Exploring the Application of Convex Optimization Theory in Deep Learning

Areeb Asad Syed, Jiaxi Lei, Natapong Stephen Jarrell

1 Introduction

In this project, we do a survey study to explore state of the art approaches of applying convex optimization to deep learning problems. We look at various previous studies, and analyze their methods of trying to make convex optimization deep learning available.

Convex optimization is the process of minimizing convex functions over convex sets. As shown in [11], a convex optimization problem takes the form:

$$\text{minimize } f_0(x)$$
$$\text{subject to } f_i(x) \leq b_i, i = 1, \ldots, m$$
$$\text{where the functions } f_0, \ldots, f_m : R^n \rightarrow R \text{ are convex,}$$
$$\text{i.e. satisfy } f_i(\alpha x = \beta y) \leq \alpha f_i(x) + \beta f_i(y)$$
$$\text{for all } x, y \in R^n \text{ and all } \alpha \text{ , } \beta \in R \text{ with } \alpha + \beta = 1, \alpha \geq 0, \beta \geq 0$$

Neural networks are not convex. However, in some cases, they can be converted to convex optimization problems. This is often useful as convex optimization can provide more efficient solutions. A convex optimization problem does not fluctuate based on hyper-parameters. A non-convex problem might get a local minimum if the initial state is not ideal, but a convex optimization problem will guarantee finding a global minimum. When training with gradient methods, convex problems can start with a higher learning rate, compared to it's non-convex counterpart, without worrying about divergence. Thus, using convex optimization helps us train neural networks faster and use less computational power. In this project we survey three different approaches of applying convex optimization to neural networks:

1. Incorporating differentiable convex optimization problems as layers within the network.

2. Using convex dual theory to provide a theoretical basis for the empirical success of Batch Normalization during Neural Network training

3. Transforming a non-convex problem such as a Two Layer ReLU network into a convex problem, by representing the neural network as a convex, feasible space.

2 Incorporating differentiable convex optimization problems as layers within the network.

2.1 Approach

This approach is from [6]. The main idea is to incorporate differentiable convex optimization problems as layers within a neural network. The purpose of this is to provide a useful inductive bias for some problems. The most major contribution of this approach is addressing the following issue: the existing systems of embedding convex optimization layers are difficult to use and apply to different settings. Existing systems like [12] are not flexible to new settings, and require manual, laborious, and cause many errors during transformations into canonical form.

Here is where domain specific languages (DSLs) become really useful. DSLs made for convex optimization are able to handle changes to canonical forms by themselves, and so users are able to skip the manual process of such transformations. Thus, DSLs are able to provide an easy format for users to specify their problems, even if they are not convex optimization experts. This research applies convex optimization DSLs to make convex optimization layers in neural networks. By using DSLs, the proposed program is able to parse through disciplined convex programs (DCPs). DCPs are a type of convex optimization problems that can be solved by the convex optimization DSLs they use. In short, DCP is just a grammar to define convex optimization problems. More information and expansion on this topic can be found at [15]. The canonicalized cone program version of the problem is then solved. Thus, for each layer that is converted into a convex optimization problem, we can represent it as a convex cone program (As shown in Problem Statement).

The novelty aspect of this approach comes in at this stage. A new grammar is introduced known as Disciplined Parametrized Programming (DPP). DPPs can be thought of as parameterized DCPs, meaning that it is a DCP that limits parameter usage, or in other words, a DCP that can perform parameter-dependent curvature analysis. So, let's say we have constructed a convex optimization problem using DPP. We can get its canonicalized form cone program using canonicalization [16]. According to [16], DCPs can be canonicalized to cone programs by expanding each nonlinear into it's graph implementation, which are affine expressions. [6] expands on how the canonicalization process works for DPPs, and introduces affine-solver-affine (ASA) form. This approach can obtain affine maps, from parameters to problem data, and from cone program solution to original problem solution. This structure of affine maps and a solver is known as affine-solver-affine (ASA). The solver is a conic solver algorithm that solves the optimization problem, and called during the forward pass, for differentiating through the DPP. According to [6], A conic solver is used to map the problem data (A, b, c) to a solution x^*, by providing a function $s : R^{m\times n} \times R^m \times R^n \to R^n$. More details on differentiating through cone programs can be found at [18]. The ASA Form allows users to differentiate through the DPP without requiring any kind of backpropagation to the canonicalizer.

