Projected Stochastic Gradient Descent with Quantum Annealed Binary Gradients

Maximilian Krahn[1,2]
Michele Sasdelli[3]
Fengyi Yang[3]
Vladislav Golyanik[4]
Juho Kannala[1]
Tat-Jun Chin[3]
Tolga Birdal[2]

[1] Aalto University, Espoo, Finland
[2] Imperial College London, London, UK
[3] University of Adelaide, Adelaide, Australia
[4] MPI for Informatics, Saarbrücken, Germany

Appendix

This appendix supplements our paper with the additional material referred to in the main text. This includes the proofs of the propositions (specifically the usage of the binary map as gradient approximation, the convergence and P-SBGD, and the QUBO derivation), supporting evidence of the directionality and additional experimental evaluations, further experiments on the UCI adult dataset, MNIST and analysis of the D-Wave QUBO problems. In addition, we furnish extra information on related work, implementation specifics, and function definitions.

1 Theoretical Results

In Tab. 1 we list the most important notations for the paper and appendix.

Notation	Explanation
f	$\mathbb{R}^n \rightarrow \mathbb{R}^m$any function (neural networks)
n	dimension of the input of f
m	dimension of the output of f
$\mathbf{y}$	evaluation of f at $\mathbf{x}$; $\mathbf{y} = f(\mathbf{x})$
$E(\mathbf{y})$	$\mathbb{R}^n \rightarrow \mathbb{R}$ differentiable, Lipschitz continuous function
$E_f(\mathbf{x})$	$\mathbb{R}^n \rightarrow \mathbb{R}$ loss function; $E_f(\mathbf{x}) := E(f(\mathbf{x}))$
K_i	Lipschitz constant for the i^{th} dimension of E_f
$\bar{K}$	$\frac{1}{n}\sum_{i=1}^{n} K_i$
t	current iteration
T	number of total iterations
$\{\alpha_t\}_t$	series of learning rates,
$\{\mathbf{x}^t\}_t$	series of iterates, *i.e.*parameters updated over iterations.
$\tilde{\nabla}_{\mathbf{x}} E_f(\mathbf{x})$	*stochastic gradient* w.r.t. $\mathbf{x}$
$\mathbf{j}_i^t$	i^{th} column of the Jacobian map $\mathbb{R}^{n\times m} \ni \mathbf{J}^t := \nabla_{\mathbf{x}} \mathbf{y}$ evaluated at $\mathbf{x}^t$
$\mathbf{Z}^t$	in $\mathbb{R}^{n\times m}$; *normalized Jacobian*
$\mathbf{z}_i^t$	in $\mathbb{R}^n$ a column vector fo the normalized Jacobian: $\mathbf{j}_i^t / \|\mathbf{j}_i^t\|^2$

Table 1: Notations

1.1 Approximating Euclidean gradients with binary gradients

Before we start with the proof we restate Prop. 1 from the paper.

Proposition 1. Let $\hat{\mathbf{\Pi}}_{\mathbf{U}} : \mathbb{R}^m \rightarrow \mathbb{R}^n$ denote the *relaxed* or continuous version of our projection map. Replacing $\mathbf{U}$ by a normalized gradient w.r.t. $\mathbf{y}$ and using $\tilde{\nabla}_{\mathbf{y}} E_f(\mathbf{x})$ as an input, $\hat{\mathbf{\Pi}}_{\mathbf{U}}$ satisfies:

$$\hat{\mathbf{\Pi}}_{\mathbf{Z}^t}(\tilde{\nabla}_{\mathbf{y}} E_f(\mathbf{x})) = \underset{\mathbf{b}\in\mathbb{R}^n}{\arg\min} \|\tilde{\nabla}_{\mathbf{x}} E_f(\mathbf{x})|_{\mathbf{x}^t} - \mathbf{b}\|_2^2. \tag{1}$$

Proof. We now give the proof for the continuous map given in Eq (1), where we optimise over $\mathbb{R}^n$. We now expand Eq (1):

$$\frac{\partial E}{\partial \mathbf{x}} = \underset{\mathbf{g}\in\mathbb{R}^n}{\arg\min} \sum_{i=1}^{m} |\frac{\partial E}{\partial y_i} - \mathbf{g}^\top \mathbf{z}_i^t|^2. \tag{2}$$

This statement is true if every term in the sum is 0. If this holds, we can replace $\mathbf{g}$ by $\frac{\partial E}{\partial \mathbf{x}}$ and seek to have

$$\frac{\partial E}{\partial y_i} - \frac{\partial E}{\partial \mathbf{x}}^\top \mathbf{z}_i^t = 0. \tag{3}$$

Assuming differentiability and compositionally, we can use the chain rule to write

$$\frac{\partial E}{\partial \mathbf{x}} = \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial \mathbf{x}}. \tag{4}$$

By Eq (4), and the definition of $\mathbf{z}_i^t$ we can reformulate the left hand side of Eq (3) as follows:

$$\frac{\partial E}{\partial y_i} - \left(\frac{\partial E}{\partial \mathbf{x}}\right)^\top \left(\frac{\partial y_i}{\partial \mathbf{x}} / \| \frac{\partial y_i}{\partial \mathbf{x}} \|^2\right) = \frac{\partial E}{\partial y_i} - \frac{\partial E}{\partial y_i}\left(\frac{\partial y_i}{\partial \mathbf{x}}\right)^\top \left(\frac{\partial y_i}{\partial \mathbf{x}} / \| \frac{\partial y_i}{\partial \mathbf{x}} \|^2\right) \tag{5}$$

$$= \frac{\partial E}{\partial y_i}\left(1 - \left(\frac{\partial y_i}{\partial \mathbf{x}} / \| \frac{\partial y_i}{\partial \mathbf{x}} \|\right)^\top \left(\frac{\partial y_i}{\partial \mathbf{x}} / \| \frac{\partial y_i}{\partial \mathbf{x}} \|\right)\right) \tag{6}$$

$$= \frac{\partial E}{\partial y_i}(1-1) = 0 \qquad \text{for all } i. \tag{7}$$

Alternatively, we can arrive at the same proof by using the fact that Jacobians map tangent vectors, *i.e.*, local isomorphisms between the tangent spaces of input and output points. Finally, by satisfying Eq (7) for all i, we see that in the continuous space, Eq (2) is fulfilled. □

Note that we can use this mapping to calculate binary updates also for multidimensional function parameters, such as matrices. This can easily be done by vectorising the parameter matrix into m vectors and computing m times the mapping of the gradient of a parameter vector.

