Contrastive Learning for Lifted Networks

Christopher Zach
zach@chalmers.se
Chalmers University
Gothenburg, Sweden

Virginia Estellers
virginia.estellers@gmail.com
Microsoft Cambridge
Cambridge, UK

Abstract

In this work we address supervised learning via lifted network formulations. Lifted networks are interesting because they allow training on massively parallel hardware and assign energy models to discriminatively trained neural networks. We demonstrate that training methods for lifted networks proposed in the literature have significant limitations, and therefore we propose to use a contrastive loss to train lifted networks. We show that this contrastive training approximates back-propagation in theory and in practice, and that it is superior to the regular training objective for lifted networks.

1 Introduction

Supervised training of deep neural networks (DNNs) relies in almost all implementations on back-propagation (in combination with stochastic gradient descent or an accelerated version thereof) to adjust the network parameters in order to minimize a given loss function. While back-propagation is highly successful in practice, it has some limitations: (i) it can suffer from the vanishing and exploding gradient problem, (ii) its implicit use of fine-grained synchronization also limits its implementation on massively parallel hardware, and (iii) there is evidence that back-propagation is not the basis of learning in biological systems.

Lifted networks have been proposed mainly to utilize massively parallel hardware to train deep neural networks [4, 13, 21, 24, 25]. Lifted networks introduce explicit variables to represent the activations of network units, and these activations are determined implicitly as minimizers of an underlying optimization problem (which we call the *network energy*). In strictly layered DNN architectures the learning problem decouples into layer-wise subproblems over weights once the activations are fixed (and vice versa). This enables massively parallel optimization of the decoupled sub-problems. We are especially interested in the setting when the network energy is strictly convex, since strict convexity of the network energy allows finding the global minimum easily and enables fully parallel updates of the activations based on (block) coordinate descent (e.g. [20]). Further, suitable stricly convex energies are able to approximate the computations in feed-forward networks with arbitrary precision for a range of non-linearities.

Training of lifted networks is conducted by augmenting the network energy with a loss term steering the network output. This objective is minimized with respect to the network weights and the activations (for a whole training set), i.e. the optimal weights lead to the smallest (average) *loss-augmented* network energies over the training set. For training data the inference of network activations has knowledge of the correct output label. At test time the network's output is inferred from the original, non-augmented network energy. Hence, there is a mismatch on how network activations and outputs are determined during the training phase and at test time, which is expected to lead to inferior prediction performance. We will also demonstrate that training lifted networks solely with loss-augmented network energies largely keeps the network in the linear regime and does therefore not leverage the expressive power of non-linear units.

In this work we propose to utilize a *contrastive* training objective to address the shortcomings of standard methods to train lifted networks. Essentially, we reconcile the training and inference phase of lifted networks by applying ideas of contrastive Hebbian learning [17, 23] to the lifted formulation. To this purpose, instead of determining the networks parameters that minimize the loss-augmented energy over the training samples, we find the network parameters that minimize a contrastive objective, which ensures that minima of the network energy agree with minima of the loss-augmented energy used to train the lifted network. Our contrastive loss generalizes to any convex loss function and substantially improves the performance of lifted networks. We will show that training lifted networks with a contrastive approach approximates back-propagation. Hence, we connect energy-based models and back-propagation based discriminative training.

Compared to traditional contrastive Hebbian learning [17, 18, 23], the convex formulation of lifted networks speeds up the computation of the contrastive loss and allows for distributed training of deep neural networks. The connection between back-propagation and energy-based models might offer a better understanding of energy landscapes and generalization ability of different network parameters.

Related Work Energy-based models in machine learning have a long history, and Hopfield nets [10], Boltzmann machines [1] and restricted Boltzmann machines [9, 19] are prominent examples. Contrastive learning methods using non-convex network energies, which relate to variants of Hebbian learning, are proposed in [17, 18, 23]. [23] proves the connection between back-propagation and contrastive loss-based optimization using a particular non-convex network energy.

[4] proposes a quadratic relaxation for the computations in a feedforward network, and thereby introduces "auxiliary coordinates", i.e. network activations as explicit variables, to obtain a highly distributed learning algorithm. Further algorithmic improvements based on this or similar quadratic relaxation were proposed in [7, 13, 21]. It was recognized in [2, 24], that ReLU and other non-linearities can be approximated using quadratic network energies with bounds constraints on the activations. [8, 16] extend this construction to larger classes of non-linearities and obtain block multi-convex network energies. In particular, [16] proposes non-convex energies, where inference of activation by minimization yields exactly the feedforward pass in standard DNNs.

Other methods aiming to replace or to modify back-propagation are difference target propagation [15], proximal back-propagation [6], and approaches based on activation statistics [5] or DC programming [3]. Synthetic gradients aim to avoid the need of synchronization in back-propagation [11].

2 Lifted Networks

This section reviews lifted networks and introduces some of the notation used in subsequent sections. Lifted networks determine the internal network activations implicitly by solving an optimization problem instead of relying on explicitly provided mappings e.g. in feed-forward networks. By solving an optimization problem, lifted networks usually also allow activations in later layer to influence earlier layers.

