The Hidden Convex Optimization Landscape of Deep Neural Networks

---

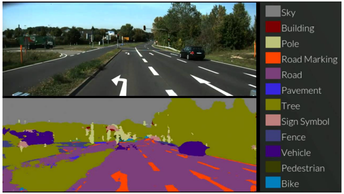
Mert Pilanci

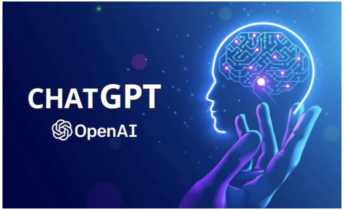
June 28, 2023

Electrical Engineering

Stanford University

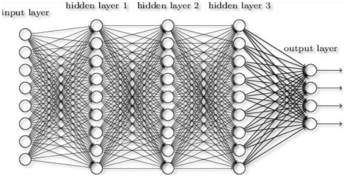
The Impact of Deep Learning





Y. LeCun, Y. Bengio, G. Hinton (2015)

Deep Neural Networks



- non-convex (stochastic) gradient descent
- extremely high-dimensional problems

  152 layer ResNet-152: 60.2 Million parameters (2015)

  GPT[1]-3 language model: 175 Billion parameters (May 2020)

  BAAI[2] multi-modal model: 1.75 Trillion parameters (June 2021)

  GPT-4 (March 2023)

[1] OpenAI General Purpose Transformer

[2] The Beijing Academy of Artificial Intelligence

deep learning models

- often provide the best performance due to their large capacity

  $\rightarrow$ **challenging to train**

deep learning models

- often provide the best performance due to their large capacity
  $\rightarrow$ **challenging to train**
- are complex black-box systems based on non-convex optimization
  $\rightarrow$ **hard to interpret what the model is actually learning**

deep learning models

- often provide the best performance due to their large capacity
  → **challenging to train**
- are complex black-box systems based on non-convex optimization
  → **hard to interpret what the model is actually learning**

nature

Letter | Published: 29 August 2018

Deep learning of aftershock patterns following large earthquakes

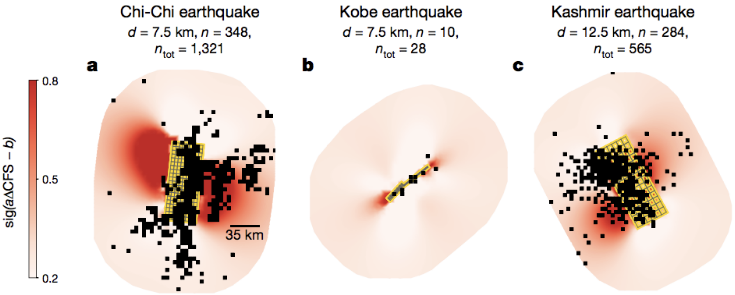
Phoebe M. R. DeVries, Fernanda Viégas, Martin Wattenberg & Brendan J. Meade

Nature **560**, 632–634(2018) | Cite this article

19k Accesses | **20** Citations | **1018** Altmetric | Metrics

deep learning models

- often provide the best performance due to their large capacity
  → **challenging to train**
- are complex black-box systems based on non-convex optimization
  → **hard to interpret what the model is actually learning**



deep learning models

- often provide the best performance due to their large capacity
  → **challenging to train**
- are complex black-box systems based on non-convex optimization
  → **hard to interpret what the model is actually learning**

one year later, another paper

nature

Matters Arising | Published: 02 October 2019

One neuron versus deep learning in aftershock prediction

Arnaud Mignan & Marco Broccardo

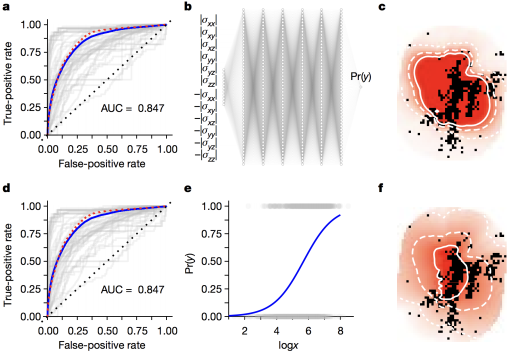
Nature **574**, E1–E3(2019) | Cite this article

6210 Accesses | **2** Citations | **367** Altmetric | Metrics

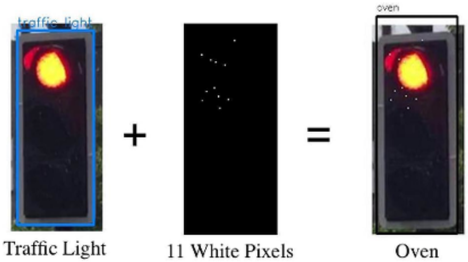
deep learning models

- often provide the best performance due to their large capacity
  $\rightarrow$ **challenging to train**
- are complex black-box systems based on non-convex optimization
  $\rightarrow$ **hard to interpret what the model is actually learning**

logistic regression (1 layer) has the same performance as the 6 layer NN for this task



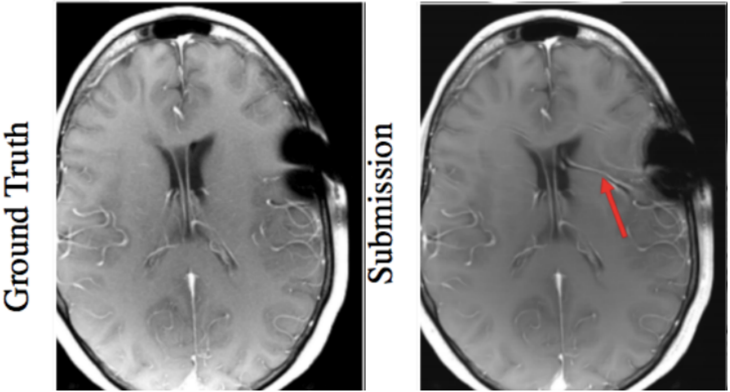
Sensitive to perturbations



- adversarial examples, Szegedy et al., 2014, Goodfellow et al., 2015
- (left) traffic light is classified as '**oven**' when 11 pixels are changed
- (right) stop sign recognized as speed limit sign, Evtimov et al, 2017

