Entangled q-convolutional neural nets

Anagiannis, V.; Cheng, M.C.N.

DOI
10.1088/2632-2153/ac2800

Publication date
2021

Document Version
Final published version

Published in
Machine Learning: Science and Technology

IOP Publishing Mach. Learn.: Sci. Technol. 2 (2021) 045026 https://doi.org/10.1088/2632-2153/ac2800

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
23 June 2021

REVISED
11 September 2021

ACCEPTED FOR PUBLICATION
17 September 2021

PUBLISHED
20 October 2021

PAPER

Entangled q-convolutional neural nets

Vassilis Anagiannis[1] **and Miranda C N Cheng**[1,2,*]

[1] Institute of Physics, University of Amsterdam, Amsterdam, The Netherlands
[2] Korteweg-de Vries Institute for Mathematics, University of Amsterdam, Amsterdam, The Netherlands
[*] Author to whom any correspondence should be addressed.

E-mail: mcheng@uva.nl

Keywords: quantum entanglement, convolutional neural network, quantum many-body system

Abstract

We introduce a machine learning model, the *q-CNN model*, sharing key features with convolutional neural networks and admitting a tensor network description. As examples, we apply q-CNN to the MNIST and Fashion MNIST classification tasks. We explain how the network associates a quantum state to each classification label, and study the entanglement structure of these network states. In both our experiments on the MNIST and Fashion-MNIST datasets, we observe a distinct increase in both the left/right as well as the up/down bipartition entanglement entropy (EE) during training as the network learns the fine features of the data. More generally, we observe a universal negative correlation between the value of the EE and the value of the cost function, suggesting that the network needs to learn the entanglement structure in order the perform the task accurately. This supports the possibility of exploiting the entanglement structure as a guide to design the machine learning algorithm suitable for given tasks.

1. Introduction

Convolutional neural networks (CNNs) have seen remarkable successes in various applications. At the same time there are tasks with similar descriptions that can nevertheless not be solved with a CNN architecture[3]. Moreover, it is not always transparent what choices of hyperparameters work the best, and why. More generally, we do not always have a precise explanation of why certain choices of machine learning architectures and hyperparameters work and do not work for a given task. This lack of a precise understanding is related to the curse of dimensionality which prevents an explicit analysis. That said, the data of a given problem typically lie in a high co-dimensional subspace. For instance, a typical point in the configuration space of all possible N-pixel grayscale pictures resembles a 'white noise' image, and looks nothing like a picture encountered in the relevant data set.

This is very reminiscent of the situation in quantum many-body systems. The high dimensionality of the Hilbert space of quantum states makes it hard to find the desired state (e.g. ground state of the given Hamiltonian) explicitly. Tensor network [2, 3] is one of the most popular tools utilised in many-body quantum physics to overcome this problem. Abstractly speaking, they provide a way to approximate high-order tensors in terms of lower-order tensors, and by doing so greatly reduce the parameters needed to describe the relevant quantum states, circumventing the curse of dimensionality. This is possible because the physically relevant states lie in a tiny 'corner of the Hilbert space'. One can quantify this using the *entanglement entropy* (EE) of a quantum state with respect to a bipartition of the system, which measures the degree to which the quantum state is entangled between two subsystems. While a typical element of the Hilbert space has an EE which scales like the volume of the sub-region, the physically relevant states tend to have entanglement entropies that scale like the boundary area (possibly with logarithmic corrections) of the sub-region. At the same time, the entanglement structure of a quantum state is precisely what constrains how effectively it could be approximated by a given tensor network architecture. See [4, 5] for a review.

[3] See, for instance, [1] for an example in physics.

The analogy between quantum many-body systems and machine learning prompts the following questions. Could we have a similar theoretical understanding in the context of machine learning architectures on how effective they are for a given task? Could we also understand the subspace of relevant data with similar tools as those used in the study of quantum many-body systems? Moreover, one might also hope that the analogy between quantum many-body systems and machine learning architectures can help the developments of natural and effective quantum machine learning architectures [6–8].

Inspired by these questions, there have been increasing efforts to build a bridge between the two fields of machine learning and quantum many-body systems, and in particular tensor networks [9–25]. In this work, we continue to strengthen this bridge, focusing on CNNs. Specifically, we build a CNN-like architecture, which we call q-CNN, which admits a description as a tensor network. In particular, our architecture has the same weight sharing property as the usual CNN, and as a result the number of parameters grows only logarithmically with the system size. Subsequently, we apply q-CNN to the classification tasks on the MNIST and Fashion-MNIST datasets, obtaining maximum test accuracy 97% and 89% respectively, comparable to [26].

With the confidence that our architecture has satisfactory performance, we then go ahead and explore its quantum mechanical properties. As mentioned before, a crucial probe of the qualitative features of a quantum states is its entanglement entropies. We compute the entanglement entropies of the network quantum states, with respect to bipartitions of the configuration space corresponding to the left/right and up/down partitions of an image. In the current context, the EE of the network given a specific bipartition of the image measures the degree to which the network captures the correlation between the two parts of the image. For instance, a vanishing EE implies that the network assigns a probability for the image to be in a certain category that can be expressed as the product of two probabilities corresponding to each part of the bipartition, which is a clearly an undesirable outcome.

Quantum EE is well-known to be an important quantity quantifying the important entanglement attributes of a quantum system. As a result, computationally it has been playing a crucial role in the design of tensor networks capable of performing computations pertaining to given quantum systems; too intricate an entanglement structure could pose a fundamental hurdle to computability via given tensor networks. See for instance [27] for a recent review. Similarly, from the analogy between quantum many-body systems and machine learning, it is reasonable to suspect that the concept of entanglement can also play the similar role as a guiding principle for choosing the right architectures and hyperparameters capable of solving a given problem. As a first step, in this work we established that entanglement is indeed a relevant structure the network needs to learn before it can start performing the tasks we designated.

In both our experiments on the MNIST and Fashion-MNIST datasets, we initialize the network in a random way which renders a quantum state with no particular entanglement structure. Subsequently, we observe a distinct increase in EE during training as the network learns the fine features of the data. More generally, we observe a negative correlation between the value of the EE and the value of the cost function, across different initialisations and choices of hyperparameters of the network. This constitutes convincing evidence that one of the structures of the data that the network needs to 'learn' is the entanglement structure. It can also be read out that the entanglement needed for the (Fashion-) MNIST classification tasks is low, which could be viewed as a 'justification' why a simple CNN is capable of performing these tasks.

1.1. Related work

The q-CNN architecture discussed in this work is based on the theoretical architecture, named deep convolutional arithmetic circuit, introduced in [23, 24]. The product pooling proposed in [23, 24] presents a challenge in training the network described above, since it can easily lead to numerical instabilities such as underflow or overflow. We would like to render the network trainable in practice, and do so without spoiling the analogy to quantum many-body systems. In [28] this architecture was trained using simnets [29], circumventing the numerical instability of product pooling by performing the calculations in logarithmic space. In q-CNN, we instead introduce additional batch normalisation layers which can easily be incorporated into the tensor description of the network, compatible with the quantum analogy.