Notations For a convex set $\mathcal{C}$ we use $\imath_{\mathcal{C}}$ to denote the indicator function, $\imath_{\mathcal{C}}(x) = 0$ iff $x \in \mathcal{C}$ and $\imath_{\mathcal{C}}(x) = \infty$ otherwise. We write $\Pi_{\mathcal{C}}(x)$ for the projection of x into a convex set $\mathcal{C}$.

Lifted networks Lifted networks were initially introduced to enable massively parallel implementations of deep neural networks [4] (with later extensions of [6, 13, 21, 24]). The core idea of using convex energies to determine hidden unit activations is by observing that ReLU non-linearities (and also other ones such as hard-sigmoid and leaky ReLU) can be stated as proximal operators [24]

$$[x]_+ := \max\{0, x\} = \operatorname{prox}_{\imath_{\geq 0}}(x) = \arg\min_{z \geq 0} \|z - x\|^2/2, \tag{1}$$

where $\operatorname{prox}_f(x) := \arg\min_z f(z) + \|z - x\|^2/2$ is called the proximal mapping for a convex function f. Hence, the feed-forward computations in ReLU networks can be approximated by determining the minimizer z^* of the following convex objective,

$$E(z; x) := \tfrac{1}{2}\|z_1 - W_0 x\|^2 + \tfrac{1}{2}\textstyle\sum_{k=1}^{L-1} \gamma^k \|z_{k+1} - W_k z_k\|^2, \tag{2}$$

subject to $z_k \in \mathcal{C}_k \subseteq \mathbb{R}^{n_k}$. x is the input to the network, and W_k are the weights connecting layers k and $k+1$. For notational simplicity we omit the bias terms. The parameter $\gamma > 0$ is the feedback weight, and $\mathcal{C}_k$ are convex sets in $\mathbb{R}^{n_k}$, where n_k is the dimension of z_k. If $\mathcal{C}_k = \mathbb{R}^{n_k}$, then the network has linear units, and if $\mathcal{C}_k = \mathbb{R}^{n_k}_{\geq 0}$, then one obtains ReLU non-linearities. For the output layer L we will always assume $C_L = \mathbb{R}^{n_L}$. We call $E(z; x)$ in Eq. 2 the *free* energy, since the output activations are not influenced by a target label or loss.

First order optimality conditions on the activations z_k^* read as

$$(\mathtt{I} + \gamma W_k^T W_k) z_k^* - W_{k-1} z_{k-1}^* - \gamma W_k^T z_{k+1}^* + \partial \imath_{\mathcal{C}_k}(z_k^*) = 0. \tag{3}$$

For linear units the subgradient $\partial \imath_{\mathcal{C}_k}(z_k^*)$ is the zero vector, and for ReLU units ($\mathcal{C}_k = \mathbb{R}^{n_k}_{\geq 0}$) the subgradient $\partial \imath_{\mathcal{C}_k}(z_k^*)$ is the non-positive orthant. For small feedback weights $\gamma \approx 0$ we have $(\mathtt{I} + \gamma W_k^T W_k) \approx (\mathtt{I} - \gamma W_k^T W_k)^{-1}$ and therefore

$$z_k^* \approx \Pi_{\mathcal{C}_k}\left(W_{k-1} z_{k-1}^* + \gamma W_k^T z_{k+1}^*\right) \overset{\gamma \to 0^+}{\to} \Pi_{\mathcal{C}_k}\left(W_{k-1} z_{k-1}^*\right). \tag{4}$$

Thus, by letting the feedback weight γ approaching 0 one can emulate the standard feedforward computation with a lifted network.

Learning with lifted networks If a training set $\{(x_i, y_i)\}_{i=1}^N$ is given, then learning with lifted networks is performed by jointly minimizing over weights and activations [2, 21, 24],

$$J_0(\mathtt{W}) = \frac{1}{N} \textstyle\sum_i \min_z \left(E(z) + \ell(z_L, y_i)\right), \tag{5}$$

where $\ell(\cdot,\cdot)$ is a task-specific loss function. Thus, the aim is to determine weight matrices $\{W_k\}$ such that the loss-augmented network energy is as small is possible. Due to the min-min structure of the objective, minimization by coordinate or block-coordinate descent is possible. Let us assume $\ell(z_L,y) = \imath\{z_L = y\}$. J_0 is then

$$J_0(\mathtt{W}) = \frac{1}{N}\sum_i \min_{z:z_L=y_i} E(z) = \frac{1}{N}\sum_i \min_z \hat{E}(z), \tag{6}$$

where we introduced the *clamped* energy,

$$\begin{aligned}\hat{E}(z;x,y) &:= E(z;x) + \imath\{z_L = y\} \\ &= \tfrac{1}{2}\|z_1 - W_0x\|^2 + \textstyle\sum_{k=1}^{L-1} \frac{\gamma^k}{2}\|z_{k+1} - W_kz_k\|^2 + \frac{\gamma^{L-1}}{2}\|y - W_{L-1}z_{L-1}\|^2\end{aligned} \tag{7}$$