Fast MRI Challenge, 2020

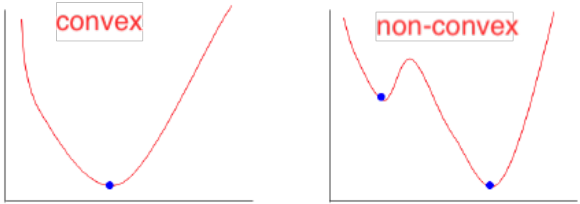
model generates a false vessel (Muckley et al.)



Open questions

- what are neural networks actually doing?
- can we make neural network models more reliable?
- can we make training energy/memory/data efficient?

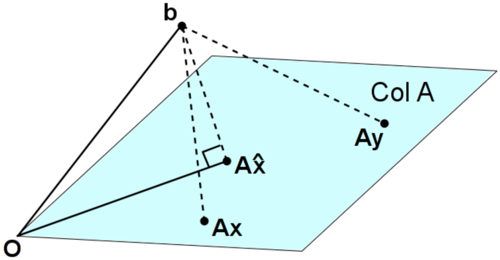
How neural networks work?



- Least-Squares, Logistic Regression, Support Vector Machines etc. are understood extremely well
- Insightful theorems for neural networks?

Least Squares

$$\min_x \|Ax - b\|_2^2$$



- optimality condition: $A^T(Ax - b) = 0$
- solvers: Cholesky/QR, Conjugate Gradient,...

Least Squares with L1 regularization

$$\min_x \|Ax - y\|_2^2 + \lambda\|x\|_1$$

- L1 norm $\|x\|_1 = \sum_{i=1}^d |x_i|$

  encourages solution x^* to be sparse

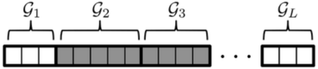
L1 regularization: mechanical interpretation with large λ

$$\min_x \quad \underbrace{\frac{1}{2}(x-y)^2}_{\text{elastic energy}} \quad + \quad \underbrace{\lambda|x|}_{\text{potential energy}}$$

red spring constant $=1$

blue ball mass $= \lambda$ (large)

Least Squares with group L1 regularization



$$\min_x \left\| \sum_{i=1}^{L} A_i x_i - y \right\|_2^2 + \lambda \sum_{i=1}^{L} \|x_i\|_2$$

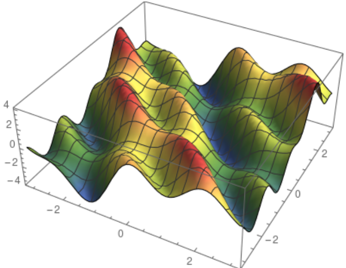
$\|x_i\|_2 = \sqrt{\sum_{j=1}^{d} x_{ij}^2}$

encourages solution x^* to be group sparse, i.e., most blocks x_i are zero

Training two-layer neural networks: Non-convex optimization

$$p_{\text{non-convex}} := \underset{\substack{W_1 \in \mathbb{R}^{d\times m} \\ W_2 \in \mathbb{R}^{m\times 1}}}{\text{minimize}} \quad L\left(\phi(XW_1)W_2, y\right) + \lambda\left(\|W_1\|_F^2 + \|W_2\|_F^2\right)$$

where $\phi(u)$ is the activation function



Rectified Linear Unit (ReLU) and Threshold Activations

$$p_{\text{non-convex}} := \underset{\substack{W_1 \in \mathbb{R}^{d\times m} \\ W_2 \in \mathbb{R}^{m\times 1}}}{\text{minimize}} \quad L\left(\phi(XW_1)W_2, y\right) + \lambda\left(\|W_1\|_F^2 + \|W_2\|_F^2\right)$$

where $\phi(u) = \text{ReLU}(u) = \max(0, u)$

or $\phi(u) = \text{sign}(u)$

Neural Networks are Convex Regularizers

$$
\begin{aligned}
p_{\textsf{non-convex}} := \quad & \textsf{minimize} \quad L\left(\phi(XW_1)W_2, y\right) + \lambda\left(\|W_1\|_F^2 + \|W_2\|_F^2\right) \\
& W_1 \in \mathbb{R}^{d\times m} \\
& W_2 \in \mathbb{R}^{m\times 1} \\
p_{\textsf{convex}} := \quad & \textsf{minimize} \quad L\left(Z, y\right) + \lambda \qquad \underbrace{R(Z)}_{\textsf{convex regularization}} \\
& Z \in \mathcal{K} \subseteq \mathbb{R}^{d\times p}
\end{aligned}
$$

$$p_{\text{non-convex}} := \underset{\substack{W_1 \in \mathbb{R}^{d\times m} \\ W_2 \in \mathbb{R}^{m\times 1}}}{\text{minimize}} \quad L\left(\phi(XW_1)W_2, y\right) + \lambda\left(\|W_1\|_F^2 + \|W_2\|_F^2\right)$$

$$p_{\text{convex}} := \underset{Z \in \mathcal{K} \subseteq \mathbb{R}^{d\times p}}{\text{minimize}} \quad L\left(Z, y\right) + \lambda R(Z)$$

Theorem $p_{\text{non-convex}} = p_{\text{convex}}$, and an optimal solution to $p_{\text{non-convex}}$ can be obtained from an optimal solution to p_{convex}.

