Decoupling Structural and Quantitative Knowledge in ReLU-based Deep Neural Networks

José Duato
jduato@qsimov.com
Qsimov Quantum Computing S.L.
Talavera de la Reina, Spain

Jose I. Mestre
jmiravet@uji.es
Universitat Jaume I
Castelló de la Plana, Spain

Manuel F. Dolz
dolzm@uji.es
Universitat Jaume I
Castelló de la Plana, Spain

Enrique S. Quintana-Ortí
quintana@disca.upv.es
Universitat Politècnica de València
Valencia, Spain

José Cano
jose.canoreyes@glasgow.ac.uk
University of Glasgow
Glasgow, United Kingdom

Abstract

The relentless growth of artificial intelligence applications has led to substantial economic and environmental costs associated with training deep neural networks (DNNs). Recognizing the challenges in further optimizing conventional DNN training, in this paper we propose a novel approach that decouples structural information (non-linear functions) from quantitative knowledge (model parameters), and provide strong experimental evidence to demonstrate that these two types of knowledge can be trained independently. We evidence that ReLU-based DNNs can be deployed as globally linear models, from which different parts of the DNN are active for each sample, thus emulating the piece-wise linear outputs generated by ReLU activation functions. Leveraging this linear model foundation, this kind of DNN supports various objectives, including faster re-training times and combining multiple copies trained on different datasets for incremental and federated re-training.

CCS Concepts: • **Computing methodologies** → **Neural networks**; **Learning linear models**.

Keywords: Artificial Intelligence, Deep Neural Networks, ReLU function, Structural and Quantitative Knowledge

ACM Reference Format:
José Duato, Jose I. Mestre, Manuel F. Dolz, Enrique S. Quintana-Ortí, and José Cano. 2025. Decoupling Structural and Quantitative Knowledge in ReLU-based Deep Neural Networks. In *The 5th Workshop on Machine Learning and Systems (EuroMLSys '25), March 30-April 3, 2025, Rotterdam, Netherlands.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3721146.3721950

EuroMLSys '25, Rotterdam, Netherlands

ACM ISBN 979-8-4007-1538-9/2025/03
https://doi.org/10.1145/3721146.3721950

1 Introduction

Deep Learning (DL) has become a highly effective solution for addressing a growing number of challenges. Nevertheless, the time and energy costs to train complex Deep Neural Networks (DNNs) are growing over time [2, 5], outpacing the computational advances contributed by hardware accelerators in recent years [10, 12, 17].

The massive training costs of DNNs stem from several factors. To achieve accurate recognition of complex features, DNNs must undergo extensive training on large datasets with sophisticated architectures that have numerous tunable parameters. Consequently, training complex DNNs requires a large number of arithmetic operations, typically using the conventional Stochastic Gradient Descent (SGD) or any of its variants. This is partly due to challenges like the vanishing gradient problem, which leads to slow convergence, and the need for techniques to avoid getting trapped in local minima. While non-linearities in DNNs are crucial for capturing real-world phenomena, they also increase the computational demands and training time for large networks.

In addition to the previous discussion, for many applications data evolves dynamically over time, necessitating re-training. The consequence is that, since DNNs learn via some method that minimizes a loss function, all the training samples must be processed every time the DNN is re-trained. (Otherwise, re-training usually leads to forgetting the contribution of the oldest samples.) Therefore, re-training is costly and difficult to implement without processing older samples or without disruption, asking for a fundamentally different approach to that employed in conventional DNN training.

Focusing on these problems, we propose a linear model that emulates the piece-wise linear behavior of DNNs that use Rectified Linear Units (ReLUs) as activation functions. The method activates a specific subset of the linear model for each sample, separating the information needed to select the active subset (referred to as Structural Knowledge (SK)) from the full set of model parameters (referred to as Quantitative Knowledge (QK)). This design enables simpler, faster, and more environmentally-friendly training and re-training.

A key feature is that the QK excludes activation functions, containing only linear relationships, which can open new advances in order to significantly reduce the cost of re-training.

