Flexible Learning of Sparse Neural Networks via Constrained L_0 Regularization

Jose Gallego-Posada
Mila, Université de Montréal*

Juan Ramirez
Universidad EAFIT

Akram Erraqabi
Mila, Université de Montréal

Abstract

We propose to approach the problem of learning L_0-sparse networks using a constrained formulation of the optimization problem. This is in contrast to commonly used penalized approaches, which combine the regularization terms additively with the (surrogate) empirical risk. Our experiments demonstrate that we can obtain approximate solutions to the constrained optimization problem with comparable performance to state-of-the art methods for L_0-sparse training. Finally, we discuss how this constrained approach provides greater (hyper-)parameter interpretability and accountability from a practitioner's point of view.

1 Introduction

The great expressive power of neural networks as function approximators (Bengio et al., 2013) comes with an inherent need for regularization. Regularization aimed at controlling the generalization behavior of the network can be realized by a wide range of mechanisms: explicitly, through the use of additive penalties in training (e.g. Structural Risk Minimization (Vapnik and Chervonenkis, 1974)); implicitly, via the choice of optimization algorithm used to train the network (Neyshabur et al., 2015; Caruana et al., 2001); or even be in-grained in the architecture of the network through stochastic computation (Srivastava et al., 2014).

The internal structure of commonly used neural networks results in *overparametrized* models, whose performance is robust to harsh levels of parameter pruning (Han et al., 2016; Ullrich et al., 2017; Frankle and Carbin, 2019). Thus, regularization techniques aimed at learning sparse models can drastically reduce the computational cost associated with the learnt model by removing unnecessary parameters, and retain good performance in the learning task. With recent research trends exploring the capabilities of ever more ambitious large-scale models (Brown et al., 2020), developing techniques which provide reliable training of *sparsified* models becomes crucial for deploying them in massively-used systems, or on resource-constrained devices.

Louizos et al. (2018) develop a framework for training classifiers with low L_0-norms. The authors propose to augment the network's classification loss with an additive surrogate penalty (scaled by a suitably selected multiplicative factor λ). At convergence, the model is expected to strike a balance (dependent on λ) between achieving a good performance in the classification task, and having a low number of active parameters, as measured by its L_0-norm. We retain their proposed reparametrization based on modified binary concrete random variables (Maddison et al., 2017), which allows for a gradient based optimization procedure, while maintaining *exact* zeros in the parameter values.

In this paper we advocate for the use of a constrained formulation of the learning problem and resort to standard saddle-point optimization techniques for optimizing the Lagrangian associated with the constrained problem. Concretely, we consider constraints of the type $\|\boldsymbol{\theta}_g\|_0 \leq \epsilon$, where $\boldsymbol{\theta}_g$ represents a group g of parameters of the network. In contrast to the penalized formulation of Louizos et al. (2018), we do not seek to reduce the L_0-norm of the model after the given threshold is respected.

*Correspondence: `gallegoj(at)mila(dot)quebec`

Published as a workshop paper at the NeurIPS 2021 - LatinX in AI Workshop

Adopting this constrained formulation provides several advantages. It enjoys an interpretable hyper-parameter: the constraint level. Unlike the multiplicative factor λ, the constraint level ϵ has straightforward semantics associated with the *density* of a block of parameters $\boldsymbol{\theta}_g$. Moreover, requiring different density levels for parameter groups with larger contributions to the overall computation time simply amounts to specifying several constraints with levels matching these desired densities, thus eliminating the successive tuning and re-balancing of various multiplicative factors.

We concentrate on the setting of training sparsified networks as a case study to illustrate the advantages of the constrained approach. However, the benefits of constrained formulations extend to many other applications beyond the training of sparsified networks.

2 Regularization via L_0 penalties

Louizos et al. (2018) propose a framework for training sparse models using the L_0-norm as an additive penalty to the usual training objective. The L_0-norm penalizes the number of non-zero entries of the parameter vector $\boldsymbol{\theta}$, inducing sparsity without attempting to shrink the parameter values.