(subject to $z_k \in \mathcal{C}_k$), in which input and output units are fixed (clamped) to given values. Due to the quadratic term $\|y - W_{L-1}z_{L-1}\|^2$, Eq. 7 corresponds to learning with a quadratic loss. For a given training sample (x,y) and inferred activations $\hat{z}$ (such that $\hat{z} = \arg\min \hat{E}(z;x,y)$) the weights are updated such that pre-activations $W_k\hat{z}_{k-1}$ and post-activations $\hat{z}_k$ match. More precisely, one has (for a single training sample (x,y))

$$\begin{aligned}\nabla_{W_k}\hat{E}(\hat{z};x,y) = 0 &\iff (W_k\hat{z}_k - \hat{z}_{k+1})\hat{z}_k^T = 0 \\ &\iff \forall j,j' : \hat{z}_{kj} = 0 \vee (W_k\hat{z}_k)_{j'} - \hat{z}_{k+1,j'} = 0.\end{aligned} \tag{8}$$

This means that $z_{k,j} = 0$ whenever there exists j' such that $(W_k\hat{z}_k)_{j'} \neq \hat{z}_{k+1,j'}$ (i.e. the network layer behaves non-linearly). The condition $\nabla_{W_k}\hat{E} = 0$ implies that the layer behaves linearly for non-vanishing layer inputs $\hat{z}_k$. Curiously, this is true for non-linearities other than the ReLU.

For a complete training dataset we can deduce that lifted networks trained via Eq. 6 have a strong tendency to yield linear networks even when trained with constrained (and therefore non-linearly behaving) activations. This explains the limited accuracies reported in the literature for such lifted networks, but in a sense this also justifies to use them for pre-training [2, 24], as they are less susceptible to vanishing gradients due to their preference for linear behavior. We validate these claims experimentally in Section 5.

3 Contrastive Learning and Lifted Networks

In this section, for a single training sample (x,y) we consider a contrastive variant of Eq. 6,

$$J_1(\mathtt{W};x,y) = \min_z \hat{E}(z) - \min_z E(z) = \min_{z:z_L=y} E(z) - \min_z E(z), \tag{9}$$

which we term *contrastive loss*. For a complete training set the contrastive loss is the average of individual contrastive losses. We call $\hat{z} := \arg\min_z \hat{E}(z)$ the *clamped solution* and $\check{z} := \arg\min_z E(z)$ the *free solution*. By construction J_1 is always non-negative (since $\hat{E}$ adds the constraint $z_L = y$ to E). J_1 is a min-max ("adversarial") loss,

$$J_1(\mathtt{W}) = \min_{\hat{z}} \hat{E}(\hat{z}) + \max_{\check{z}} -E(\check{z}) = \min_{\hat{z}}\max_{\check{z}} \left\{\hat{E}(\hat{z}) - E(\check{z})\right\}. \tag{10}$$

Hence, optimization of J_1 by alternating minimization of $\mathtt{W}$, $\hat{z}$ and $\check{z}$ is not possible. Using convex duality, one can replace $\min_{\check{z}} E(\check{z})$ with $\max_\lambda E^*(\lambda)$, yielding

$$J_1(\mathtt{W}) = \min_{\hat{z}} \hat{E}(\hat{z}) - \min_{\check{z}} E(\check{z}) = \min_{\hat{z}} \hat{E}(\hat{z}) - \max_\lambda E^*(\lambda) = \min_{\hat{z},\lambda}\{E(\hat{z}) - E^*(\lambda)\}. \tag{11}$$

Recall that E^* is concave and therefore $-E^*$ convex. For the usual choice of $\mathcal{C}_k$ constraint qualification holds and therefore one has strong duality, $\min_{\check{z}} E(\check{z}) = \max_\lambda E^*(\lambda)$. Below we state the duals programs for $\hat{E}$ and E.

Dual programs Via Fenchel or Lagrange duality it can be shown that the dual programs corresponding to the free and clamped energy, respectively, are given by

$$E^*(\lambda) = -\sum_{k=1}^{L-1} \gamma^{k-1} \left(\frac{\|\lambda_k\|^2}{2} + \phi_k^*(\gamma W_k^T \lambda_{k+1} - \lambda_k) \right) - \lambda_1^T W_0 x - \iota\{\lambda_L = 0\} \tag{12}$$

$$\hat{E}^*(\lambda) = -\sum_{k=1}^{L} \frac{\gamma^{k-1}}{2} \|\lambda_k\|^2 - \sum_{k=1}^{L-1} \gamma^{k-1} \phi_k^*(\gamma W_k^T \lambda_{k+1} - \lambda_k) - \lambda_1^T W_0 x + \gamma^{L-1} \lambda_L^T y, \tag{13}$$

where ϕ_k^* is the convex conjugate of $\iota_{\mathcal{C}_k}(\cdot)$. For linear units ($\mathcal{C}_k = \mathbb{R}^{n_k}$) we have $\phi_k^* = \iota_{\{0\}}$ (corresponding to equality constraints), for ReLU units ($\mathcal{C}_k = \mathbb{R}^{n_k}$) we obtain $\phi_k^* = \iota_{\leq 0}$ (i.e. inequality constraints), and for the hard sigmoid non-linearity $\mathcal{C}_k = [0,1]^{n_k}$ we obtain an L^1-like penalizer, $\phi_k^*(\cdot) = \|[\lambda_k - \gamma W_k^T \lambda_{k+1}]_+\|_1$. The important relation used in the following is the connection between optimal primal and dual variables,

$$\lambda_k = z_k - a_k = z_k - W_{k-1} z_{k-1}, \tag{14}$$

which holds for the dual free and clamped energy. $a_k := W_{k-1} z_{k-1}$ is the pre-activation, i.e. the unconstrained signal propagated from layer $k-1$.