The pivotal question we address in this work is whether the SK can be kept unmodified when re-training the QK of a DNN without significant loss in validation accuracy. In doing so, we make the following contributions:

- We define some fundamental DNN concepts related to active paths and activation patterns that form the foundation of a Proof-of-Concept (PoC) Artificial Intelligence (AI) system.
- We motivate our hypothesis for decoupling the SK and QK of a DNN. Basically, SK refers to the ability of the network to activate and deactivate paths, while QK refers to the numerical weight and bias values.
- We validate our hypothesis through a PoC AI system that separates the SK and QK, comparing the evolution of both types of knowledge along the training process, using LeNet-5, AlexNet and VGG8 convolutional DNNs on the CIFAR-10 dataset.

The rest of the paper is organized as follows. Section 2 provides an overview of related work in the field. Section 3 introduces several fundamental concepts and defines the notions of SK and QK. In Section 4, we explore the separation of SK and QK within DNNs, starting with a PoC AI system. Section 5 presents an evaluation and comparative analysis of the convergence of different DNNs using the knowledge-separated system, as opposed to the traditional approach. Finally, Section 6 concludes with some remarks and discusses future research directions.

2 Related Work

This section provides an overview of the state-of-the-art and the challenges in DNN re-training algorithms, emphasizing efficient methods and the incremental learning process.

Incremental learning is a crucial task in many key applications with evolving data [15]. Significant examples arise in financial applications such as stock market forecasting [4], algorithmic trading [9], credit risk assessment [16], portfolio allocation [1], and asset pricing [20], as well as insurance risk assessment [13], preventive maintenance [7], and fine-tuning in natural language processing [21]. In these scenarios, models need to be periodically fine-tuned or re-trained using both old and new data. However, effective solutions for incremental training remain an open challenge.

Transfer learning is a widely used technique that establishes a solid foundation by pre-training the DNN on a different use case and dataset, and then refining its knowledge with a specific dataset, significantly reducing training time. However, a key challenge with this approach is catastrophic forgetting, where previously acquired knowledge is lost [8, 14]. Several strategies have been proposed to address this issue, such as memory replay, which combines previous and new data during re-training or periodic re-trains using all prior data [18, 22]; parameter adjustment based on importance [14]; and dynamic adjustments to the DNN architecture [19, 24].

A key factor in DNN training is the choice of activation functions. The Gated Linear Unit (GLU) improves upon traditional functions such as ReLU by introducing a gating mechanism, defined as $GLU(a, b) = a \times \sigma(b)$, where σ is the sigmoid function. This mechanism allows dynamic control of the information flow, mitigating the vanishing gradients and enhancing the model's ability to capture complex relationships [6, 23]. Unlike ReLU, which can suffer from dead neurons, GLUs improve gradient flow during backpropagation, leading to more stable and efficient training.

In conclusion, important gaps still remain in the design of efficient re-training methods for DNNs and in addressing challenges of incremental learning such as catastrophic forgetting. This work proposes a radically different solution that separates the structural and quantitative knowledge of the Neural Network (NN) to circumvent the aforementioned problems.

3 Structural versus Quantitative Knowledge

In this section, we introduce a few concepts that are used to formulate our hypothesis of separating SK and QK.

3.1 Fundamental concepts

In the following, we use the simple DNN displayed in Figure 1 to define a series of concepts.

Activation function. A neuron computes a non-linear activation function of the sum of the weighted neuron inputs plus a bias. The non-linearity enables the NN to learn complex relationships among the data. The PoC presented in this work assumes ReLU as the activation function.

Active neuron refers to a neuron whose activation function produces a positive output for a given input sample. An inactive neuron produces a zero output, indicating inactivity.

Activation pattern is the set of active neurons for a given input sample [11]. In practice, samples with similar features are expected to produce the same pattern.

Active path is a sequence of active neurons in consecutive layers, with each active neuron being connected to the next one in the path. The activation pattern for a specific input sample is defined by the collection of paths it activates. Each input-output pair may be connected via multiple active paths. We distinguish between **full active paths**, which connect a specific input-output pair; and **bias active paths**, which connect the bias from a neuron to a given output; see Figure 1

Path weight is defined as the product of the weights along a path (including the bias of the source neuron in the product for bias paths). For a given sample, an output of a NN can be expressed as the sum of contributions from all active paths