As opposed to other works aiming to study and/or practice machine learning in ways inspired by quantum many-body physics [26, 30, 31], we train the network as a usual neural network with Pytorch instead of optimization schemes commonly used for tensor networks, such as DMRG. Also note that the number of parameters of our network grows merely logarithmically with the size of the image, as opposed to linearly as in the aforementioned approaches, since we retain the weight sharing feature of the usual CNN in our architecture. In [14], the EE of the final trained network was computed for a very different architecture directly related to tensor networks, but not its evolution during training and the correlation between entanglement and accuracy. In [32] and [24], the possibility was mentioned that the requirement of being capable to accommodate the entanglement could be used to guide the design of the network. We note that

the values of the EE in our experiments on the MNIST and F-MNIST datasets is of the order of $\log 2$. As a result, the entanglement is unlikely to be a bottleneck of the network performance for such tasks.

2. The q-CNN architecture

In this section we describe the architecture of q-CNN, as summarised in figure 1.

Consider a grayscale image with N pixels, corresponding to a point in the configuration space $\mathbf{x} = (x_1, \ldots, x_N) \in [0,1]^{\otimes N}$. For simplicity we have flattened the 2d image into a chain. We will take $N = 2^L$ with integral L, using padding if necessary. The first layer of the q-CNN is a (non-linear) representation layer:

$$\begin{aligned} \text{Rep}: \ & [0,1]^{\otimes N} \to (\mathbb{R}^{d_0})^{\otimes N} \\ & (x_1, \ldots, x_N) \mapsto (\boldsymbol{\zeta}_1^{(0)}, \boldsymbol{\zeta}_2^{(0)}, \ldots, \boldsymbol{\zeta}_N^{(0)}). \end{aligned} \tag{1}$$

Explicitly, we write $\zeta_{p,i}^{(0)}$ to be the ith component of the d_0-dimensional vector $\boldsymbol{\zeta}_p^{(0)}$, for $p \in \{1, \ldots, N\}$. For instance, $\zeta_{p,i}^{(0)} = f_i(x_p)$ for $i \in \{1, 2, \ldots, 2n\}$ in (19).

Subsequently, we will have L iterations of feature learning. Each iteration consists of three operations: batch normalisation, convolution, and pooling. In what follows we will discuss them individually.

Batch normalisation:

In q-CNN, we employ a somewhat unusual product pooling to correlate information captured in different spacial locations of the image. This product operation is prone to numerical instability such as overflow and underflow. To remedy this, we use batch normalisation to standardise the input features of each layer such that they have a specific (learnable) mean $\boldsymbol{\mu}^{(\ell)}$ and variance $\boldsymbol{\sigma}^{(\ell)}$ over the spacial and batch dimensions. In other words, in the ℓth iteration and for a given batch, let $\boldsymbol{\mu}_b^{(\ell)}$ and $\boldsymbol{\sigma}_b^{(\ell)}$ be the mean and variance of the input $\boldsymbol{\zeta}^{(\ell)}$ over the spatial (denoted by p above) dimensions and over the data points in the batch b. Then the batch normalisation layer amounts to the following affine transformation:

$$\begin{aligned} \text{BN}: \ & (\mathbb{R}^{d_\ell})^{\otimes 2^{L-\ell}} \to (\mathbb{R}^{d_\ell})^{\otimes 2^{L-\ell}} \\ & (\boldsymbol{\zeta}_1^{(\ell)}, \boldsymbol{\zeta}_2^{(\ell)}, \ldots, \boldsymbol{\zeta}_{2^{L-\ell}}^{(\ell)}) \mapsto (\boldsymbol{\zeta}'^{(\ell)}_1, \boldsymbol{\zeta}'^{(\ell)}_2, \ldots, \boldsymbol{\zeta}'^{(\ell)}_{2^{L-\ell}}), \end{aligned} \tag{2}$$

with $\boldsymbol{\zeta}'^{(\ell)}_p = \mathbf{w}^{(\ell)} \boldsymbol{\zeta}_p^{(\ell)} + \mathbf{z}^{(\ell)}$, or

$$\zeta'^{(\ell)}_{p,i} = w_i^{(\ell)} \zeta_{p,i}^{(\ell)} + z_i^{(\ell)} \tag{3}$$

in components. In the above, we have that $\mathbf{w}^{(\ell)} = \text{diag}(w_1^{(\ell)}, \ldots, w_{d_\ell}^{(\ell)})$ is a diagonal matrix and $\mathbf{z}^{(\ell)}$ is a vector, with entries

$$w_i^{(\ell)} := \frac{\sigma_{b,i}^{(\ell)}}{\sqrt{\sigma_{b,i}^2 + \epsilon}}, \quad z_i^{(\ell)} := \mu_i^{(l)} - \frac{\mu_{b,i}\sigma_i^{(\ell)}}{\sqrt{\sigma_{b,i}^2 + \epsilon}} \tag{4}$$

where ε is a small regularizing constant.

Convolution:

We consider a convolution of window size 1×1. Note that this is nevertheless non-trivial since we have $d_\ell > 1$ channels. As a result the convolution mixes information carried in different channels though not in different spacial locations. Writing the weight tensor in the ℓth layer as a matrix $\mathbf{a}^{(\ell)}$ of size $d_{\ell+1} \times d_\ell$, we have

$$\begin{aligned} \text{Conv}: \ & (\mathbb{R}^{d_\ell})^{\otimes 2^{L-\ell}} \to (\mathbb{R}^{d_{\ell+1}})^{\otimes 2^{L-\ell}} \\ & (\boldsymbol{\zeta}'^{(\ell)}_1, \boldsymbol{\zeta}'^{(\ell)}_2, \ldots, \boldsymbol{\zeta}'^{(\ell)}_{2^{L-\ell}}) \mapsto (\boldsymbol{\xi}_1^{(\ell)}, \boldsymbol{\xi}_2^{(\ell)}, \ldots, \boldsymbol{\xi}_{2^{L-\ell}}^{(\ell)}), \end{aligned} \tag{5}$$

with $\boldsymbol{\xi}_p^{(\ell)} = \mathbf{a}^{(\ell)} \boldsymbol{\zeta}'^{(\ell)}_p$, or

$$\xi_{p,j}^{(\ell)} = \sum_i a_{ji}^{(\ell)} \zeta'^{(\ell)}_{p,i} \tag{6}$$

in components. Note that the tensor $\mathbf{a}^{(\ell)}$ does not depend on the spacial location p, a feature often referred to as weight sharing in the context of the usual CNN architecture.