A connection between the contrastive loss and learning the posterior By observing that optimization over unknowns corresponds approximately to marginalization (via log-sum-exp), one can restate Eq. 9 as follows,

$$J_1(\mathtt{W}; x, y) = -\lim_{\tau \to 0} \frac{1}{\tau} \log \left(\sum\nolimits_z e^{-\tau E(z;x,y,\mathtt{W})} \right) - \frac{1}{\tau} \log \left(\sum\nolimits_{y',z} e^{-\tau E(z;x,y',\mathtt{W})} \right), \tag{15}$$

where (x,y) is a training sample and $\mathtt{W}$ are the weights made explicit. Let $p(x,y,z;\mathtt{W},\tau) = \exp(-\tau E(z;x,y,\mathtt{W}))/Z(\mathtt{W},\tau)$ be the induced joint probability (at inverse temperature $\tau > 0$), then marginalization yields

$$p(x,y;\mathtt{W},\tau) = \frac{1}{Z(\mathtt{W},\tau)} \sum\nolimits_z e^{-\tau E(x,y,z,\mathtt{W})} \quad \text{and} \quad p(x;\mathtt{W},\tau) = \frac{1}{Z(\mathtt{W},\tau)} \sum\nolimits_{y',z} e^{-\tau E(x,y',z,\mathtt{W})},$$

and J_1 can be restated as

$$J_1(\mathtt{W}; x, y) = -\lim_{\tau \to 0} \frac{1}{\tau} \log \frac{p(x,y;\mathtt{W},\tau)}{p(x;\mathtt{W})} = -\lim_{\tau \to 0} \frac{1}{\tau} \log p(y|x;\mathtt{W},\tau). \tag{16}$$

This means that the contrastive loss (i.e. energy of clamped solution minus energy of free solution) is essentially maximizing the posterior of the (given) output conditioned on the input. It also means that the free energy can be interpreted as the unnormalized negative log-likelihood of the input x. Our initial experiments indicate, that $\min_z E(z;x)$ (with weights obtained by contrastive training) is only of limited use to directly score inputs x, i.e. to assign log-likelihoods. Due to the shape of the free energy Eq. 2 only the first layer activations contribute to $E(z;x)$ in the weak feedback setting. Nevertheless, it opens a possible new perspective of understanding DNNs.

4 Contrastive Learning Approximates Back-propagation

Contrastive learning of weights by minimizing $J_1(\mathtt{W})$ approximates back-propagation for small values of γ. This was shown for a particular non-convex free energy (related to contrastive Hebbian learning [17]) in [23]. [18] establishes establishes a general connection between nested optimization problems and contrastive objectives. In this framework the contrastive cost $J_1(\mathtt{W})$ is interpreted as a finite difference approximation,

$$\tfrac{1}{\beta}J_1(\mathtt{W}) = \tfrac{1}{\beta}\left(\min_z \left\{E(z;x,\mathtt{W}) + \beta\|z_L - y\|^2\right\} - \min_z E(z;x,\mathtt{W})\right) \tag{17}$$

evaluated at $\beta = 0$. In our formulation $\beta = \gamma^{L-1}$. Letting $\beta \to 0$ (i.e. $\gamma \to 0$) it is shown that

$$\lim_{\beta\to 0} \tfrac{1}{\beta}\nabla_{\mathtt{W}} J_1(\mathtt{W}) = \nabla_{\mathtt{W}}\|z_L^* - y\|^2 \qquad \text{s.t. } z^* = \arg\min_z E(z;\mathtt{W}), \tag{18}$$

i.e. the gradient of the contrastive loss approaches the gradient of a nested optimization problem. Since for $\gamma \to 0$, $z^* = \arg\min_z E(z;\mathtt{W})$ converges to the activations obtained by a standard forward pass, one can conclude that $\frac{1}{\gamma^{L-1}}\nabla_{\mathtt{W}} J_1(\mathtt{W}) \overset{\gamma\to 0}{\to} \nabla_{\mathtt{W}}\|z_L^* - y\|^2/2$.[1]

The main shortcoming of the above “algebraic” and indirect derivation is that the structural properties of E are largely ignored, and that for finite (non-infinitesimal) values of γ it is not clear how contrastive learning deviates from back-propagation. Hence, in the following we provide a direct and constructive proof.