M. Pilanci, T. Ergen, **Neural Networks are Convex Regularizers: Exact Polynomial-time Convex Optimization Formulations for Two-Layer Networks. ICML 2020**, T. Ergen, I. Gulluk, J. Lacotte, M. Pilanci, **ICLR 2023**

ReLU Network using squared loss = group Lasso using fixed features

data matrix $X \in \mathbb{R}^{n\times d}$ and label vector $y \in \mathbb{R}^n$ $X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$

$$p_{\text{non-convex}} = \text{ minimize}_{W_1,W_2} \left\| \sum_{j=1}^m \phi(XW_{1j})W_{2j} - y \right\|_2^2 + \lambda \left(\|W_1\|_F^2 + \|W_2\|_F^2 \right)$$

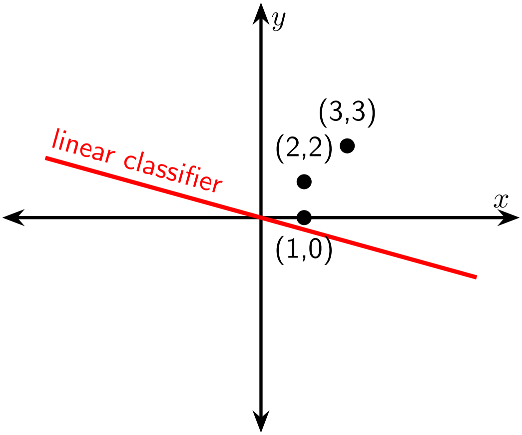
$$p_{\text{convex}} = \text{ minimize}_{u_1,v_1\ldots u_p,v_p\in\mathcal{K}} \left\| \sum_{i=1}^p D_i X(u_i - v_i) - y \right\|_2^2 + \lambda \left(\sum_{i=1}^p \|u_i\|_2 + \|v_i\|_2 \right)$$

$D_1, ..., D_p$ are fixed diagonal matrices

Theorem $p_{\text{non-convex}} = p_{\text{convex}}$, and an optimal solution to $p_{\text{non-convex}}$ can be recovered from optimal non-zero u_i^*, v_i^*, $i = 1, ..., p$ as

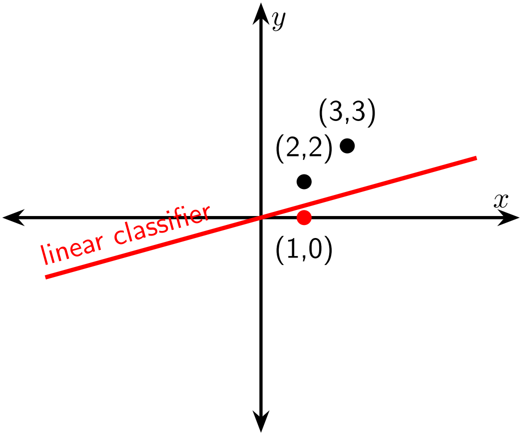
$W_{1j}^* = \frac{u_i^*}{\sqrt{\|u_i^*\|_2}}$, $W_{2j} = \sqrt{\|u_i^*\|_2}$ or $W_{1j}^* = \frac{v_j^*}{\sqrt{\|v_j^*\|_2}}$, $W_{2j} = -\sqrt{\|v_j^*\|_2}$.

$$n = 3 \text{ samples in } \mathbb{R}^d,\ d = 2 \quad X = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$



$$D_1 X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} X = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}$$

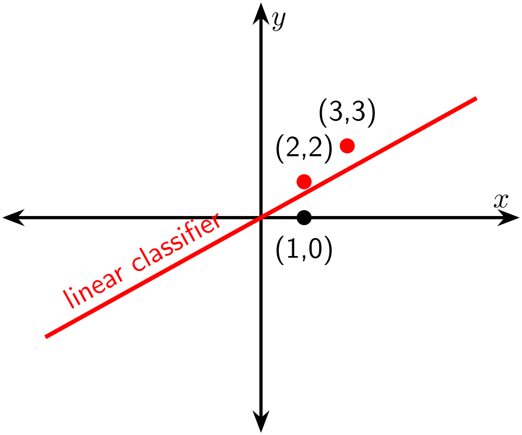
$n = 3$ samples in $\mathbb{R}^d$, $d = 2$ $\quad X = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$



$$D_1 X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} X = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}$$

$$D_2 X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} X = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 0 & 0 \end{bmatrix}$$

$n = 3$ samples in $\mathbb{R}^d$, $d = 2$ $\quad X = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$



$$D_1 X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} X = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}$$

$$D_2 X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} X = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 0 & 0 \end{bmatrix}$$

$$D_4 X = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} X = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}$$

Example: Convex Program for $n = 3, d = 2$

$$n = 3 \text{ samples} \quad X = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$\min \left\| \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \end{bmatrix} (u_1 - v_1) + \begin{bmatrix} x_1^T \\ x_2^T \\ 0 \end{bmatrix} (u_2 - v_2) + \begin{bmatrix} 0 \\ 0 \\ x_3^T \end{bmatrix} (u_3 - v_3) - y \right\|_2^2$$

subject to

$$+ \lambda \Big(\sum_{i=1}^{3} \|u_i\|_2 + \|v_i\|_2 \Big)$$

$D_1 X u_1 \geq 0, D_1 X v_1 \geq 0$

$D_2 X u_2 \geq 0, D_2 X v_2 \geq 0$

$D_4 X u_3 \geq 0, D_4 X v_3 \geq 0$

equivalent to the non-convex two-layer NN problem

Computational Complexity

Learning two-layer ReLU neural networks with m neurons

$f(x) = \sum_{j=1}^{m} W_{2j}\phi(W_{j1}x)$

Previous results: ◦ Combinatorial $O(2^m n^{dm})$ (Arora et al., ICLR 2018)