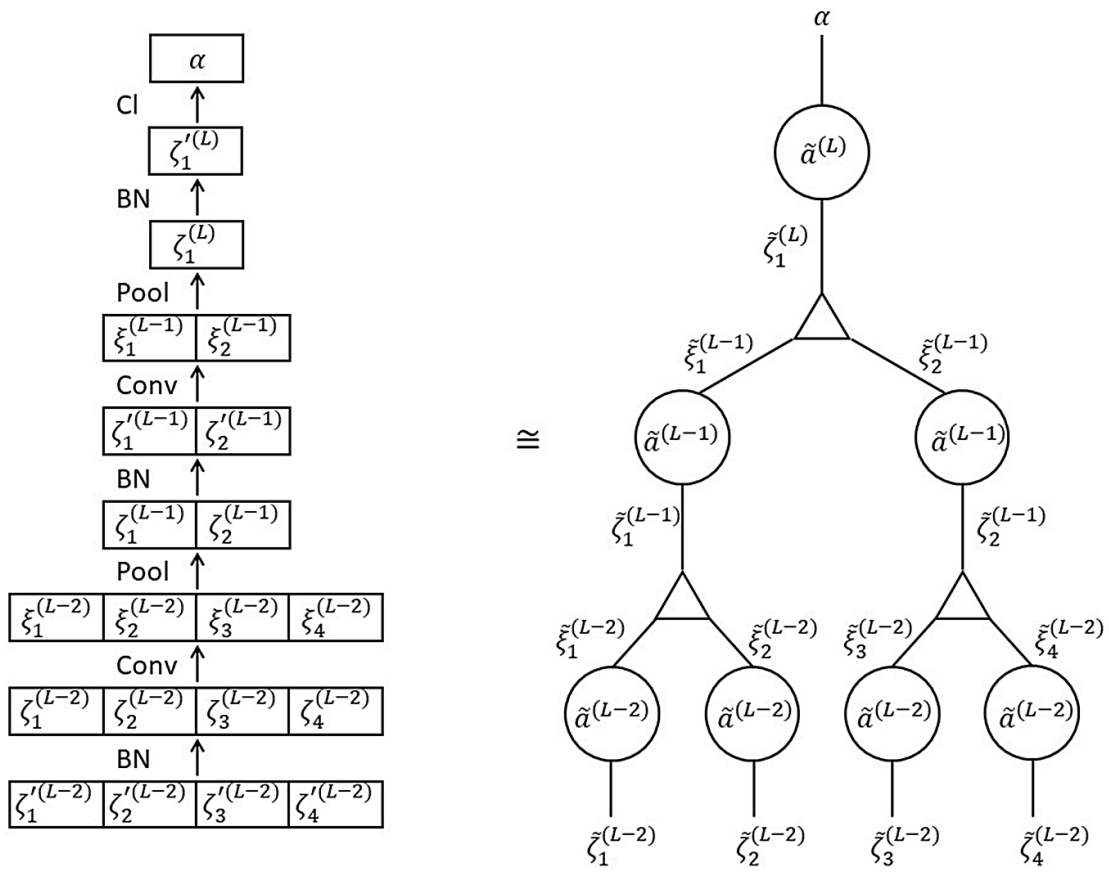

Figure 1. The neural network architecture used in this paper (left), and its tensorial description (right). The triangles represent the delta tensor implementing the same-channel product pooling (8). The circles represent the matrix multiplication as in (6).

Product pooling:

Following each such convolution there is a product pooling operation which perform the same-channel product of the corresponding (one-dimensional) features in non-overlapping spacial windows of size 2, thus reducing the spacial size of the feature map by a factor of 2. In other words, we have

$$\begin{aligned} \text{Pool}: \ & (\mathbb{R}^{d_{\ell+1}})^{\otimes 2^{L-\ell}} \to (\mathbb{R}^{d_{\ell+1}})^{\otimes 2^{L-\ell-1}} \\ & (\boldsymbol{\xi}_1^{(\ell)}, \boldsymbol{\xi}_2^{(\ell)}, \ldots, \boldsymbol{\xi}_{2^{L-\ell}}^{(\ell)}) \mapsto (\boldsymbol{\zeta}_1^{(\ell+1)}, \boldsymbol{\zeta}_2^{(\ell+1)}, \ldots, \boldsymbol{\zeta}_{2^{L-\ell-1}}^{(\ell+1)}), \end{aligned} \tag{7}$$

where

$$\zeta_{p,i}^{(\ell+1)} = \xi_{2p-1,i}^{(\ell)} \xi_{2p,i}^{(\ell)} = \sum_{j,k} \delta_{i,j,k} \xi_{2p-1,j}^{(\ell)} \xi_{2p,k}^{(\ell)}. \tag{8}$$

Classification:

The last pooling layer is followed by a batch normalisation operation and a dense linear layer (trivially a convolution on a 1×1 feature map), which maps the remaining features to the classification space of $|\mathcal{C}|$ labels,

$$\begin{aligned} \text{Cl}: \ & \mathbb{R}^{d_L} \to \mathbb{R}^{|\mathcal{C}|} \\ & \boldsymbol{\zeta}'^{(L)}_1 \mapsto \boldsymbol{\alpha}, \end{aligned} \tag{9}$$

with $\boldsymbol{\alpha} = \mathbf{a}^{(L)} \boldsymbol{\zeta}'^{(L)}_1$, or

$$\alpha_y = \sum_i a_{yi}^{(L)} \zeta'^{(L)}_{1,i} \tag{10}$$

in components.

2.1. Tensor description

As mentioned before, without batch normalisation, the architecture consisting of the specific form of convolutional and product pooling layers described above is the convolutional arithmetic circuit that the authors proposed in [23], and further studied in [24, 33]. It was also pointed out that such an architecture implements a (hierarchical) Tucker decomposition of the network tensors. To have a trainable network we additionally apply batch normalisation. Naively, this destroys the description of the network as a tensor operation, since batch normalisation (3) is an affine instead of a linear transformation. However, this can be easily remedied by adding an additional dimension, corresponding to the 'constant term', in all layers. Concretely, we can equivalently describe the affine transformation (2) as the following linear transformation:

$$\begin{aligned} \mathrm{BN}: \ & (\mathbb{R}^{d_\ell+1})^{\otimes 2^{L-\ell}} \to (\mathbb{R}^{d_\ell+1})^{\otimes 2^{L-\ell}} \\ & (\tilde{\zeta}_1^{(\ell)}, \tilde{\zeta}_2^{(\ell)}, \ldots, \tilde{\zeta}_{2^{L-\ell}}^{(\ell)}) \mapsto (\tilde{\zeta'}_1^{(\ell)}, \tilde{\zeta'}_2^{(\ell)}, \ldots, \tilde{\zeta'}_{2^{L-\ell}}^{(\ell)}), \end{aligned} \tag{11}$$

where

$$\tilde{\zeta}_p^{(\ell)} = \begin{pmatrix} 1 \\ \zeta_p^{(\ell)} \end{pmatrix}, \quad \tilde{\zeta'}_p^{(\ell)} = \begin{pmatrix} 1 \\ \zeta'^{(\ell)}_p \end{pmatrix}, \tag{12}$$

and $\tilde{\zeta'}_p^{(\ell)} = \tilde{\mathbf{w}}^{(\ell)} \tilde{\zeta}_p^{(\ell)}$, with

$$\tilde{\mathbf{w}}^{(\ell)} = \begin{pmatrix} 1 & 0_{1\times d_\ell} \\ \mathbf{z} & \mathbf{w}^{(\ell)} \end{pmatrix}. \tag{13}$$

At the same time, the convolution and product pooling layers can also be described in terms of the $(d+1)$-dimensional vectors $\tilde{\zeta}_p^{(\ell)}$, $\tilde{\zeta'}_p^{(\ell)}$ and $\tilde{\xi}_p^{(\ell)}$ in a straightforward way. For instance, the batch normalisation and the convolutional layers can be described as a combined tensorial operation:

$$\begin{aligned} \mathrm{BN}+\mathrm{Conv}: \ & (\mathbb{R}^{d_\ell+1})^{\otimes 2^{L-\ell}} \to (\mathbb{R}^{d_{\ell+1}+1})^{\otimes 2^{L-\ell}} \\ & (\tilde{\zeta}_1^{(\ell)}, \tilde{\zeta}_2^{(\ell)}, \ldots, \tilde{\zeta}_{2^{L-\ell}}^{(\ell)}) \mapsto (\tilde{\xi}_1^{(\ell)}, \tilde{\xi}_2^{(\ell)}, \ldots, \tilde{\xi}_{2^{L-\ell}}^{(\ell)}) \end{aligned} \tag{14}$$

with $\tilde{\xi}_p^{(\ell)} = \tilde{\mathbf{a}}^{(\ell)} \tilde{\zeta'}_p^{(\ell)}$, where

$$\tilde{\mathbf{a}}^{(\ell)} = \begin{pmatrix} 1 & 0_{1\times d_\ell} \\ \mathbf{a}^{(\ell)}\mathbf{z} & \mathbf{a}^{(\ell)}\mathbf{w}^{(\ell)} \end{pmatrix} \tag{15}$$

for $\ell = 0, 1, \ldots, L-1$. In the final classification layer, we simply have

$$\tilde{\mathbf{a}}^{(L)} = \begin{pmatrix} \mathbf{a}^{(\ell)}\mathbf{z} & \mathbf{a}^{(\ell)}\mathbf{w}^{(\ell)} \end{pmatrix} \tag{16}$$

since we do not need the additional constant channel in the final output.