1.2 Proof of the fixed point theorem

Before we prove the fixed point theorem, we present the definition of a fixed point and reiterating the theorem.

Definition 1 (Fixed point)**.** $\mathbf{s} \in \{\pm 1\}^n$ is a fixed point for P-SBGD, if $\mathbf{s}^0 = \mathbf{s}$ in Dfn. 2 implies that $\mathbf{s}^t = \mathbf{s}$ for all $t = 1, 2, \dots$, where $\mathbf{s}^t = \text{sign}(\mathbf{x}^t)$. We say that P-SBGD converges if there exists $t < \infty$ such that $\mathbf{s}^t$ is a fixed point.

Theorem 1 (Fixed point of P-SBGD)**.** Let $\mathbf{Z}^s$ be the normalised Jacobian of the function f at $\mathbf{s} \in \{\pm 1\}^n$. Let $\mathbf{x}^t$ be the iterative produced by Eq. 7 from the paper. We further assume, that for $i \in [n]$ $(\nabla_\mathbf{x} E_f(\mathbf{x}^t))_i \neq 0$ and $(\nabla_\mathbf{x} E_f(\mathbf{s}))_i \neq 0$. Under the assumption of a diverging learning rate ($\sum_{i \in \mathbb{N}} \alpha_i = \infty$ and $\alpha_i > 0$) , $\mathbf{s}$ is a fixed point for P-SBGD if and only if ($\Leftrightarrow$) $-\mathbf{\Pi}_{\mathbf{Z}^s}(\nabla_\mathbf{y} E_f(\mathbf{s})) = \mathbf{s}$. This point might not exist, in which case we cannot state the convergence of P-SBGD for any starting point $\mathbf{x}^0 \in \mathbb{R}^n$.

Proof. Our proof follows closely the proof of [4]. We first define the *fixed point* as a stationary point. We start with the direction $\Rightarrow$. Hence we assume that $\mathbf{s} \in \{\pm 1\}^n$ is a fixed point. We start by restating the update rule for point $\mathbf{x}^T$ starting from a **fixed point** $\mathbf{s}$:

$$\mathbf{x}^T = \mathbf{x}^0 - \sum_{t=0}^{T-1} \alpha_i \mathbf{\Pi}_{\mathbf{Z}^t}(\nabla_\mathbf{y} E_f(\mathbf{s})). \tag{8}$$

Hence $\mathbf{x}^T = \mathbf{x}^0$ holds by definition for all $T \in \mathbb{N}_+$. As the fixed point is the same in the space of reals or binary variables, we are allowed to take the sign of both sides:

$$\mathbf{s} := \text{sign}(\mathbf{x}^T) = \text{sign}\left(\mathbf{x}^0 - \mathbf{\Pi}_{\mathbf{Z}^{T-1}}(\nabla_\mathbf{y} E_f(\mathbf{s})) \sum_{i=0}^{T-1} \alpha_i\right). \tag{9}$$

Taking the limit $T \longrightarrow \infty$ and applying the assumption that $\sum_t \alpha_t = \infty$ yields

$$s_i = \lim_{T \to \infty} \text{sign}\left(\mathbf{x}^0 - \mathbf{\Pi}_{\mathbf{Z}^{T-1}}(\nabla_\mathbf{y} E_f(\mathbf{s})) \sum_{t=0}^{T} \alpha_t\right)_i = \text{sign}(-\mathbf{\Pi}_{\mathbf{Z}^{T-1}}(\nabla_\mathbf{y} E_f(\mathbf{s})))_i) = -\mathbf{\Pi}_{\mathbf{Z}^{T-1}}(\nabla_\mathbf{y} E_f(\mathbf{s})))_i$$

concluding that $\mathbf{s} = -\mathbf{\Pi}_{\mathbf{Z}^{T-1}}(\nabla_{\mathbf{y}} E_f(\mathbf{s}))$.

Next, we prove the direction " $\Leftarrow$ ". If $\mathbf{s}$ obeys that $-\mathbf{\Pi}_{\mathbf{Z}^{T-1}}(\nabla_{\mathbf{y}} E_f(\mathbf{s})) = \mathbf{s}$ for all $i \in [n]$, then if we take any $\mathbf{x}^0$ such that $\text{sign}(\mathbf{x}^0) = \mathbf{s}, \mathbf{x}^t$ will move in a straight line towards the direction of $-\mathbf{\Pi}_{\mathbf{Z}^{T-1}}(\nabla_{\mathbf{y}} E_f(\mathbf{s}))$. In other words, $\text{sign}(\mathbf{x}^t) = \text{sign}(\mathbf{x}^0) = \mathbf{s}$ for all $t = 0, 1, 2, \ldots$. Therefore, by definition, $\mathbf{s}$ is a fixed point. □

1.3 QUBO term of BM

Proposition 2 (BM as QUBO)**.** The binary projection $\mathbf{\Pi}_{\mathbf{Z}^t}(\mathbf{v})$ in Dfn. 1 admits the following *Ising Model* or quadratic unconstrained binary optimisation (QUBO) form:

$$\mathbf{\Pi}_{\mathbf{U}}(\mathbf{v}) = \underset{\mathbf{g} \in \{-1,1\}^n}{\arg\min}\ \mathbf{g}^\top \sum_{i=1}^{m} \mathbf{Q}_i \mathbf{g} + \mathbf{s}^\top \mathbf{g} \tag{10}$$

where $\mathbf{s} = -2\sum_{i=1}^m v_i \mathbf{u}_i$, $\mathbf{Q}_i = \mathbf{u}_i \mathbf{u}_i^\top$ and $\mathbf{Q} = \sum_{i=1}^m \mathbf{Q}_i$.

