf{h}_i^{(\ell)}) - \min(\mathbf{h}_i^{(\ell)})$ is the range for $\mathbf{h}_i^{(\ell)}$, $\bar{\mathbf{h}}$ is the normalized activation map, and $\lfloor \cdot \rceil$ is the stochastic rounding (SR) operation [40]. SR rounds a number to its nearest integer with a probability inversely proportional to the distance from the quantization boundaries.[6] It can be shown that SR is an unbiased operator as the rounding probabilities are dependent on distance of $\lfloor \bar{\mathbf{h}} \rceil$ to the nearest integers [41].[7] Figure 8-A) illustrates SR with uniform bin widths.

The inverse process of dequantization is defined as:

$$\hat{\mathbf{h}}_i^{(\ell)} = \text{Dequant}\left(\mathbf{h}_{i_{\texttt{INT}}}^{(\ell)}\right) = r_i^{(\ell)} \mathbf{h}_{i_{\texttt{INT}}}^{(\ell)} / B + Z_i^{(\ell)} \tag{10}$$

which linearly transforms the quantized values from $[0, B]$ back to their original ranges. Note that we still have some information-loss, since $\mathbf{h}_{i\texttt{INT}}^{(\ell)}$ is only an estimate of $\mathbf{h}_i^{(\ell)}$.[8]

Random Projection: Another way of minimizing memory footprint of activation maps is to perform dimensionality reduction on them. This is done via random projection in EXACT as:

$$\mathbf{h}_{i_{\text{proj}}}^{(\ell)} = \text{RP}(\mathbf{h}_i^{(\ell)}) = \mathbf{h}_i^{(\ell)} \mathbf{R} \tag{11}$$

[6] For any scalar activation map, h, SR is given by:

$$\lfloor h \rceil = \begin{cases} \lfloor h \rfloor + \delta, \text{with probability } (h - \lfloor h \rfloor)/\delta \\ \lfloor h \rfloor, \text{with probability } 1 - (h - \lfloor h \rfloor)/\delta \end{cases}$$

where δ is the uniform bin width, $\lfloor \cdot \rfloor$ is the floor operator.

[7] Consider, the expectation according to the definition of SR:

$$\begin{aligned} \mathbb{E}[\lfloor h \rceil] &= (\lfloor h \rfloor + \delta) \cdot (h - \lfloor h \rfloor) / \delta + \lfloor h \rfloor \cdot (1 - ((h - \lfloor h \rfloor) / \delta)) \\ &= h - \lfloor h \rfloor + \lfloor h \rfloor = h. \end{aligned}$$

This proves that SR with uniform bin widths is unbiased.

[8] Note that quantization followed by dequantization is unbiased due to stochastic rounding, i.e., $\mathbb{E}[\hat{\mathbf{h}}_i^{(\ell)}] = \mathbb{E}[\text{Dequant}(\text{Quant}(\mathbf{h}_i^{(\ell)}))] = \mathbf{h}_i^{(\ell)}$.

where $\mathbf{R} \in \mathbb{R}^{D \times R}$ with $R < D$ is the normalized Rademacher random matrix [42] that satisfies $\mathbb{E}[\mathbf{R}\mathbf{R}^\top] = \mathbf{I}$.

The random projected node embeddings are inversely transformed by

$$\hat{\mathbf{h}}_i^{(\ell)} = \text{IRP}\left(\mathbf{h}_{i_{\text{proj}}}^{(\ell)}\right) = \mathbf{h}_{i_{\text{proj}}}^{(\ell)} \mathbf{R}^T. \tag{12}$$

The matrix containing all projected and recovered activation maps are defined as $\mathbf{H}_{\text{proj}}^{(\ell)}$ and $\hat{\mathbf{H}}^{(\ell)}$, respectively.[9]

EXACT method combines random projection and quantization to obtain compounding reductions in memory consumption. Specifically, node embeddings are compressed as $\tilde{\mathbf{h}}_i^{(\ell)} = \text{Quant}\left(\text{RP}\left(\mathbf{h}_i^{(\ell)}\right)\right)$, are stored in memory during the forward pass, and during the backward pass they are recovered as $\hat{\mathbf{h}}_i^{(\ell)} = \text{IRP}\left(\text{Dequant}\left(\tilde{\mathbf{h}}_i^{(\ell)}\right)\right)$.