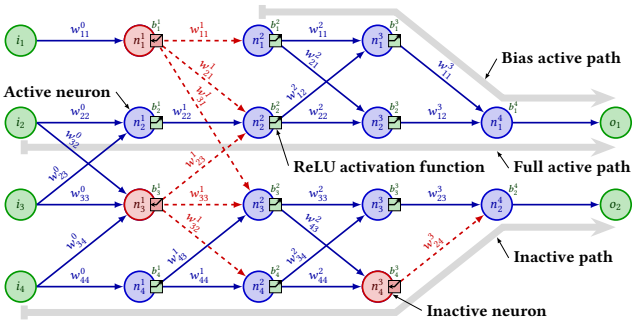


Figure 1. Simple NN with active/inactive neurons and paths. The neurons (n), the weights (w) and the bias (b) are annotated with the layer number (superscript) and neuron identifier within the layer (subscript). In this example, the blue neurons are active and the red neurons inactive.

leading to it, where each contribution is either the product of the corresponding path weight times the input value for a full path or just the path weight for a bias path. For example, the value of the output o_1 in the NN in Figure 1 is given by:

$$\begin{aligned} o_1 = b_1^4 \;+\; & w_{11}^3 b_1^3 + \boxed{w_{11}^3 w_{11}^2 b_1^2} + w_{11}^3 w_{12}^2 b_2^2 + w_{11}^3 w_{12}^2 w_{22}^1 b_2^1 \\ + \; & \boxed{w_{11}^3 w_{12}^2 w_{22}^1 w_{22}^0} i_2 + w_{11}^3 w_{12}^2 w_{22}^1 w_{23}^0 i_3 \\ + \; & w_{12}^3 b_2^3 + \boxed{w_{12}^3 w_{21}^2 b_1^2} + w_{12}^3 w_{22}^2 b_2^2 + w_{12}^3 w_{22}^2 w_{22}^1 b_2^1 \\ + \; & \boxed{w_{12}^3 w_{22}^2 w_{22}^1 w_{22}^0} i_2 + w_{12}^3 w_{22}^2 w_{22}^1 w_{23}^0 i_3. \end{aligned}$$

In this expression, we can observe the contribution of the two full paths from i_2 to o_1, with the corresponding path weights given by the product of the weights along the paths (marked with blue boxes) as well as those of the bias paths from n_1^2 to o_1 (marked with green boxes). The above expression also exposes that the inactive paths have no contribution to the output.

In general, the information in the NN flows along the paths and, for a given sample, the associated output value corresponds to the sum of the contributions from the active paths leading to that output, regardless of whether or not these paths partially overlap. The full paths model the contribution of the inputs to the outputs; the bias paths model the contribution of the neuron biases to the outputs.

3.2 Structural and quantitative knowledge

The proposed PoC AI system formulates the hypothesis that the knowledge learned by a trained NN can be separated into two components: **Structural Knowledge (SK)** and **Quantitative Knowledge (QK)**. **SK** refers to the information that the NN has acquired to activate or deactivate a collection of paths, thus generating specific activation patterns for specific input samples. It encompasses the ability learned by the NN to control the information flow through its layers via the activation functions, resulting in a distinct activation pattern for each specific input sample. Figure 2 shows this separation from a mathematical point of view; the SK can be represented by the values of the derivatives of the neuron activation functions for the different samples. The SK allows the NN to effectively leverage its architecture to discriminate among samples.

In contrast, the **QK** only encompasses the numerical values of the neuron weights and biases learnt by the NN that allow it to generate accurate predictions for specific input samples, given the corresponding activation patterns. The QK does not involve the processing of activation functions (including their derivatives), since the active paths (and active neurons) are determined by the SK. Note that when considering the set of active paths for a given sample, the NN can be regarded as a *linear function*.

In general, combining SK and QK enables a NN to effectively model complex non-linear systems. By decoupling the acquisition of the SK from that of the QK, it becomes possible 1) to design new systems that initially acquire the SK; 2) to fine-tune the QK while avoiding the complexity of dealing with non-linearities; and 3) to re-train the NN multiple times, keeping the SK constant and updating only the QK, yielding a lower computational cost.