By recalling the connection between primal and dual variables ($\lambda_k = z_k - a_k$, see Section 2) we can write the gradient of J_1 w.r.t. W_k concisely as follows:

$$\begin{aligned}\nabla_{W_k} J_1(\mathtt{W}) &= \nabla_{W_k}\left(E(\hat z;\mathtt{W}) - E(\check z;\mathtt{W})\right) = \gamma^k(W_k\hat z_k - \hat z_{k+1})\hat z_k^T - \gamma^{k-1}(W_k\check z_k - \check z_{k+1})\check z_k^T \\ &= -\gamma^k\left(\hat\lambda_{k+1}\hat z_k^T - \check\lambda_{k+1}\check z_k^T\right).\end{aligned} \tag{19}$$

Weight updates for linear networks Let us focus on $C_k = \mathbb{R}^{n_k}$, i.e. network activations z_k are unconstrained. First note that in this setting the free phase has zero cost, $\min_z E(z) = 0$. Hence, $J_1(\mathtt{W})$ reduces to $\min_z \hat E(z;\mathtt{W})$ and

$$\nabla_{W_k} J_1(\mathtt{W}) = -\gamma^k \lambda_{k+1}\hat z_k^T. \tag{20}$$

From $C_k = \mathbb{R}^{n_k}$ we deduce that $\phi_k^* = \imath_0$ and therefore the dual clamped energy reads as

$$\hat E^*(\lambda) = -\textstyle\sum_{k=1}^L \dfrac{\gamma^{k-1}}{2}\|\lambda_k\|^2 - \lambda_1^T W_0 x + \gamma^{L-1}\lambda_L^T y, \tag{21}$$

subject to $\lambda_k = \gamma W_k^T \lambda_{k+1}$. By recursively expanding this constraint we can express the dual variables solely in terms of λ_L,

$$\lambda_k = \gamma^{L-k}\textstyle\prod_{j=k}^{L-1} W_j^T \lambda_L. \tag{22}$$

Since $\lambda_L = \hat z_L - \hat a_L = y - W_{L-1}\hat z_{L-1}$, i.e. the difference between target and predicted output, we introduce $\Delta y := \lambda_L = y - W_{L-1}\hat z_{L-1}$. Consequently, Eq. 20 can be restated as

$$\nabla_{W_k} J_1(\mathtt{W}) = -\gamma^k\gamma^{L-k-1}\textstyle\prod_{j=k+1}^{L-1} W_j^T \Delta y\, \hat z_k^T = \gamma^{L-1}\prod_{j=k+1}^{L-1} W_j^T (W_{L-1}\hat z_{L-1} - y)\hat z_k^T. \tag{23}$$

[1]A further small complication is that E is required to be differentiable in z. This can be achieved by replacing e.g. non-negativity constraints on z by penalizers or barriers such as $-\varepsilon\log(z_k)$ for $\varepsilon \approx 0$.

The error signal arriving at layer k is $\Delta y^{(k)} := \prod_{j=k+1}^{L-1} W_j^T (W_{L-1}\hat{z}_{L-1} - y)$, which is the same as the error signal used in back-propagation. If we denote the forward propagated value by $x^{(k)} := W_{k-1} \cdots W_0 x$, then the only difference (besides the constant scaling γ^{L-1}) to the gradient induced by back-propagation, $\nabla_{W_k}^{\text{back-prop}} = \Delta y^{(k)} (x^{(k)})^T$, is the occurrence of $\hat{z}_k$ instead of $x^{(k)}$. In the weak feedback setting ($\gamma \ll 1$) we have $\hat{z}_k \approx x^{(k)}$. Thus, for finite (non-infinitesimal) values of γ, the difference between back-propagation induced parameter updates and the ones given by contrastive learning lies in the difference of the utilized activations (pure forward vs. inferred). Further, in light of Eq. 22 we also expect the contrastive loss-based gradients for earlier layers to be closer to back-propagation gradients than later layers.

Weight updates for ReLU networks If we add non-negativity constraints on the hiddens, i.e. $C_k = \mathbb{R}^{n_k}_{\geq 0}$, then $\phi_k^* = \imath_{\leq 0}$ and the dual objective is given by Eq. 21, but with different constraints, $\lambda_k \geq \gamma W_k^T \lambda_{k+1}$. These constraints can be stated as $\lambda_k = \gamma W_k^T \lambda_{k+1} + \nu_k$ for $\nu_k \geq 0$. By inserting the connection between primal and dual variables, $\lambda_k = z_k - a_k$, we obtain

$$\begin{aligned} z_k - W_{k-1} z_{k-1} &= \gamma W_k^T (z_{k+1} - W_k z_k) + \nu_k \qquad \text{or} \\ (\mathtt{I} + \gamma W_k^T W_k) z_k - W_{k-1} z_{k-1} &- \gamma W_k^T z_{k+1} - \nu_k = 0, \end{aligned} \tag{24}$$

which can be identified as first order optimality condition for z_k. Hence, $-\nu_k \in \partial \imath_{\geq 0}(z_k)$, and we also have complementary slackness: $z_{kj} > 0$ implies $\nu_{kj} = 0$, and if the constraint $z_{kj} \geq 0$ is active, then $\nu_{kj} > 0$.