Let $h(x;\ \boldsymbol{\theta})$ be a predictor with parameters $\boldsymbol{\theta}$. Given a supervised learning problem defined by a dataset of N i.i.d. pairs $\mathcal{D} = \{(x_i,\ y_i)\}_{i=1}^N$, a loss function ℓ and a regularization coefficient $\lambda \geq 0$, they formulate the following L_0-regularized empirical risk objective:

$$\mathcal{R}(\boldsymbol{\theta}) = \frac{1}{N}\left(\sum_{i=1}^{N} \ell\left(h(x_i;\ \boldsymbol{\theta}),\ y_i\right)\right) + \lambda\|\boldsymbol{\theta}\|_0, \qquad \|\boldsymbol{\theta}\|_0 = \sum_{j=1}^{|\boldsymbol{\theta}|} \mathbb{1}\{\theta_j \neq 0\}. \tag{1}$$

The non-differentiability of the L_0-norm makes it poorly suited to gradient-based optimization. The authors propose a reparametrization $\boldsymbol{\theta} = \tilde{\boldsymbol{\theta}} \odot \mathbf{z}$, where $\tilde{\boldsymbol{\theta}}$ are free parameter magnitudes, and $\mathbf{z}$ represent independent binary random gates indicating whether a parameter is active or not. To guarantee differentiability, they model each of these gates using a modified version of the concrete distribution (Maddison et al., 2017; Jang et al., 2017).

Consider a concrete random variable $s_j \sim q(\cdot|\phi_j \triangleq (\alpha_j, \beta_j))$. This variable is then stretched and clamped, resulting in a mixed distribution with point masses at 0 and 1, and a continuous density over $(0, 1)$. Formally, given $U_j \sim \text{Unif}(0, 1)$ and hyper-parameters $\gamma < 0 < 1 < \zeta$,

$$s_j = \text{Sigmoid}\left(\frac{1}{\beta_j}\log\left(\frac{\alpha_j\, U_j}{1-U_j}\right)\right), \qquad \mathbf{z} = \text{clamp}_{[0,1]}(\mathbf{s}(\zeta-\gamma)+\gamma)). \tag{2}$$

Given this stochastic reparameterization of the network parameters, the authors re-define the training objective to be the mean (under the gate distribution) of the L_0-regularized empirical risk in Eq. (1):

$$\mathcal{R}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\phi}) \triangleq \mathbb{E}_{\mathbf{z}\,|\,\boldsymbol{\phi}}\left[\mathcal{R}(\tilde{\boldsymbol{\theta}} \odot \mathbf{z})\right] = \underbrace{\mathbb{E}_{\mathbf{z}\,|\,\boldsymbol{\phi}}\left[\frac{1}{N}\sum_{i=1}^{N} \ell\left(h(x_i;\ \tilde{\boldsymbol{\theta}} \odot \mathbf{z}),\ y_i\right)\right]}_{\triangleq\ \mathfrak{f}_{\text{obj}}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\phi})} + \lambda\ \underbrace{\mathbb{E}_{\mathbf{z}\,|\,\boldsymbol{\phi}}\left[\ \|\mathbf{z}\|_0\right]}_{\triangleq\ \mathfrak{g}_{\text{const}}(\boldsymbol{\phi})}. \tag{3}$$

In practice, the first term is estimated using Monte Carlo samples. Notably, under the specific choice of re-parametrization in Eq. (2), the expected L_0-norm can be expressed in closed-form as:

$$\mathbb{E}_{\mathbf{z}|\boldsymbol{\phi}}\left[\ \|\boldsymbol{\theta}\|_0\right] = \sum_{j=1}^{|\boldsymbol{\theta}|} \mathbb{P}[z_j \neq 0] = \sum_{j=1}^{|\boldsymbol{\theta}|} \text{Sigmoid}\left(\log\alpha_j - \beta\log\frac{-\gamma}{\zeta}\right) \tag{4}$$

where $\beta_j = \beta$ is shared across gates. Note that one parameter ϕ_j is required for each gate, and thus using one gate per model parameter, effectively doubling the number of training parameters. Instead of considering a gate for each individual parameter, parameters may be gathered under a shared gate, thus leading to a reduction in the number of training parameters. All experiments presented in this work match the setting of (Louizos et al., 2018): i) a gate per *input* neuron for fully connected layers, and ii) a gate per *output* feature map for convolutional layers.