As a result, the q-CNN we defined above can also be described in terms of tensor networks, as we will further describe in the following section.

3. Quantum properties

In this section we discuss the description the data and the q-CNN network in the language of quantum many-body systems, which then enables the definition and calculation of the EE of the network states.

3.1. A quantum description

To describe the neural network introduced in section 2, let us first describe our quantum Hilbert space in terms of a space of L^2-integrable complex functions[4]. Explicitly, consider

$$L^2(S^1, \mathbb{C}) := \{f: [0,1] \to \mathbb{C} \mid f(0) = f(1), \int_0^1 dx \overline{f(x)} f(x) \ \text{exists}\}. \tag{17}$$

[4] In this current work we will only use real functions in practice, though the generalisation allowing for complex functions is straightforward, at least in theory.

As is well-known, this space is equipped with a natural norm

$$\langle f,g\rangle := \int_0^1 dx\overline{f(x)}g(x), \tag{18}$$

and an orthonormal basis is given by the Fourier basis[5]

$$f_0(x)=1,\ \ f_{2k-1}(x)=\sqrt{2}\cos(2\pi kx),\ \ f_{2k}(x)=\sqrt{2}\sin(2\pi kx) \tag{19}$$

for $k\in\mathbb{N}$. To obtain a finite-dimensional Hilbert space, we restrict to a subspace by truncating the modes with frequency larger than a given $n\in\mathbb{N}$:

$$L^2_n(S^1) := \{f\in L^2(S^1,\mathbb{C})|\langle f,f_k\rangle = 0\ \ \forall\, k>2n\}. \tag{20}$$

We will identify this space with the our *local Hilbert space*

$$\mathcal{H}_{\rm loc}\cong L^2_n(S^1)\cong\mathbb{C}^{2n+1} \tag{21}$$

describing the state of the local pixel (or spin in the physics analogy). Explicitly, corresponding to the orthonormal basis (19) for $L^2_n(S^1)$, we introduce an orthonormal basis $|f_0\rangle,\ldots,|f_{2n}\rangle$ for $\mathcal{H}_{\rm loc}$, and it follows from the orthonormality that

$$\sum_{i=0}^{2n}|f_i\rangle\langle f_i| = \mathbf{1}_{\mathcal{H}_{\rm loc}}.$$

We also introduce $\langle x|\in\mathcal{H}^*_{\rm loc}$ to be giving the *evaluation map*:

$$\langle x|f\rangle = \mathbf{ev}(f,x)=f(x). \tag{22}$$

In other words, we can think of $|x\rangle$ as the position eigenstates, and $f(x)$ can be thought of as the corresponding wavefunction associated with the state $|f\rangle$. Explicitly, one has

$$\langle x| = \sum_{i=0}^{2k} f_i(x)\langle f_i|, \tag{23}$$

and

$$\int_0^1 dx\,|x\rangle\langle x| = \sum_{i=0}^{2n}|f_i\rangle\langle f_i| = \mathbf{1}_{\mathcal{H}_{\rm loc}}. \tag{24}$$

Note also that

$$\langle x|x'\rangle = D_n(x-x')\ ,\ \ D_n(z)=1+2\sum_{k=1}^{n}\cos(2\pi kz), \tag{25}$$

is the so-called Dirichlet kernel on $L^2_n(S^1)$, and has the Dirac delta function $\delta(x)$ as the limit when $n\to\infty$.

Now, consider a system with N pixels (or lattice sites in the physics analogy), and we have

$$\mathcal{H}=\mathcal{H}_{\rm loc}^{\otimes N}\cong L^2_n(T^N)\cong(L^2_n(S^1))^{\otimes N}. \tag{26}$$

We will write the coordinates $\mathbf{x}=(x_1,\ldots,x_N)$ for the N-torus $T^N=[0,1)^{\otimes N}$, and introduce the following natural (orthonormal) basis for $\mathcal{H}$,

$$|f_{i_1\ldots i_N}\rangle = \bigotimes_{p=1}^{N}|f_{i_p}\rangle_p,\quad i_p\in\{0,1,\ldots,2n\}, \tag{27}$$

[5] Apart from the Fourier basis, other bases may be used as well. Concretely, we have implemented network with the Legendre basis for $L^2([-1,1])$, with which we obtain similar results in terms of accuracies.

corresponding to the basis $|f_i\rangle_p$ for the local Hilbert space of pth pixel.

Identifying a greyscale image with N pixels with a point in $[0,1]^{\otimes N}$, the 'position eigenstate' $|\mathbf{x}\rangle$ corresponds in our case to a specific image. Note that, by working with the space (26), we restrict ourselves to a periodic representation that is invariant under the action of swapping a zero with an one[6]. Just as in the single pixel/particle case, its expansion in the orthonormal basis (27) reads

$$|\mathbf{x}\rangle = \sum_{\mathbf{i}\in\{0,1,\ldots,2n\}^{\otimes N}} \Psi_{\mathbf{i}}(\mathbf{x})|f_{\mathbf{i}}\rangle \tag{28}$$

where we have written $\mathbf{i} = (i_1,\ldots,i_N)$, and the coefficient function reads[7]

$$\Psi_{\mathbf{i}}(\mathbf{x}) = \prod_{p=1}^{N} f_{i_p}(x_p). \tag{29}$$

It then follows immediately that the final score function output from our neural network can be viewed as the value of the wavefunction corresponding to the 'network state'[8]

$$|\Psi_y\rangle = \sum_{\mathbf{i}\in\{0,1,\ldots,2n\}^{\otimes N}} \Psi_{y;\mathbf{i}}|f_{\mathbf{i}}\rangle \tag{30}$$

corresponding to the label $y \in \mathcal{C}$:

$$\alpha_y(\mathbf{x}) := \sum_{\mathbf{i}\in\{0,1,\ldots,2n\}^{\otimes N}} \Psi_{\mathbf{i}}(\mathbf{x})\,\Psi_{y;\mathbf{i}} = \langle\mathbf{x}|\Psi_y\rangle. \tag{31}$$