We group the activations z_k into strictly positive elements and clamped (i.e. zero) ones. After permuting indices, such that strictly positive activations come first, one can write $z_k = (\tilde{z}_k^T, \mathbf{0}^T)^T$, where $\tilde{z}_k > 0$ corresponds to elements in z_k where the non-negativity constraints are inactive. Hence, the relation between primal and dual variables is

$$\begin{pmatrix} \tilde{\lambda}_k \\ \lambda_k^0 \end{pmatrix} = \begin{pmatrix} \tilde{z}_k \\ \mathbf{0} \end{pmatrix} - \begin{pmatrix} \tilde{a}_k \\ a_k^0 \end{pmatrix}, \tag{25}$$

where a_k^0 and λ_k^0 correspond to clamped component in z_k. Since $\tilde{z}_k > 0$, the corresponding dual variable $\tilde{\nu}_k$ are zero (via complementary slackness), and we obtain $\tilde{\lambda}_k = \gamma \tilde{W}_k^T \lambda_{k+1}$, where we grouped $W_k^T = [\tilde{W}_k^T; (W_k^0)^T]$. We recall Eq. 19,

$$\nabla_{W_k} J_1(\mathtt{W}) = -\gamma^k \left(\hat{\lambda}_{k+1} \hat{z}_k^T - \check{\lambda}_{k+1} \check{z}_k^T \right) \approx -\gamma^k (\hat{\lambda}_{k+1} - \check{\lambda}_{k+1}) \hat{z}_k^T, \tag{26}$$

where we assume weak feedback ($\gamma \ll 1$), and therefore the activations of the free and clamped phase are close, i.e. $\check{z}_k \approx \hat{z}_k$ and $\check{z}_{k+1} \approx \hat{z}_{k+1}$. This assumption also implies that the free and clamped solutions share the set of clamped activations. By recursively expanding the relation $\lambda_{k+1} = \gamma W_{k+1}^T \lambda_{k+2} + \nu_{k+1}$ we therefore obtain

$$\begin{aligned} \hat{\lambda}_{k+1} - \check{\lambda}_{k+1} &= \gamma \begin{pmatrix} \tilde{W}_{k+1}^T (\hat{\lambda}_{k+2} - \check{\lambda}_{k+2}) \\ \mathbf{0} \end{pmatrix} = \gamma \operatorname{diag}(\hat{z}_{k+1} > 0) W_{k+1}^T (\hat{\lambda}_{k+2} - \check{\lambda}_{k+2}) \\ &= \gamma^2 \operatorname{diag}(\hat{z}_{k+1} > 0) W_{k+1}^T \operatorname{diag}(\hat{z}_{k+2} > 0) W_{k+2}^T (\hat{\lambda}_{k+3} - \check{\lambda}_{k+3}) \\ &= \gamma^{L-k-1} \left(\prod_{k'=k+1}^{L-1} \operatorname{diag}(\hat{z}_{k'} > 0) W_{k'}^T \right) \underbrace{(\hat{\lambda}_L - \check{\lambda}_L)}_{=\Delta y}. \end{aligned} \tag{27}$$

This equation is almost exactly the error signal used in back-propagation (the difference being that $\hat{z}_k$ is appearing instead of the purely forward propagated $x^{(k)}$). This implies that the contrastive loss leads (approximately) to the correct cancellation of error signals propagated backwards. Inserting this into Eq. 26 yields

$$\begin{aligned}\nabla_{W_k} J_1(\mathtt{W}) &\approx -\gamma^k(\hat{\lambda}_{k+1} - \check{\lambda}_{k+1})\hat{z}_k^T = -\gamma^k\gamma^{L-k-1}\left(\prod_{k'=k+1}^{L-1} \operatorname{diag}(\hat{z}_{k'} > 0)W_{k'}^T\right)(\hat{\lambda}_L - \check{\lambda}_L)\hat{z}_k^T \\ &= -\gamma^{L-1}\left(\prod_{k'=k+1}^{L-1} \operatorname{diag}(\hat{z}_{k'} > 0)W_{k'}^T\right)(\hat{\lambda}_L - \check{\lambda}_L)\hat{z}_k^T \qquad (28) \\ &\approx -\gamma^{L-1}\left(\prod_{k'=k+1}^{L-1} \operatorname{diag}(\hat{z}_{k'} > 0)W_{k'}^T\right)\Delta y\,(x^{(k)})^T, \qquad (29)\end{aligned}$$

where the last line corresponds to standard back-propagation. We can summarize this section as follows: in contrast to regular losses for lifted networks the use of the contrastive loss yields (approximately) the correct backward signal and weight update. Standard lifted networks do not approximate back-propagation even in the small feedback setting.

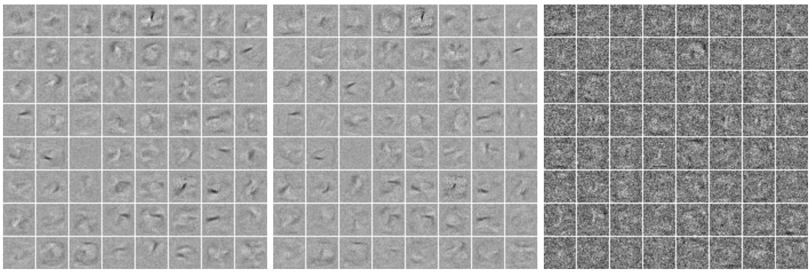
Remark 1. The analogous result can be obtained if non-negativity constraints are replaced by more general element-wise bounds constraints, such as $z_k \in [0,1]^{n_k}$.