Moreover, the use of this "structured sparsity" results in practical computational improvements as whole parameter groups (e.g. entire slices of a convolution kernel) can be discarded. It is challenging to achieve comparable savings when using "unstructured/parameter sparsity" (with individual, scattered parameters being sparse) without specialized hardware.

3 Regularization via L_0 constraints

We favor formulating regularization goals as constraints, rather than as additive penalties. We refer to these two approaches as *constrained* and *penalized*, respectively. Although an ubiquitous tool in machine learning, penalized formulations may come at the cost of hyper-parameter interpretability and are susceptible to intricate dynamics when trying to incorporate multiple sources of regularization. In the following we show how training L_0-sparse models using a constrained formulation allows us to overcome these challenges.

3.1 Constrained formulation

In contrast to the penalized objective of (Louizos et al., 2018) presented in Eq (4), we propose to incorporate the L_0 sparsity through constraints. We formulate an optimization problem that aims to minimize the model's empirical risk, subject to constraints on the expected L_0-norm of pre-determined parameter groups:

$$\begin{aligned} \min_{\tilde{\boldsymbol{\theta}}, \boldsymbol{\phi}} \quad & \mathfrak{f}_{\text{obj}}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\phi}) \\ \text{s.t.} \quad & \mathfrak{g}_{\text{const}}(\boldsymbol{\phi}[g]) \leq \epsilon_g \cdot \texttt{parcount}(\boldsymbol{\phi}[g]) \;\; \text{for } g \in [1, \dots G], \end{aligned} \tag{5}$$

where g denotes a subset of gates, $\boldsymbol{\phi}[g]$ selects the parameters of the stochastic gates in g, and $\texttt{parcount}(\boldsymbol{\phi}[g])$ counts the total number of parameters associated with gates in g[2].

Note how the $\texttt{parcount}(\boldsymbol{\phi}[g])$ factor in the constraint levels allows us to interpret ϵ_g as the maximum proportion of gates that are allowed to be active (in expectation) within group g. We refer to ϵ_g as the *target density* of group g.

Lowering the target density demands a sparser model and thus a (not necessarily strictly) more challenging optimization problem from the point of view of the best feasible empirical risk. However, for any choice of $\epsilon_g \geq 0$, the feasible set in Eq (5) is always non-empty; while values of $\epsilon_g \geq 1$ result in vacuous constraints.

We highlight one important difference between the constrained and penalized formulations. The penalized approach is optimizing "jointly" the training loss $\mathfrak{f}_{\text{obj}}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\phi})$ and the expected L_0-norm $\mathfrak{g}_{\text{const}}(\boldsymbol{\phi})$, due to their additive combination (mediated by λ). Meanwhile, the constrained method focuses on obtaining the best possible model within a certain sparsity level ϵ: given two *feasible* solutions, the formulation in Eq (5) only discriminates based on the training losses. In other words, we aim to *satisfy* the constraints, not to *optimize* them.

3.2 Constraints can be liberating

We use the training of L_0-sparse networks to illustrate the advantages that constrained formulations of regularized problems can provide. Note that we do not intend to claim that constrained approaches are universally better than their penalized counterparts.

First, the semantics of the constraint function $\mathfrak{g}_{\text{const}}(\boldsymbol{\phi}[g])$ provide a straightforward interpretation of the "hyper-parameter" in the constrained approach as the maximum proportion of active gates within group g. In contrast, the value of the multiplier λ in the penalized formulation of (Louizos et al., 2018) lacks a comparably clear interpretation. This challenge becomes even stronger in the presence of multiple sources of regularization (i.e., a separate λ for each parameter group).