2.2 Problem Statement

For each layer that is converted into a convex optimization problem, we can represent it as a convex cone program [6]:

Primal Formulation of cone program:

$$\begin{gathered}\text{minimize } c^T x \\ \text{subject to } Ax + s = b\,,\, s \in K\end{gathered}$$

Where the set $K \subseteq R^m$ is a convex cone, $x \in R^n$ is the primal variable, $s \in R^m$ is the slack variable , and the problem data is : $A \in R^{m\times n}$, $b \in R^m$.

Dual Formulation of the cone program:

$$\begin{gathered}\text{minimize } b^T y \\ \text{subject to } A^T y + c = 0,\, y \in K^*\end{gathered}$$

Where K^* is the dual cone of K, and $y \in R^m$ is the dual variable.

KKT Conditions:

The solution to this cone program is (x, y, s), if the following KKT conditions are satisfied:

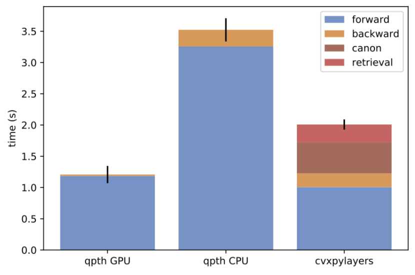
$$Ax + s = b, A^T y + c = 0, s \in K, y \in K*, s^T y = 0$$

To find the derivative of the conic solver, the implicit function theorem [17] is used on the optimality conditions of the cone program. In the case that the cone program cannot be differentiated, at non differentiable points, if the linear system of derivative is not invertible, a heuristic (least-squares solution) is calculated instead.

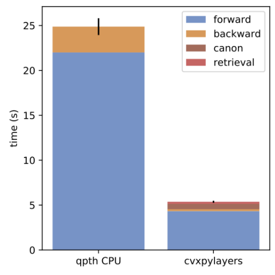
2.3 Results and Conclusion

[6] is able to provide an implementation of DPP and ASA form in CVXPY 1.1. CVXPY is a convex optimization DSL based in Python [13]. Moreover, [6] provides code examples of how to define DPP problems in CVXPY, and also an implementation of using DPP and ASA in Pytorch layers.

As shown in [6], when comparing performance between Pytorch CvxpyLayer (CVXPY 1.1) to qpth (CVXPY 1.0.23), we can see that CvxpyLayer is able to finish the training process much faster (Figure 1), compared to qpth on CPU for dense QPs, and also for sparse QPs.



(a) Dense QP, batch size of 128.



(b) Sparse QP, batch size of 32.

Figure 1: Performance time: CvxpyLayer vs qpth [6]

Thus, using CVXPY and Pytorch, this can be applied to many neural networks, and help users train neural networks faster and use less computational power.

2.4 Applications

Here are some researches/projects where these systems have been used to apply convex optimization to neural network layers:

1. In [19], the researchers propose a graph neural networks (GNNs) approach to solving large-scale radio resource management problems. They formulate radio resource management problems as graph optimization problems and apply a type of GNNs known as message passing graph neural networks (MPGNNs) for this task. During the specific implementation of an MPGNN known as wireless channel graph convolution network (WCGCN), they calculate a differentiable normalization function known as β. For some general constraints, β can be calculated as differentiable projections at each layer, and this research suggests to apply the methods and implementations of DPP for these calculations.

2. In [20], the researchers propose new approaches to use machine learning to solve online mixed-integer optimization (MIO) problems at faster speeds. For this, they build an extension to a Machine Learning Optimizer used previously in another project. Here, they use the DPP language implementation in CVXPY 1.1 for parameteric MIQO, to fasten it's multiple iterations of canonicalizations.

2.5 Future Works

1. One possible future direction from [6] is to extend DPP and add more types of convex programs to be used, as CVXPY allows users to add custom/ specialized solvers. One potential solver to add is QP solvers.

2. Another possible future work could be implementing/developing a deep learning program for specific networks, that are built on CVXPY DPP and ASA, and abstract away the process of creating deep learning pipelines that use DPP.

3 Using convex dual theory to provide a theoretical basis for the empirical success of Batch Normalization during Neural Network training

3.1 Approach

State-of-the-art neural networks are trained using optimizers with some variant of Stochastic Gradient Descent (SGD). However, due to the large number of epochs and large datasets required to optimize heavily-parameterized networks to complex tasks such as visual perception or natural language understanding, data is separated into batches. This accelerates training convergence, but sampling the gradient updates from batches during optimization is a trade-off of gradient stability (instability can lead to divergence) for performance. BN alleviates this by normalizing layer outputs for a given batch as an operator between network layers, which is shown to mimic the whitening effect that reshapes the loss landscape, ultimately helping network optimization [5].

