CONVEX NEURAL AUTOREGRESSIVE MODELS: TOWARDS TRACTABLE, EXPRESSIVE, AND THEORETICALLY-BACKED MODELS FOR SEQUENTIAL FORECASTING AND GENERATION

Vikul Gupta, Burak Bartan, Tolga Ergen, Mert Pilanci

Department of Electrical Engineering, Stanford University

ABSTRACT

Three features are crucial for sequential forecasting and generation models: tractability, expressiveness, and theoretical backing. While neural autoregressive models are relatively tractable and offer powerful predictive and generative capabilities, they often have complex optimization landscapes, and their theoretical properties are not well understood. To address these issues, we present convex formulations of autoregressive models with one hidden layer. Specifically, we prove an exact equivalence between these models and constrained, regularized logistic regression by using semi-infinite duality to embed the data matrix onto a higher dimensional space and introducing inequality constraints. To make this formulation tractable, we approximate the constraints using a hinge loss or drop them altogether. Furthermore, we demonstrate faster training and competitive performance of these implementations compared to their neural network counterparts on a variety of data sets. Consequently, we introduce techniques to derive tractable, expressive, and theoretically-interpretable models that are nearly equivalent to neural autoregressive models.

***Index Terms*—** Neural Autoregressive Models, Convex Optimization, Sequential Forecasting and Generation

1. INTRODUCTION

Sequential data is everywhere, from language to finance to weather. By modeling this data, we can solve downstream tasks such as predicting the weather and generating speech. One idea is to model the probability distribution the data is sampled from. However, these models, such as the restricted Boltzmann machine [1], are intractable for large data sets. Other models, such as traditional autoregressive models, i.e., AR, ARMA, and ARIMA models [2], make assumptions about the data. These models are tractable but limit their expressiveness by making too many assumptions. Still other models, such as deep autoregressive models [3, 4, 5, 6, 7], variational autoencoders (VAEs) [8], and generative adversarial networks (GANs) [9], make fewer assumptions and let neural networks learn the probability distributions. While these models are expressive and usually tractable, we have little theoretical insight into them, a barrier preventing their use in the real world. We tackle this last category of models by attempting to develop the theory and improve the tractability of deep autoregressive models. In particular, we analyze two-layer autoregressive models instead of state-of-the-art models such as PixelRNN [5], WaveNet [7], and Image-GPT [6] to build the theoretical foundations of these advanced models.

There are several directions of research to better understand neural networks. Some approaches attempt to understand the optimization landscape of such networks by proving theoretical guarantees for algorithms [10, 11, 12] and specific models [13, 14]. However, these methods do not allow for analyses of neural networks besides convergence guarantees. The mean field view [15] does not provide sufficient analysis for a finite number of hidden units, whereas the neural tangent kernel view [16] provides an insufficient interpretation of empirical results. In [17, 18, 19, 20, 21], the authors introduce equivalent convex programs for two-layer, dense neural networks with convex loss. We extend their theory to deep, autoregressive models in Section 2.2. Further, we introduce tractable convex models by adapting the convex programs to allow for mini-batching.

Our Contribution: We introduce exact and tractable convex formulations for shallow neural autoregressive models. Using semi-infinite duality [22], we show an equivalence between the non-convex, binary cross-entropy loss of a two-layer autoregressive model and the convex loss of a constrained and regularized logistic regression problem (Section 2). By proving this equivalence, we demonstrate that the two-layer autoregressive model essentially solves a logistic regression problem in a constrained, higher dimensional space. This insight can be used in future theoretical analysis of such models, such as better interpretations of the learned distributions. Moreover, solving the convex problem is easier than solving the non-convex one, since optimization algorithms for the former efficiently reach the global minimum instead of getting stuck in local optima.