4.2 Methods

Quantizing activation maps of GNNs reduces the memory consumption when training GNNs but does introduce an additional overhead in the computation time due to the quantization/dequantization steps. We propose to perform large block-wise quantization [39] in place of quantizing individual tensors in order to recover some of the slowdown and to further reduce the memory consumption.

4.2.1 Block-wise Quantization of Activation maps

The quantization in Eq. (8) is performed over each node embedding, which is a tensor $\mathbf{h}_i^{(\ell)} \in \mathbb{R}^D$ resulting in a sequence of b-bit integers i.e., $\mathbf{h}_{i_{\texttt{INT}}}^{(\ell)} \in \{0, \ldots, B-1\}^D$. Instead of quantizing each node embedding, block-wise quantization takes a larger chunk of tensors and performs the quantization on them which further reduces the memory footprint and yields speedup. Block-wise quantization has been shown to be effective in reducing the memory footprint as demonstrated in [39] where optimizer states are block-wise quantized to 8-bits (`INT8`)[43].

Consider the complete node embedding matrix after random projection, $\mathbf{H}_{\texttt{proj}}^{(\ell)} \in \mathbb{R}^{N \times R}$. To perform block-wise quantization first the node embedding matrix is reshaped into a stack of tensor blocks of length G:

$$\mathbf{H}_{\texttt{block}}^{(\ell)} \in \mathbb{R}^{\cdot \times G} := \text{reshape}\left(\mathbf{H}_{\texttt{proj}}^{(\ell)}, G\right). \tag{13}$$

The sequence of random projection and quantization as described in Section 4.1 are performed on each block in $\mathbf{h}_{i_{\texttt{block}}}^{(\ell)} \in \mathbb{R}^G \; \forall \; i = [1, \ldots, (N \cdot R/G)]$. Performing quantization using larger blocks of tensors is shown to improve training stability, as block-wise quantization localizes the effects of outliers to within its own block [39]. In this work, we experiment different block sizes to study the impact on memory consumption and test performance.

4.2.2 Improved Variance Minimization

Starting from the observation that $\mathbf{h}_{i_{\texttt{INT}}}^{(\ell)}$ is an unbiased estimate, we want to find the quantization boundaries such that its variance, $\text{Var}(\mathbf{h}_{i_{\texttt{INT}}}^{(\ell)})$, is minimized to further reduce the effect of quantization. To achieve this we need three components: 1) distribution of activation maps, 2) variance as a function of

[9] Also note that the RP and IRP operations are also unbiased. i.e., $\mathbb{E}[\hat{\mathbf{H}}^{(\ell)}] = \mathbb{E}[\text{IRP}(\text{RP})(\mathbf{H}^{(\ell)})] = \mathbf{H}^{(\ell)}$.

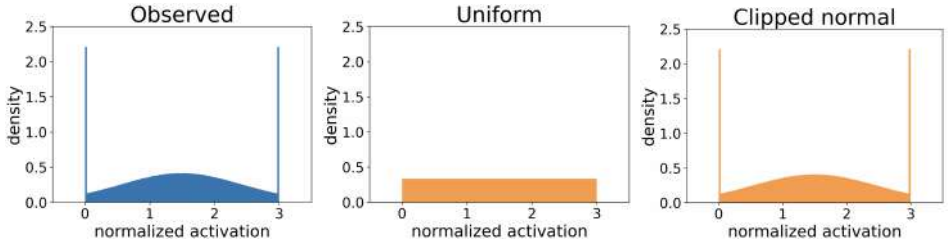


Figure 9: *The observed normalized activations in a GNN model on the OGB-Arxiv data (left) compared to different modelled distributions: uniform (center), and clipped normal (right). Notice the clipped normal is able to model the observed distribution more accurately, including the edges where the spikes are caused due to clipping at the boundaries.*

the activation maps, and 3) minimization of the expected variance as a function of quantization boundaries.