Previously, there was no theoretical basis for the empirical success of BN in neural network optimization. Recent work reveals that we can use convex optimization to show the implicit whitening affect of BN, which is a data transformation that improves the reliability of convergence by reshaping the geometry of the loss landscape. We can investigate an example of BN and how it affects a downstream activated layer and prove that BN, when applied to regularized two-layer network training problems, is a convex finite-dimensional problem with the data matrix whitened. First, let's look at BN broadly.

The convexity of Batch Normalization can be shown to hold for arbitrary convex loss functions, including current state-of-the-art loss functions such as hinge loss and cross entropy loss. The parameters of BN are represented by

$$\theta := \{W^{(l)}, \gamma_l, \alpha_l\}_{l=1}^{L}$$

and the parameter space

$$\Theta := \{W^{(l)}, \gamma^{(l}), \alpha^{(l)}\}_{l=1}^{L} : W^{(L)} \in \mathbb{R}^{m_{l-1} x m_l}, \gamma_l \in \mathbb{R}^{m_l}, \alpha_l \in \mathbb{R}^{m_l}$$

With this simpliciation, we can give a primal form of the basic optimization task with L1 regularization for an arbitrary loss function.

$$\min_{\theta \in \Theta_s} L(f_{\theta,L}(X), y) + \beta ||w||_1^{(L)}$$

The subscript L is the number of layers, L the loss function, f the model with its parameters θ, y the target variable, β the regularization weight, and w the weight vector.

The dual formulation can be expressed using the Fenchel conjugate function[21]:

$$L^*(v) = \max_z z^T v - L(z, y)$$

such that the dual form is:

$$\max_v -L^*(v)$$
$$\max_{\theta \in \Theta} |v^T BN_{\gamma,\alpha}(A^{(L-2)} w^{(1)})| \leq \beta$$

3.2 Primal Form

Now that we've elaborated the general case, we can extend the derivations to two-layer networks with MSE loss and L1 regularization. The problem can be stated as:

$$p_2^* = \min_{\theta} ||f_\theta(X) - y||_2^2 + \beta||w||_1$$

where $\Theta_s := \theta \in \Theta : \gamma_j^{(L-1)^2} + \alpha_j^{(L-1)^2} = 1$

Both γ and α are the hyperparameters of BN that are set before training.

3.3 Dual Form

Using this equivalence we can derivative the dual w.r.t the two output layers [7].

$$p_2^* \geq d_2^* = \max_v -\frac{1}{2}||v - y||_2^2 + \frac{1}{2}||y||_2^2, \text{ s.t. } \max_{\theta \in \Theta} v^T(BN_{\gamma,\alpha}(Xw^{(1)}))_+| \leq \beta$$

where $\Theta_s := \theta \in \Theta : w^{(1)} \in \mathbb{R}, \gamma_j^{(1)^2} + \alpha_j^{(1)^2} = 1$

3.4 Closed Form Solution

The convexity of Batch Normalization in a two-layer network is proven as a combination of linear models, resulting in the optimal solution as sparse piece-wise linear functions. Thus, a non-convex training problem shown in the primal form can be stated as a finite-dimensional convex program. We can do this by defining a diagonal matrix $D := diag(\mathbb{1}[Xw \geq 0])$ and represent the ReLU activation as $DXw \geq 0$ and $(I_n - D)Xw \leq 1$. These two constraints can be condensed to $(2D - I_n)Xw \geq 0$. The cardinality of the set of possible diagonal matrices is dependent upon the rank of X. So for the set of all possible convex combinations P and data U:

$\min_{s_i, s_i' i=1} \frac{1}{2}||\Sigma_{i=1}^P D_i U'(s_i - s_i') - y||_2^2 + \beta\Sigma_{i=1}^P(||s_i||_2 + ||s_i'||_2)$ s.t.

$$\begin{cases} (2D_i - I_n)U's_i \geq 0 \\ (2D_i - I_n)U's_i' \geq 0 \end{cases}$$

where r is the rank of the data matrix. $U \in \mathbb{R}^{nxr}$ and $U' \in \mathbb{R}^{nxr+1}$ are computed using compact singular value decomposition: $(I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^T)X := U\Sigma V^T$ and $U' := [U\frac{1}{\sqrt{n}}\mathbf{1}]$

3.5 Results and Future Work

Through the finite-dimensional convex program, we can arrive at values of the transformed dataset, U, where the covariance matrices result in the identity matrix, which is the definition of the whitening effect: $U^T U' = I_{r+1}$ [14]. By doing so, we prove that Batch Normalization causes an implicit whitening effect on the data, explaining its success as an operation between layers during neural network optimization. This analysis provides the theoretical basis for incorporating Batch Normalization in SOTA and future model architectures that rely on over-parameterization and ReLU activations. Future work will hopefully include utilizing convex optimization to provide closed-form solutions for other hyperparameters and architecture designs, such as explaining the empirical success of cosine annealing learning rate for model convergence, and the basis for the unexplained yet empirical success of these methodologies in neural network optimization.

