

Quantum Deep Learning: Sampling Neural Nets with a Quantum Annealer

Catherine F. Higham
School of Computing Science
University of Glasgow
Glasgow G12 8QQ
`Catherine.Higham@glasgow.ac.uk`

Adrian Bedford
OxBrdgRbtx Ltd
Stratford-upon-Avon CV37 6XU

July 20, 2021

Abstract

We demonstrate the feasibility of framing a classically learned deep neural network as an energy based model that can be processed on a one-step quantum annealer in order to exploit fast sampling times. We propose approaches to overcome two hurdles for high resolution image classification on a quantum processing unit (QPU): the required number and binary nature of the model states. With this novel method we successfully transfer a convolutional neural network to the QPU and show the potential for classification speedup of at least one order of magnitude.

1 Motivation

Deep learning approaches are being applied and refined for classification tasks across many data types (images, video and audio) with high levels of success [14, 11, 2]. There are many applications where classically trained convolutional neural networks (CNNs) perform very well, sometimes better than human experts [7, 3, 16]. However in some circumstances, including security, defence and automated transport, safety-critical benefits would arise if classifications can be computed more quickly. This suggests an important role for quantum computing and we give a proof of principle demonstration that quantum annealing has the potential to address this issue.

The manuscript is organised as follows. In Section 2 we discuss the quantum annealing technology, the underlying quantum physics and the machine learning

tool that we implement. We also discuss related work and summarise our main contributions. In Section 3 we give more detail about the implementation and justify our algorithmic choices. Computational results are given in Section 4. We finish in Section 5 with brief discussion.

2 Background and Related Work

2.1 Quantum Annealing Technology

The current D-Wave Advantage System 1.1 contains around 5,000 qubits and 35,000 couplers [5]. These qubits can be physically coupled to form networks with real-valued coefficients denoting coupling strength and individual on/off biases. Connectivity per qubit is limited to 15 couplers but can be extended by forming chains of qubits. These coefficients define and constrain relationships between qubits and form a quadratic binary model capable of expressing a range of behaviours. The parameter coefficients of a given model are embedded on the D-Wave network and by posing the problem as energy minimisation, quantum annealing is used to find the low energy states (on/off) of the qubits and hence the most likely formations. The ability to rapidly sample from many states and hence characterise the shape of the energy landscape is a key benefit of this technology.

2.2 Underlying Quantum Physics: The Hamiltonian

A classical Hamiltonian gives a mathematical description of some physical system in terms of its energies. For most non-convex Hamiltonians, finding the minimum energy state is an NP-hard problem that classical computers cannot solve efficiently. In quantum annealing, the system begins in the lowest-energy state of an initial Hamiltonian, A , and as it anneals introduces the problem Hamiltonian, B . To do this, a time-like parameter s and annealing functions $A(s)$ and $B(s)$ are introduced such that $A(0) \gg B(0)$ and $B(1) \gg A(1)$. Hence, as the system is annealed, $A(s)$ decreases, $B(s)$ increases and we approach the desired solution states. This approach has the potential for significant benefits in terms of both speed and accuracy, compared with classical computing technology.

For the D-Wave system [5], the Hamiltonian is expressed as follows

$$H = \frac{A(s)}{2} \left(\sum_i \hat{\sigma}_x^{(i)} \right) + \frac{B(s)}{2} \left(\sum_i h_i \hat{\sigma}_z^{(i)} + \sum_{i,j} J_{i,j} \hat{\sigma}_z^{(i)} \hat{\sigma}_z^{(j)} \right), \quad (1)$$

where $\hat{\sigma}_{x,z}^{(i)}$ are Pauli matrices operating on a qubit q_i , and h_i and $J_{i,j}$ are the qubit local field values or biases and coupling strengths or weights, respectively. In the final state, the qubits take values of either 0 or 1. Hence this provides a classical solution for the problem Hamiltonian B defined by the biases, h_i and the coupling weights, $J_{i,j}$.