In the EXACT [38] paper, the quantization boundaries are always set to integer values i.e., bins are of constant width. This stems from the assumption that the underlying distribution of activation maps are *uniformly* distributed [38] (Figure 9-center). In this work we show, on multiple datasets, that the activation maps are more accurately distributed as a variation of the normal distribution which we call the clipped normal.

Letting $B = 2^b - 1$ define the number of quantization bins, and Φ^{-1} the Probability Point Function, we describe the clipped normal distribution as

$$\begin{aligned} \mathcal{CN}_{[1/D]}(\mu, \sigma) &= \min\left(\max\left(0, \mathcal{N}(\mu, \sigma)\right), B\right), \\ &\text{where } \mu = B/2 \text{ and } \sigma = -\mu/\Phi^{-1}(1/D). \end{aligned} \tag{14}$$

The similarity between the observed and the modelled activation maps are visualized in Figure 9, where we can see that the clipped normal distribution is better at approximating the activation maps compared to the uniform distribution.

We next expand SR to use irregular bin widths. Consider the normalized activation, $h \in \bar{\mathbf{h}}$ within the bin-i, stochastic rounding when using irregular bin widths, $\delta_i \ \forall \ i = [1, \ldots, B]$, is given by:

$$\lfloor h \rceil = \begin{cases} \lceil h \rceil, \text{with probability } (h - \lfloor h \rfloor)/\delta_i \\ \lfloor h \rfloor, \text{with probability } 1 - ((h - \lfloor h \rfloor)/\delta_i). \end{cases} \tag{15}$$

Following the variance estimation from [41][10] and assuming a normalized activation h, we calculate its SR variance as

$$\text{Var}(\lfloor h \rceil) = \sum_{i=1}^{i=B} \left(\delta_i(h - \alpha_{i-1}) - (h - \alpha_{i-1})^2\right), \tag{16}$$

where δ_i is the width of the bin containing h, and α_i is the starting position of the bin.

Assuming `INT2` quantization i.e., with $B = 3$ bins, the expected variance of the SR operation under the

[10] Check Eq. 4.4 onwards in [41] for detailed derivation.

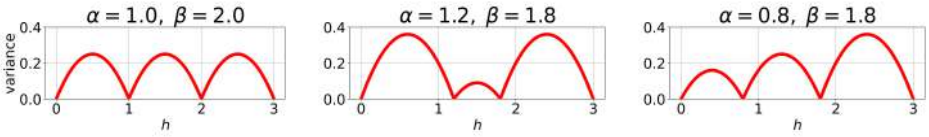


Figure 10: *Variance of SR for* `INT2` *quantization with different quantization boundaries* $[\alpha, \beta]$ *based on Eq.* (16). *When* $[\alpha = 1.0, \beta = 2.0]$ *uniform bin width is obtained.*

clipped normal distribution is obtained from Eq. (16) and Eq. (14):

$$\begin{aligned}\mathbb{E}[\mathrm{Var}(\lfloor h \rceil)] &= \int_0^{\alpha} (\alpha \cdot h - h^2)\mathcal{CN}(h;\mu,\sigma)\, dh \\ &+ \int_{\alpha}^{\beta} \left((\beta-\alpha)(h-\alpha) - (h-\alpha)^2\right)\mathcal{CN}(h;\mu,\sigma)\, dh \\ &+ \int_{\beta}^{B} \left((B-\beta)(h-\beta) - (h-\beta)^2\right)\mathcal{CN}(h;\mu,\sigma)\, dh \end{aligned} \tag{17}$$

where $[\alpha, \beta]$ are the tunable edges of the central bin (see Figure 8-B). Given this expected variance in Eq. (17), we optimize the boundaries $[\alpha, \beta]$ that minimize the variance due to SR. This can be done using standard numerical solvers implemented in Python.