Proof.

$$\begin{aligned}
\mathbf{\Pi}_{\mathbf{U}}(\mathbf{v}) &= \underset{\mathbf{g} \in \{-1,1\}^n}{\arg\min} \sum_{i=1}^{m} ||v_i - \mathbf{g}^\top \mathbf{u}_i||_2^2 && (11)\\
&= \underset{\mathbf{g} \in \{-1,1\}^n}{\arg\min} \sum_{i=1}^{m} (v_i - \mathbf{u}_i{}^\top \mathbf{g})^\top (v_i - \mathbf{u}_i{}^\top \mathbf{g}) && (12)\\
&= \underset{\mathbf{g} \in \{-1,1\}^n}{\arg\min} \sum_{i=1}^{m} \mathbf{g}^\top \mathbf{u}_i \mathbf{u}_i{}^\top \mathbf{g} - 2\sum_{i=1}^{m} v_i \mathbf{u}_i{}^\top \mathbf{g} + \sum_{i=1}^{m} v_i^2 && (13)\\
&= \underset{\mathbf{g} \in \{-1,1\}^n}{\arg\min}\ \mathbf{g}^\top \sum_{i=1}^{m} \mathbf{Q}_i \mathbf{g} + \mathbf{s}^\top \mathbf{g} && (14)
\end{aligned}$$

where $\mathbf{s} = -2\sum_{i=1}^m v_i \mathbf{u}_i$, $\mathbf{Q}_i = \mathbf{u}_i \mathbf{u}_i^\top$ and $\mathbf{Q} = \sum_{i=1}^m \mathbf{Q}_i$.

□

Runtime analysis on quantum and central processing units. In general the experiments are relatively easy on the hardware and are run within minutes on a single CPU. Hence we will only focus on the QPU time, as well as on the graph benchmarks. For all five Rosenbrock runs we use a total of one minute QPU time. The small experiments on the Adult dataset as well as the Karateclub network are solved within an hour. Experiments on graph convolutional neural networks for the bigger datasets Cora and Pubmed run on the CPU for six hours. In those experiments, we did not need to use any GPU time during training. The experiments on D-Wave for MLPs are running for 6 hours and one run consumes about a minute of QPU time. This is explainable with the minor embeddings. Calculating those minor embeddings is a non-critical combinatorial optimisation problem and, at the moment, is *known* to be a bottleneck for quantum annealers. We are hopeful that those can be accelerated and make all QA algorithms faster, in the future.

To generate all the experiments in our paper we spent an hour of QPU time on a D-Wave quantum annealer as well as 1000h of CPU time.

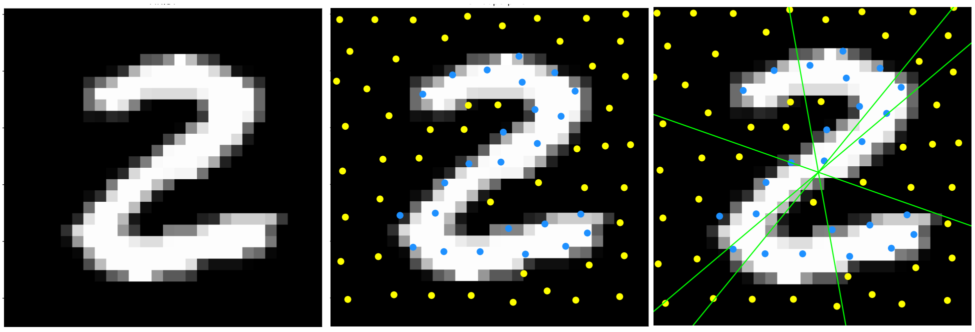


Figure 1: Label generation on MNIST [27] (example). *Left:* Initial image of the digit "2"; *Middle:* Extracted keypoint features with the superpixel approach [33]; *Right:* sampling four lines to determine binary features.

2 Implementation Details

Our code, which we will release, is implemented in PyTorch [35]. To calculate all gradients in all the experiments, we use the torch autograd library. In the graph experiments with Cora and Pubmed we use the Adam optimiser [26] with the PyTorch implementation. For BinaryConnect [16] we use our own implementation. However, contrary to the BinnaryConnect [16] paper we are using our definition of the sign function and not theirs, which projects $\text{sign}(0) = 1$ or does some random assigning. For ProxQuant we also use our own implementation, following the paper. The code will be released upon acceptance.

2.1 MNIST Feature Extraction

In the paper, we already described how we extract the features of the MNIST images. In this section, we give a more detailed overview of this procedure as well as a qualitative example in Fig. 1. To this end, we follow a three-step procedure to generate the 16-bit features: (i) We use the key points generated from the method described in [33]. They are marked in Fig. 1 as blue dots in the middle and right image. Afterwards in step (ii) we sample 16 lines, which all run through the centroid of the image. Fig. 1 in the right image contains four out of the 16 lines drawn in green. For the last and (iii) step, we count the number of feature points above and below the line. If there are more key points above line i than below, then we assign the feature i to 1; otherwise to -1.