2.3 Machine Learning Approaches: Boltzmann Machines

Boltzmann machines [10] are probabilistic models with an energy-based distribution that define a probability for each of the N discrete states in a binary vector given by.

$$p(x) = \frac{1}{Z} \exp(-E(x; \theta)). \quad (2)$$

Here $E(x; \theta)$ is an energy function parameterised by θ , and $Z = \sum_x \exp(-E(x; \theta))$ is the normalizing coefficient, also known as the partition function, which ensures that $p(x)$ sums to 1 over all the possible states of x . The energy function can be represented via a quadratic form $x^T Q x$ in which the upper-triangular matrix $Q = Q_\theta$ encapsulates the parameters of the quadratic energy function defined by

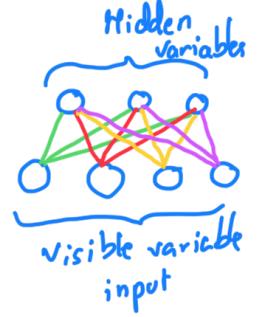
$$E(x; \theta) = x^T Q x = \sum_i^N q_{i,i} x_i + \sum_{i < j}^N q_{i,j} x_i x_j, \quad (3)$$

where q_i and $q_{i,j}$ are the biases and correlation weights respectively. This expression makes clear the connection between Boltzmann machines, quadratic binary models, and the final Hamiltonian in the second part of equation (1). This suggests that Boltzmann machines are good candidates for training and evaluation on the D-Wave quantum annealer as the problem can be framed in such a way that the solution to the final Hamiltonian is the low energy state of the problem. In this work we are interested in transferring classically learned weights to a quantum system for speed up and hence we focus on the evaluation task. A restricted Boltzmann machine (RBM) is a special type of Boltzmann machine with a symmetrical bipartite structure [9]. The set of binary variables is divided into visible (input), v , and hidden, h , variables. These are analogous to the retina and brain, respectively. The hidden variables allow for more complex dependencies among visible variables and are used to learn a stochastic generative model over a set of inputs. All visible variables connect to all hidden variables, but no variables in the same layer are linked. In the classical setting this limited connectivity makes inference and therefore learning easier because analytical expressions can be found for the conditional probabilities Gibbs sampling may be used to sample h from v and then v from h , leading to an estimated expected value of the model required to update the weights, a process known as contrastive divergence (CD) [9].

2.4 Related Work

Finding the optimal parameter values for an artificial neural network requires large amounts of data and many cycles of updates which costs time and energy. Training on a QPU has the potential to produce better models and more quickly than a digital processor. [1] investigated estimating the model expectations of RBMs using samples on a 512-qubit D-Wave machine and successfully trained a model with up to 32 visible nodes and 32 hidden nodes per RBM layer. In their tests, they found that this approach achieves comparable or better accuracy with significantly fewer iterations of generative training than conventional

*Minimize the energy
of the neurons's
connection*



CD-based training on a coarse-grained version of the MNIST data set. [15] also investigated the feasibility of using the D-Wave as a sampler for machine learning. Their work described a hybrid system that combined a classical deep neural network autoencoder with a quantum annealing RBM. Their method overcame two key limitations in the 2000-qubit D-Wave processor, namely the limited number of qubits available to accommodate typical problem sizes for fully connected quantum objective functions and samples that are binary pixel representations. Their hybrid autoencoder approach indicated advantage for quantum annealing relative to the use of a classical computer implementation for image-based machine learning and hinted at even more promising results for the next generation D-Wave quantum system.

In [6], the model expectation of gradient learning for RBM was calculated using a quantum annealer (D-Wave 2000Q), giving much faster results than Markov chain Monte Carlo used in contrastive divergence. Most Boltzmann machines use restricted topologies that exclude looping connectivity, as such connectivity creates complex distributions that are difficult to sample. [13] used an open-system quantum annealer to sample from complex distributions and implemented Boltzmann machines with looping connectivity.

In this work our starting point is a classically trained convolutional neural network (CNN) that maps real valued features to a classification label. Unlike [1], [15] and [6] we do not train on the QPU but investigate transferring the CNN problem to the QPU in order to obtain solutions. However we encounter similar issues: qubit number, binary nature and connectivity. This work looks at how these issues can be addressed on the recent D-Wave Advantage in the context of deep learning transfer.