5 Numerical Validation

The aim of this section is to numerically verify that (i) standard training for lifted networks has a strong bias towards linear behavior, and (ii) that contrastive training of lifted networks yields results comparable to back-propagation.

Implementation We use a straightforward C++ implementation with multi-threading acceleration. Inference in lifted networks requires solving a convex quadratic program subject to optional bound constraints. We use a coordinate descent method that traverses the layers and updates a single element z_{kj} in each step. Updating z_{kj} can be done in closed form, and each activation z_{kj} is updated 15 times. Since we use weak feedback ($\gamma = 1/8$ in our experiments), we initialize z via a regular forward pass (i.e. $z_{k+1} = \Pi_{\mathcal{C}_{k+1}}(W_k z_k)$ with $z_0 = x$). For this choice of γ the classification accuracies obtained by pure forward passes and by minimization of activations are almost identical.

ReLU networks are obtained by setting $\mathcal{C}_k = \mathbb{R}^{n_k}_{\geq 0}$, and hard sigmoid non-linearities use $\mathcal{C}_k = [0,1]^{n_k}$. We also include results for linear regression to further support our claim that standard, non-contrastively trained lifted networks essentially behave like linear regressors. Although we omitted bias terms in the equations, they are used in our implementation. For training the weight matrices are initialized element-wise with random values from a normal distribution, and biases are initialized to 0. The different training approaches start from the same initial network weights. After the clamped and free activations are determined, the weights are updated using stochastic gradient descent (with mini-batches of size 50). In view of Eqs. 23 and 28 suitable learning rates η^{BP} for back-propagation and $\eta^{\text{Contr.}}$ for contrastive learning are approximately related by $\eta^{\text{Contr.}} = \eta^{\text{BP}}/\gamma^{L-1}$. We use constant learning rates $\eta^{\text{BP}} = 1/20$ for back-propagation and contrastive learning. Standard training of lifted networks is more difficult: in order to at least match the performance of linear regression, the first epochs utilized a smaller learning rate. We use 100 epochs for the MNIST and Fashion-MNIST datasets, and 50 epochs for the CIFAR-10 dataset.



(a) Back-prop (99%/97.5%) (b) Contr. lifted (99%/97.3%) (c) Standard lifted (85%/86%)

Figure 1: First layer filters of a 784-64-64-10 ReLU network trained on MNIST. Training and test accuracies are given in parentheses.

MNIST and Fashion-MNIST In Fig. 1 the first layer weights of a fully connected 784-64-64-10 ReLU network trained on MNIST [14] are visualized. It can be observed that the weights obtained by back-propagation (Fig. 1(a)) and the ones obtained by contrastive training for lifted networks (Fig. 1(b)) are visually close, whereas the weights returned by standard traininig of lifted networks (Fig. 1(c)) are visually different. This is also reflected in the achieved training and test accuracies. Table 1 illustrates results for a 3-layer 784-64-64-64-10 network with either ReLU or hard sigmoid non-linearities. Back-propagation and contrastive learning achieve again similar prediction accuracies (substantially better than regular lifted training). The most interesting in Table 1 aspect is, that contrastive training leads to around 50% active non-linearities, and regularly trained lifted networks are almost entirely in their linear regime. Table 2 and 3 depict the corresponding results for the Fashion-MNIST dataset [22] (using four and five-layer networks, respectively). The results follow the same pattern as the ones for the standard MNIST dataset.

CIFAR-10 Table 4 illustrates the analogous results for a grayscale version of the CIFAR-10 dataset [12]. We observe qualitatively similar results (although at substantially lower accuracy levels compared to MNIST and Fashion-MNIST) for this dataset. No augmentation (such as horizontal flipping of input images) is employed.

6 Conclusion

The aim of this work is to draw the attention to lifted networks, which—as easy-to-train energy-based models—are attractive for understanding DNNs. Lifted networks have seen somewhat limited use in the literature, and we hypothesize that this is due to the standard training procedures for lifted networks essentially lead to linear network behavior. We demonstrate that using a contrastive training objective leads to more competitive lifted networks, which also better leverage the expressive power of the network non-linearities.