The tuning of ϵ and λ is closely linked to the previous discussion on interpretability. Application specific requirements can provide information on what the desired network sparsity may be. For

[2]For example, consider a $[d_{\text{in}}, d_{\text{hid}}, d_{\text{out}}]$ 1-hidden layer MLP with input-neuron sparsity on both of its two fully connected layers. For simplicity, we ignore the bias in the description below.

- **Grouping at the layer level** (akin to "λ sep." in (Louizos et al., 2018)), would yield $G = 2$ groups, with $\#\{g = 1\} = d_{\text{in}}$ gates in group $g = 1$. Note that each gate in group 1 is shared across d_{hid} parameters in $\boldsymbol{\theta}$, thus $\texttt{parcount}(\boldsymbol{\phi}[1]) = d_{\text{in}} \cdot d_{\text{hid}}$. Similarly for group $g = 2$.
- Alternatively, it is possible to **group at the model level** (akin to "$\lambda = \frac{\#}{N}$" in (Louizos et al., 2018)). In this case we have $G = 1$ group, with $\#\{g = 1\} = d_{\text{in}} + d_{\text{hid}}$ gates. Finally, $\texttt{parcount}(\boldsymbol{\phi}[1]) = d_{\text{in}} \cdot d_{\text{hid}} + d_{\text{hid}} \cdot d_{\text{out}}$ gives the total number of parameters in the network, as expected.

example, if the network will be deployed on a resource constrained device, achieving low latency gives bounds on the allowed number of FLOPS, which directly depend on the number of active parameters in the network. In the constrained formulation, such requirements can be easily translated from number of FLOPS into target sparsity levels for (each layer of) the model.

On the other hand, achieving a pre-specified level of sparsity using the penalized approach may require running several experiments, to find a suitable value of λ which yields an acceptably dense network with good predictive performance. Finally, employing several multipliers in order to impose different density levels across different layers in the model is challenging as these multipliers interact in intricate ways in the penalized objective. Degrave and Korshunova (2021) provide a brief and modern discussion of the failure modes of gradient descent at thoroughly exploring all trade-offs between regularized objectives for problems with non-convex Pareto fronts.

3.3 Solving the constrained problem

There is vast literature in solving constrained convex optimization problems (Boyd and Vandenberghe, 2004), with notable examples including interior point methods or the Frank-Wolfe algorithm (Frank and Wolfe, 1956; Jaggi, 2013). However, many of these methods assume convexity in the objective/constraints, or specific structure of the feasible set. In this work, we focus on applications that involve neural networks, typically leading to the violation of both of these assumptions.

We start by considering the (nonconvex-concave) Lagrangian associated with the constrained formulation in Eq (5), along with the corresponding min-max game:

$$\underset{\tilde{\boldsymbol{\theta}},\boldsymbol{\phi}}{\operatorname{argmin}} \underset{\boldsymbol{\lambda}\geq 0}{\operatorname{argmax}} \; \mathcal{L}(\tilde{\boldsymbol{\theta}},\boldsymbol{\phi},\boldsymbol{\lambda}) \triangleq \mathfrak{f}_{\text{obj}}(\tilde{\boldsymbol{\theta}},\boldsymbol{\phi}) + \sum_g \boldsymbol{\lambda}_g \left(\frac{\mathfrak{g}_{\text{const}}(\boldsymbol{\phi}[g])}{\texttt{parcount}(\boldsymbol{\phi}[g])} - \epsilon_g \right), \tag{6}$$

where $\boldsymbol{\lambda} = [\lambda_g]_{g=1}^G \geq 0$ are Lagrange multipliers associated with each of the constraints.