2.5 Contributions

The main contribution of this work is to show that a trained artificial neural network can be transferred to a quantum computing setting. To do this we use the framework of energy minimisation involving an appropriate quadratic form, where quantum annealing provides samples from the low energy, most likely model states. Results, obtainable in microseconds rather than milliseconds, can be used to estimate a predictive class score. We show how to design a quadratic binary model, the engine of quantum annealing, to behave like a neural network, combining deep learned parameters and layer structures from a classically trained neural network with quadratic binary model parameters. Constraint parameters are adapted so that designated class units act together as a softmax classification unit. We then test this approach on digit image data by finding an appropriate embedding on the D-Wave QPU and transferring the coupling and bias parameters from the classical model to the qubits. Key barriers to scaling up to high resolution image/video processing are the binary nature of the variables, the number of qubits that could be assigned to visible units and the limited number of couplers per qubits. We address these issues by showing how real-valued features can be introduced into the system and by introducing novel pooling qubits that reduce the number of connections required.

These pooling qubits also add nonlinear behaviour required to model data. Our approach is outlined in Section 3 and the results of experiments on the D-Wave in Section 4.

3 Methods: Framing Classification as a Quadratic Binary Model with Constraints

3.1 Neural Network Layers

A typical neural network classification model takes features, x , as input and passes these features through a series of mappings and activations leading to a classification score [8]. These mappings typically include a pooling function, such as max pooling, $\max(x)$, a linear transform function, $x^T W + b$, where W is a matrix and b is a vector, and softmax activation, $S(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$, to determine a class score. The values of the parameters W and b in each layer are determined through training with a cost function to minimise the class score error. In our work we set $b = 0$ to reduce the number of parameters.

↳ no bias !

3.2 Quadratic Binary Model with Constraints

A quadratic binary model defines an energy-based network of binary random variables with real-valued parameters for biases and correlation weights. Constraints can be added to the biases. Energy minimisation involves finding the binary values of the model states that result in the lowest energy levels. This minimisation problem is also referred to as quadratic binary optimisation or QUBO.

Let $v = \{v^{ij}\}_{i=1\dots N_p, j=1\dots M_p}$, $h_p = \{h_p^j\}_{j=1\dots M_p}$, and $h_c = \{h_c^k\}_{k=1\dots M_c}$ be binary random variables where v denote visible units, h_p and h_c denote hidden pooling and classification units, respectively. N , N_p , M_p , M_c are the number of visible units, the number of visible units in a pool, the number of hidden pooling units and the number of class units, respectively. For pooling, M_p subsets of v , each containing four neighbouring elements are connected to a h_p reducing the possible connections from NM_p to $4M_p$. All h_p are connected to all h_c forming a fully connected classification layer. Further, h_c are interconnected so that a decision can be made about the most likely state equivalent to a softmax activation.

The energy of this system involving v , h_p and h_c and the connections described above can be introduced, as in equation (3), as follows

$$E(v, h_p, h_c) = -q_v v - \sum_j^{M_p} \sum_i^{N_p} (h_p^j q_{vp}^{ij} v^{ij} + q_p^j h_p^j) - h_c^T q_{pc} h_p - q_c^T h_c - \sum_{j>k}^{M_c, M_c-1} h_c^j q_{cc}^{jk} h_c^k, \quad (4)$$

where q_v , q_{hp} , q_{hc} are the biases of v , h_p , h_c respectively, and q_{vp} , q_{pc} , q_{cc} are the coupling strengths between v and h_p , h_p and h_c and between h_c and h_c respectively.

*find the minimal configuration of neurons with energy approach
Great explanation of RBM book
at the video of Luis Serrano on YouTube*

In order that the quadratic binary model behaves like the neural network classification model, the parameters, $Q = \{q_v, q_{hp}, q_{hc}, q_{vp}, q_{cp}, q_{cc}\}$, are set as follows to achieve pooling and classification.