◦ Approximate $O(2^{\sqrt{m}})$ (Goel et al., COLT 2017)

$\boxed{\text{Convex program } O((\frac{n}{r})^r) \text{ where } r = \mathrm{rank}(X)}$

Computational Complexity

Learning two-layer ReLU neural networks with m neurons

$f(x) = \sum_{j=1}^{m} W_{2j}\phi(W_{j1}x)$

Previous results: ◦ Combinatorial $O(2^m n^{dm})$ (Arora et al., ICLR 2018)

◦ Approximate $O(2^{\sqrt{m}})$ (Goel et al., COLT 2017)

Convex program $O((\frac{n}{r})^r)$ where $r = \mathrm{rank}(X)$

n : number of samples, d : dimension

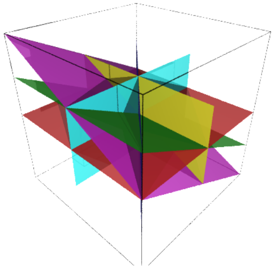
(i) polynomial in n and m for fixed rank r

(ii) exponential in d for full rank data $r = d$. This can not be improved unless $P = NP$ even for $m = 1$.

Number of variables = number of hyperplane arrangements

- convex program has at most $\left((\frac{n}{r})^r\right)$ variables

  #activation patterns of a **one neuron**
  $= \left|\{\mathbf{sign}(Xw)\ : w \in \mathbb{R}^d\}\right| \leq O\left((\frac{n}{r})^r\right)$ where $r = \mathbf{rank}(X)$.



- rank is constant for convolutional networks

  e.g., $3 \times 3 \times 1024$ convolution $\implies$ $r = 9$ $\implies$ polynomial-time

ReLU Networks with Batch Normalization (BN)

- BN transforms a batch of data to zero mean and standard deviation one, and has two trainable parameters α, γ

$$\textbf{BN}_{\alpha,\gamma}(x) = \frac{(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)x}{\|(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)x\|_2}\gamma + \alpha$$

$$p_{\text{non-convex}} = \min_{W_1, W_2, \alpha, \gamma} \left\| \mathbf{BN}_{\alpha,\gamma}(\phi(XW_1))W_2 - y \right\|_2^2 + \lambda \left(\|W_1\|_F^2 + \|W_2\|_F^2 \right)$$

$$\|$$

$$p_{\text{convex}} = \min_{w_1, v_1 \ldots w_p, v_p \in \mathcal{K}} \left\| \sum_{i=1}^{p} U_i(w_i - v_i) - y \right\|_2^2 + \lambda \left(\sum_{i=1}^{p} \|w_i\|_2 + \|v_i\|_2 \right)$$

where $U_i \Sigma_i V_i^T = D_i X$ is the SVD of DX_i, i.e., BatchNorm whitens local data

T. Ergen, A. Sahiner, B. Ozturkler, J. Pauly, M. Mardani, M. Pilanci
Demystifying Batch Normalization in ReLU Networks, ICLR 2022

Vector Output Two-layer ReLU: equivalent to nuclear norm penalty

$$p_{\text{non-convex}} = \min_{W_1 \in \mathbb{R}^{d\times m}, W_2 \in \mathbb{R}^{m\times c}} \left\| \sum_{j=1}^{m} \phi(XW_{1j})W_{2j} - Y \right\|_2^2 + \lambda \left(\|W_1\|_F^2 + \|W_2\|_F^2 \right)$$

$$p_{\text{convex}} = \min_{U_1, V_1 \ldots U_p, V_p \in \mathcal{K}} \left\| \sum_{i=1}^{p} D_i X (U_i - V_i) - y \right\|_2^2 + \lambda \left(\sum_{i=1}^{p} \|U_i\|_* + \|V_i\|_* \right)$$

$D_1, ..., D_p$ are fixed diagonal matrices

Theorem $p_{\text{non-convex}} = p_{\text{convex}}$, and an optimal solution to $p_{\text{non-convex}}$ can be recovered from optimal non-zero U_i^*, V_i^*, $i = 1, ..., p$.

A. Sahiner, T. Ergen, J. Pauly, M. Pilanci **Vector-output ReLU Neural Network Problems are Copositive Programs, ICLR 2021**

Three layer NN: FC-Relu-FC-Relu-FC is equivalent to a convex program with double hyperplane arrangements

$$p_3^* = \min_{\substack{\{W_j, u_j, w_{1j}, w_{2j}\}_{j=1}^m \\ u_j \in \mathcal{B}_2, \forall j}} \frac{1}{2} \left\| \sum_{j=1}^m \left((\mathbf{X} W_j)_+ w_{1j}\right)_+ w_{2j} - y \right\|_2^2 + \frac{\beta}{2} \sum_{j=1}^m \left(\|W_j\|_F^2 + \|w_{1j}\|_2^2 + w_{2j}^2 \right),$$