2.2 Definitions

Hard tanh. Hard tanh is defined as follows:

$$\mathsf{hardTanh}(x) = \begin{cases} -1 & \text{if } x \leq -1, \\ x & \text{if } -1 < x < 1, \\ 1 & \text{if } x \geq 1, \end{cases}. \tag{15}$$

sign. We use the following definition of the sign function:

$$\operatorname{sign}(x) = \begin{cases} -1 & \text{if } x < 0, \\ 1 & \text{if } x \geq 0. \end{cases} \tag{16}$$

3 CDP hypothesis test for neural networks

We will use the H_0 hypothesis Z test to determine if our algorithm is more likely to point in the same direction as the gradient. As a comparison, we will conduct the same test on the established binary baseline signSGD. We choose as our hypothesis:

$\mathbf{H_0}$ that the projected gradient points in a random direction, that means:

$$\rho_i(\mathbf{x}^t) = \operatorname{Prob}\left(\mathbf{\Pi}_{\mathbf{Z}^t}(\tilde{\nabla}_{\mathbf{y}} E_f(\mathbf{x}^t))_i = \operatorname{sign}(\nabla_{\mathbf{x}} E_f(\mathbf{x}^t))_i\right) = 0.5 \tag{17}$$

$\mathbf{H_1}$ that the projected gradient points in the same/opposite direction

$$\rho_i(\mathbf{x}^t) = \operatorname{Prob}\left(\mathbf{\Pi}_{\mathbf{Z}^t}(\tilde{\nabla}_{\mathbf{y}} E_f(\mathbf{x}^t))_i = \operatorname{sign}(\nabla_{\mathbf{x}} E_f(\mathbf{x}^t))_i\right) > 0.5 \tag{18}$$

or

$$\rho_i(\mathbf{x}^t) = \operatorname{Prob}\left(\mathbf{\Pi}_{\mathbf{Z}^t}(\tilde{\nabla}_{\mathbf{y}} E_f(\mathbf{x}^t))_i = -\operatorname{sign}(\nabla_{\mathbf{x}} E_f(\mathbf{x}^t))_i\right) > 0.5 \tag{19}$$

We can clearly see that either $\mathbf{H_0}$ or $\mathbf{H_1}$ is true.

The data collection is done by measuring the gradients during training on the MNIST experiment in the paper, where we use the gradient on the full training set as the real non-stochastic gradient.

Let k be the number of occurrences where the gradient points in the same direction as our projected gradient, $p = 0.5$ as our H_0 probability and n as the number of compared gradients. The Z test for binary variables is formulated as follows:

$$Z = \frac{k - np}{\sqrt{np(1-p)}}. \tag{20}$$

A Z value in the interval of $[-1.96, 1.96]$ provides a 95% certainty that $\mathbf{H_0}$ is fulfilled.

The following two tables Tab. 2 and Tab. 3 show a sample collection from 5 runs. In the first column, we describe the number of samples, where our projected gradient and the real gradient point in the same direction, in the second column we have all samples and in the last column we note the Z value for the experimental run, which was calculated with Eq (20).

Discussion. As we can see in Tab. 2 and Tab. 3, when calculating the gradient with QP-SBGD the Z-value is always bigger than 1.96. This shows that we reject the $\mathbf{H_0}$ hypothesis and accept the $\mathbf{H_1}$ hypothesis. Hence we can be confident that the CDP holds in our case. As the tables further present, the projected gradients point in the same direction as the real gradient, then away. Hence we can narrow down the $\mathbf{H_1}$ hypothesis to the CDP assumption.

Further to verify our experiments we also tested signSGD, where the CDP assumption is indicated by a few lemmas. Here we observe that $\mathbf{H_0}$ can be rejected, but on a weaker basis, as the Z values are smaller. This further indicates that our algorithm fulfils the CDP assumption.

4 Additional experiments

# of same direction	total # of samples	Z-value
2285	3915	10.4
2199	3647	12.4
2311	3886	11.8
2334	3984	10.8
2356	3990	11.4

Table 2: Samples from QP-SBGD directions.

# of same direction	total # of samples	Z-value
2334	4626	0.61
732	1176	8.39
339	630	1.91
2559	4702	6.06
2103	4683	-6.9

Table 3: Samples from signSGD.

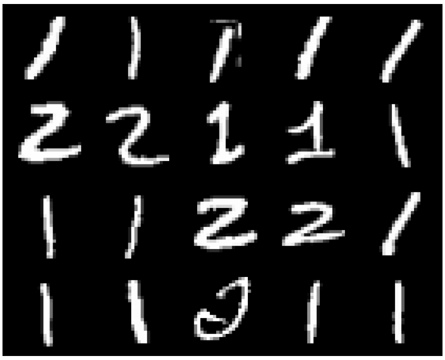


Figure 3: Classification of 1s

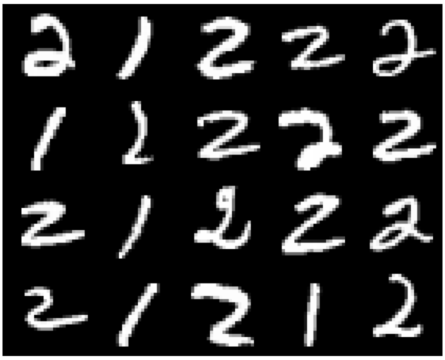


Figure 4: Classification of 2s

In this section, we present additional experiments on the UCI adult dataset, some qualitative examples on the MNIST dataset, and some D-Wave analysis.

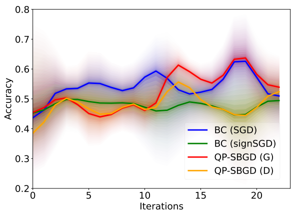


Figure 2: Training accuracy on a subset of the UCI Adult dataset for binary classification.

4.1 Quantitative results on UCI Adult

We now use three manually selected features and 30 batches with a batch size of 20 to train a two-layer neural network with the input dimension of 3, hidden dimension of 5 and the output of 2. Fig. 2 reports the training accuracy where we observe that QP-SBGD (G) outperforms the baselines as well as our D-Wave optimiser, QP-SBGD (D), which nevertheless performs better than or on par with BC (signSGD).

4.2 Qualitative Results of MNIST classification

In Fig. 3 and Fig. 4, we show a few qualitative results on the MINIST digits dataset. We select one of the networks trained to classify the digits ’1’ and ’2’ with QP-SBGD (D-Wave). Similar to the numbers reported in Tab 1 in the paper, the accuracy on those samples is 72.5%.

4.3 Additional analysis of the QUBO problems using D-Wave annealer

Embedding. To solve QUBO problems on D-Wave, we minor-embed them onto the Pegasus architecture. As shown in Fig. 5, this operation interprets the logical graph (left) and implements each *logical bit* with multiple, redundant qubits (right). The graph in the figure is an exemplary QUBO corresponding to training the third layer in our MNIST experiment.

