Optimization for Robust Deep Learning

M. Pawan Kumar

University of Oxford

Joint work with Rudy Bunel, Alban Desmaison, Krishnamurthy Dvijotham, Pushmeet Kohli and Philip Torr
ImageNet Challenge

~1 M images, 1000 classes
Deep Learning

Linear functions followed by piecewise linear non-linearities
ImageNet Classification Error (Top 5)

- 2011 (XRCE): 26.0
- 2012 (AlexNet): 16.4
- 2013 (ZF): 11.7
- 2014 (VGG): 7.3
- 2014 (GoogLeNet): 6.7
- Human: 5.0
- 2015 (ResNet): 3.6
- 2016 (GoogLeNet-v4): 3.1
Human drivers replaced by deep neural networks
Deep Learning

Road sign recognition

Parameters $W$ perform multiclass classification

Estimate $W$ using training data
Parameters $W$ select the class of a new input image
Deep Learning

Small deformations can cause fatal errors

Blurring, Saturation  Aung et al., 2017

Pixel Errors  Evtimov et al., 2017
Spot the Difference
Spot the Difference

“pig” (91%)

“airliner” (99%)
0.005 x
3D Object

classified as turtle  classified as rifle  classified as other

Athalye, Engstrom, Ilyas and Kwok, 2017
Audio

Carlini and Wagner, 2018

"it was the best of times, it was the worst of times"

"it is a truth universally acknowledged that a single"
Robust Deep Learning

Training Data

Parameters $W$

SVM
Robust Deep Learning

Training Data → Augment Training Data Set → Identify Deformation With Error

Parameters $W$
Robust Deep Learning

Is there an erroneous output?

Image Deformations

Condition on inputs

Safe

Error

Classifications
Robust Deep Learning

Is there an erroneous output?

Image Deformations

Condition on inputs

Re-estimate parameters $W$

Safe

Error

Classifications
Robust Deep Learning

Is there an erroneous output?

Image Deformations

Condition on inputs

Re-estimate parameters $W$

Safe

Error

Classifications
Robust Deep Learning

Is there an erroneous output?

Condition on inputs

Image Deformations

Classifications

Re-estimate parameters $W$
Robust Deep Learning

Is there an erroneous output?

Image Deformations

Terminate when all putative outputs are safe
Robust Deep Learning

Is there an erroneous output?

Terminate when all putative outputs are safe
Outline

• Neural Network Bounds

• Lagrangian Decomposition
• Proximal Minimization

Method of Multipliers

• Results
Robust Deep Learning

Is there an erroneous output?

Non-convexity makes the problem NP-hard
Robust Deep Learning

Is there an erroneous output?

Replace by a convex superset
Find the minimum valid output

\[
\begin{align*}
\text{Find the minimum valid output} & \\
\min & \quad y \\
\text{s.t.} & \quad -2 \leq x_1 \leq 2 \\
& \quad -2 \leq x_2 \leq 2 \\
& \quad a_{in} = x_1 + x_2 \\
& \quad b_{in} = x_1 - x_2 \\
& \quad a_{out} = \max\{a_{in}, 0\} \\
& \quad b_{out} = \max\{b_{in}, 0\} \\
& \quad y = -a_{out} - b_{out}
\end{align*}
\]
Example

**Linear constraints**

\[
\begin{align*}
\text{s.t.} & \quad -2 \leq x_1 \leq 2 \\
& \quad -2 \leq x_2 \leq 2 \\
& \quad a_{\text{in}} = x_1 + x_2 \\
& \quad b_{\text{in}} = x_1 - x_2 \\
& \quad a_{\text{out}} = \max\{a_{\text{in}}, 0\} \\
& \quad b_{\text{out}} = \max\{b_{\text{in}}, 0\} \\
& \quad y = -a_{\text{out}} - b_{\text{out}}
\end{align*}
\]