At this point, it is tempting to associate to our wavefunction the usual probabilitic interpretation à la Born's rule. To do this, introduce the normalised network states

$$|\Psi_{y,0}\rangle := \frac{1}{\sqrt{\sum_{y\in\mathcal{C}}\langle\Psi_y|\Psi_y\rangle}}|\Psi_y\rangle \tag{32}$$

and define the joint probability density function

$$p(y,\mathbf{x}) := |\langle\mathbf{x}|\Psi_{y,0}\rangle|^2. \tag{33}$$

Building a (generative) network learning the states $|\Psi_{y,0}\rangle$ that gives a good approximation to the above probability density function is beyond the scope of the current paper; here we focus on the classification task and hence we can only trust $p(y,\mathbf{x})$ in the subspace of $\mathcal{H}$ where $\mathbf{x}$ resembles the training data in some way. Instead, for the classification task at hand, the relevant probability is the conditional probability

$$p(y|\mathbf{x}) = \frac{p(y,\mathbf{x})}{p(\mathbf{x})} = \frac{|\langle\mathbf{x}|\Psi_{y,0}\rangle|^2}{\sum_{y'\in\mathcal{C}}|\langle\mathbf{x}|\Psi_{y',0}\rangle|^2} = \frac{|\langle\mathbf{x}|\Psi_y\rangle|^2}{\sum_{y'\in\mathcal{C}}|\langle\mathbf{x}|\Psi_{y'}\rangle|^2}. \tag{34}$$

As is manifest from the last equality, the conditional probability is insensitive to the normalisation (32) of the network state $|\Psi_y\rangle$. This justifies our classification given the network output $\alpha_y(\mathbf{x}) = \langle\mathbf{x}|\Psi_y\rangle$:

$$y(\mathbf{x}) := \arg\max_y |\alpha_y(\mathbf{x})|. \tag{35}$$

Note that this is different from the more common ways of assigning probabilities to the outputs of such a classification network, such as through a softmax function.

3.2. Entanglement entropy

Equipped with the interpretation described in section 3.1 of our architecture as quantum states, we are now ready to 'measure' the behaviour of the the neural network using quantum mechanical tools. In particular, in this subsection we will discuss the EE of the network.

[6] But not invariant under swapping 0.05 and 0.95, For this reason the periodicity is not so significant in practice for real data. The choice for a periodic representation is simply made for convenience and does not seem to hinder the classification in our experiments.

[7] Note that $|\mathbf{x}\rangle$ is a product state and is in particular not entangled.

[8] Note that a wavefunction is complex-valued in general. As mentioned earlier, here we work with real functions for convenience and have $\Psi^*_{y;\mathbf{i}} = \Psi_{y;\mathbf{i}}$.

As the name suggests, the EE measures the extent to which a quantum state is entangled across two different subsystems, $\mathtt{U}$ and $\overline{\mathtt{U}}$. It is defined as the von Neumann entropy of the reduced density matrix. For instance, if the state is a product state of two separate systems, then its EE (with respect to the subsystems) vanishes.

Here, we consider a bipartition $(\mathtt{U}, \overline{\mathtt{U}})$ of the images that spatially splits the pixels $\mathbf{x} = (\mathbf{x}_{\mathtt{U}}, \mathbf{x}_{\overline{\mathtt{U}}})$ into two groups, with the corresponding Hilbert space decomposition $\mathcal{H} = \mathcal{H}_{\mathtt{U}} \otimes \mathcal{H}_{\overline{\mathtt{U}}}$. To discuss EE of different network states corresponding to different classification labels $y \in \mathcal{C}$, we normalise the network state as

$$|\Psi_{y,*}\rangle := \frac{1}{\sqrt{\langle\Psi_y|\Psi_y\rangle}}\,|\Psi_y\rangle. \tag{36}$$

Note that this is a different normalisation from (32). The corresponding reduced density matrix, obtained by tracing the density matrix over the subspace $\mathcal{H}_{\overline{\mathtt{U}}}$, reads

$$\rho_{y;\mathtt{U}} := \mathrm{Tr}_{\mathcal{H}_{\overline{\mathtt{U}}}}\left[|\Psi_{y,*}\rangle\langle\Psi_{y,*}|\right], \tag{37}$$

which satisfies $\mathrm{Tr}_{\mathcal{H}_{\mathtt{U}}}\rho_{y;\mathtt{U}} = \langle\Psi_{y,*}|\Psi_{y,*}\rangle = 1$. Then the EE of the network state $|\Psi_{y,*}\rangle$ corresponding to the bipartition $(\mathtt{U}, \overline{\mathtt{U}})$ is given by

$$\mathcal{S}(\rho_{y;\mathtt{U}}) = -\mathrm{Tr}_{\mathcal{H}_{\mathtt{U}}}\left(\rho_{y;\mathtt{U}}\log\rho_{y;\mathtt{U}}\right). \tag{38}$$

This quantity measures the extent to which the quantum state is entangled across the subspaces $\mathtt{U}$ and $\overline{\mathtt{U}}$, i.e. the degree to which the quantum state fails to be separable into two parts, belonging to two subsystems. For instance, if $|\Psi_y\rangle$ is a product state $|\Psi_y\rangle = |\Psi_y\rangle_{\mathtt{U}} \otimes |\Psi_y\rangle_{\overline{\mathtt{U}}}$ with $|\Psi_y\rangle_{\mathtt{U}} \in \mathcal{H}_{\mathtt{U}}$ and $|\Psi_y\rangle_{\overline{\mathtt{U}}} \in \mathcal{H}_{\overline{\mathtt{U}}}$, then the probability (34) also takes the form of a product of contribution from $\mathtt{U}$ and $\overline{\mathtt{U}}$, and we have $\mathcal{S}(\rho_{y;\mathtt{U}}) = 0$.

In what follows we will focus on the bipartition with $\mathtt{U}$ and $\overline{\mathtt{U}}$ each contains $N/2$ pixels (or lattice sites), corresponding to the two inputs of the top pooling tensor. Computationally, this is the biparition whose EE is easiest to compute, corresponding to the fact that one needs to cut the least number of legs in the diagram. In the next section we flatten the input images in ways such that this bipartition corresponds to either the left/right or the up/down separation of the images. In this case, analogous to the basis (27) for the total Hilbert space $\mathcal{H}$, we introduce the corresponding orthonormal basis for $\mathcal{H}_{\mathtt{U}} \cong \mathcal{H}_{\overline{\mathtt{U}}} \cong (\mathbb{C}^{2n+1})^{\otimes N/2}$:

$$|f_{\mathbf{i}}\rangle = |f_{i_1}\rangle \otimes \cdots \otimes |f_{i_{\frac{N}{2}}}\rangle, \tag{39}$$

for $\mathbf{i} = (i_1, \ldots, i_{\frac{N}{2}}) \in \{0, 1, \ldots, 2n\}^{\otimes\frac{N}{2}}$. The top pooling layer of the tensor decomposition gives the following expression

$$|\Psi_y\rangle = \sum_I \tilde{\mathbf{a}}^{(L)}_{yI}\,|\phi_I\rangle_{\mathtt{U}} \otimes |\phi_I\rangle_{\overline{\mathtt{U}}}\,, \tag{40}$$

where

$$|\phi_I\rangle = \sum_{\mathbf{i}\in\{0,1,\ldots,2n\}^{\otimes\frac{N}{2}}} \phi_{I,\mathbf{i}}|f_{\mathbf{i}}\rangle \tag{41}$$

for both $|\phi_I\rangle_{\mathtt{U}}$ and $|\phi_I\rangle_{\overline{\mathtt{U}}}$. Recall that $|\phi_I\rangle_{\mathtt{U}} = |\phi_I\rangle_{\overline{\mathtt{U}}}$ under the natural isomorphism $\mathcal{H}_{\mathtt{U}} \cong \mathcal{H}_{\overline{\mathtt{U}}}$, and this is a consequence of the weight sharing feature of our architecture. Also note that the y-dependence in $|\Psi_y\rangle$ comes entirely from the top layer tensor $\mathbf{a}^{(L)}$.