A commonly used approach to optimize this Lagrangian is *simultaneous* gradient descent on $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, and projected (to $\mathbb{R}^+$) gradient ascent on $\boldsymbol{\lambda}$:

$$\begin{aligned}
[\tilde{\boldsymbol{\theta}}_{t+1}, \boldsymbol{\phi}_{t+1}] &= [\tilde{\boldsymbol{\theta}}_t, \boldsymbol{\phi}_t] - \eta_{\text{primal}} \nabla_{[\tilde{\boldsymbol{\theta}}_t, \boldsymbol{\phi}_t]} \mathcal{L}(\tilde{\boldsymbol{\theta}}_t, \boldsymbol{\phi}_t, \boldsymbol{\lambda}_t) \\
\hat{\boldsymbol{\lambda}}_{t+1} &= \boldsymbol{\lambda}_t + \eta_{\text{dual}} \nabla_{\boldsymbol{\lambda}_t} \mathcal{L}(\tilde{\boldsymbol{\theta}}_t, \boldsymbol{\phi}_t, \boldsymbol{\lambda}_t) = \boldsymbol{\lambda}_t + \eta_{\text{dual}} \left[\frac{\mathfrak{g}_{\text{const}}(\boldsymbol{\phi}[g])}{\texttt{parcount}(\boldsymbol{\phi}[g])} - \epsilon_g \right]_{g=1}^G \\
\boldsymbol{\lambda}_{t+1} &= \max\left(0, \hat{\boldsymbol{\lambda}}_{t+1}\right)
\end{aligned} \tag{7}$$

Note how the gradient update for $\boldsymbol{\lambda}$ matches the size of the violation of each constraint. When a constraint is satisfied, the gradient for its corresponding Lagrange multiplier is non-positive, leading to a reduction in the value of the multiplier. On the other hand, just like in the penalized formulation of (Louizos et al., 2018), the update for $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ requires the gradient of the prediction loss and that of the re-parameterized L_0-norm. Hence, the cost of executing this update scheme is the same (up to the negligible cost of updating the multiplier values) as the cost of a gradient descent update on the penalized formulation in Eq (4).

Recall that a pure Nash equilibrium of this min-max game corresponds to a saddle point of the Lagrangian and determines an optimal, feasible solution. However, for non-convex problems such pure Nash equilibria might not exist, potentially leading to oscillations in parameter space when performing gradient descent-ascent (GDA) updates. Cotter et al. (2019) propose an algorithm that returns a pair of mixed strategies which configure a "semi-coarse correlated equilibrium". Experimentally we observed stable, reliable, non-oscillatory behavior when using GDA.

Other approaches, such as extra-gradient (Korpelevich, 1976), provide better convergence guarantees for games like Eq (6), compared to GDA. However, extra-gradient requires twice as many gradient computations per parameter update and the storage of an auxiliary copy of all the trainable parameters. Nonetheless, extrapolation from the past (Gidel et al., 2019) enjoys similar convergence properties to extra-gradient without requiring a second gradient computation. Our experiments showed no significant difference in performance between GDA and extra-gradient.

One drawback of these techniques is that the constraint violations accumulate in the value of the Lagrange multipliers, which continues to affect the optimization dynamics, even after a constraint

is being satisfied. This results in an excessive regularization, moving the parameters *towards the interior of the feasible set*, which can be detrimental if we are only concerned about *satisfying* (not minimizing) the constraint. To address this, we propose a **dual restart** scheme in which the Lagrange multiplier associated with a constraint is set to 0 whenever the constraint is satisfied, rather than waiting for the (negative gradient updates) to reduce its value. This removes the contribution of the expected L_0-norm to the Lagrangian for groups g whose constraints are satisfied, so that the optimization may focus on improving the predictive performance. Any further, more restrictive sparsity requirements, should be specified as a tighter constraint level.

4 Related work

Cotter et al. (2019) train models under constraints on the prediction rates of the model over different datasets. The authors argue this approach enables them to take advantage of side information by incorporating this in a straightforward manner into a training pipeline. While our work shares a similar motivation, the sparsity-inducing regularization we study in this paper depends only on properties of the model *parameters* and not on the model *predictions*.

The hard-concrete reparameterization in Eq (2) allows us to consider a non-convex, (almost everywhere) differentiable, constrained optimization problem. It would be interesting to incorporate the idea of *proxy constraints* by Cotter et al. (2019) in measuring the actual sparsity of the model used at test time, rather than the surrogate metric given by the model's expected L_0 norm.

In recent work, Lin et al. (2020) present non-asymptotic complexity results showing that two-timescale GDA can find stationary points for nonconvex-concave minimax problems efficiently.