4 Proof-of-Concept AI System

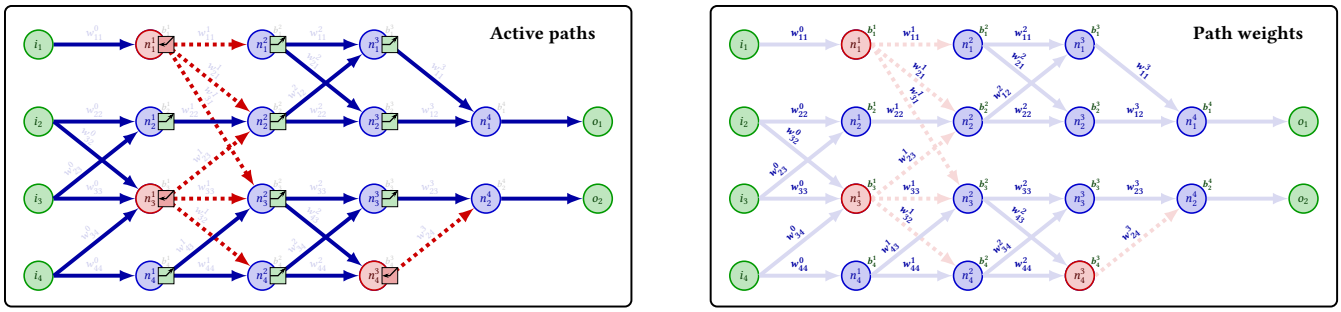
This section presents a PoC AI system that decouples SK and QK within a DNN, enabling us to re-train the QK while keeping constant the SK. We outline the key stages of the QK re-training process. The PoC uses a DNN-like structure and a re-training mechanism resembling the one in DNNs. This approach aims to validate the hypothesis that the two types of knowledge can indeed be decoupled.

4.1 System Definition

The goal is to reduce the amount of computations required to re-train a PoC AI system by decoupling SK and QK, preserving the SK while re-training only the QK. This is fundamentally different from traditional training, for which both types of knowledge are updated during re-training.

The new approach is viable because we introduce a strategy to decouple SK from QK and, as proved later, the SK exhibits a high degree of stability across successive re-training operations with additional samples [11]. The reason for this behavior lies in that the high-level features of the samples are typically captured in the early stages of the training process, while the refinement of the low-level features occurs in later stages. The stabilization of the activation patterns is thus an indication that the high-level features have been effectively learned. In contrast, the fine-tuning of weights and biases, constituting the QK, enables the network to reduce the loss function, once the activation patterns are mostly stabilized.

Starting from a given NN, the PoC AI system builds two copies of such a network. The first copy is used for computing the derivatives of the neuron activation functions and



(a) Structural knowledge. **(b)** Quantitative knowledge.

Figure 2. Comparison of structural and quantitative knowledge for the example neural network in Figure 1.

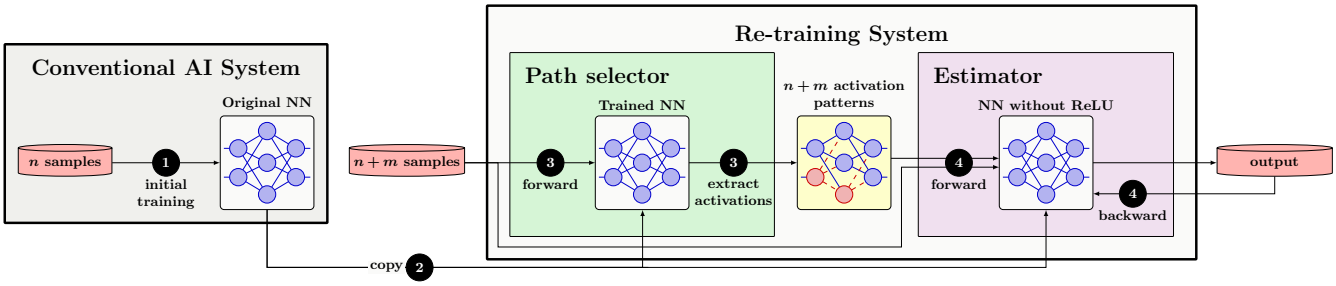


Figure 3. Diagram of the PoC AI system, including the initial training, path selector and the estimator.

is not updated when re-training the system. Those derivatives (whose values for the ReLU are either 0 or 1) indicate the activation status of the corresponding neurons, thereby forming sequences of active neurons known as active paths. Hence, this copy will be referred to as the **path selector**. The second copy uses the derivatives computed by the first one for both inference and re-training, and its tunable parameters are updated during re-training. Since the second copy is used to estimate the system output values, we will refer to it as the **estimator**.