In future work we intend to leverage the network energy to estimate the likelihood of input data, e.g. for anomaly detection. We also plan to investigate early stopping criteria for activation inference, which will allow faster training procedures for lifted networks.

	accuracy (%)		linear activations (%)		
models	train	test	layer 1	layer 2	layer 3
ReLU back-prop	99.8	97.7	43.8	39.3	38.2
ReLU lifted	85.2	86.3	99.9	99.9	99.9
ReLU contr.	99.8	97.6	37.9	43.6	48.4
Hard sigm. back-prop	99.7	97.0	53.5	45.1	34.6
Hard sigm. lifted	85.4	86.3	99.9	99.9	99.9
Hard sigm. contr.	99.8	97.4	47.0	43.7	50.7
Linear regression	85.3	86.0			

Table 1: Training and test accuracies and fraction of activations in the linear range for 784-64-64-64-10 networks trained on MNIST.

	accuracy (%)		linear activations (%)		
models	train	test	layer 1	layer 2	layer 3
ReLU back-prop	95.6	88.0	38.0	39.6	38.4
ReLU lifted	81.4	80.0	99.9	99.9	99.9
ReLU contr.	94.2	88.2	30.0	48.2	66.1
Hard sigm. back-prop	95.5	88.1	46.7	40.5	41.7
Hard sigm. lifted	81.6	80.1	99.9	99.9	99.9
Hard sigm. contr.	94.7	88.2	44.9	35.6	66.3
Linear regression	82.3	80.4			

Table 2: Training and test accuracies and fraction of activations in the linear range for 784-64-64-64-10 networks trained on Fashion-MNIST.

	accuracy (%)		linear activations (%)			
models	train	test	layer 1	layer 2	layer 3	layer 4
ReLU back-prop	96.2	88.4	33.5	42.0	44.3	40.3
ReLU lifted	72.7	72.3	99.9	99.9	99.9	99.9
ReLU contr.	97.0	88.6	36.1	44.4	45.5	67.9

Table 3: Training and test accuracies and fraction of activations in the linear range for 784-128-64-64-32-10 networks trained on Fashion-MNIST.

	accuracy (%)		linear activations (%)	
models	train	test	layer 1	layer 2
ReLU back-prop	71.1	48.2	28.5	28.9
ReLU lifted	28.0	27.7	99.9	99.9
ReLU contr.	61.7	43.1	15.2	90.8
Linear regression	24.2	21.8		

Table 4: Training and test accuracies and fraction of activations in the linear range for 1024-512-512-10 ReLU network trained on (grayscale) CIFAR-10.

References

[1] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

[2] Armin Askari, Geoffrey Negiar, Rajiv Sambharya, and Laurent El Ghaoui. Lifted neural networks. *arXiv preprint arXiv:1805.01532*, 2018.

[3] Leonard Berrada, Andrew Zisserman, and M Pawan Kumar. Trusting svm for piecewise linear cnns. *arXiv preprint arXiv:1611.02185*, 2016.

[4] Miguel Carreira-Perpinan and Weiran Wang. Distributed optimization of deeply nested systems. In *Artificial Intelligence and Statistics*, pages 10–19, 2014.

[5] Anna Choromanska, Sadhana Kumaravel, Ronny Luss, Irina Rish, Brian Kingsbury, Ravi Tejwani, and Djallel Bouneffouf. Beyond backprop: Alternating minimization with co-activation memory. *arXiv preprint arXiv:1806.09077*, 2018.

[6] Thomas Frerix, Thomas Möllenhoff, Michael Moeller, and Daniel Cremers. Proximal backpropagation. *arXiv preprint arXiv:1706.04638*, 2017.

[7] Akhilesh Gotmare, Valentin Thomas, Johanni Brea, and Martin Jaggi. Decoupling backpropagation using constrained optimization methods. 2018.

[8] Fangda Gu, Armin Askari, and Laurent El Ghaoui. Fenchel lifted networks: A lagrange relaxation of neural network training. *arXiv preprint arXiv:1811.08039*, 2018.

[9] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[10] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[11] Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1627–1635. JMLR. org, 2017.

[12] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[13] Tim Tsz-Kit Lau, Jinshan Zeng, Baoyuan Wu, and Yuan Yao. A proximal block coordinate descent algorithm for deep neural network training. *arXiv preprint arXiv:1803.09082*, 2018.

[14] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[15] Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 498–515. Springer, 2015.

[16] Jia Li, Cong Fang, and Zhouchen Lin. Lifted proximal operator machines. *arXiv preprint arXiv:1811.01501*, 2018.

[17] Javier R Movellan. Contrastive hebbian learning in the continuous hopfield model. In *Connectionist Models*, pages 10–17. Elsevier, 1991.

[18] Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.

[19] Paul Smolensky. *Information processing in dynamical systems: Foundations of harmony theory*, volume 1, chapter 6, pages 194–281. MIT Press, Cambridge, 1986.

[20] David Sontag and Tommi Jaakkola. Tree block coordinate descent for MAP in graphical models. *Journal of Machine Learning Research*, 2009.

[21] Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, and Tom Goldstein. Training neural networks without gradients: A scalable admm approach. In *International Conference on Machine Learning*, pages 2722–2731, 2016.

[22] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[23] Xiaohui Xie and H Sebastian Seung. Equivalence of backpropagation and contrastive hebbian learning in a layered network. *Neural computation*, 15(2):441–454, 2003.

[24] Ziming Zhang and Matthew Brand. Convergent block coordinate descent for training tikhonov regularized deep neural networks. In *Advances in Neural Information Processing Systems*, pages 1721–1730, 2017.

[25] Ziming Zhang, Yuting Chen, and Venkatesh Saligrama. Efficient training of very deep neural networks for supervised hashing. *CVPR*, 2016.