4.3 Experiments and Results

Data: Experiments are performed on two large-scale graph benchmark datasets for inductive learning tasks. The open graph benchmark (OGB) Arxiv dataset [44] consisting of graph with $\approx 170k$ nodes and $> 1M$ edges, and the Flickr dataset [45] consisting of $\approx 90k$ nodes and $\approx 900k$ edges.
Experimental Setup: The GNNs used in our experiments use the popular GraphSAGE architecture [37] implemented in Pytorch [46], which is also the baseline model with no activation compression i.e., operating in `FP32` precision. EXACT is used in `INT2` precision and $D/R = 8$ as the second baseline which uses extreme compression. We experiment our proposed compression methods in `INT2` precision and different group sizes $G/R = [2, 4, 8, 16, 32, 64]$ to demonstrate the influence of block-wise quantization. To keep the dimensionality proportion between the GNN layers, we scale the dimensionality of each layer equally when performing grouping, hence the block size is presented using the G/R. The influence of variance minimization (VM) on the test performance is also reported.
Results: Performance of the baseline methods and different configurations of the method presented in this work for two datasets are reported in Table 4. The most astonishing trend is that there is no noticeable difference in test performance on both datasets, across all models, even with extreme quantization (`INT2`) and any of the reported block sizes. With our proposed method there is a further improvement in memory consumption compared with EXACT by about 15% (97% with baseline `FP32`) and about 8% (97% with baseline `FP32`) for the Arxiv and Flickr datasets, respectively, when using the largest block size (G/R=64). We also gain a small speedup in training time per epoch: 5% for Arxiv, and 2.5% for Flickr, compared to EXACT.
Use of clipped normal distribution in Eq. (14) to model the activation maps is better than uniform distribution. This is captured using the Jensen-Shannon divergence measure, reported in Table 5 where we observe that for all layers, in both datasets, the distance to the observed distribution is smaller for clipped normal distribution.
Variance minimization does indeed decrease the variance induced by SR when performed with EXACT (Table 5), but shows no change in performance (Table 4).

Based on the experiments and the results in Table 4, we notice that block-wise quantization of activation

Dataset	Quant.	G/R	Accuracy ↑	S (e/s) ↑	M(MB) ↓
Arxiv	FP32 [37]	–	71.95 ± 0.16	13.07	786.22
	INT2 [38]	–	71.16 ± 0.21	10.03	30.47
	INT2	2	71.16 ± 0.34	10.23	27.89
		4	71.17 ± 0.22	10.46	26.60
		8	71.21 ± 0.39	10.54	25.95
		16	71.01 ± 0.19	10.55	25.72
		32	70.87 ± 0.29	10.54	25.60
		64	71.28 ± 0.25	10.54	25.56
	INT2+VM	–	71.20 ± 0.19	9.16	30.47
Flickr	FP32[37]	–	51.81 ± 0.16	17.95	546.92
	INT2[38]	–	51.65 ± 0.23	11.26	20.39
	INT2	2	51.58 ± 0.24	11.38	19.54
		4	51.57 ± 0.29	11.50	19.12
		8	51.60 ± 0.25	11.55	18.95
		16	51.65 ± 0.21	11.54	18.86
		32	51.61 ± 0.19	11.53	18.84
		64	51.72 ± 0.24	11.53	18.84
	INT2+VM	–	51.71 ± 0.18	10.78	20.39

Table 4: *Performance of block-wise quantization with $D/R = 8$, different quantization precision (*FP32, INT2*), block size (G), and with variance minimization (VM). We report the following metrics on the* Arxiv *[44] and* Flickr *[45] datasets: accuracy (%), speed (S) reported as epochs/second and memory (M) consumption in MB. Standard deviations of test accuracy is computed over 10 runs.*

Dataset	Layer	R	$\mathcal{U}$	$\mathcal{CN}$	Var. Reduction (%)
Arxiv	layer 1	16	0.0495	0.0213	3.17
	layer 2	16	0.0446	0.0016	2.09
	layer 3	16	0.0451	0.0041	2.19
Flickr	layer 1	63	0.0674	0.0017	6.14
	layer 2	32	0.0504	0.0033	4.37

Table 5: *Jensen-Shannon divergence measure for Uniform and Clipped Normal distributions compared to the normalized activations $\bar{\mathbf{h}}$ at each layer of the GNN for Arxiv and Flickr datasets. In all cases we see a smaller divergence measure between the clipped normal and the empiricial distribution of activation maps.*

maps on top of random projection and SR yields a further reduction in memory consumption and a small speedup in runtime. Increasing block size does not hamper the test performance but progressively yields further reduction in memory consumption.