In summary, the PoC AI system has the following two modules: The **path selector** maintains the SK and is responsible for obtaining the set of active paths for a sample. This information is next fed to the estimator. The **estimator** contains the QK and is responsible for delivering accurate inference results for the input sample using the subset of parameters related to the activation pattern determined by the path selector.

4.2 Implementation

Next, we describe the experiments conducted during the initial setup of the PoC AI system and subsequent re-training iterations with additional samples. Figure 3 illustrates the workflow devised for the system, delineated into the following four phases:

Phase 1: Initial training. In this phase, the NN is initialized either from scratch or leveraging a previous version in case of re-training. Using an optimizer like SGD and a data subset of size n, the NN learns high-level features and sample distinctions through training a given number of epochs.

Phase 2: Initialization of path selector and estimator. After the initial training, the path selector receives a copy of the trained network, that will be frozen unless the SK becomes outdated. The estimator obtains a version of the trained NN without activation functions.

Phase 3: Activation pattern acquisition. Before re-training the estimator in Phase 4, the activation patterns for all samples, including n original inputs and m new ones, are obtained by performing inference using the static copy of the NN in the path selector.

Phase 4: Estimator re-training. The estimator undergoes knowledge-only re-training using the expanded set of $n + m$ samples and the activation patterns computed by the path selector in Phase 3. For each batch, a forward pass uses the activation patterns to determine neuron activity. The loss function computes deviations, and in the backward pass, the gradients are obtained using the derivatives extracted from the activation patterns. SGD is used to re-adjust estimator weights.

This re-training process implements the fundamental concepts of the new AI system, while being similar to the conventional DNN re-training process. In practice, this re-training can be significantly faster since the estimator does not include nonlinear activation functions.

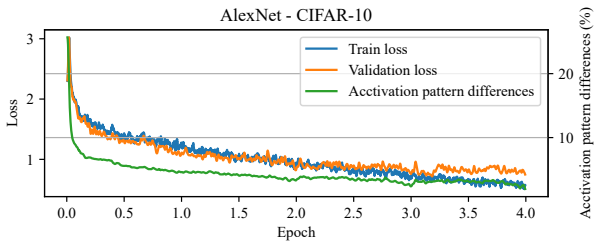


Figure 4. Activation pattern convergence for AlexNet using CIFAR-10.

5 System Validation

In order to validate the proposed re-training system, in this section we conduct several experiments using the NNs, LeNet, AlexNet and VGG8, on the CIFAR-10 dataset. The implementation leverages basic codes that incorporate the forward and backward methods from the PyDTNN training framework [3] while TensorFlow v2.6.2 was used for the initialization and training.

For the PoC hypothesis evaluation, initially each of these models was pre-trained using the conventional training approach for 200 epochs, with n =1,024 for LeNet and AlexNet, and 512 samples for VGG8. Additionally, to analyze the impact of the quality of the path selector on the accuracy of the estimator after the re-training process, the same experiments were repeated for the same models, pre-trained with 4,096 randomly selected samples from the dataset. For each subsequent re-training experiment with an increasing number of additional samples (m), our PoC AI system was initialized with two copies of the NN: one for the path selector and the other for the estimator. The path selector performed inference using the set of $n + m$ samples and recorded the activation pattern for each sample. Subsequently, the estimator re-trained the QK with $n + m$ samples, using the SGD optimizer for 50 epochs. All trainable layers, including convolutional and fully-connected layers, as well as batch normalization layers, were trained. Furthermore, the NN copy used by the estimator does not include activation functions. Instead, it relies on the activation patterns obtained from the path selector to determine the active and inactive neurons for each sample.

To compare the results of our PoC with traditional DNNs in consecutive re-training experiments with an increasing number of samples, we re-trained the same models using the traditional approach, with $n + m$ samples for 50 epochs. This allows us to assess the effectiveness of our approach against the conventional training method.

5.1 SK stabilization analysis

One of our hypotheses is that the SK stabilizes during the initial stages of the training process. This enables the preservation of the SK while re-training only the QK. To validate this hypothesis, we compare the variations in the activation patterns throughout the training process.