5 Experiments

Following Louizos et al. (2018), we evaluate our approach on the MNIST classification task using two different architectures: i) an MLP with 2 hidden layers with 300 and 100 units respectively, and ii) a convolutional LeNet-5-Caffe network. We followed their reparametrization approach and used their protocol for network initialization and for the hard-concrete reparametrization of the gates.

The experiments shown below use Adam (Kingma and Ba, 2015) descent scheme for the model parameters with learning rate $\eta_{\text{primal}} = 7 \cdot 10^{-4}$, and gradient ascent on the Lagrange multipliers with learning rate $\eta_{\text{dual}} = 10^{-4}$. Experiments shown in Table 1 employ dual restarts.

To ensure a fair comparison with the models obtained by Louizos et al. (2018), we estimate the density levels of their pruned architectures and used these as target density levels for our approach, hence leveraging the interpretability of the constraint formulation. Similarly to Louizos et al. (2018), we regularize our models with two grouping schemes: a) with a single constraint at the model level, or b) with one constraint per layer.

Performance comparison. We report in Table 1 the resulting pruned architectures and their corresponding performances. Previous works reported the best performance throughout training only. Regarding this evaluation metric, our method proves to be competitive compared to previous work, and in worst cases (MLP with single constraint) as good as (Louizos et al., 2018). We also report the average error at the end of training, across 5 different runs.

Training dynamics. Figure 1 portrays the training dynamics of our approach. During the first epochs of training, the optimization focuses on improving the model's prediction capabilities and disregards sparsity. When the multiplier increases, the density of the model is reduced, which also causes the validation error to increase. Afterwards, when the model reaches the desired sparsity level, λ_g decreases. For optimizations with dual restarts, this amounts to a decrease back to 0, while for no dual restarts the decrease is gradual and slow. Then, the model trained with dual restarts stays near the desired density level during the rest of training, whilst the other becomes more sparse at the cost of prediction performance.

Architecture	Approach	Pruned architecture	Error (%) best	Error (%) at 200 epochs (avg ± 95% CI)
MLP 784-300-100	Sparse VD (Molchanov et al., 2017)	512-114-72	1.8	–
	BC-GNJ (Louizos et al., 2017)	278-98-13	1.8	–
	BC-GHS (Louizos et al., 2017)	311-86-14	1.8	–
	$\lambda = 0.1/N$ (Louizos et al., 2018)	219-214-100	1.4	–
	Ours: $\epsilon = 33\%$	198-233-100	1.4	1.69 ± 0.11
	$\lambda = [0.1, 0.1, 0.1]/N$ (Louizos et al., 2018)	266-88-33	1.8	–
	Ours: $\epsilon = [30\%, 30\%, 30\%]$	243-89-29	1.63	2.18 ± 0.10
LeNet5 20-50-800-500	Sparse VD (Molchanov et al., 2017)	14-19-242-131	1.0	–
	GL (Wen et al., 2016)	3-12-192-500	1.0	–
	GD (Srinivas and Babu, 2016)	7-13-208-16	1.1	–
	SBP (Neklyudov et al., 2017)	3-18-284-283	0.9	–
	BC-GNJ (Louizos et al., 2017)	8-13-88-13	1.0	–
	BC-GHS (Louizos et al., 2017)	5-10-76-16	1.0	–
	$\lambda = 0.1/N$ (Louizos et al., 2018)	20-25-45-462	0.9	–
	Ours: $\epsilon = 10\%$	20-21-34-407	0.53	1.04 ± 0.09
	$\lambda = [10, 0.5, 0.1, 0.1]/N$ (Louizos et al., 2018)	9-18-65-25	1.0	–
	Ours: $\epsilon = [50\%, 30\%, 70\%, 10\%]$	10-14-224-29	0.7	0.97 ± 0.06

Table 1: Pruned MLP and LeNet5 Caffe architectures and their respective performances. Previous works only reported best performance throughout training. In addition, for our models, we also report the average final performance across 5 runs of 200 epochs each, along with 95% confidence intervals.