3.3 Downsampling with Pooling Units

The objective of pooling for the hidden units h_p is to establish whether features are present. If features are present, then h_p is on and contributions are made to the classification units h_c . If features are not present then h_p is off and no contributions are made to the classification units. Consider the contribution a pooling subset, h_p^j makes to the energy equation,

$$-\sum_i (q_v^i v^{ij} + h_p^j q_{vp}^{ij} v^{ij} + q_p^j h_p). \quad (5)$$

If q_v^i are the normalised feature values of v , real values between $-\alpha$ and α , then neutralising h_p by setting $q_p^j = 0$ and weighting the interaction between v and h_p by setting $q_{vp} = \alpha$ lowers the energy of the system and increases the likelihood that h_p is on if a feature is present.

Same as regularization

3.4 Establishing a Classification Unit

We assume that the weights of a CNN classification layer, W , have been optimised so that the class position of the maximum value of the predicted class vector $y = x^T W$, where x is a feature vector, agrees with the class of x . The contribution on the classification units h_c comes from an interaction with h_p ($h_c^T q_{pc} h_p$) and interactions with other h_c ($q_c^T h_c + \sum_{j>k}^{M_c, M_c-1} h_c^j h_c^k$), see equation (4). Concerning the interaction between h_p and h_c , we set $q_{pc} = W$.

For the interaction of h_c values with each other we constrain the model so that $\sum_k h_c^k = 1$. This is achieved by finding the values of q_c and q_{cc} that minimise $(\sum_k h_c^k - 1)^2$ using the following expansion

$$(\sum_k h_c^k - 1)^2 = 2 \sum_{j>k}^{M_c, M_c-1} h_c^j q_c c^{jk} h_c^k - \sum_j^{M_c} h_c^j + 1, \quad (6)$$

with solutions $q_{cc} = 2$ and $q_c = -1$.

3.5 Transfer to QPU

The weights and biases are physically transferred to the QPU in a sparse matrix format, Q_{IJ} , with the biases for each node q_v, q_p, q_c along the diagonal in position Q_{II} and the weights connecting each pair q_{vp}, q_{pc}, q_{cc} in position Q_{IJ} . An illustrative example of Q is shown in Figure 1. Before the quantum annealing process, a mapping has to be found that physically embeds the Q matrix on the D-Wave graph. This embedding is found using the available

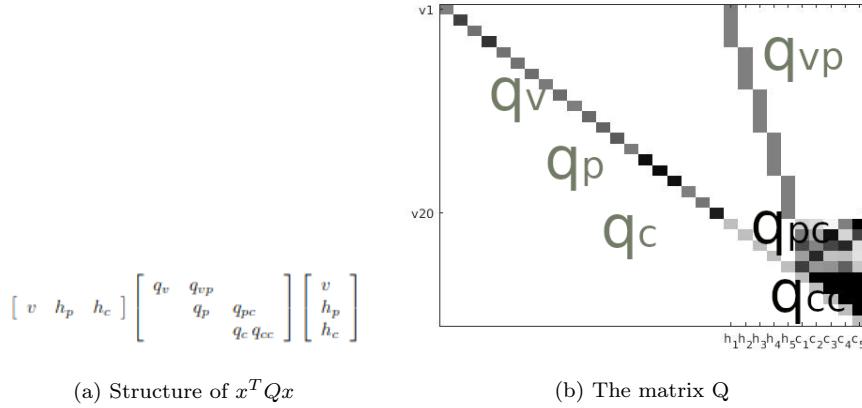


Figure 1: **Illustrative Q matrix** for feature pooling and classification. Five sets of four features are each coupled, with weights q_{vp} , to a pooling node, see columns labelled h_1 to h_5 . These nodes are further fully connected to five class nodes with coupling strength determined by the learned CNN weights, q_{pc} . Further connections are added between the class nodes in order to constrain the class states to just one being on, q_{cc} . The biases, $\{q_v, q_p, q_c\}$ associated with each node, including the input features, are placed along the diagonals.

Need more details on how it's done

heuristic tool, **minorminer** by [4], at the beginning of the first run and then reused in subsequent runs.

The D-Wave sampler is called and the returned samples are ordered lowest energy to highest energy and summed across classes to provide a consensus class score and hence classification.