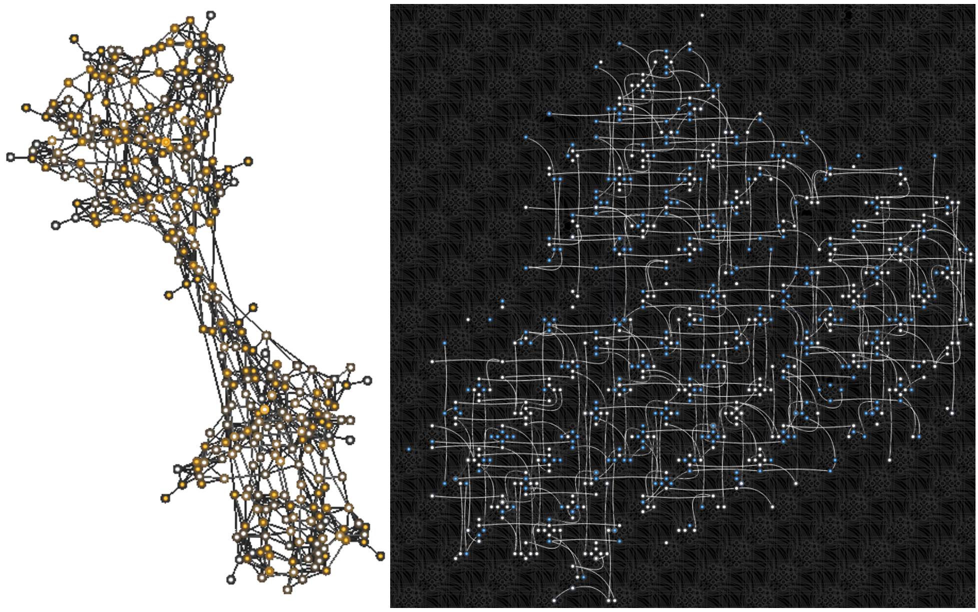


Figure 5: *Left*: QUBO problem graph to update the weights of the second layer in the first training iteration *Right*: Embedded problem graph on the Pegasus [9] D-Wave architecture.

Evolution of the QUBO Next, we investigate the development of the QUBO during training. In Fig. 6 we observe how the QUBO changes over time/iterations. The QUBO matrices shown in the figure are examples from the MNIST experiment (Sec. 3.3.2) taken from the first layer. We display one QUBO after every 10th update. We observe here that all QUBOs have a rather densely connected support of the otherwise sparse graph. This overall sparsity of the graph enables us to embed those with minor embeddings and deploy them to the D-Wave QPU.

Sampling solutions. After embedding each QUBO problem in the D-Wave Advantage architecture, we query the quantum annealer multiple times and retrieve the solution corresponding to the lowest energy. However, it is still interesting to see the solutions' distribution. To this end, we analyze QUBOs, which calculate the weight updates of the MNIST experiment in the third layer, using D-Wave Explorer and measure a series of QUBOS, which are 10 updates apart. Sec. 4.3 plots the different energies for the sampled solutions. The histogram in the upper left corner corresponds to the QUBO update problem in the first iteration and the histogram in the lower right corner corresponds to the QUBO problem after 50 updates (*e.g.*, the 51st update). In total, we collected 1000 samples from D-Wave and compared the energy. The resulting distribution can be described as a tail-heavy Poisson distribution, which is shifted towards negative numbers. This type of resulting distribution is typical for problems, which are difficult to compute.

Similarity. In Fig. 7 we compare the similarities of the sampled D-Wave solutions. Here the 10 scores with the lowest energy are compared with the Jaccard Similarity. We can see that all QUBO solutions are fairly dissimilar to each other. As seen in the picture only the first and fourth/fifth solutions have a relatively high similarity score of around 0.4. The indications of that are two-fold: (i) it demonstrates the importance of finding the optimal solution, and

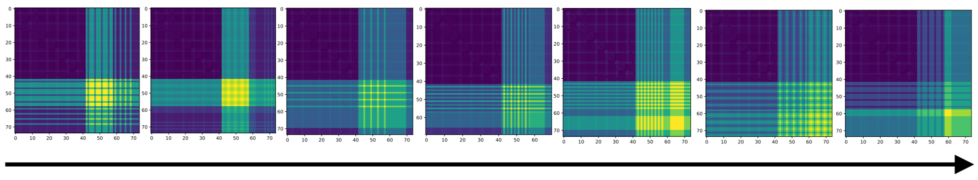


Figure 6: Evolution of the QCBO formulation to calculate the weight updates for the first layer with the Quantum Projected Stochastic Binary-Gradient Descent algorithm.

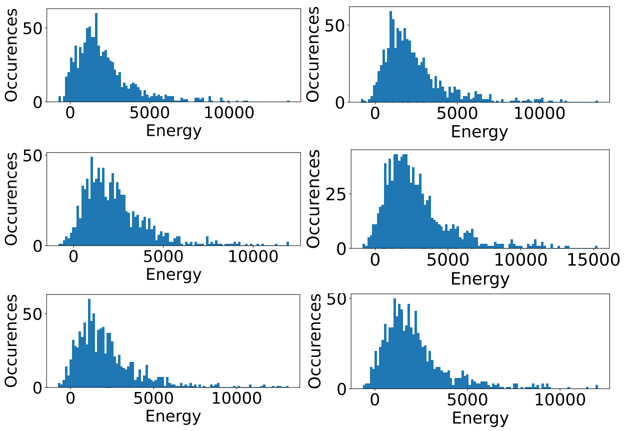


Figure 7: Histogram of the different sampling energies of the weight update QUBO of the third layer. In the first column, we show the histogram of the zeroth, tenth, and twentieth updates, and in the second column the histogram of the thirtieth, fortieth, and fiftieth updates from top to bottom

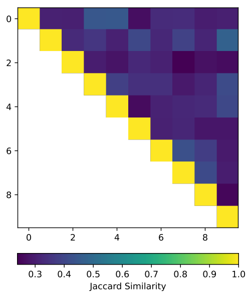


Figure 8: Jaccard similarity of the 10 samples with the lowest energy.