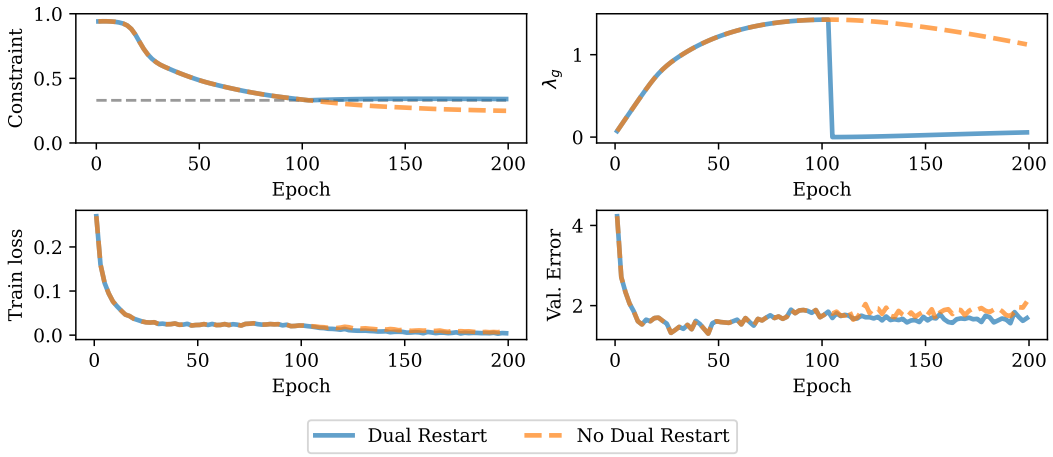

Figure 1: Training dynamics for an MLP model with $\epsilon = 33\%$ (at the model-level) along 200 epochs, one trained employing dual restarts and another without them.

Acknowledgements

We thank Manuel del Verme for many insightful discussions and his contributions to the implementation of some constrained optimization methods used in this paper.

JGP was partially supported by the Canada CIFAR AI Chair Program and by an IVADO PhD Excellence Scholarship.

References

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE TPAMI*, 35(8):1798–1828, 2013.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language Models are Few-Shot Learners. In *NeurIPS*, 2020.

R. Caruana, S. Lawrence, and C. Giles. Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping. In *NeurIPS*, 2001.

A. Cotter, H. Jiang, M. Gupta, S. Wang, T. Narayan, S. You, and K. Sridharan. Optimization with Non-Differentiable Constraints with Applications to Fairness, Recall, Churn, and Other Goals. In *JMLR*, 2019.

J. Degrave and I. Korshunova. Why machine learning algorithms are hard to tune and how to fix it. Engraved, blog: `www.engraved.blog/why-machine-learning-algorithms-are-hard-to-tune/`, 2021.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

J. Frankle and M. Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *ICLR*, 2019.

G. Gidel, H. Berard, G. Vignoud, P. Vincent, and S. Lacoste-Julien. A Variational Inequality Perspective on Generative Adversarial Networks. In *ICLR*, 2019.

S. Han, H. Mao, and W. J. Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In *ICLR*, 2016.

M. Jaggi. Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization. In *ICML*, 2013.

E. Jang, S. Gu, and B. Poole. Categorical Reparameterization with Gumbel-Softmax. In *ICLR*, 2017.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

G. M. Korpelevich. The extragradient method for finding saddle points and other problems. *Matecon*, 1976.

T. Lin, C. Jin, and M. Jordan. On Gradient Descent Ascent for Nonconvex-Concave Minimax Problems. In *ICML*, 2020.

C. Louizos, K. Ullrich, and M. Welling. Bayesian Compression for Deep Learning. In *NeurIPS*, 2017.

C. Louizos, M. Welling, and D. P. Kingma. Learning Sparse Neural Networks through L_0 Regularization. In *ICLR*, 2018.

C. J. Maddison, A. Mnih, and Y. W. Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *ICLR*, 2017.

D. Molchanov, A. Ashukha, and D. Vetrov. Variational Dropout Sparsifies Deep Neural Networks. In *ICML*, 2017.

K. Neklyudov, D. Molchanov, A. Ashukha, and D. Vetrov. Structured Bayesian Pruning via Log-Normal Multiplicative Noise. In *NeurIPS*, 2017.