3.6 Real Valued Features and Scaling Up

In work, e.g. [1, 6], where RBMs are trained on D-Wave the visible units are clamped by adding clamp strength as a constraint on the biases ensuring that v will retain its value in the annealing process. Here, we use the fact that v is known, so the second term in equation (3) becomes dependent on x_i only and $q_{i,j}$ can be seen as biases. In summary, real valued information about the input features entered the system through the biases q_v placed on the visible units v .

Fully connected layers involving large numbers of units are not suitable for the QPU. Instead we investigate the feasibility of using convolutional connections where connections between layers are reduced to a subset of units in each layer but where weights are shared for local and global connectivity. How these weights form part of the energy system Q is illustrated in Figure 2.

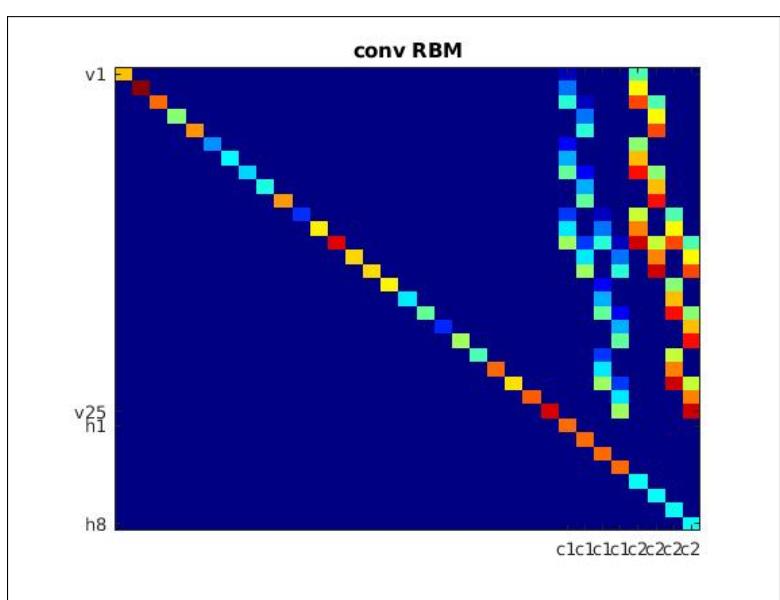


Figure 2: **Illustrative Q Matrix for a Convolutional Layer** comprising two convolutional filters, W_1 and W_2 , each containing 3×3 weights, passing over an image input (5×5) with stride 2. This results in two downsized features (2×2) where each interaction between input nodes, denoted $v_1 \dots v_{25}$, and output nodes, denoted $h_1 \dots h_8$, can be represented in matrix form as illustrated above. The interactions include biases (local field values), b , along the diagonal, and shared weights, W_1 and W_2 , down the columns $h_1 \dots h_4$ and $h_5 \dots h_8$ respectively. Each column denotes one of the eight output features.

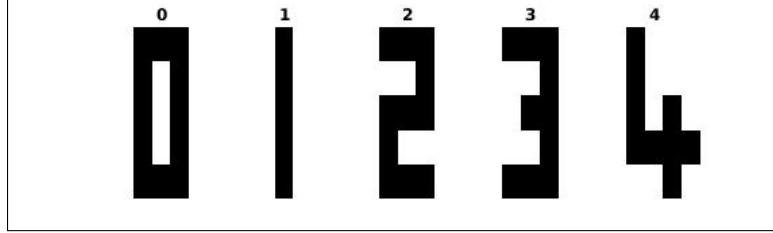


Figure 3: Binary digits 5×5 .

4 Results

In this Section we present results from sampling with a quantum annealer for two classification neural networks with five and ten classes, respectively. We finish the Section with a discussion on scaling up and timings.

4.1 Binary Digits 0:4 Classification

A small classification neural network was classically trained to distinguish between binary 5×5 images for digits $\{0, 1, 2, 3, 4\}$ and optimised model parameter values, W , were obtained. The network comprised 30 nodes: 20 visible feature nodes, 5 hidden nodes and 5 classification nodes (see Appendix for more details). The Q values for the QUBO were set as follows: $q_p c = W$, $q_v p = 1$, $q_{cc} = 2$ and $q_c = -1$. Input to this model were binary 5×5 images for digits $\{0, 1, 2, 3, 4\}$, see Figure 3, and the features included in the quadratic binary model were extracted from the convolutional layer after sigmoid activation. The biases and weights were then transferred to the QPU and 1000 samples taken. The 100 samples with the lowest energy were averaged to estimate the expected value of each classification node. The results for the five nodes assigned to the five classes clearly show dominance in the correct class, see Table 1. This simple example shows proof of principle that the weights from a classically trained CNN can be used to connect qubits for classification tasks.