(ii) even though the D-Wave annealer cannot act like a true Bayesian posterior sampler, the samples from its (modified) posterior are diverse and span the solution space. This also confirms the findings of [8] in which these samples are used as alternative solutions.

Ablation studies. Fig. 9 tests the effect of different hyperparameters on Cora [30]. Our hybrid optimiser, QP-SBGD, is robust to batch size changes, reported as training data percentages, and slightly more impacted by the step size.

Spectral gap. A crucial feature is the *spectral gap* in QUBOs solved by a quantum annealer. A larger spectral gap significantly enhances the likelihood of the annealer converging to the accurate ground truth solution [32]. We now examine, via simulation, the spectral gap corresponding to the QUBO in Prop. 2 for small MLPs. Fig. 10 plots the eigenvalues (blue lines) of the time-varying Hamiltonian as a function of t/T, for a single binary linear layer l with up to four neurons and batch size 1 (besides these parameters, the size of H, and hence the number of distinct eigenvalues, depend on other factors; more on the feasibility of simulation below). The smallest spectral gap is highlighted by a red bar in the plots. With the same annealing time, a larger spectral gap yields a final state closer to the ground state and, thus, a higher-quality solution. The results show that, as expected, the spectral gap shrinks gradually with increasing problem size. While two data points are not enough to extrapolate a trend, the exercise shows how the performance of quantum annealing depends on the hyperparameters. In particular, the observations suggest keeping the number of neurons in each layer small.

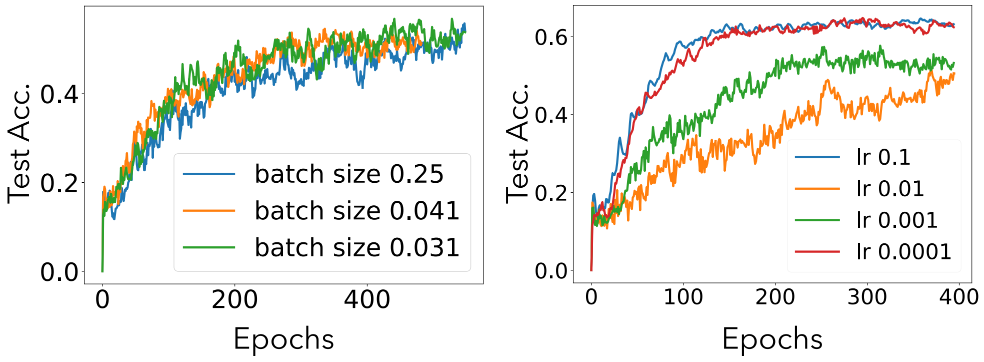
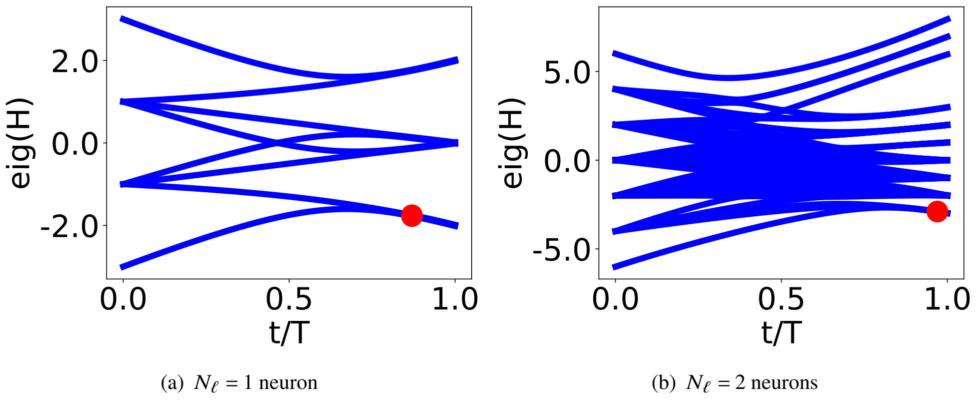


Figure 9: Impact of learning rate (LR) and batch sizes on the test accuracy. We plot the mean accuracy over six runs.



(a) $N_\ell = 1$ neuron (b) $N_\ell = 2$ neurons

Figure 10: The eigenvalues of the QUBO Hamiltonian in Prop. 2 plotted as a function of annealing time (t/T) while training a binary linear layer with two neurons on the UCI-adult. The red bar represents the eigenvalue gap between the ground level and the first excited level not evolving into the ground state. The minimal spectral gap amounts to 0.03 and 0.01 in (a) and (b), respectively.

5 Extended Related Work

Binary neural networks (BNNs). Compared to full-precision neural networks, BNNs [22] consume less memory and provide faster inference at the cost of certain information loss. Much effort has been devoted to mitigating performance degradation due to binarisation, as detailed in recent surveys [36, 49]. Here, we summarise important developments based on the recent surveys. We refer the reader to said surveys for a more complete overview.

Based on their binarisation strategies, BNN models can be categorised into *naive* or *optimisation-based* ones. BinaryConnect [16] is an early naive training strategy which binarises the weights by a sign function. In this way, lightweight bitwise XNOR operations replace most real-valued arithmetic operations. BinaryNetg [22] is an extension that binarises both weights and activations during inference and training. Dorefa-Net [53] defines new quantisation functions for weights, activations and gradients to accelerate network training.