4 Non-Convex Two Layer ReLU Network as Convex Problem in Finite Higher Dimension

4.1 Introduction

Previous work has limited convexity of two-layer structure to infinite width.[9] More recent research has proposed theory on almost-convex like characteristics exhibited by over-parameterized two-layer networks. [10] This section is mainly derived from Mert Pilanci and Tolga Ergen's work [1]. In short, a finite dimensional two-layer network can be transformed into an identical convex problem with finite higher dimensional space, where the optimal solution to this convex problem can be used to construct the solution (set) to the non-convex problem.

4.2 Method

Let a two-layered neural network (one hidden layer) with ReLU activation and L2 regularization be defined as:

$$p_{\text{non-convex}} = \text{minimize } L(\phi(XW_1)W_2, y) + \lambda(||W_1||_F^2 + ||W_2||_F^2)$$

Where $W_1 \in R^{d\times m}, W_2 \in R^{m\times 1}$ are the weights being optimized, L is the loss function, ϕ is ReLU activation.

This non-convex problem can be transformed into a convex problem in the general form of: [1] [2]

$$p_{\text{convex}} = \text{minimize } L(Z, y) + \lambda R(Z)$$

Where L is the same loss function used in the non-convex scenario. $Z \in K \subset R^{d\times p}$, R(Z) is some convex regularization. Mert Pilanci and Tolga Ergen in their paper propose that the convex problem and non-convex problem are identical, and the optimal solution to the convex problem can be used to obtain the optimal solution to the non-convex problem.

4.2.1 Primal Form

The paper gives an in-depth proof to the above theorem. The example in the proof considers using squared loss:

$$p_{\text{non-convex}} = \min_{\{W_{1j},W_{2j}\}_1^j} \frac{1}{2}||\sum_{j=1}^{m} \phi(XW_{1j})W_{2j} - y||_2^2 + \frac{\lambda}{2}\sum_{j=1}^{m}(||W_{1j}||_2^2 + W_{2j}^2)$$

This is proven to be **equivalent** to the L1 penalized neural network training cost[4]:

$$p_{\text{non-convex}} = \min_{||W_{1j}||_2 \le 1} \min_{\{W_{2j}\}_1^j} \frac{1}{2}||\sum_{j=1}^{m} \phi(XW_{1j})W_{2j} - y||_2^2 + \lambda\sum_{j=1}^{m}|W_{2j}|$$

4.2.2 Dual Form

The **dual** form of this network is proven in the appendix of the paper as:

$$p^* = \min_{||W_{1j}||_2\le 1} \max_{v\in R \text{ s.t.} |v^T\phi(XW_{1j})|\le\lambda} -||y-v||_2^2 + ||y||_2^2$$

4.2.3 Convex Program

The convex problem can be formulated as:

$$p_{\text{convex}} = \min_{u_1,\dots u_p, v_1,\dots,v_p \in K} \frac{1}{2}||\sum_{i=1}^{p} D_iX(u_i - v_i) - y||_2^2 + \frac{\lambda}{2}(\sum_{i=1}^{p}||u_i||_2 + ||v_i||_2)$$

Note the transformed regularization term takes the form of a block L1 Norm.
The closed formed formula of the optimal solution to the non-convex problem can be constructed as:

$$W_{1i}^* = \frac{u_i^*}{\sqrt{||u_i^*||_2}}, W_{2i} = \sqrt{||u_i^*||_2}$$

Or

$$W_{1i}^* = \frac{v_i^*}{\sqrt{||v_i^*||_2}}, W_{2i} = \sqrt{||v_i^*||_2}$$

4.3 Results and Possible Future Work

The authors of [1] provided promising result on the training time of such two-layer network in variety of structures, showing guaranteed optimality in polynomial time. To extend the result of [1] [2], recent research has proposed a deep neural networking training technique called layer-wised training that trains 2-layer network progressively to build up a deep network.[3] For each progression, layer-wised training keeps the hidden layer of the optimized 2-layer network and uses its output as input to the next 2-layer network. In the same research, it is shown that layer-wised training of deep neural network can perform on-par or even outperforms classical training approach by achieving 90.4% accuracy on CIFAR-10 dataset and 88.7% accuracy on ImageNet dataset. Using well-defined convex optimization solvers, it is promising that we can train even deep neural networks faster.