Figure 4 illustrates these variations alongside the training and validation loss at each step for AlexNet. The variations are measured as the difference between active and inactive paths for each sample in the validation dataset, quantifying the percentage of paths that have changed compared to the previously trained NN, using the same random seed for weight initialization and batch shuffling. As shown in the plot, the activation pattern stabilizes much earlier than the validation loss, even while the weights are still being updated. Thus, we can infer that the SK stabilizes within a few epochs, while the QK continues to evolve.

5.2 QK analysis with successive re-trainings

Figures 5 illustrates the validation accuracy and Categorical Cross Entropy (CCE) validation loss achieved for the re-training of all classification layers of LeNet-5, AlexNet and VGG8 using: 1) Traditional SGD (blue line); and 2) SGD-based QK-only re-training (red and orange lines).

For both LeNet and AlexNet, as new samples are added (from 1,024 to 32,768), the accuracy improves at a slightly slower rate when only the QK is re-trained. This was expected since the QK-only approach relies on a fixed SK obtained from a small number of samples (1,024). However, when the SK is re-initialized using 4,096 samples, the accuracy achieved is comparable to that of the traditional SGD algorithm. Nevertheless, it is important to note that the case that only uses 1,024 samples to obtain the SK still improves with the re-training step, even if the accuracy line falls below that of the traditional SGD algorithm. The curves of the CCE validation loss for both the traditional SGD and the QK-only re-training, using the SK obtained from 1,024 samples, exhibit similar behavior.

A similar trend is observed for the VGG8 model. However, due to its more complex architecture, the SK stabilizes with 512 training samples. Beyond that point, the QK re-training shows slower convergence compared to the traditional SGD-based approach. While positive results are achieved, a re-initialization of the SK may be needed to address obsolete knowledge. Notably, re-training with 4,096 samples yields nearly similar results to the SGD-based approach.

In summary, these experiments demonstrate the feasibility and effectiveness of the PoC AI system. The combination of the path selector module and the estimator has the potential to drastically reduce DNN training time compared to traditional methods, especially when frequent re-training is required. This advancement paves the way for optimizing the training process and achieving better results in DNN applications across various domains.

6 Concluding Remarks

In this work, we have presented and validated the hypothesis that SK and QK can be decoupled in ReLU-based DNNs through a novel PoC AI system, improving and reducing energy consumption performance compared to conventional