Due to the constraints in this formulation, optimization algorithms that take advantage of mini-batching cannot be applied directly. As such, we introduce two tractable versions of the convex problem: (1) we approximate the constraints using a hinge loss and (2) we drop the constraints to form a logistic regression with group lasso problem [23] (Section 3). We then

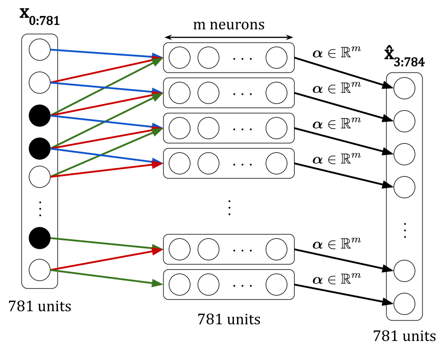

Fig. 1. Two-layer autoregressive model. Output probability for an entry is predicted using previous $r = 3$ entries. All arrows with the same color are shared parameters.

evaluate efficient implementations of these formulations on binary vector data sets and a binarized MNIST data set [24] (Section 4). Lastly, we discuss how to extend this work to other two-layer autoregressive models and beyond (Section 5).

2. EXACT FORMULATION

Suppose we have N sequences of M-dimensional, binary data, $\boldsymbol{X} \in \{0,1\}^{N\times M}$. We would like to model the probability distribution, $p(\boldsymbol{x})$, the sequences were sampled from. Section 2.1 sets up the non-convex formulation for an explanatory subset of autoregressive models, and Section 2.2 derives an equivalent convex formulation. Section 5 discusses extensions to other two-layer autoregressive models.

2.1. Autoregressive Model

We first factorize the distribution $p(\boldsymbol{x})$ using the chain rule:

$$p(\boldsymbol{x}) = \prod_{d=1}^{M} p(x_d|x_1, x_2, \dots, x_{d-1}) = \prod_{d=1}^{M} p(x_d|\boldsymbol{x}_{<d}),$$

where $\boldsymbol{x}_{<d} = [x_1, x_2, \dots, x_{d-1}]$. We model each of these conditional distributions with a Bernoulli random variable, i.e., $p_\theta(x_d|\boldsymbol{x}_{<d}) = \text{Bern}(\mu(\boldsymbol{x}_{<d}))$, where θ are the parameters (weights) of the mean function $\mu : \{0,1\}^{d-1} \to [0,1]$. This function is modeled by a two-layer neural network (Figure 1), whose parameters are learned by minimizing a regularized negative log likelihood loss.

We note here that several assumptions that result in slightly worse-performing models are made for ease of explanation. First, only the previous r elements of a sequence are used to model the next element, instead of all previous elements. Second, the mean function of each element's conditional probability distribution is the same. Third, the conditional probabilities for the first r elements of the sequence are ignored. These assumptions are likely not to be reflected in real data, but allow for a more digestible introduction to the techniques below.

Mathematically, the model is constructed as follows:

$$\begin{aligned} \boldsymbol{x}_{<d,(r)} &= [x_{d-r-1}, x_{d-r}, \dots, x_{d-1}] \in \mathbb{R}^r \\ \mu(\boldsymbol{x}_{<d}) &= \sigma(\boldsymbol{\alpha}^T \boldsymbol{h}) = \sigma(\boldsymbol{\alpha}^T (\boldsymbol{U}\boldsymbol{x}_{<d,(r)})_+). \end{aligned} \quad (1)$$

The output is the expectation of the d-th element of the sequence, x_d, and the inputs are the previous r elements, $\boldsymbol{x}_{<d,(r)}$. The first-layer weights are $\boldsymbol{U} \in \mathbb{R}^{m\times r}$, and the second-layer weights are $\boldsymbol{\alpha} \in \mathbb{R}^m$. The hidden layer, $\boldsymbol{h} = (\boldsymbol{U}\boldsymbol{x}_{<d,(r)})_+ \in \mathbb{R}^m$, uses ReLU activation, $(\cdot)_+ = \max(0, \cdot)$. The expectation, $\mu(\boldsymbol{x}_{<d})$, uses sigmoid activation, $\sigma(t) = \frac{1}{1+e^{-t}}$. To learn the data distribution, we simply minimize the binary cross-entropy loss, CE, over the weights $\boldsymbol{U}$ and $\boldsymbol{\alpha}$.