5 Conclusion and Discussion

In this project, we have surveyed three different approaches of how convex optimization concepts are applied to deep learning neural networks to help improve their efficiency, optimality and to reduce required computing power. In the first case, they use differentiable convex optimization layers as layers in neural networks to provide a useful inductive bias for some problems. Existing systems that apply layers of convex optimization problems are slow and find it difficult to adapt to new settings, and this new approach abstracts away this process by implementing DPP in CVXPY, reducing error and labor. In the second case, convex optimization can be used to provide a theoretical basis for the success of Batch Normalization, which has led to large scale success of overparametrized neural networks that are leading the state-of-the-art, by revealing how Batch Normalization implicitly transforms the data to have unique covariance properties. In the third case the authors present an approach of transforming a finite dimensional two-layer network with ReLU activation into an identical convex problem with finite higher dimensional space. This approach opens up the possibility for users to use a variety of well-defined convex optimization approaches to guide neural network training into finding the optimal solution. In all cases, convex optimization provides a framework for understanding and utilizing deep learning beyond empirical results, which will hopefully fuel further research into created better convex and non-convex optimization approaches.

References

[1] Pilanci, Mert, and Tolga Ergen. ”Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks.” International Conference on Machine Learning. PMLR, 2020.

[2] Wang, Yifei, Jonathan Lacotte, and Mert Pilanci. ”The Hidden Convex Optimization Landscape of Two-Layer ReLU Neural Networks: an Exact Characterization of the Optimal Solutions.” arXiv preprint arXiv:2006.05900 (2020).

[3] Belilovsky, Eugene, Michael Eickenberg, and Edouard Oyallon. ”Greedy layerwise learning can scale to imagenet.” International conference on machine learning. PMLR, 2019.

[4] Neyshabur, B., Tomioka, R., and Srebro, N. In search of the real inductive bias: On the role of implicit regularization in deep learning. arXiv preprint arXiv:1412.6614, 2014.

[5] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. ”How does batch normalization help optimization?” (2018)

[6] Agrawal, Akshay, et al. ”Differentiable convex optimization layers.” Advances in neural information processing systems 32 (2019).

[7] Ergen, Sahiner, et al. ”Demystifying Batch Normalization in ReLU Networks: Equivalent Convex Optimization Models and Implicit Regularization.” arXiv:2103.01499 (2021).

[8] Wei, Stokes, Scwab. ”Mean-Field Analysis of Batch Normalization” arXiv:1903.02606 (2019).

[9] Bengio, Yoshua, et al. ”Convex neural networks.” Advances in neural information processing systems 18 (2005).

[10] Allen-Zhu, Zeyuan, Yuanzhi Li, and Zhao Song. ”A convergence theory for deep learning via over-parameterization.” International Conference on Machine Learning. PMLR, 2019.

[11] Boyd, Stephen, Stephen P. Boyd, and Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004. pp 7.

[12] Amos, Brandon, and J. Zico Kolter. ”Optnet: Differentiable optimization as a layer in neural networks.” International Conference on Machine Learning. PMLR, 2017.

[13] Diamond, Steven, and Stephen Boyd. ”CVXPY: A Python-embedded modeling language for convex optimization.” The Journal of Machine Learning Research 17.1 (2016): 2909-2913.

[14] Tolga Ergen and Mert Pilanci. ”Revealing the structure of deep neural networks via convex duality.” (2020)

[15] Grant, M., S. Boyd, and Y. Ye. ”Disciplined Convex Programming Global Optimization: From Theory to Implementation ed L Liberti and N Maculan.” (2006): 155-210.

[16] Grant, Michael C., and Stephen P. Boyd. ”Graph implementations for nonsmooth convex programs.” Recent advances in learning and control. Springer, London, 2008. 95-110.

[17] Dontchev, Asen L., and R. Tyrrell Rockafellar. Implicit functions and solution mappings. Vol. 543. Berlin: Springer, 2009.

[18] Agrawal, Akshay, et al. ”Differentiating through a cone program.” arXiv preprint arXiv:1904.09043 (2019).

[19] Shen, Yifei, et al. ”Graph neural networks for scalable radio resource management: Architecture design and theoretical analysis.” IEEE Journal on Selected Areas in Communications 39.1 (2020): 101-115.

[20] Bertsimas, Dimitris, and Bartolomeo Stellato. ”Online mixed-integer optimization in milliseconds.” arXiv preprint arXiv:1907.02206 (2019).

[21] Boyd and Vandenberghe. ”Convex optimization”. Cambridge university press, 2004