Theorem

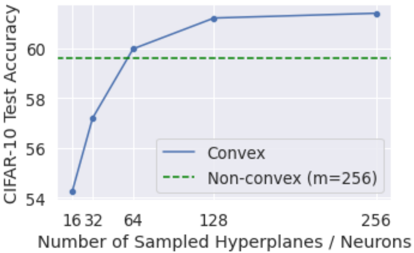
The equivalent convex problem is

$$\min_{\{W_i, W_i'\}_{i=1}^p \in \mathcal{K}} \frac{1}{2} \left\| \sum_{i=1}^p \sum_{j=1}^P D_i D_j \tilde{\mathbf{X}} \left(W_{ij}' - W_{ij}\right) - y \right\|_2^2 + \frac{\beta}{2} \sum_{i,j=1}^p \|W_{ij}\|_F + \|W_{ij}'\|_F$$

Reducing Complexity: Approximating Convex Programs by Sampling

$$\tilde{p}_{\text{sampled-cvx}} = \text{minimize}_{u_1, v_1 \ldots u_{\tilde{p}}, v_{\tilde{p}} \in \mathcal{K}} \left\| \sum_{i=1}^{\tilde{p}} D_i X(u_i - v_i) - y \right\|_2^2 + \lambda \left(\sum_{i=1}^{\tilde{p}} \|u_i\|_2 + \|v_i\|_2 \right)$$

- sampled convex model: sample $D_1, \ldots, D_{\tilde{p}}$ as $\text{Diag}(Xu \geq 0)$ where $u \sim N(0, I)$
- guarantee for two-layer ReLU NNs: $(1 + \frac{\sigma_{k+1}(X)}{\lambda})$ relative objective value approximation using $O\left((\frac{n}{k})^k\right)$ samples



All stationary points correspond to sampled convex models

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

Theorem Stationary points $\left\{x \ : 0 \in \operatorname{conv}\left\{\lim_{k\to\infty} \nabla f(x_k) \mid \lim_{k\to\infty} x_k = x,\, x_k \in D\right\}\right\}$ of $p_{\text{non-convex}}$ are optimal solutions of the sampled convex program $p_{\text{sampled-cvx}}$

Y. Wang, J. Lacotte, M. Pilanci. **The Hidden Convex Optimization Landscape of Two-Layer ReLU Neural Networks: an Exact Characterization of the Optimal Solutions ICLR, 2022**

Exact Convex Program: Two-Layer ReLU NN

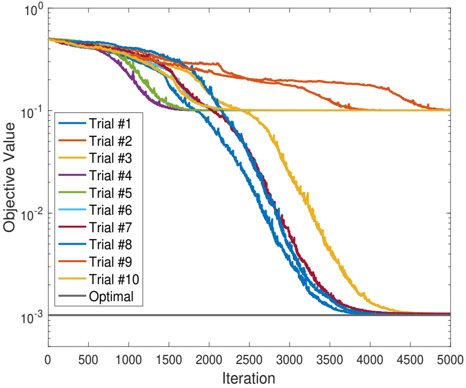


Figure: $m = 8$

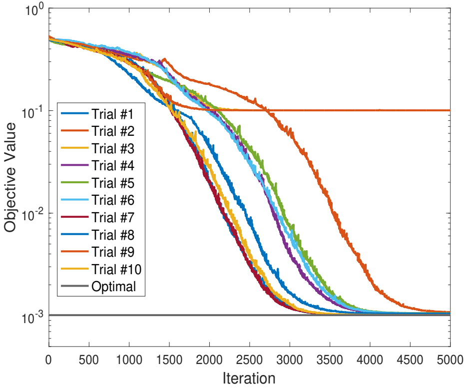
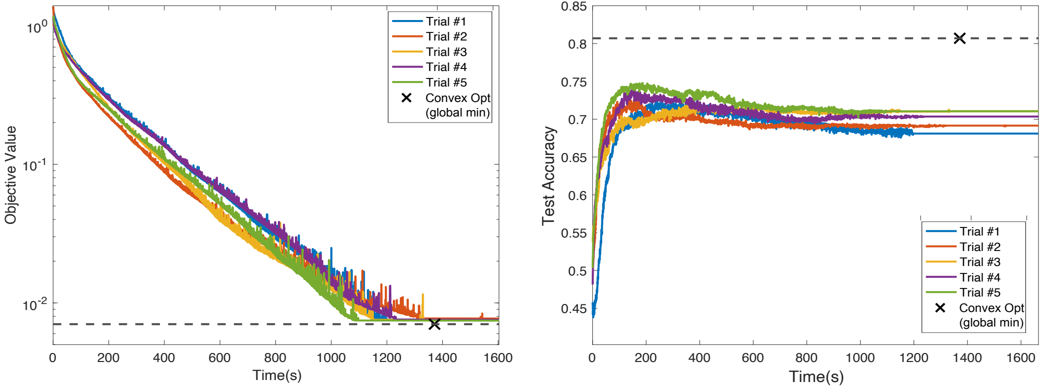


Figure: $m = 15$

Training cost of a two-layer ReLU network trained with SGD (10 initialization trials) and the convex program on a toy dataset ($d = 2$)

Exact Convex Program: Classifying a subset of CIFAR-10



(a) $m = 8$ **(b)** $m = 50$

Figure: Two-layer ReLU network trained with SGD (10 initialization trials) and the convex program on a subset of CIFAR-10 for binary classification $(n = 195)$

Sampled Convex Model vs Non-convex Model (Stochastic Gradient Descent)