Define $\mathcal{L}(\boldsymbol{a}, \boldsymbol{b}) = CE(\sigma(\boldsymbol{a}), \boldsymbol{b})$, $D = M - r$, $F = ND$, and $\tilde{\boldsymbol{x}} \in \mathbb{R}^D$ as the last D elements of $\boldsymbol{x}$. Construct $\boldsymbol{y} \in \{0,1\}^F$ by stacking $\tilde{x}_d^{(n)}$ on top of each other for all $d = 1, \dots, D, n = 1, \dots, N$. Construct $\boldsymbol{X}_{(r)} \in \{0,1\}^{F\times r}$ by stacking $\tilde{\boldsymbol{x}}_{<d,(r)}^{(n)}$ in the same order. We introduce l_2-regularization terms for the weights and formulate the non-convex objective:

$$p^* := \min_{\{\alpha_j, \boldsymbol{u}_j\}_{j=1}^m} \mathcal{L}\left(\boldsymbol{\alpha}^T (\boldsymbol{U}\boldsymbol{X}_{(r)}^T)_+, \boldsymbol{y}\right) + \frac{\beta}{2} \sum_{j=1}^{m} (||\boldsymbol{u}_j||_2^2 + \alpha_j^2), \quad (2)$$

where $\beta > 0$ is a hyper-parameter, $\boldsymbol{u}_j \in \mathbb{R}^r$ is the j-th row of $\boldsymbol{U}$, and α_j is the j-th element of $\boldsymbol{\alpha}$.

2.2. Exact Convex Formulation

We construct a set of diagonal matrices. Let $\boldsymbol{z} \in \mathbb{R}^r$ be an arbitrary vector and $1[\boldsymbol{X}_{(r)}\boldsymbol{z} \geq 0] \in \{0,1\}^F$ be an indicator vector such that the j-th entry is Boolean: $1[\boldsymbol{X}_{(r)}\boldsymbol{z} \geq 0]_j := 1[\boldsymbol{x}_{(r),j}^T \boldsymbol{z} \geq 0]$. We define diagonal matrices $\boldsymbol{D}_1, \dots, \boldsymbol{D}_P$[1] as the set of all possible matrices of the form $\text{Diag}(1[\boldsymbol{X}_{(r)}\boldsymbol{z} \geq 0])$. Construct $\boldsymbol{G}_i = (2\boldsymbol{D}_i - \boldsymbol{I}_n)\boldsymbol{X}_{(r)}$ and the objective,

$$\begin{aligned} f_{\{\boldsymbol{v}_i, \boldsymbol{w}_i\}_{i=1}^P} = \mathcal{L}\Big(\sum_{i=1}^{P} \boldsymbol{D}_i \boldsymbol{X}_{(r)}(\boldsymbol{v}_i - \boldsymbol{w}_i), \boldsymbol{y}\Big) \\ + \beta \sum_{i=1}^{P} (||\boldsymbol{v}_i||_2 + ||\boldsymbol{w}_i||_2), \end{aligned} \quad (3)$$

to get the convex problem below ($\boldsymbol{v}_i, \boldsymbol{w}_i \in \mathbb{R}^r$):

$$\begin{aligned} &\min_{\{\boldsymbol{v}_i, \boldsymbol{w}_i\}_{i=1}^P} f_{\{\boldsymbol{v}_i, \boldsymbol{w}_i\}_{i=1}^P} \\ &\text{subject to } \boldsymbol{G}_i\boldsymbol{v}_i \geq 0, \boldsymbol{G}_i\boldsymbol{w}_i \geq 0 \text{ for } i = 1, \dots, P. \end{aligned} \quad (4)$$

Theorem 1 *The non-convex (2) and convex (4) problems have identical optimal values if* $m \geq m^*$.[2] *Furthermore, given*

[1]See section 3.3 for details about the number of matrices, P.