4.2 MNIST classification

We investigate using the high level abstracted features from the penultimate layers of a neural network model trained using the MNIST database [12]. The quadratic binary model comprised 50 nodes: 40 feature nodes and 10 classification nodes. The Q values were $q_c = -1$, $q_{cc} = 2$ and $q_p c = W$ where W was obtained from the classification layer of the MNIST CNN. The biases and weights were then transferred to the QPU and 1000 samples taken. The 100 samples with the lowest energy were averaged to estimate the expected value of each classification node. The results for the ten nodes assigned to the ten classes show dominance in the correct class for digits 0, 3, 5 and 6. There was confusion between digits 1, 7 and 9. Digits 2 and 4 were never predicted whereas digit 8 was sometimes

Table 1: Binary Digit Classification. Class predictions *columns* based on 100 samples with the lowest energy out of 1000 samples are shown for each input class *rows*. All digits show dominance in the correct class.

Input Classes	Class Predictions				
	0	1	2	3	4
0	100	0	4	0	17
1	0	100	0	68	0
2	0	0	100	2	0
3	0	0	0	100	86
4	47	0	27	22	100

Table 2: MNIST Classification. Class predictions *columns* based on 100 samples with the lowest energy out of 1000 samples are shown for each input class *rows*. Digits 0,1,3,5,6 and 7 show dominance in the correct class. However there is confusion between 1,7 and 9. Digits 2, 4 and 8 are wrongly classified.

Input Classes	Class Predictions									
	0	1	2	3	4	5	6	7	8	9
0	98	1	29	0	0	0	17	60	0	0
1	10	100	0	0	0	0	0	100	0	0
2	36	100	0	0	0	0	0	100	0	0
3	0	14	1	86	0	79	2	29	0	0
4	55	1	2	12	0	14	20	1	31	46
5	0	0	0	69	0	100	40	0	0	0
6	88	0	0	0	0	100	100	0	0	0
7	0	100	0	0	0	0	0	100	0	0
8	96	38	0	0	0	0	0	50	42	0
9	0	100	0	0	0	0	0	100	0	0

predicted but with a high error rate, see Table 1. This experiment was performed on only one randomly chosen digit from each class and with one embedding for Q. With physical experiments, variations are to be expected with the set-up, for example within the individual qubits and couplers. Future work will run this experiment many more times to reduce this variation and we would expect to see more reliable predictions. This example shows that including high level abstracted features arising from one source or a combination as sources as biases in the quadratic binary models opens up the classification task to a broader field.

4.3 Scaling Up and Timing

Using our approach we were able to find an embedding and obtain samples from a four layer model comprising 358 nodes: 196 visible nodes, 128 convolutional feature nodes, 24 fully connected hidden nodes and 10 classification nodes. The convolutional window was 5×5 and eight windows passed over the input in strides of two, resulting in 8×16 outputs. We repeated the experiment on a classical computer and estimated that classification and printing out would take about 8000 microseconds whereas samples can be obtained from the QPU in 168 microseconds (anneal 20 μs Readout 127 μs Delay 21 μs) suggesting a possible speed up of at least one order of magnitude.

Problem here is we haven't the infra of the classical computer \rightarrow we cannot conclude on a speedup

5 Discussion and Conclusions

We have shown that it is possible to frame a classification task as a quadratic binary model using the weights from a classically trained classification neural network. This problem can then be sent to D-Wave's QPU for one-step quantum annealing. Given a trained network, D-Wave can be used to evaluate it.

Regarding timings, we have focused on the annealing time (around 20 μs). Currently, there are time overheads (access, programming, sampling and post-processing) involved with this process but these could be addressed by engineering and pipelines that improve streaming to and from the QPU for specific tasks such as classification.