Optimisation-based BNNs attempt to address the accuracy drop more directly resulting from weight and activation binarisation in the naive models while preserving the compact nature. Following Qin et al. [36], we further categorise such works based on their optimisation goals:

- *Minimising the quantisation error.* XNOR-Net introduces a scaling factor for the binary weights and activations to generate a better approximation to the corresponding real-valued parameters [37]. XNOR-Net++ builds on this work and fuses the scaling factors of the weights and activations into one. The fused scaling factor is gained from discriminative training via backpropagation [11]. Dorefa-Net defines new quantisation functions for weights, activations and gradients instead of a pre-defined uniform quantiser to accelerate network training [53]. LQ-Nets trains the quantisers for activations in each layer and weights in each channel jointly with the neural network [52]. Parameterised Clipping Activation (PACT) shifts the activation distribution via learning a quantisation scaling factor during network training to achieve low bit-width precision [14]. LAB2 directly minimises binarisation loss for weights using a proximal Newton algorithm [20].
- *Improving the loss function.* How-to-Train adds a regularisation term to encourage weight bipolarisation in addition to the task-specific loss term [42]. LAB directly develops a loss function during layer-wise weight binarisation [20]. ReActNet introduces a spatial distributional loss to enforce the BNN output distribution to approximate that of the full-precision model, on top of activation distribution shift and reshaping using new quantisation functions [29].
- *Reducing the gradient approximation error.* Another stream of work focuses on the nature of quantisation and activation functions to enable backpropagation and enhance forward propagation. Bi-Real Net approximates the derivative of the sign function for activations using a piecewise linear function to overcome the intrinsic discontinuity and proposes a magnitude-aware gradient for weights [28]. HWGQ-Net designs a forward quantised approximation and a backward continuous approximation of the ReLU activation function to address the learning inefficiency of BNNs [12].

Various training techniques including asymptotic optimisation, quantisation, gradient approximation and network structure transformation, are adopted in some of the learning-based models above. Standard techniques are used in optimisation, involving Adam, AdaMax, SGD and RMSprop. For more details please refer to Qin *et al.* [36].

Quantum annealing. Recent progress in *quantum computer vision* [8] has shown that various computer vision algorithms could benefit from QUBO-formulations and quantum annealing [1, 6, 8, 13, 17, 31, 41, 50, 51]. Arthur *et al.* [2] has similarly shown the advantage for several machine learning models. A new paradigm was developed recently to harness the computational power of quantum annealers for the training of BNNs [38] as well as for optimising in binary variables [50]. As an early work in this direction, a simple version of BNN was examined. As alluded to above, our work improves the scalability of their approach by adopting a hybrid and incremental training scheme.

Quantum neural networks. As an emerging field, *quantum machine learning* [7, 10, 40, 47] has shown that the training of linear regression, support vector machines and k-means clustering admit QUBO-like formulations [2], whereas the first neural network variants trained on quantum hardware were Boltzmann Machines. In parallel, *quantum deep learning* [18, 24, 46], the problem of creating quantum circuits that enhance the operations of neural networks by physically realising them, has emerged to alleviate some of the computational limitations of classical deep learning, thanks to the efficient training algorithms [5, 25]. However, a severe challenge remains to be implementing non-linearities and other non-unitary operations with quantum unitaries [24, 39]. Similarly, *quantum convolutional neural networks* (Q-CNNs) [15] have been developed with the mindset of implementing the analogue of classical CNN operations via quantum gates. These works are fundamentally different to ours, as they try to realise quantum implementations of machine learning algorithms, whereas we leverage quantum computation to solve the classical problem of neural network training.

Network quantisation. Instead of training BNNs from data, network quantisation techniques aim to reduce the precision (including to binary) of a trained full-precision neural network in a manner that preserves accuracy; see Gholami et al. [19] for a recent survey. We stress that *quantisation* fundamentally differs in formulating a BNN for execution or training on *quantum* computers. In the current work, we are focusing on the most extreme kind of network quantisation: binary neural networks. More general types of quantisation can be formulated as a QUBO in principle [34].

Binary graph neural networks. As opposed to the plethora of methods dealing with the binarisation and quantisation of ordinary neural networks, binary graph neural networks are under-explored. Only a few works exist [3, 43, 44] and most of them operate on the principles of XNOR-Net and its variants, aiming to binarise the multiplication. Still, re-quantisation remains to be a necessary operation to avoid full-precision computations [48]. On a parallel track, developing efficient binary graph operators [23] or quantising graph neural networks [21, 23, 54] as well as GNN-compression are somewhat studied. We refer the reader to a recent survey [45] and find it worth mentioning that none of these procedures are friendly for implementation on quantum hardware.

References

[1] Federica Arrigoni, Willi Menapace, Marcel Seelbach Benkner, Elisa Ricci, and Vladislav Golyanik. Quantum motion segmentation. In *European Conference on Computer Vision (ECCV)*, 2022.

[2] Davis Arthur, Lauren Pusey-Nazzaro, et al. Qubo formulations for training machine learning models. *Scientific reports*, 11(1):1–10, 2021.

[3] Mehdi Bahri, Gaétan Bahl, and Stefanos Zafeiriou. Binary graph neural networks. In *CVPR*, pages 9492–9501, 2021.

[4] Yu Bai, Yu-Xiang Wang, and Edo Liberty. Proxquant: Quantized neural networks via proximal operators. *arXiv preprint arXiv:1810.00861*, 2018.

[5] Kerstin Beer, Dmytro Bondarenko, Terry Farrelly, Tobias J Osborne, Robert Salzmann, Daniel Scheiermann, and Ramona Wolf. Training deep quantum neural networks. *Nature communications*, 11(1):1–6, 2020.

[6] Harshil Bhatia, Edith Tretschk, Zorah Lähner, Marcel Seelbach Benkner, Michael Möller, Christian Theobalt, and Vladislav Golyanik. Ccuantumm: Cycle-consistent quantum-hybrid matching of multiple shapes. In *Computer Vision and Pattern Recognition (CVPR)*, 2023.

[7] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[8] Tolga Birdal, Vladislav Golyanik, Christian Theobalt, and Leonidas Guibas. Quantum permutation synchronization. In *CVPR*, 2021.

[9] Kelly Boothby, Paul Bunyk, Jack Raymond, and Aidan Roy. Next-generation topology of d-wave quantum processors. *arXiv e-prints*, 2020.

[10] Michael Broughton, Guillaume Verdon, Trevor McCourt, Antonio J Martinez, Jae Hyeon Yoo, Sergei V Isakov, Philip Massey, Ramin Halavati, Murphy Yuezhen Niu, Alexander Zlokapa, et al. Tensorflow quantum: A software framework for quantum machine learning. *arXiv preprint arXiv:2003.02989*, 2020.