[2]m^* is bounded above by $F+1$. Please see [22] for a detailed explanation.

optimal solutions $\boldsymbol{v}_i^*, \boldsymbol{w}_i^*$ for (4) such that at most one of $\boldsymbol{v}_i^*$ or $\boldsymbol{w}_i^*$ is non-zero for all $i = 1, \ldots, P$, an optimal solution for (2) with m^* neurons is

$$(\boldsymbol{u}_{j_i}^*, \alpha_{j_i}^*) = \begin{cases} \left(\frac{\boldsymbol{v}_i^*}{\sqrt{\|\boldsymbol{v}_i^*\|_2}}, \sqrt{\|\boldsymbol{v}_i^*\|_2}\right) & \text{if } \|\boldsymbol{v}_i^*\|_2 > 0 \\ \left(\frac{\boldsymbol{w}_i^*}{\sqrt{\|\boldsymbol{w}_i^*\|_2}}, -\sqrt{\|\boldsymbol{w}_i^*\|_2}\right) & \text{if } \|\boldsymbol{w}_i^*\|_2 > 0 \end{cases}.$$

By Theorem 1, we see that solving the convex problem (4) is equivalent to solving the non-convex problem (2). Thus, since (4) is a constrained, regularized logistic regression problem (Section 3.1), training the deep autoregressive model in Section 2.1 is equivalent to solving a logistic regression problem.

3. TRACTABLE FORMULATIONS

Efficient mini-batch optimizers such as stochastic gradient descent (SGD) are not immediately applicable to solving (4) due to its constraints. To resolve this issue, we introduce the relaxed convex problem and hinge loss approximation below. We also discuss sampling the diagonal matrices.

3.1. Relaxed Convex Problem

Our first method to solve (4) efficiently is to remove the constraints. Specifically, given the same matrices $\boldsymbol{D}_1, \ldots, \boldsymbol{D}_P$, we define the following problem:

$$\min_{\{\boldsymbol{v}_i, \boldsymbol{w}_i\}_{i=1}^P} f_{\{\boldsymbol{v}_i, \boldsymbol{w}_i\}_{i=1}^P}. \tag{5}$$

We now provide an interpretation of this convex program. Define a set of data matrices $\boldsymbol{M}_i = \boldsymbol{D}_i \boldsymbol{X}_{(r)}$ for $i = 1, \ldots, P$. Then, each $\boldsymbol{M}_i$ is an arbitrary masking of $\boldsymbol{X}_{(r)}$. This gives us exactly the logistic regression with group lasso problem, where each $\boldsymbol{M}_i$ is the local feature matrix. Thus, we can view the first term in (5) as the aggregate binary cross-entropy loss over P local linear models, where the local feature matrix is a randomly masked version of the global feature matrix. The second term is a group lasso regularization that combines the local models into a global model.

Note that the constraints in (4) ensure that the dot products of the local variables with the local data points are positive, while those with the other data points are negative.

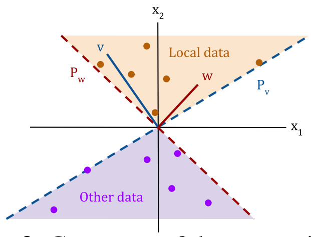

Fig. 2. Geometry of the constraints.

Geometrically, we construct two planes perpendicular to the two local variables to create four half-spaces. The local data are in the intersection of the two half-spaces containing the variables, whereas all other data are in the intersection of the two half-spaces that don't contain the variables (Figure 2). Moreover, the local models use precisely the data in the intersection of the two half-spaces containing the local variables to compute their contribution to the global loss.

3.2. Hinge Loss Approximation

As an alternative to simply dropping the constraints, we approximate them via a hinge loss. The modified minimization problem, with hyper-parameter $\rho > 0$, is

$$\min_{\{\boldsymbol{v}_i, \boldsymbol{w}_i\}_{i=1}^P} f_{\{\boldsymbol{v}_i, \boldsymbol{w}_i\}_{i=1}^P} + \rho \sum_{i=1}^P \mathbf{1}^T \left((\boldsymbol{G}_i \boldsymbol{v}_i)_+ + (\boldsymbol{G}_i \boldsymbol{w}_i)_+ \right). \tag{6}$$