Activation maps in GNNs are not uniformly distributed; we demonstrated this using empirical visualizations in Figure 9. We quantified this using the clipped normal distribution which had a smaller Jensen-Shannon divergence to the observed distribution, as seen in Table 5. This implies that using uniform quantization bin width could be sub-optimal. We presented an extension to stochastic rounding that accounts for variable bin widths in Eq. (15). The influence on quantization variance using Eq. (16) visualized in Figure 10 clearly demonstrates the value of using non-uniform bin widths.

Limitations: The compute overhead even with the proposed modifications do not fully recover the reduction in speedup compared to the baseline i.e., when using FP32. While the variance estimation improvement introduced by modelling the activation maps with clipped normal distribution better models the activation maps, minimizing the variance of SR under this distribution does not yield a noticeable improvement in test performance. This could simply be due to the fact that the overall drop in performance even with block-wise quantization is small, and there is no room for further improvement. The software implementations of the quantization and variance minimization strategies are not highly optimized and

there is room for further fine-tuning.

5 Conclusion

This deliverable documents three unique methods within the domain of efficient machine learning, carried out in relation to T3.2 within the SustainML project. Each method has been designed to enhance the efficiency of the training process, but they do so in different ways, addressing different aspects of the process.

The first method focuses on the volume of training data, the second method addresses the number of labels and the third method focuses on the memory required for training.

Each of these methods represents a significant improvement over existing methods. They have been designed to be easily integrated into existing machine learning frameworks, making them highly practical and usable. Together, these three methods represent a significant step forward in the quest for more efficient machine learning.

References

[1] Tong Chen and Raghavendra Selvan. "Is Adversarial Training with Compressed Datasets Effective?" In: *arXiv preprint arXiv:2402.05675* (2024).

[2] Bo Zhao and Hakan Bilen. "Dataset Condensation with Distribution Matching". In: *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2023, pp. 6503–6512.

[3] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. "Dataset Condensation with Gradient Matching". In: *International Conference on Learning Representations*. 2021.

[4] George Cazenavette et al. "Dataset Distillation by Matching Training Trajectories". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.

[5] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. "Coresets for Data-efficient Training of Machine Learning Models". In: *Proceedings of the 37th International Conference on Machine Learning*. 2020, pp. 6950–6960.

[6] Krishnateja Killamsetty et al. "GRAD-MATCH: Gradient Matching based Data Subset Selection for Efficient Deep Model Training". In: *Proceedings of the 38th International Conference on Machine Learning*. 2021, pp. 5464–5474.

[7] Rishabh Iyer et al. "Submodular combinatorial information measures with applications in machine learning". In: *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*. PMLR, 16–19 Mar 2021.

[8] Francesco Croce and Matthias Hein. "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks". In: *Proceedings of the 37th International Conference on Machine Learning*. 2020, pp. 2206–2216.

[9] Francesco Croce et al. "RobustBench: a standardized adversarial robustness benchmark". In: *arXiv preprint arXiv:2010.09670* (2020).

[10] Justin Cui et al. "DC-BENCH: Dataset Condensation Benchmark". In: *arXiv preprint arXiv:2207.09639* (2022).

[11] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Instance normalization: The missing ingredient for fast stylization". In: *arXiv preprint arXiv:1607.08022* (2016).

[12] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[13] Ting Chen et al. "A simple framework for contrastive learning of visual representations". In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[14] Jiahao Xie et al. "Unsupervised Object-Level Representation Learning from Scene Images". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer et al. 2021. URL: https://openreview.net/forum?id=X2K8KVEaAXG.

[15] Alexei Baevski et al. "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 12449–12460. URL: https://proceedings.neurips.cc/paper/2020/file/92d1e1eb1cd6f9fba3227870bb6d7f07-Paper.pdf.

[16] Xinpeng Xie et al. "Instance-aware self-supervised learning for nuclei segmentation". In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2020, pp. 341–350.