We addressed two barriers to scaling up: binary input states which restrict types of data that can be analysed and number of couplings between nodes limited by the physical D-Wave graph; and required number of the model variables. We showed that real value features can be considered as biases that can be added directly to the biases of connecting nodes or subject to pooling and connecting node constraints. This illustration opens up the possibility of different types of input features including features derived from other systems or multi-faceted features from complex systems. We introduced pooling and convolutional couplings. Pooling serves two main purposes. First, down-sampling the input which can reduce the size of the network and hence the number of parameters needed (especially important for quantum computers), and reduce the computational cost, and can improve generalisation. Second, adding non-linearity, required for better model expressiveness, to the linear maps. Convolutional filters also reduce the connectivity load, and by extracting features from spatial settings improve model expressiveness. The largest network we were able run had four layers and 358 nodes.

In summary, providing neural networks with a quantum engine has the potential, assuming the pipeline for streaming data to the quantum computer can be made more efficient, to obtain classification results from high dimensional sources with speeds at least an order of magnitude.

References

- [1] Steven H. Adachi and Maxwell P. Henderson. Application of quantum annealing to training of deep neural networks. arXiv, 2015.
- [2] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. arXiv, 2016.
- [3] Antoine Buetti-Dinh, Vanni Galli, Sören Bellenberg, Olga Ilie, Malte Herold, Stephan Christel, Mariia Boretska, Igor V. Pivkin, Paul Wilmes, Wolfgang Sand, Mario Vera, and Mark Dopson. Deep neural networks outperform human expert's capacity in characterizing bioleaching bacterial biofilm composition. *Biotechnology Reports*, 22:e00321, 2019.
- [4] Jun Cai, William G. Macready, and Aidan Roy. A practical heuristic for finding graph minors. arXiv, 2014.
- [5] D-Wave Systems, 2021. <http://www.dwavesys.com/>, 2017-11-30.
- [6] Vivek Dixit, Raja Selvarajan, Muhammad A. Alam, Travis S. Humble, and Sabre Kais. Training and classification using a restricted Boltzmann machine on the d-wave 2000q, 2020.
- [7] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542:115 – 118, 2017.
- [8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016.
- [9] G. Hinton. A practical guide to training restricted Boltzmann machines. Technical Report UTM 2010–003, University of Toronto, 2010.
- [10] G. E. Hinton. Boltzmann machine. *Scholarpedia*, 2(5):1668, 2007. revision #91076.
- [11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.
- [12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.
- [13] Jeremy Liu, Ke-Thia Yao, and Federico Spedalieri. Dynamic topology reconfiguration of Boltzmann machines on quantum annealers. *Entropy*, 22(11), 2020.

- [14] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In Mark W. Jones Xianghua Xie and Gary K. L. Tam, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 41.1–41.12. BMVA Press, September 2015.
- [15] Jennifer Sleeman, John Dorband, and Milton Halem. A hybrid quantum enabled RBM advantage: Convolutional autoencoders for quantum image compression and generative learning, 2020.
- [16] Wenyi Zhou, Yang Yang, Cheng Yu, Juxian Liu, Xingxing Duan, Zongjie Weng, Dan Chen, Qianhong Liang, Qin Fang, Jiaoqiao Zhou, Hao Ju, Zhenhua Luo, Weihao Guo, Xiaoyan Ma, Xiaoyan Xie, Ruixuan Wang, and Luyao Zhou. Ensembled deep learning model outperforms human experts in diagnosing biliary atresia from sonographic gallbladder images. *Nat Commun*, 112(1259), 2021.

6 Appendix

6.1 Classically Trained Neural Network Specifications

6.1.1 Binary 5 x 5 Digits 0:4 Classification

The classically trained convolutional neural network comprised input, convolutional (filter size $W = 3$, $H = 3$ and $D = 5$, stride=2) sigmoid activation, max pooling (window $W = 2$ and $H = 2$, stride=2), fully connected (filter size $W = 5$ and $H = 5$, and softmax activation layers where W denotes width and H height. For input sized 5×5 the resulting features after the convolutional layer are sized $2 \times 2 \times 5$ and after the pooling layer are sized 1×5 .