Putting it together, we have the following expression for the reduced density matrix (37)

$$\begin{aligned}\rho_{y;\mathtt{U}} &= \frac{1}{\langle\Psi_y|\Psi_y\rangle}\sum_{I,J}(\tilde{a}^{(L)}_{yJ})^*\,\tilde{a}^{(L)}_{yI}\,\mathrm{Tr}_{\mathcal{H}_{\overline{\mathtt{U}}}}\left(|\phi_I\rangle_{\overline{\mathtt{U}}} \otimes |\phi_I\rangle_{\mathtt{U}}\,{}_{\mathtt{U}}\langle\phi_J| \otimes {}_{\overline{\mathtt{U}}}\langle\phi_J|\right)\\ &= \frac{1}{\langle\Psi_y|\Psi_y\rangle}\sum_{\mathbf{i},\mathbf{i}'\in\{0,1,\ldots,2n\}^{\otimes\frac{N}{2}}}(M_y)_{\mathbf{i},\mathbf{i}'}|f_{\mathbf{i}}\rangle\langle f_{\mathbf{i}'}|\end{aligned} \tag{42}$$

where the matrix elements are given by

$$(M_y)_{\mathbf{i},\mathbf{i}'} = \sum_{I,J}\sum_{\mathbf{i}''\in\{0,1,\ldots,2n\}^{\otimes\frac{N}{2}}}(\tilde{a}^{(L)}_{yJ})^*\,\tilde{a}^{(L)}_{yI}\,\phi_{I,\mathbf{i}''}\,\phi^*_{J,\mathbf{i}''}\,\phi_{I,\mathbf{i}}\,\phi^*_{J,\mathbf{i}'}. \tag{43}$$

From the eigenvalues $m_{\mathbf{i}}$, $\mathbf{i} \in \{0, 1, \ldots, 2n\}^{\otimes \frac{N}{2}}$ of the above matrix, we readily compute the EE (38) of the network states to be

$$\mathcal{S}(\rho_{y;\mathrm{U}}) = -\sum_{\mathbf{i}\in\{0,1,\ldots,2n\}^{\otimes \frac{N}{2}}} p_{\mathbf{i}} \log p_{\mathbf{i}}, \tag{44}$$

with

$$p_{\mathbf{i}} := \frac{m_{\mathbf{i}}}{\sum_{\mathbf{i}'} m_{\mathbf{i}'}}. \tag{45}$$

4. Experiments

4.1. Setup

We trained the network on two datasets, MNIST [34] and Fashion-MNIST [35], using PyTorch [36]. Each contains 60 000 training images, 10 000 test images, and a total number of $|\mathcal{C}| = 10$ classes. The images are resized to 16×16, so that the flattened array contains $N = 256 = 2^8$ elements, resulting in a depth $L = 8$ network.

The number of parameters of the network can easily be calculated,

$$P(N; \{d_\ell\}) = \sum_{l=0}^{L=\log_2 N} (d_\ell d_{\ell+1} + 2d_\ell), \tag{46}$$

where the first term comes from the convolution (without bias), and the second from the batch normalisation layers. Note that, in the case that d_ℓ is independent of ℓ, the number of parameters grows like $\log_2 N$ with N.

Our main examples will be networks with number of channels increasing with the depth, $d_\ell = d(\ell + 1)$ for $\ell = 0, \ldots, L$, and $d_{L+1} = |\mathcal{C}| = 10$ for the final layer. This results in the following total number of parameters

$$\begin{aligned} P(N) &= \frac{d^2}{3}(\log_2 N)^3 + (d^2 + d)(\log_2 N)^2 + \left(\frac{2}{3}d^2 + 13d\right)\log_2 N + 12d, \\ P(2^8) &= 240d^2 + 180d, \end{aligned} \tag{47}$$

which scales quadratically with d and merely with (powers of) $\log_2 N$, a result of the tree tensor network structure, which eliminates quadratic growth, as well as weight-sharing, which eliminates linear growth with N . We used values of d from 4 up to 40, resulting in a number of parameters in the range $(1320, 391\,200)$.

Classification of an input $\mathbf{x}$ is done by choosing the label y for which the output $|\alpha_y(\mathbf{x})|$ is the largest (cf (35)), consistent with the quantum interpretation discussed in section 3.1. We used the following square distance loss function,

$$\mathcal{L} = \frac{1}{|\mathcal{B}|}\sum_{b=1}^{|\mathcal{B}|}\sum_{y=1}^{|\mathcal{C}|}\left(\alpha_y(\mathbf{x}) - \delta_{y,\mathfrak{l}(\mathbf{x})}\right)^2, \tag{48}$$

where $\mathfrak{l}(\mathbf{x})$ is the correct label of $\mathbf{x}$, the first sum is over the images in the batch and the second is over the different labels. Reducing this cost means that the output in the correct label channel moves closer to 1, while the outputs in the rest of the channels move closer to 0[9].

Optimization was done by using the AdamW optimizer [37], with weight decay parameter 0.01 and learning rate parameter 0.01, which was reduced by half every ten epochs. Batch size was chosen equal to 50 and we trained for a total of 90 epochs. Tensor network computations for evaluating the EE were done using the Tensornetwork library [38]. The corresponding code can be found here.

4.2. Results and discussion

The best (average) test accuracies achieved among all experiments were 97% for MNIST and 89% for Fahsion-MNIST, which are comparable to the results obtained in [26]. Note that since our main goal is the

[9] Given (34) and (35) it might be tempting to use the loss function $|\alpha_y(\mathbf{x})|$ or $|\alpha_y(\mathbf{x})|^2$, or even just $-\log\left(\frac{|\alpha_{\mathfrak{l}(\mathbf{x})}(\mathbf{x})|^2}{\sum_y |\alpha_y(\mathbf{x})|^2}\right)$. However, these choices all give worse training results than (48). One simple reason is just that the above loss is invariant under a sign change of α_y and hence does not admit a unique minimum.