Easy to handle
Example

\[
\begin{align*}
\text{min} & \quad y \\
\text{s.t.} & \quad -2 \leq x_1 \leq 2 \\
& \quad -2 \leq x_2 \leq 2 \\
& \quad a_{\text{in}} = x_1 + x_2 \\
& \quad b_{\text{in}} = x_1 - x_2 \\
& \quad a_{\text{out}} = \max\{a_{\text{in}},0\} \\
& \quad b_{\text{out}} = \max\{b_{\text{in}},0\} \\
\end{align*}
\]

Non-linear constraints

\[
y = - a_{\text{out}} - b_{\text{out}}
\]
Post ReLU

\[ a_{out} = \max\{a_{in}, 0\} \quad a_{in} \in [l, u] \]

Set of feasible \((a_{in}, a_{out})\) is not convex
\[ a_{\text{out}} = \max\{a_{\text{in}}, 0\} \quad a_{\text{in}} \in [l,u] \]

Replace with smallest convex superset

Ehlers 2017
min \ y

\text{s.t.} \quad -2 \leq x_1 \leq 2

\quad -2 \leq x_2 \leq 2

a_{in} = x_1 + x_2

b_{in} = x_1 - x_2

a_{out} = \max\{a_{in}, 0\}

b_{out} = \max\{b_{in}, 0\}

y = -a_{out} - b_{out}
Convex Relaxation

Linear Program

\[ \begin{align*}
\text{min} & \quad y \\
\text{s.t.} & \quad -2 \leq x_1 \leq 2 \\
& \quad -2 \leq x_2 \leq 2 \\
& \quad a_{\text{in}} = x_1 + x_2 \\
& \quad b_{\text{in}} = x_1 - x_2 \\
& \quad a_{\text{out}} \geq 0, \quad a_{\text{out}} \geq a_{\text{in}}, \quad a_{\text{out}} \leq 0.5a_{\text{in}} + 2 \\
& \quad b_{\text{out}} \geq 0, \quad b_{\text{out}} \geq b_{\text{in}}, \quad b_{\text{out}} \leq 0.5b_{\text{in}} + 2 \\
& \quad y = -a_{\text{out}} - b_{\text{out}}
\end{align*} \]
General Formulation

\[ \min_x \quad c^T x \]

s.t. \[ A_k(x) \quad k = 1, \ldots, L \]

\[ B_k(x) \quad k = 1, \ldots, L \]

Linear mapping of the k-th layer ("easy" constraints)
General Formulation

$$\min_x \ c^T x$$

s.t. $$A_k(x) \quad k = 1, \ldots, L$$

$$B_k(x) \quad k = 1, \ldots, L$$

Linear mapping of the k-th layer ("easy" constraints)

Convex hull of activations (relaxations)
Outline

• Neural Network Bounds

• **Lagrangian Decomposition**

• Proximal Minimization

• Results
Additional Variables

\[
\min_{x, x_k, z_k} \quad c^T x
\]

s.t. \quad A_k(x) \quad k = 1, \ldots, L

\quad B_k(x) \quad k = 1, \ldots, L
\[
\min_{x, x_k, z_k} \quad c^T x \\
\text{s.t.} \quad A_k(x_k) \quad k = 1, \ldots, L \\
B_k(z_k) \quad k = 1, \ldots, L \\
x_k = x \quad k = 1, \ldots, L \\
z_k = x \quad k = 1, \ldots, L
\]
Partial Lagrangian

\[
\begin{align*}
\max_{\lambda, \mu} \quad & \min_{x, x_k, z_k} \quad c^T x + \sum_k \lambda_k^T (x_k - x) + \sum_k \mu_k^T (z_k - x) \\
\text{s.t.} \quad & A_k(x_k) \quad k = 1, \ldots, L \\
& B_k(z_k) \quad k = 1, \ldots, L
\end{align*}
\]