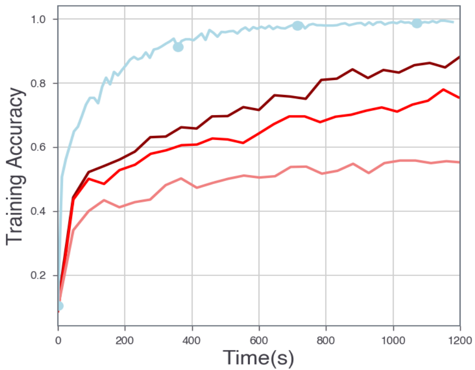


Figure: training accuracy

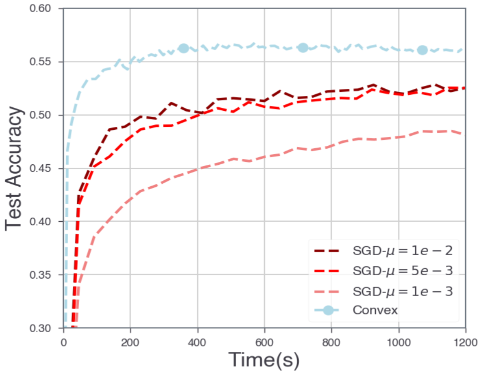
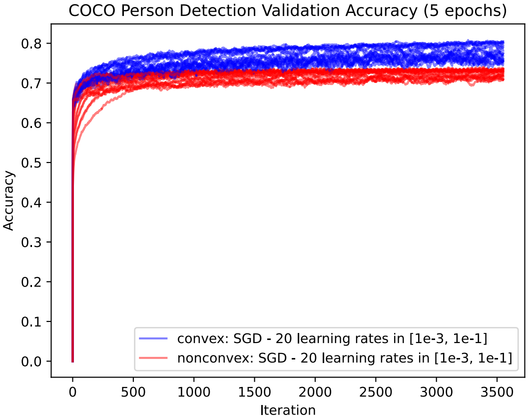


Figure: test accuracy

10-class classification on the CIFAR Dataset ($n = 50,000$, $d = 3072$) with randomly sampled arrangement patterns for the convex program

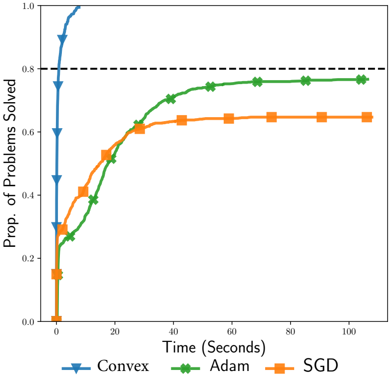
Re-training Final Convolutional Layers of Pretrained Deep Nets



Person detection task on the COCO Dataset containing $110,000$ images of median resolution 640 x 480. Two-layer ReLU CNN trained on pretrained MobileNetV3 features (convex PyTorch model: `https://github.com/pilancilab/convex_nn`)

- **baseline:** gradient based non-convex optimization: SGD, ADAM (best of 10 random initializations and 10 learning rates)
- **convex:** proximal gradient with adaptive acceleration

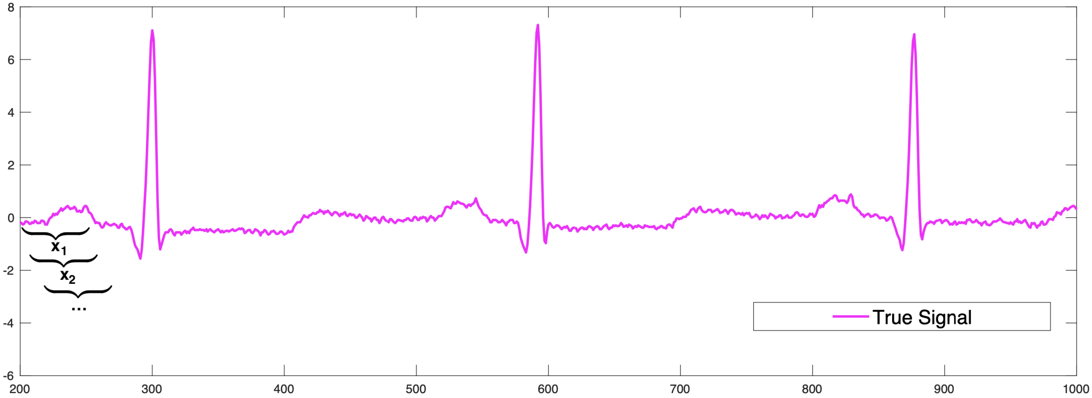
  $O(1/T^2)$ convergence rate



Performance profile showing the percentage of problems solved over a collection of 400 UC Irvine datasets up to 10^{-3} training error vs time [4]

[4] A. Mishkin, A. Sahiner, M. Pilanci. **Fast Convex Optimization for Two-Layer ReLU Networks, ICML 2022**. `github.com/pilancilab/scnn`

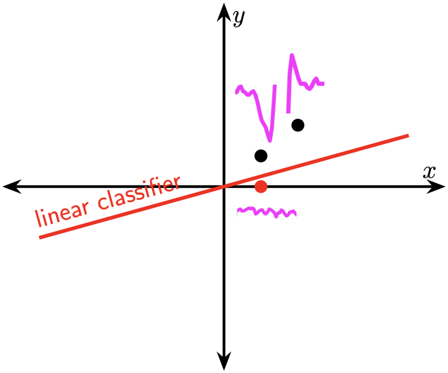
Interpreting Neural Networks via Convexity: Time Series Prediction



$$X = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix} = \begin{bmatrix} x[1] & \dots & x[d] \\ x[2] & \dots & x[d+1] \\ \vdots & & \\ x[n] & \dots & x[d+n-1] \end{bmatrix}, \quad y = \begin{bmatrix} x[d+1] \\ x[d+2] \\ \vdots \\ x[d+n] \end{bmatrix}$$