B. Neyshabur, R. Tomioka, and N. Srebro. In Search of the Real Inductive Bias: On the Role of Implicit Regularization in Deep Learning. In *ICLR - Workshop*, 2015.

S. Srinivas and R. V. Babu. Generalized Dropout. *arXiv preprint arXiv:1611.06791*, 2016.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. In *JMLR*, volume 15, pages 1929–1958, 2014.

K. Ullrich, E. Meeds, and M. Welling. Soft Weight-Sharing for Neural Network Compression. In *ICLR*, 2017.

V. Vapnik and A. Chervonenkis. Theory of Pattern Recognition, 1974.

W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning Structured Sparsity in Deep Neural Networks. In *NeurIPS*, 2016.

A Varying the target density

Figure 2 shows the performance of our approach when setting different target densities. The initial model architectures and the optimization hyper-parameters are the same as presented in Section 5. Note that while many of the results correspond to models which are *unfeasible* (due to the use of dual restarts), the magnitude of the constraint violation is very low. We remark that our proposed constrained method mostly results in models whose density is reasonably close the desired target density, for a wide range of values of the target density.

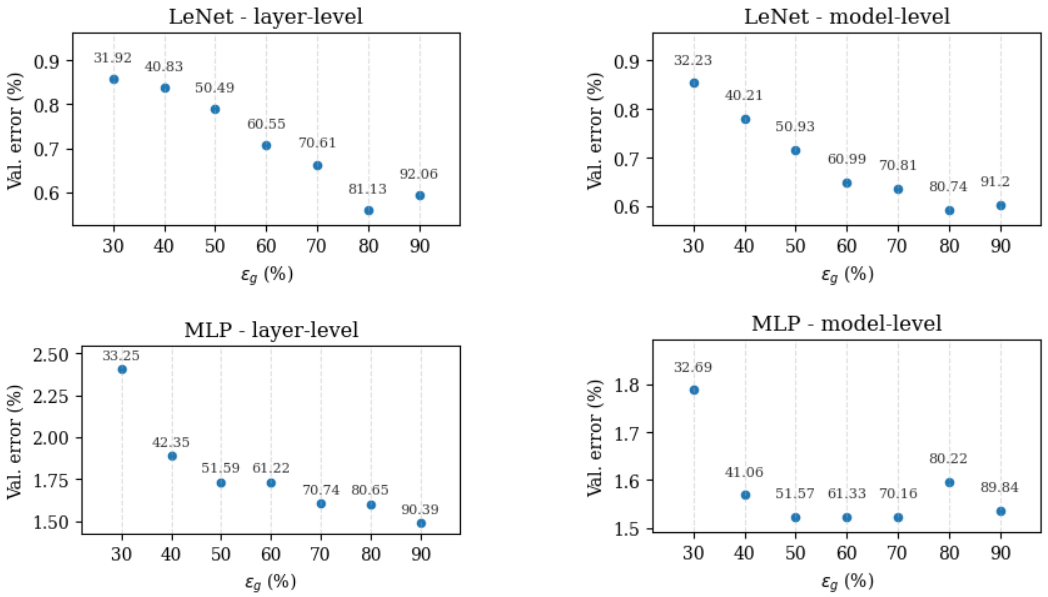

Figure 2: Average error after 100 epochs across three runs for different configurations of the constraint level. Top panels correspond to the LeNet architecture, and inferior panels to MLPs; left panels comprise runs where parameters are grouped at a layer level, those on the right where groups at a model level.
Annotations indicate the *worst* achieved constraint value across runs. For layer-level experiments, this is selected according to the most dense layer.

As expected, low values of ϵ_g demand more sparse models which have worse predictive performance. This is more notorious for layer-wise experiments, as this grouping results in constraints which reduce the flexibility of the model in comparison to model-layer groupings.