3.3. Sampling Diagonal Matrices

The number of diagonal matrices P needed in the convex formulations is bounded above by a term on the order of $k(\frac{N}{k})^k$, where k is the rank of the matrix [22]. For instance, with $N = 1000$ sequences of dimension $D = 100$ and $r = 10$ (a small dataset), $P \leq O(10^{50})$. It is infeasible to solve (4) (5), or (6) with this many diagonal matrices. Instead, we sample $\tilde{P}$ diagonal matrices by generating $\tilde{P}$ uniformly random vectors $\boldsymbol{u}_i$ and picking $\boldsymbol{D}_i = 1[\boldsymbol{X}_{(r)} \boldsymbol{u}_i \geq 0]$ for $i = 1, \ldots, \tilde{P}$.

4. EXPERIMENTS

We consider the following implementations of the convex formulations: 'nn-full' minimizes the non-convex objective (2), 'exact-subsampled' solves the exact convex problem (4), 'hinge-full' minimizes the hinge loss-based problem (6), 'relaxed-subsampled' solves the relaxed convex problem (5), and 'relaxed-full' minimizes the relaxed convex problem (5). The 'full' implementations are trained on the entire dataset with SGD (PyTorch), whereas the 'subsampled' ones are trained on subsampled versions (1000 and 100,000 examples for exact-subsampled and relaxed-subsampled) because training on the entire dataset is infeasible without SGD (CVXPY). We evaluate the implementations on several benchmark data sets [4], namely Adult, Connect4, DNA, Mushrooms, RCV1, OCR-letters, and NIPS-0-12 (some results shown). These data sets are sampled from distributions of binary vectors of dimension 100 to 500. We also evaluate the implementations on a binarized MNIST data set of dimension 784.

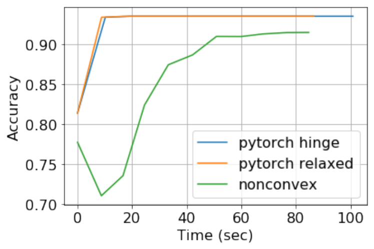

Fig. 3. Accuracy on MNIST validation data during training for the three PyTorch models (10 iterations).

PyTorch models: In general, the PyTorch implementations (nn-full, hinge-full, and relaxed-full) performed the best on the test set (Figure 4). Since relaxed-full performed as well

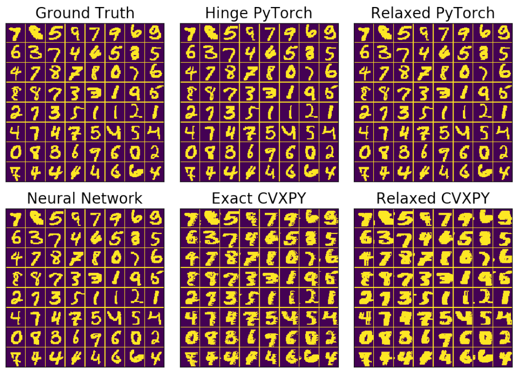

Fig. 4. Ground truth and generated images from MNIST test data. CVXPY implementations trained on 10 vectors (less than an image). Images generated by predicting each pixel given previous 10 pixels in the ground truth image (not recursively).

as hinge-full and nn-full, the inequality constraints did not affect the trained model's performance. Furthermore, for some data sets like Adult and Mushrooms, nn-full performed worse than hinge-full and relaxed-full, suggesting that SGD occasionally converged to a local minima in the non-convex landscape. The PyTorch implementations for the convex approaches converged faster than all other implementations. In particular, they converged in fewer iterations than the neural network (Figure 3), due to the relative simplicity of optimizing a convex landscape.Additionally, hinge-full converged in fewer iterations than relaxed-full, which we hypothesize to be because the constraints affect the convergence rate. However, each individual iteration of hinge-full was slower than those for relaxed-full and nn-full, which took the same time. Further analysis suggests that this is due to additional loss and gradient calculations from the hinge loss term.