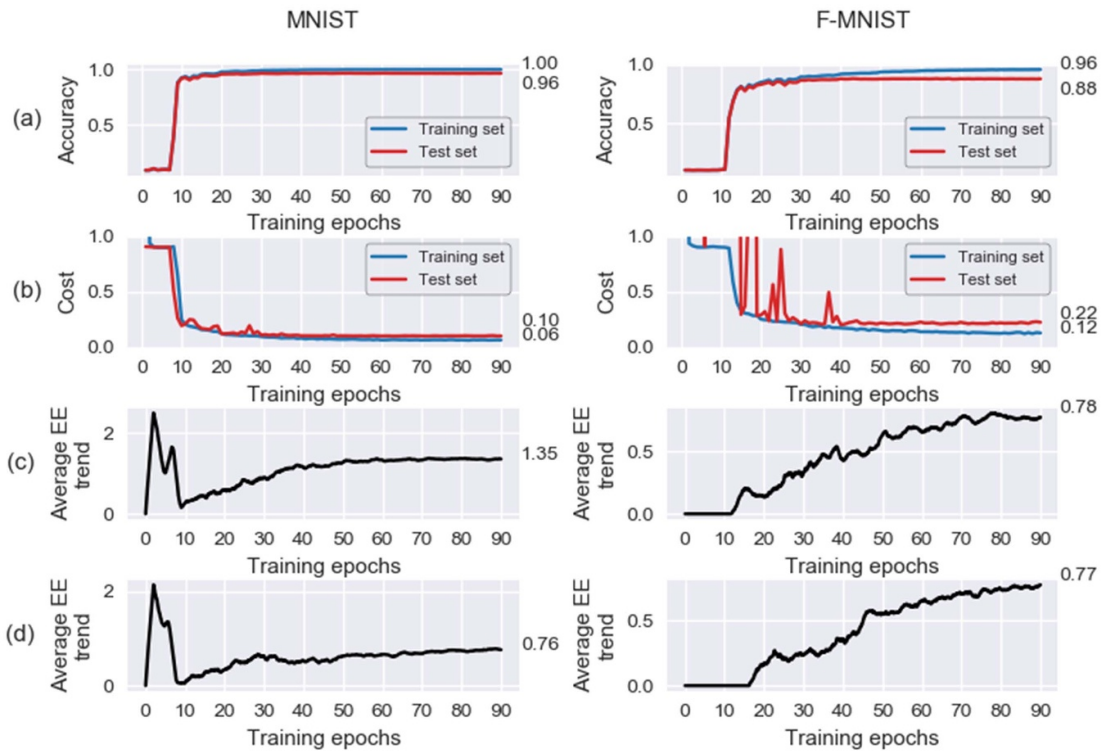

Figure 2. (a), (b) Accuracy and cost during training for a typical run of an experiment with $d = 18$ for both datasets. (c) The corresponding left/right average EE trend, where we average over the classes with an averaging window of two epochs. (d) The up/down average EE trend for a different run with the same hyperparameters.

study of the entanglement structure of this type of network architecture instead of achieving the very best classification results, we did not perform an exhaustive hyperparameter search, so it is likely that the achieved accuracies can be improved further. Figure 2 depicts a typical (as opposed to optimal in terms of attained accuracies) run of an experiment with $d = 18$ (81 000 parameters), for both datasets[10]. Notice that after epochs 30–50 the test accuracy and cost value start to slowly approach their final values. Figure 2(c) shows the trend (averaged over two epochs) of the average EE of all ten output channels, according to a left/right partition, as it develops during training. Upon initialization of the network the EE is practically zero, and during the first few epochs its value can vary widely among different initialization seeds, optimizers and number of hyperparameters. However, after the accuracy and the cost value has stopped changing rapidly, which happens after around ten epochs, the EE starts to increase steadily. This signifies a rise in the degree of correlations that are built between the left/right parts of the network in order to classify the images with more precision. This is to be contrasted with the 'accidental' entanglement that appear during the first ten epochs in the MNIST case, when the network is not yet able to correctly classify the images systematically.

Note that in the MNIST case the value of the EE seems to stabilise during the final epochs, whereas it appears to keep growing in the case of F-MNIST. A possible explanation is that at an accuracy of ~88% there are many more misclassifications in the latter case compared to the former case, and thus the minimum EE needed for near-optimal classification has not yet been reached. In support of this, we note that after around epoch 30 the F-MNIST plot resembles the region between epochs 10 to 40 of the MNIST plot, where the corresponding accuracy is still lower than the final achieved value.

Figure 2(d) shows the average EE trend for a different run with the same hyperparameters, but with a different flattening of the images such that the top pooling corresponds to an up/down division of the images. Note that the development trend of the up/down EE is very similar to that of the left/right EE. We believe that the heuristic explanation of the trend mentioned above for the left/right EE also applies to the case of up/down EE.

As mentioned before, in the (final value of the) EE of a rather different neural network has been measured in the context of MNIST dataset. We note that the order of magnitude of the final EE we measured

[10] In the experiments, we do not notice a very significant difference in performance in the range $d = 10$–40.

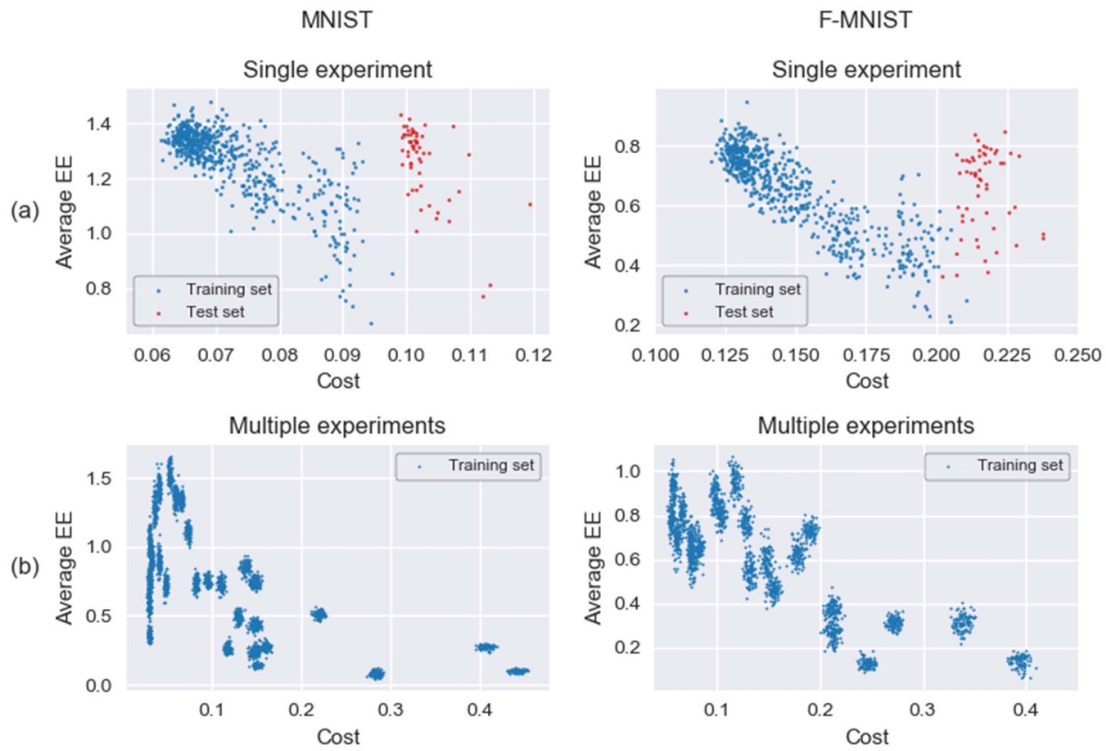

Figure 3. (a) The average EE trend versus the value of the cost function for a typical run of an experiment with $d = 18$, for both datasets. (b) The corresponding plots across multiple experiments.

is similar to that in suggesting that EE is indeed a robust quantitative measure of certain key properties of the given tasks.

Finally, in figure 3(a) we depict the average EE versus the value of the cost function, for the same run throughout the training process, as recorded in figures 2(a)–(c). The data before the 30th epoch are discarded due to their large fluctuations, as noted earlier. We observe a distinct negative correlation between EE and the value of the cost function, which is to say that higher values of the EE tend to appear when the network has lower values for the cost function. This can be seen more easily in figure 3(b), where the plots contain data from many different experiments, across different initialization seeds and choices of hyperparameters, which appear as 'islands' in a larger landscape that also exhibits some degree of negative correlation. This is consistent with our interpretation that the EE of the network starts increasing steadily only as the network starts learning the finer features of the data. Also note that this negative correlation is less evident in the test cost, here depicted by the red dots; we believe that this is due to the fact that the test cost drops much slower than the training cost after a certain point during training. This suggests that the aforementioned increase of the EE is in part also tied to overfitting.

In this work, we have introduced the q-CNN architecture and demonstrated that it is capable to perform the classification tasks on the MNIST and Fashion-MNIST datasets. We have subsequently computed the entanglement entropies given the bipartition corresponding to the left/right and up/down of the images. The results of these experiments provide evidence that (1) the entanglement structure is necessary for the classification task (2) the network does learn this structure during training.