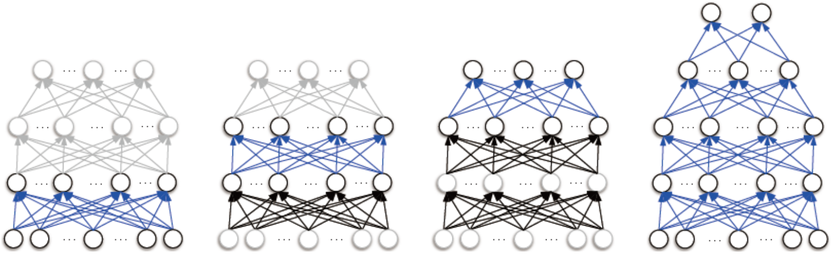
Interpreting Neural Networks via Convexity: Time Series Prediction

$$p_{\text{convex}} = \text{minimize}_{u_1, v_1 \dots u_p, v_p \in \mathcal{K}} \left\| \sum_{i=1}^{p} D_i X(u_i - v_i) - y \right\|_2^2 + \lambda \left(\sum_{i=1}^{p} \|u_i\|_2 + \|v_i\|_2 \right)$$



- sampled convex program: $D_i = \text{diag}(Xu_i \geq 0)$, $u_i \sim \mathcal{N}(0, I)$ forms a Locality Sensitive Hash (Charikar, 2002)

Layer-Wise Training of Deep Networks



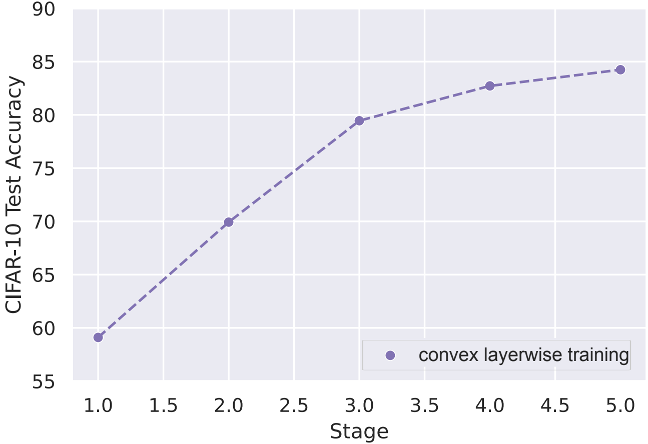
(i) train a two-layer network convex optimization

(ii) fix the hidden layer to use as feature embedding

(ii) repeat two-layer network training on these features

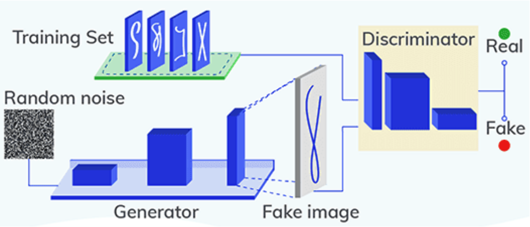
- ideal for **edge AI:** low memory and low communication between blocks
- modular: networks can keep evolving, can terminate early during inference
- each convex model is trained to global optimality efficiently with no hyperparameter tuning

Numerical results for layer-wise convex learning: CIFAR-10 image classification



- end-to-end trained 5 layer CNN accuracy: 89%, 16 layer VGG accuracy: 92%

Convex Generative Adversarial Networks (GANs)



- Wasserstein GAN parameterized with neural networks

$$p^* = \min_{\theta_g} \max_{D:\,\text{1-Lipschitz}} \mathbb{E}_{x\sim p_x}[D(x)] - \mathbb{E}_{z\sim p_z}[D(G_{\theta_g}(z))]$$

$$\cong \min_{\theta_g} \max_{\theta_d} \mathbb{E}_{x\sim p_x}[D_{\theta_d}(x)] - \mathbb{E}_{z\sim p_z}[D_{\theta_d}(G_{\theta_g}(z))]$$

Theorem: Two-layer generator two-layer discriminator WGAN problems are convex-concave games. Saddle-points exists and globally solvable under convex parameterization. (Sahiner et al. **Hidden Convexity of Wasserstein GANs, ICLR 2022.)**

Conclusion and Open Problems

- neural networks are high-dimensional convex models. Convex optimization theory & solvers can be applied.
- we can have better specialized solvers (e.g., accelerated proximal gradient)
- Extensions: autoencoders, transformers, diffusion models
- Open problems: improving the sampler

Ref 1 M. Pilanci, T. Ergen, Neural Networks are Convex Regularizers: Exact Polynomial-time Convex Optimization Formulations for Two-Layer Networks. ICML 2020

Ref 2 T. Ergen, M. Pilanci, Convex Geometry and Duality of Over-parameterized Neural Networks. JMLR 2021

- two-layer ReLU-activation generator $G_{\theta_g}(Z) = (ZW_1)_+ W_2$
- two-layer quadratic activation discriminator $D_{\theta_d}(X) = (XV_1)^2 V_2$

  Wasserstein GAN problem is equivalent to a convex-concave game, which can be solved via convex optimization