CVXPY models: The CVXPY models (exact-subsampled and relaxed-subsampled) were generally slower and performed worse on the test sets than the PyTorch models (Figure 4). However, they both performed much better on the train sets. These results imply that, given a small subset of the train data, the global solution to the convex problem (and thus the non-convex problem) generalizes poorly to unseen data. Furthermore, relaxed-subsampled consistently performed even worse than exact-subsampled, suggesting that the constraints are essential when training on a small subset of the data. Lastly, since relaxed-subsampled was trained on many more examples and its training times were similar to those for exact-subsampled, the relaxed CVXPY implementation was substantially faster per example than the exact implementation.

5. DISCUSSION

We prove that a class of two-layer autoregressive models are equivalent to constrained, regularized logistic regression problems and propose methods of solving these problems efficiently. Through evaluation on several binary data sets, we demonstrate more efficient training and better performance of these programs compared to the non-convex neural network. Hence, we develop theoretically-backed, tractable, and powerful models for sequential forecasting and generation tasks.

We now discuss how to apply these techniques to other two-layer autoregressive models. First, we note that the construction of (4) only depends on the loss function $\mathcal{L}$, not the data $\boldsymbol{X}_{(r)}, \boldsymbol{y}$. Consequently, the relaxed convex problem and hinge loss approximation apply for arbitrary data. Second, Theorem 1 holds as long as the structure of (2) and (4) stays the same. Furthermore, by applying the results from [22], we can prove the equivalence for any convex $\mathcal{L}$. Thus, as long as we can represent the loss (without regularization) of any two-layer autoregressive model as $\mathcal{L}(\sum_{j=1}^{m}(\boldsymbol{X}\boldsymbol{u}_j)_+\alpha_j, \boldsymbol{y})$ with convex $\mathcal{L}$, we can apply these techniques.

There are several clear directions to further pursue this work. Research can be done into extending these techniques to deeper autoregressive models, such as PixelRNN [5], WaveNet [7], and Image-GPT [6]. Randomized dimension reduction [25, 26], iterative sketching [27, 28, 29, 30] and preconditioning methods [31, 32] can be used to address high-dimensional convex training problems. If successful, this could lead to better interpretation and training stability of such models.

Table 1. Results for binary vector data sets. Negative log likelihood (top), training time in seconds (bottom).

Model	Adult	DNA	OCR	RCV1
nn-full (baseline)	70.43	102.8	56.58	102.4
exact-subsampled	77.58	116.6	81.74	189.8
hinge-full	**68.34**	**101.6**	57.12	**100.7**
relaxed-subsampled	78.26	148.2	114.0	139.7
relaxed-full	**68.84**	**101.9**	**55.89**	**101.3**
nn-full (baseline)	593.7	235.6	583.4	1037
exact-subsampled	475.1	547.7	1009	4622
hinge-full	354.4	**105.8**	**449.9**	**185.5**
relaxed-subsampled	447.6	439.2	592.4	70868
relaxed-full	**299.8**	121.6	1121	678.0

6. REFERENCES

[1] Ruslan Salakhutdinov and Iain Murray, "On the quantitative analysis of deep belief networks," in *International Conference on Machine Learning*.

[2] Robert Shumway and David Stoffer, *Time Series Analysis and its Applications*, Springer Texts in Statistics, 4th edition.

[3] Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra, "Deep autoregressive networks," in *International Conference on Machine Learning*.

[4] Benigno Uria, Marc-Alexandre Cote, Karol Gregor, Iain Murray, and Hugo Larochelle, "Neural autoregressive distribution estimation," *Journal of Machine Learning Research*.

[5] "Aaron van den oord, nal kalchbrenner, and koray kavukcuoglu. pixel recurrent neural networks," in *International Conference on Machine Learning*.

[6] Mark Chen, Alec Radford, Rewon Child, Jeff Wu, Heewoo Jun, David Luan, and Ilya Sutskever, "Generative pretraining from pixels," in *International Conference on Machine Learning*.

[7] Aaron Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, and Alex Graves, "Nal kalchbrenner, andrew senior, and koray kavukcuoglu," WaveNet: A Generative Model for Raw Audio. arXiv:1609.03499,.