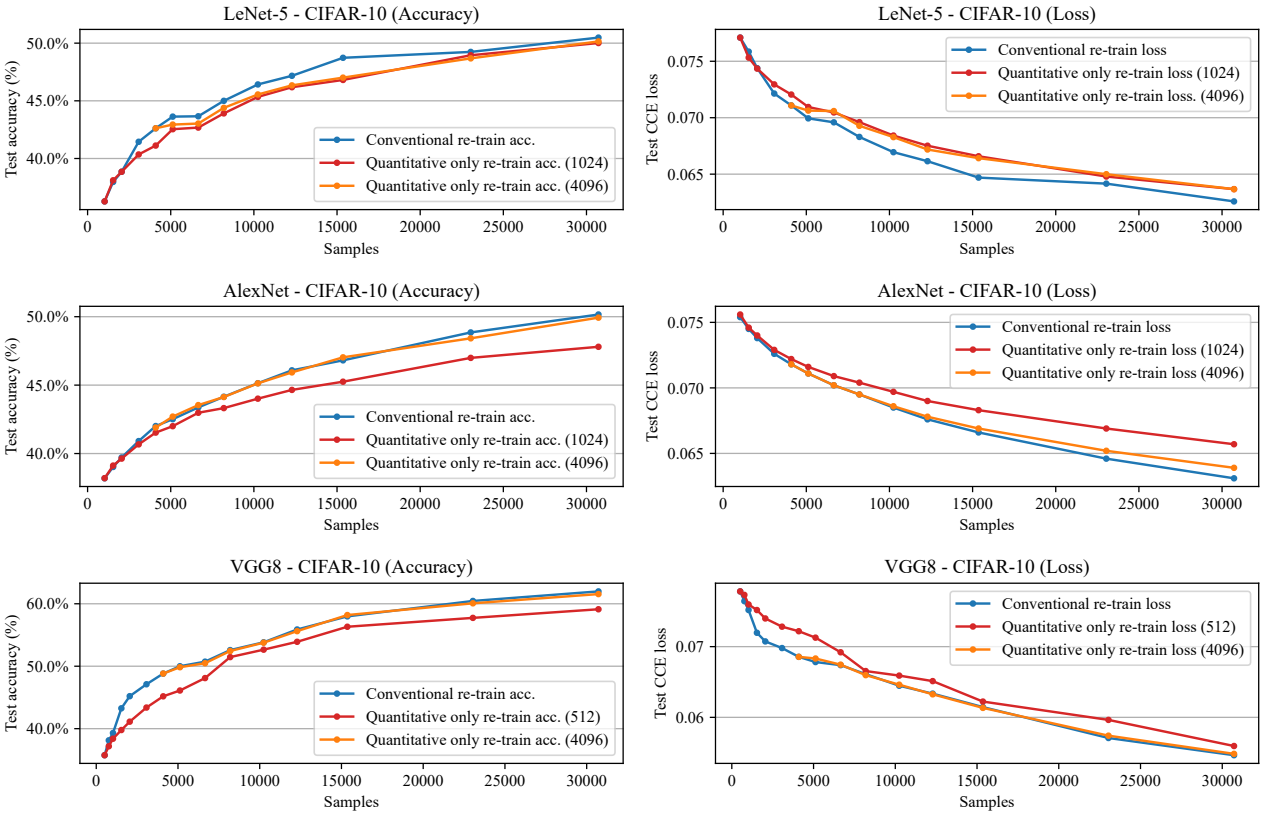


Figure 5. Validation accuracy (left column) and CCE loss (right column) with traditional SGD and quantitative-only re-training using LeNet-5 (top row), AlexNet (middle row), and VGG8 (bottom row) for the CIFAR-10 dataset.

DNN training and re-training. The system design includes a path selector, which determines the activation pattern for each sample; and an estimator, which fine-tunes the weights and biases of the active paths identified by the path selector for the samples in the dataset.

The experimental evaluation using AlexNet, LeNet-5 and VGG8 on the CIFAR-10 dataset offers valuable insights into the behavior and convergence of the proposed system and the consequences of decoupling the SK and the QK. First, we can confirm that during the initial training phase of the NN, the SK converges to a stable state faster than the QK, and this SK is sufficient to generate activation patterns that keep the whole model convergence rate close to that obtained by the original SGD-based algorithm. Moreover, when re-training with quantitative information only, the accuracy improvement rate was slightly lower compared to the traditional SGD-based re-training. This can be attributed both to keeping the SK constant and to obtaining it from a smaller number of samples. However, it is important to note that the quantitative-only approach showed continuous knowledge improvement as more samples were added. These results therefore demonstrate the high potential of updating only the QK for efficient re-training.

The experiments also revealed that by initializing the SK using a larger number of samples, the accuracy matches to that of the traditional SGD-based approach can be achieved. This finding emphasizes the significance of correctly selecting the dataset for the initial training of the path selector. On a related topic, we plan to analyze the effect of updating the SK in line with potentially evolving data to keep achieving high accuracy over time.

Building upon the promising results demonstrated in this PoC system, we are currently focused on implementing a fully functional AI system for efficient re-training with new samples without requiring full dataset processing. Our future work involves optimizing and comprehensively evaluating the system's performance, accuracy, and scalability, comparing it against traditional re-training approaches on diverse datasets and DNNs architectures. Special attention will be given to its adaptability towards evolving data distributions while preserving or improving model accuracy as the system scales to larger and more complex tasks.

Acknowledgments

This research was funded by the Qsimov Quantum Computing S.L. company through the Research and Development contract 10556/2022 "Development of alternative techniques

for the acceleration of deep neural network training" with Universitat Jaume I and by the project PID2023-146569NB-C22 supported by MCIU/AEI/10.13039/501100011033 and ERDF/UE. Manuel F. Dolz was supported by the Plan Gen–T grant CIDEXG/2022/013 of the *Generalitat Valenciana*. Jose I. Mestre was supported by the predoctoral grant ACIF/2021/281 of the *Generalitat Valenciana*.

References

[1] Saud Almahdi and Steve Y Yang. 2017. An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications* 87 (2017), 267–279.

[2] Dario Amodei and Danny Hernandez. 2018. AI and compute. https://openai.com/blog/ai-and-compute. Last accessed: April 2023.