[11] Adrian Bulat and Georgios Tzimiropoulos. Xnor-net++: Improved binary neural networks. *arXiv preprint arXiv:1909.13863*, 2019.

[12] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. In *CVPR*, pages 5918–5926, 2017.

[13] Tat-Jun Chin, David Suter, Shin-Fang Chng, and James Quach. Quantum robust fitting. In *ACCV*, 2020.

[14] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018.

[15] Iris Cong, Soonwon Choi, and Mikhail D Lukin. Quantum convolutional neural networks. *Nature Physics*, 15:1273–1278, 2019.

[16] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NIPS*, pages 3123–3131, 2015.

[17] Matteo Farina, Luca Magri, Willi Menapace, Elisa Ricci, Vladislav Golyanik, and Federica Arrigoni. Quantum multi-model fitting. In *Computer Vision and Pattern Recognition (CVPR)*, 2023.

[18] Siddhant Garg and Goutham Ramakrishnan. Advances in quantum deep learning: An overview. *arXiv preprint arXiv:2005.04316*, 2020.

[19] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

[20] Lu Hou, Quanming Yao, and James T Kwok. Loss-aware binarization of deep networks. *arXiv preprint arXiv:1611.01600*, 2016.

[21] Linyong Huang, Zhe Zhang, Zhaoyang Du, Shuangchen Li, Hongzhong Zheng, Yuan Xie, and Nianxiong Tan. Epquant: A graph neural network compression approach based on product quantization. *Neurocomputing*, 503:49–61, 2022.

[22] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. *NIPS*, 29, 2016.

[23] Yongcheng Jing, Yiding Yang, Xinchao Wang, Mingli Song, and Dacheng Tao. Meta-aggregator: learning to aggregate for 1-bit graph neural networks. In *CVPR*, pages 5301–5310, 2021.

[24] Iordanis Kerenidis, Jonas Landman, and Anupam Prakash. Quantum algorithms for deep convolutional neural networks. *arXiv preprint arXiv:1911.01117*, 2019.

[25] B Kerstin, B Dmytro, F Terry, O Tobias, S Robert, and W Ramona. Efficient learning for deep quantum neural networks. *Nature*, 2019.

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[27] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010.

[28] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *ECCV*, pages 722–737, 2018.

[29] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In *ECCV*, pages 143–159. Springer, 2020.

[30] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

[31] Timothy M McCormick, Zipporah Klain, Ian Herbert, Anthony M Charles, R Blair Angle, Bryan R Osborn, and Roy L Streit. Multiple target tracking and filtering using bayesian diabatic quantum annealing. *arXiv preprint arXiv:2209.00615*, 2022.

[32] Cameron Robert McLeod and Michele Sasdelli. Benchmarking d-wave quantum annealers: Spectral gap scaling of maximum cardinality matching problems. In *International Conference on Computational Science*, pages 150–163. Springer, 2022.

[33] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, pages 5115–5124, 2017.

[34] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8024–8035. Curran Associates, Inc., 2019.

[36] Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. Binary neural networks: A survey. *Pattern Recognition*, 105, 2020. ISSN 00313203. doi: 10.1016/j.patcog.2020.107281. arXiv: 2004.03333.

[37] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[38] Michele Sasdelli and Tat-Jun Chin. Quantum annealing formulation for binary neural networks. In *2021 Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–10. IEEE, 2021.

[39] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. The quest for a quantum neural network. *Quantum Information Processing*, 13(11):2567–2586, 2014.

[40] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015.

[41] Marcel Seelbach Benkner, Vladislav Golyanik, Christian Theobalt, and Michael Moeller. Adiabatic quantum graph matching with permutation matrix constraints. In *International Conference on 3D Vision (3DV)*, 2020.

[42] Wei Tang, Gang Hua, and Liang Wang. How to train a compact binary neural network with high accuracy? In *Thirty-First AAAI conference on artificial intelligence*, 2017.

[43] Hanchen Wang, Defu Lian, Ying Zhang, Lu Qin, Xiangjian He, Yiguang Lin, and Xuemin Lin. Binarized graph neural network. *World Wide Web*, 24:825–848, 2021.

[44] Junfu Wang, Yunhong Wang, Zhen Yang, Liang Yang, and Yuanfang Guo. Bi-gcn: Binary graph convolutional network. In *CVPR*, pages 1561–1570, 2021.

[45] Shaopeng Wei and Yu Zhao. Graph learning: A comprehensive survey and future directions. *arXiv preprint arXiv:2212.08966*, 2022.

[46] Nathan Wiebe, Ashish Kapoor, and Krysta M Svore. Quantum deep learning. *arXiv preprint arXiv:1412.3489*, 2014.

[47] Peter Wittek. *Quantum machine learning: what quantum computing means to data mining*. Academic Press, 2014.

[48] Kai-Lang Yao and Wu-Jun Li. Full-precision free binary graph neural networks, 2022.

[49] Chunyu Yuan and Sos S Agaian. A comprehensive review of binary neural network. *arXiv preprint arXiv:2110.06804*, 2021.

[50] Alp Yurtsever, Tolga Birdal, and Vladislav Golyanik. Q-fw: A hybrid classical-quantum frank-wolfe for quadratic binary optimization. *arXiv preprint arXiv:2203.12633*, 2022.

[51] Jan-Nico Zaech, Alexander Liniger, Martin Danelljan, Dengxin Dai, and Luc Van Gool. Adiabatic quantum computing for multi object tracking. In *CVPR*, 2022.

[52] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *ECCV*, pages 365–382, 2018.

[53] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

[54] Zeyu Zhu, Fanrong Li, Zitao Mo, Qinghao Hu, Gang Li, Zejian Liu, Xiaoyao Liang, and Jian Cheng. A2q: Aggregation-aware quantization for graph neural networks. *arXiv preprint arXiv:2302.00193*, 2023.