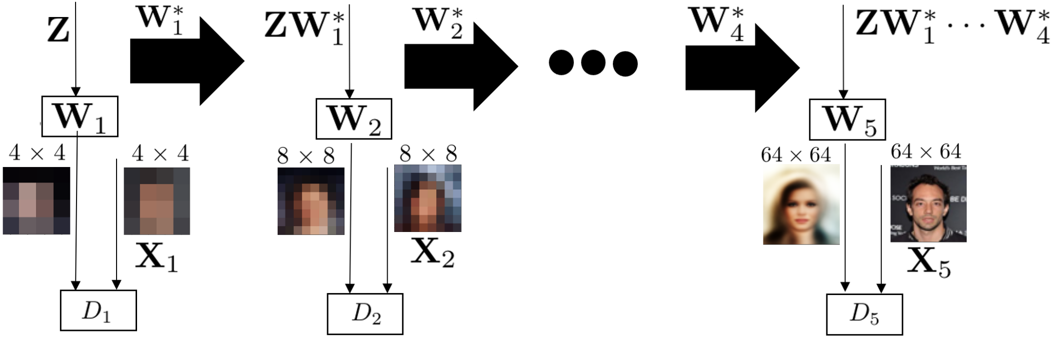
$$G^* = \mathrm{argmin}_G \|G\|_F^2 \text{ s.t. } \|X^\top X - G^\top G\|_2 \leq \lambda$$

$$W_1^*, W_2^* = \mathrm{argmin}_{W_1, W_2} \|W_1\|_F^2 + \|W_2\|_F^2 \text{ s.t. } G^* = (ZW_1)_+ W_2,$$

- the first problem can be solved via singular value thresholding as $G^* = U(\Sigma^2 - \lambda I)_+^{1/2} V^\top$ where $X = U\Sigma V^\top$ is the SVD of X.
- the second problem can be solved via convex optimization as shown earlier

Progressive GANs

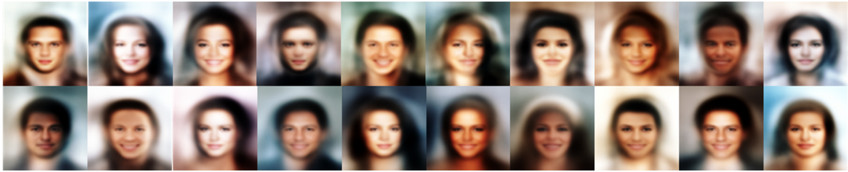
deeper architectures can be trained layerwise



Numerical Results

- real faces from the CelebA dataset

- fake faces generated using convex optimization



two-layer quadratic activation discriminator and linear generator trained via closed form optimal solution progressively for a total of 4 layers

A. Sahiner et al. **Hidden Convexity of Wasserstein GANs, preprint 2021**

Transformer and Attention-based Architectures

- based on the attention module

$$f(X) = \sigma(XQ^T KX)XV$$

- Q, K, V are trainable parameters: Q : query, K : key, V : value
- used in transformers, vision transformers, mixer models...
- **There is a convex formulation**[1]

[1] A. Sahiner, T. Ergen, B. Ozturkler, M. Mardani, J. Pauly, M. Pilanci, ICML 2022

Transfer Learning

- transfer learning *without fine-tuning existing weights of the backbone network*
- generate embeddings from an ImageNet pre-trained deep transformer model
- then finetune by training a two-layer attention block using convex optimization to classify images from CIFAR-100, while leaving the pre-trained backbone fixed

Transfer Learning

- transfer learning *without fine-tuning existing weights of the backbone network*
- generate embeddings from an ImageNet pre-trained deep transformer model
- then finetune by training a two-layer attention block using convex optimization to classify images from CIFAR-100, while leaving the pre-trained backbone fixed
- **unified architecture:** can be applied to any data (text, images, time series, tabular data, multimodal data...)
- **ideal for fine-tuning edge devices**

Two layer CNN with pooling: Conv-Pooling-Relu-FC is equivalent to ℓ_1 penalty, i.e., constrained Lasso $\min_{w\in\mathcal{K}} \|\Phi w - y\|_2^2 + \lambda\|w\|_1$

$$p_2^* = \min_{\substack{\{u_j, w_{1j}, w_{2j}\}_{j=1}^m \\ u_j \in \mathcal{B}_2, \forall j}} \frac{1}{2}\left\|\sum_{j=1}^m (\mathbf{X}U_j w_{1j})_+ w_{2j} - y\right\|_2^2 + \frac{\beta}{2}\sum_{j=1}^m \left(\|w_{1j}\|_2^2 + w_{2j}^2\right),$$

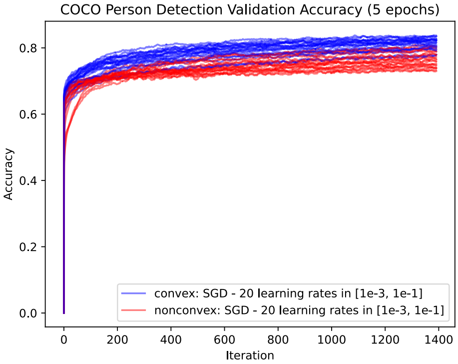
Theorem

Let $\tilde{\mathbf{X}} = \mathbf{X}F$ and $F \in \mathbb{C}^{d\times d}$ be the DFT matrix. The equivalent convex problem is

$$\min_{\substack{\{w_i, w_i'\}_{i=1}^p \\ w_i, w_i' \in \mathbb{C}^d, \forall i}} \frac{1}{2}\left\|\sum_{i=1}^p \mathit{diag}(S_i)\tilde{\mathbf{X}}\left(w_i' - w_i\right) - y\right\|_2^2 + \frac{\beta}{\sqrt{d}}\sum_{i=1}^p \left(\|w_i\|_1 + \|w_i'\|_1\right)$$

$$\text{s.t. } (2\mathit{diag}(S_i) - I_n)\tilde{\mathbf{X}}w_i \geq 0,\ (2\mathit{diag}(S_i) - I_n)\tilde{\mathbf{X}}w_i' \geq 0, \forall i,$$

Sampled Convex Model vs Non-convex Model for fine-tuning



Person detection task on the Common Objects in Context Dataset ($110,000$ images of median resolution 640 x 480).

Fine-tuning all layers of MobileNetV3 $+$ convex and non-convex CNN head