[3] Sergio Barrachina, Adrián Castelló, Mar Catalán, Manuel F. Dolz, and Jose I. Mestre. 2021. A Flexible Research-Oriented Framework for Distributed Training of Deep Neural Networks. In *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 730–739. doi:10.1109/IPDPSW52791.2021.00110

[4] Sotirios P Chatzis, Vassilis Siakoulis, Anastasios Petropoulos, Evangelos Stavroulakis, and Nikos Vlachogiannakis. 2018. Forecasting stock market crisis events using deep and statistical machine learning techniques. *Expert Systems with Applications* 112 (2018), 353–371.

[5] NVIDIA Corp. 2022. NVIDIA H100 Tensor Core GPU Architecture. https://nvdam.widen.net/s/9bz6dw7dqr/gtc22-whitepaper-hopper. Last accessed: April 2023.

[6] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language Modeling with Gated Convolutional Networks. arXiv:arXiv:1612.08083

[7] Kevin Dick, Luke Russell, Yasmina Souley Dosso, Felix Kwamena, and James R Green. 2019. Deep learning for critical infrastructure resilience. *Journal of Infrastructure Systems* 25, 2 (2019), 05019003.

[8] Robert M. French. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences* 3, 4 (April 1999), 128–135. doi:10.1016/s1364-6613(99)01294-2

[9] Eduardo A Gerlein, Martin McGinnity, Ammar Belatreche, and Sonya Coleman. 2016. Evaluating machine learning classification for financial trading: An empirical approach. *Expert Systems with Applications* 54 (2016), 193–207.

[10] Perry Gibson, José Cano, Elliot. J. Crowley, Amos Storkey, and Michael O'Boyle. 2024. DLAS: A Conceptual Model for Across-Stack Deep Learning Acceleration. *ACM Transactions on Architecture and Code Optimization (TACO)* (2024).

[11] David Hartmann, Daniel Franzen, and Sebastian Brodehl. 2021. Studying the Evolution of Neural Activation Patterns During Training of Feed-Forward ReLU Networks. *Frontiers in Artificial Intelligence* 4 (2021). doi:10.3389/frai.2021.642374

[12] John L Hennessy and David A Patterson. 2019. A new golden age for computer architecture. *Commun. ACM* 62, 2 (2019), 48–60.

[13] Kevin H Kelley, Lisa M Fontanetta, Mark Heintzman, and Nikki Pereira. 2018. Artificial intelligence: Implications for social inflation and insurance. *Risk Management and Insurance Review* 21, 3 (2018), 373–387.

[14] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.

[15] Diego Klabjan and Xiaofeng Zhu. 2020. Neural Network Retraining for Model Serving. arXiv:2004.14203 [cs.LG]

[16] Xiaomeng Ma and Shuliang Lv. 2019. Financial credit risk prediction in internet finance driven by machine learning. *Neural Computing and Applications* 31 (2019), 8359–8367.

[17] Albert Reuther, Peter Michaleas, Michael Jones, Vijay Gadepally, Siddharth Samsi, and Jeremy Kepner. 2021. AI accelerator survey and trends. In *2021 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 1–9.

[18] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. 2019. Experience replay for continual learning. *Advances in Neural Information Processing Sys.* 32 (2019).

[19] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671* (2016).

[20] Jantje Sönksen. 2022. Machine Learning for Asset Pricing. *Econometrics with Machine Learning* (2022), 337–366.

[21] Robert Tinn, Hao Cheng, Yu Gu, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2023. Fine-tuning large neural language models for biomedical natural language processing. *Patterns* 4, 4 (2023).

[22] Gido M Van de Ven, Hava T Siegelmann, and Andreas S Tolias. 2020. Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications* 11, 1 (2020), 4069.

[23] Joel Veness, Tor Lattimore, David Budden, Avishkar Bhoopchand, Christopher Mattern, Agnieszka Grabska-Barwinska, Eren Sezener, Jianan Wang, Peter Toth, Simon Schmitt, and Marcus Hutter. 2021. Gated Linear Networks. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 11 (May 2021), 10015–10023. doi:10.1609/aaai.v35i11.17202

[24] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. 2018. Slimmable neural networks. *arXiv preprint arXiv:1812.08928* (2018).