Potentially, the entanglement structure that a network needs to attain before it can start performing the designated task successfully can be a crucial hint to the architecture design and the choice of hyperparameters, as is prominently the case in the analogous context of designing tensor networks to represent specific quantum states. See also our comments section 1.1. In case the knowledge of the necessary level of entanglement is unavailable, a reasonable proxy could be the (estimated) mutual information of the relevant dataset. In particular, lessons from quantum physics indicate that how entanglement varies with the length scale of the partition could be an important feature to consider. See [27] for a recent review.

It would be interesting to explore entanglement structure of other bipartitions, corresponding to shorter-ranged entanglement for instance, apart from the left/right and up/down bipartitions. It would also be interesting to investigate the entanglement structure of a generative network approximating the probability density function (34), as such a network will have a more global knowledge of the Hilbert space $\mathcal{H}$. Finally, it would be interesting to compute the EE of the network trained by physical data, for instance in

the problem of phase classification of physical systems such as the Ising model, and compare that to the EE of generic quantum states in the corresponding physical system.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/vanagia/entangled-q-cnn.

Acknowledgments

We thank Gabriele Cesa, Ying-Jer Kao, Max Welling, and in particular Tolya Dymarsky and Vasily Pestun for helpful conversations. The work of V A is supported by ERC starting Grant H2020 # 640159. The work of M C is supported by ERC starting Grant H2020 # 640159 and NWO vidi Grant (Number 016.Vidi.189.182).

ORCID iD

Miranda C N Cheng https://orcid.org/0000-0002-4651-095X

References

[1] Beach M J S, Golubeva A and Melko R G 2018 Handmachine learning vortices at the Kosterlitz-Thouless transition *Phys. Rev.* B **97** 045207
[2] Bridgeman J C and Chubb C T 2017 Hand-waving and interpretive dance: an introductory course on tensor networks *J. Phys. A: Math. Gen.* **50** 223001
[3] Biamonte J and Bergholm V 2017 Tensor networks in a nutshell (arXiv:1708.00006)
[4] Eisert J 2013 Entanglement and tensor network states (arXiv:1308.3318)
[5] Eisert J, Cramer M and Plenio M B 2010 Coll.: area laws for the entanglement entropy *Rev. Mod. Phys.* **82** 277
[6] Cong I, Choi S and Lukin M D 2019 Quantum convolutional neural networks *Nat. Phys.* **15** 1273–8
[7] Bondesan R and Welling M 2020 Quantum deformed neural networks (arXiv:2010.11189)
[8] Huggins W, Patil P, Mitchell B, Whaley K B and Stoudenmire E M 2019 Towards quantum machine learning with tensor networks *Quantum Sci. Technol.* **4** 024001
[9] Carrasquilla J and Melko R G 2017 Machine learning phases of matter *Nat. Phys.* **13** 431–4
[10] Amin M H, Andriyash E, Rolfe J, Kulchytskyy B and Melko R 2018 Quantum Boltzmann machine *Phys. Rev.* X **8** 021050
[11] Novikov A, Trofimov M and Oseledets I 2016 Exponential machines (arXiv:1605.03795)
[12] Cohen N and Shashua A 2016 Inductive bias of deep convolutional networks through pooling geometry (arXiv:1605.06743)
[13] Cohen N and Shashua A 2016 Convolutional rectifier networks as generalized tensor decompositions *Int. Conf. on Machine Learning* pp 955–63
[14] Liu D, Ran S-J, Wittek P, Peng C, García R Bazquez, Su G and Lewenstein M 2019 Machine learning by unitary tensor network of hierarchical tree structure *New J. Phys.* **21** 073059
[15] Zhaokai Li, Liu X, Nanyang X and Jiangfeng D 2015 Experimental realization of a quantum support vector machine *Phys. Rev. Lett.* **114** 140504
[16] Cichocki A, Lee N, Oseledets I, Phan A-H, Zhao Q and Mandic D P 2016 Tensor networks for dimensionality reduction and large-scale optimization: part 1 low-rank tensor decompositions *Found. Trends Mach. Learn.* **9** 249–429
[17] Cichocki A, Phan A-H, Zhao Q, Lee N, Oseledets I V, Sugiyama M and Mandic D 2017 Tensor networks for dimensionality reduction and large-scale optimizations. Part 2 applications and future perspectives (arXiv:1708.09165)
[18] Han Z-Y, Wang J, Fan H, Wang L and Zhang P 2018 Unsupervised generative modeling using matrix product states *Phys. Rev.* X **8** 031012
[19] Carleo G and Troyer M 2017 Solving the quantum many-body problem with artificial neural networks *Science* **355** 602–6
[20] Chen J, Cheng S, Xie H, Wang L and Xiang T 2018 Equivalence of restricted Boltzmann machines and tensor network states *Phys. Rev.* B **97** 085104
[21] Huang Y and Moore J E 2017 Neural network representation of tensor network and chiral states (arXiv:1701.06246)
[22] Glasser I, Pancotti N, August M, Rodriguez I D and Cirac J I 2018 Neural-network quantum states, string-bond states and chiral topological states *Phys. Rev.* X **8** 011006
[23] Cohen N, Sharir O and Shashua A 2016 On the expressive power of deep learning: a tensor analysis *Conf. on Learning Theory* pp 698–728
[24] Levine Y, Yakira D, Cohen N and Shashua A 2017 Deep learning and quantum entanglement: fundamental connections with implications to network design (arXiv:1704.01552)
[25] Pestun V and Vlassopoulos Y 2017 Tensor network language model (arXiv:1710.10248)
[26] Efthymiou S, Hidary J and Leichenauer S 2019 Tensornetwork for machine learning (arXiv:1906.06329)
[27] Orús R 2019 Tensor networks for complex quantum systems *Nat. Rev. Phys.* **1** 538–50
[28] Sharir O, Tamari R, Cohen N and Shashua A 2016 Tensorial mixture models (arXiv:1610.04167)
[29] Cohen N, Sharir O and Shashua A 2016 Deep simnets *Proc. Conf. on Computer Vision and Pattern Recognition* pp 4782–91
[30] Stoudenmire E and Schwab D J 2016 Supervised learning with tensor networks *Advances in Neural Information Processing Systems* pp 4799–807
[31] Stoudenmire E M 2018 Learning relevant features of data with multi-scale tensor networks *Quantum Sci. Technol.* **3** 034003
[32] Liu Y, Zhang X, Lewenstein M and Ran S-J 2018 Entanglement-guided architectures of machine learning by quantum tensor network (arXiv:1803.09111)
[33] Levine Y, Sharir O, Cohen N and Shashua A 2019 Quantum entanglement in deep learning architectures *Phys. Rev. Lett.* **122** 065301

[34] LeCun Y and Cortes C 2010 MNIST handwritten digit database
[35] Xiao H, Rasul K and Vollgraf R 2017 Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms (arXiv:1708.07747)
[36] Paszke A *et al* 2019 PyTorch: an imperative style, high-performance deep learning library *Advances in Neural Information Processing Systems* pp 8026–37
[37] Loshchilov I and Hutter F 2017 Decoupled weight decay regularization (arXiv:1711.05101)
[38] Roberts C, Milsted A, Ganahl M, Zalcman A, Fontaine B, Zou Y, Hidary J, Vidal G and Leichenauer S 2019 Tensornetwork: a library for physics and machine learning (arXiv:1905.01330)