[17] Mehdi Noroozi and Paolo Favaro. "Unsupervised learning of visual representations by solving jigsaw puzzles". In: *European conference on computer vision*. Springer. 2016, pp. 69–84.

[18] Mihir Sahasrabudhe et al. "Self-supervised nuclei segmentation in histopathological images using attention". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2020, pp. 393–402.

[19] Ishan Misra and Laurens van der Maaten. "Self-supervised learning of pretext-invariant representations". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 6707–6717.

[20] Neeraj Kumar et al. "A Dataset and a Technique for Generalized Nuclear Segmentation for Computational Pathology". In: *IEEE Transactions on Medical Imaging* 36.7 (2017), pp. 1550–1560. DOI: 10.1109/TMI.2017.2677499.

[21] Simon Graham et al. "Hover-net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images". In: *Medical Image Analysis* 58 (2019), p. 101563.

[22] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035.

[23] William Falcon and The PyTorch Lightning team. *PyTorch Lightning*. Mar. 2019. DOI: 10.5281/zenodo.3828935. URL: https://www.pytorchlightning.ai.

[24] Franco Scarselli et al. "The graph neural network model". In: *IEEE transactions on Neural Networks* 20 (2008).

[25] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *International Conference on Learning Representations (ICLR)*. 2017.

[26] Tien Huu Do et al. "Matrix completion with variational graph autoencoders: Application in hyperlocal air quality inference". In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019.

[27] Zhongyuan Zhao et al. "Distributed scheduling using graph neural networks". In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2021.

[28] Panagiotis Tzirakis, Anurag Kumar, and Jacob Donley. "Multi-channel speech enhancement using graph neural networks". In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2021.

[29] Juan Cervino, Luana Ruiz, and Alejandro Ribeiro. "Training Graph Neural Networks on Growing Stochastic Graphs". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2023.

[30] Keyu Duan et al. "A comprehensive study on large-scale graph training: Benchmarking and rethinking". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2022).

[31] Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. *Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models*. ICML Workshop on Challenges in Deploying and monitoring Machine Learning Systems. arXiv:2007.03051. July 2020.

[32] Jaime Sevilla et al. "Compute trends across three eras of machine learning". In: *arXiv preprint arXiv:2202.05924* (2022).

[33] Brian R Bartoldson, Bhavya Kailkhura, and Davis Blalock. "Compute-Efficient Deep Learning: Algorithmic Trends and Opportunities". In: *Journal of Machine Learning Research* 24 (2023), pp. 1–77.

[34] Dustin Wright et al. "Efficiency is Not Enough: A Critical Perspective of Environmentally Sustainable AI". In: *Arxiv* (2023).

[35] Itay Hubara et al. "Binarized neural networks". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2016).

[36] Jianfei Chen et al. "Actnn: Reducing training memory footprint via 2-bit activation compressed training". In: *International Conference on Machine Learning (ICML)*. 2021.

[37] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs". In: *Advances in neural information processing systems (NeurIPS)* (2017).

[38] Zirui Liu et al. "EXACT: Scalable Graph Neural Networks Training via Extreme Activation Compression". In: *International Conference on Learning Representations (ICLR)*. 2022.

[39] Tim Dettmers et al. "8-bit Optimizers via Block-wise Quantization". In: *International Conference on Learning Representations (ICLR)*. 2022.

[40] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. "Binaryconnect: Training deep neural networks with binary weights during propagations". In: *Advances in neural information processing systems (NeurIPS)* (2015).

[41] Lu Xia et al. *Improved stochastic rounding*. Arxiv. 2020.

[42] Dimitris Achlioptas. “Database-Friendly Random Projections”. In: *Symposium on Principles of Database Systems (PODS)*. 2001.
[43] Tim Dettmers. “8-bit approximations for parallelism in deep learning”. In: *International Conference on Learning Representations (ICLR)*. 2016.
[44] Weihua Hu et al. “Open graph benchmark: Datasets for machine learning on graphs”. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2020).
[45] Matthias Fey and Jan Eric Lenssen. *Fast Graph Representation Learning with PyTorch Geometric*. ICLR Workshop on Representation Learning on Graphs and Manifolds (RLGM). 2019.
[46] Adam Paszke et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2019).