[8] Diederik Kingma and Max Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations*.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative adversarial networks," arXiv:1406.2661,.

[10] Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma, "Provable bounds for learning some deep representations," in *International Conference on Machine Learning*.

[11] Majid Janzamin, Hanie Sedghi, and Anima Anandkumar, "Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods," arXiv:1506.08473,.

[12] Hanie Sedghi and Anima Anandkumar, "Provable methods for training neural networks with sparse connectivity," in *International Conference on Learning Representation*.

[13] Alon, "Brutzkus and amir globerson. globally optimal gradient descent for a convnet with gaussian inputs," arXiv:1702.07966,.

[14] Rong Ge, Jason D. Lee, and Tengyu Ma, "Learning one-hidden-layer neural networks with landscape design," arXiv:1711.00501,.

[15] Song Mei, Andrea Montanari, and Phan-Minh Nguyen, "A mean field view of the landscape of two-layer neural networks," arXiv:1804.06561,.

[16] Arthur Jacot, Franck Gabriel, and Clément Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Neural Information Processing Systems*.

[17] Tolga Ergen and Mert Pilanci, "Implicit convex regularizers of cnn architectures: Convex optimization of two- and three-layer networks in polynomial time," *International Conference on Learning Representations (ICLR)*, 2021.

[18] Arda Sahiner, Tolga Ergen, John Pauly, and Mert Pilanci, "Vector-output relu neural network problems are copositive programs: Convex analysis of two layer networks and polynomial-time algorithms," *International Conference on Learning Representations (ICLR)*, 2021.

[19] Arda Sahiner, Morteza Mardani, Batu Ozturkler, Mert Pilanci, and John Pauly, "Convex regularization behind neural reconstruction," *International Conference on Learning Representations (ICLR)*, 2021.

[20] Burak Bartan and Mert Pilanci, "Neural spectrahedra and semidefinite lifts: Global convex optimization of polynomial activation neural networks in fully polynomial-time," *arXiv preprint arXiv:2101.02429*, 2021.

[21] Jonathan Lacotte and Mert Pilanci, "All local minima are global for two-layer relu neural networks: The hidden convex optimization landscape," *arXiv preprint arXiv:2006.05900*, 2020.

[22] Mert Pilanci and Tolga Ergen, "Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7695–7705.

[23] Lukas Meier, Sara Geer, and Peter Bühlmann, "The group lasso for logistic regression," *Journal of the Royal Statistical Society, Series B*.

[24] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proc. of the IEEE*, vol. 86, no. 11, pp. 2278–2324.

[25] Jonathan Lacotte, Mert Pilanci, and Marco Pavone, "High-dimensional optimization in adaptive random subspaces," in *Advances in Neural Information Processing Systems*, 2019, pp. 10847–10857.

[26] Jonathan Lacotte and Mert Pilanci, "Adaptive and oblivious randomized subspace methods for high-dimensional optimization: Sharp analysis and lower bounds," *arXiv preprint arXiv:2012.07054*, 2020.

[27] Jonathan Lacotte and Mert Pilanci, "Optimal randomized first-order methods for least-squares problems," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5587–5597.

[28] Jonathan Lacotte, Sifan Liu, Edgar Dobriban, and Mert Pilanci, "Optimal iterative sketching with the subsampled randomized hadamard transform," in *Advances in Neural Information Processing Systems*, 2020.

[29] Jonathan Lacotte and Mert Pilanci, "Faster least squares optimization," *arXiv preprint arXiv:1911.02675*, 2019.

[30] Jonathan Lacotte and Mert Pilanci, "Effective dimension adaptive sketching methods for faster regularized least-squares optimization," *Advances in neural information processing systems*, 2020.

[31] Ibrahim Kurban Ozaslan, Mert Pilanci, and Orhan Arikan, "Iterative hessian sketch with momentum," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 7470–7474.

[32] Ibrahim Kurban Ozaslan, Mert Pilanci, and Orhan Arikan, "M-ihs: An accelerated randomized preconditioning method avoiding costly matrix decompositions," *arXiv preprint arXiv:1912.03514*, 2019.