Eliminating \( x \) using KKT conditions
Lagrangian Decomposition

\[
\begin{align*}
\max_{\lambda, \mu} \quad & \min_{x_k, z_k} \quad \sum_k \lambda_k^T x_k + \sum_k \mu_k^T z_k \\
\text{s.t.} \quad & A_k(x_k) \quad k = 1, \ldots, L \\
& B_k(z_k) \quad k = 1, \ldots, L \\
& \sum_k \lambda_k + \sum_k \mu_k = c
\end{align*}
\]

Projected subgradient descent (Dvijotham et al., 2018)

Tuning step-size schedule is hard
Outline

• Neural Network Bounds

• Lagrangian Decomposition

• Proximal Minimization

• Results
Proximal Minimization

\[ \max_{\lambda, \mu} \min_{x_k, z_k} \quad \sum_k \lambda_k^T x_k + \sum_k \mu_k^T z_k \]
\[ + \eta \| \lambda - \lambda' \|^2 + \eta \| \mu - \mu' \|^2 \]

s.t. \[ A_k(x_k) \quad k = 1, \ldots, L \]
\[ B_k(z_k) \quad k = 1, \ldots, L \]
\[ \sum_k \lambda_k + \sum_k \mu_k = c \]

Initialize \( \lambda = \lambda' \) and \( \mu = \mu' \) \quad Solve proximal problem
Proximal Minimization

\[ \max_\lambda, \mu \quad \min_{x_k, z_k} \quad \sum_k \lambda_k^T x_k + \sum_k \mu_k^T z_k \]

\[ + \eta \| \lambda - \lambda' \|^2 + \eta \| \mu - \mu' \|^2 \]

s.t.
\[ A_k(x_k) \quad k = 1, \ldots, L \]
\[ B_k(z_k) \quad k = 1, \ldots, L \]

\[ \sum_k \lambda_k + \sum_k \mu_k = c \]

Smooth quadratic dual    Conditional gradient

Repeatedly solve proximal problems until convergence
Advantages

Conditional gradient of dual is the subgradient
Bach, 2015

Analytically computable optimal step-size

A single easy-to-tune parameter $\eta$

Highly parallelizable over GPUs
Outline

• Neural Network Bounds

• Lagrangian Decomposition

• Proximal Minimization

• Results
<table>
<thead>
<tr>
<th>Layer Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv2D(channels=16, kernel_size=4, stride=2)</td>
</tr>
<tr>
<td>ReLU</td>
</tr>
<tr>
<td>Conv2D(channels=32, kernel_size=4, stride=2)</td>
</tr>
<tr>
<td>ReLU</td>
</tr>
<tr>
<td>Linear(channels=100)</td>
</tr>
<tr>
<td>ReLU</td>
</tr>
<tr>
<td>Linear(channels=10)</td>
</tr>
</tbody>
</table>
Results

The graph shows the performance of different methods over time. The x-axis represents time, and the y-axis shows the value. Different methods are represented by distinct line types and colors. The legend provides details about the methods used:

- **Subgradient**: Various initial step sizes (3e-2, 1e-2, 3e-3, 1e-3, 3e-4, 1e-4).
- **Proximal**: Various initial eta values (1e4, 3e3, 1e3, 3e2, 1e2).

The line styles indicate the method used in the following order:

- **Proximal**: Black solid line.
- **Subgradient**: Dotted line.

The graph illustrates how each method converges to a value over time.
Results

Timing (in s)

Subgradient 1000 steps vs. Proximal 500 steps

Bound

Subgradient 1000 steps vs. Proximal 500 steps
Results

Time

10^3
10^1
10^{-1}

Gap to best bound

10^2
10^1
10^0

KW  Gurobi  Subgradient 100 steps  Proximal 100 steps  Subgradient 500 steps  Proximal 500 steps  Subgradient 1000 steps  Proximal 1000 steps
Questions?

Paper + Code + Data Available*

http://www.robots.ox.ac.uk/~oval