CONTEXT

• Adversarial attacks of Neural Networks

• Local Lipschitz robustness: \[ \|f(x + \varepsilon) - f(x)\| \leq k \|\varepsilon\| \]

• Building 1-Lipschitz network

• Classification with Lipschitz Neural Networks
• Adversarial attacks
ADVERSARIAL ATTACK

- Adversarial example: closest example with the opposite class

$$\text{adv}(f, x) = \arg\min_{z \in \Omega} \| x - z \| \quad \text{subject to} \quad \text{sign}(f(z)) = -\text{sign}(f(x))$$

- Why neural networks are structurally weak to attack?
LIPSCHITZ CONSTANT

• $f: E \rightarrow F$ is $k$-Lipschitz iif:

$$\| f(x_1) - f(x_2) \| \leq k \| x_1 - x_2 \|$$

• Lipschitz constant: smallest value of $k$
  • 1D case: $k = \max f'(x)$
  • Intuition: how much the output of the function $f$ may vary when I change the input
LIPSCHITZ CONSTANT NEURAL NETWORK

• Very hard to evaluate accurately (np-hard)
• Multilayer perceptron

\[ f(x) = \phi_k(W_k \cdot (\phi_{k-1}(W_{k-1} \ldots \phi_1(W_1 \cdot x)))) \]

• Lipschitz constant upper-bound

\[ L(f) \leq L(\phi_k) \times L(W_k) \times L(\phi_{k-1}) \times L(W_{k-1}) \times \ldots \times L(\phi_1) \times L(W_1 \cdot x) \]

• Activation function are usualy 1-Lischitz
• If all linear layer are 1-Lischitz, the network is 1-Lischitz (but the constant can be smaller than one)
WHY ARE NN WEAK TO ATTACK?

• Maximizing cross entropy => increase the Lipschitz constant

• Lipschitz constant of NN grows exponentially

• High Lipschitz constant: small variation of inputs leads to high variation of the output -> principle of adversarial attack

• Illustration
• Building 1-Lipschitz neural networks
1-LIPSCHITZ NEURAL NETWORKS

• Principles: all the layers have to be 1-lipschitz
• Dense Layer with weights $W$

\[ L(W) = \|W\| \leq \|W\|_F \leq \max_{ij} \|W_{ij}\| \ast \sqrt{nm} \]

• Constraining Lipschitz constant
  • WGAN: weight clipping (last term of the equation)
  • Weight normalization with Frobenius norm $\|W\|_F$
  • Spectral normalization with spectral norm $\|W\|$

• Convolutional layers: same approach
1-LIPSCHITZ NEURAL NETWORKS

- Pooling layer: use L2 norm pooling
- Activation function:
  - Max min
  - Group sort
  - Full-sort
- BatchNormalization: Not lipschitz
- Dropout: Not Lipschitz
DEEL LIP LAYER LIBRARY

- Keras/tensflow and pythorch implementation
- Open source
- Keras extension
  - k-lischitz layers
  - activation functions
  - weight initializers
  - monitoring tools
- layer exportation (optimization for inference)
Standard neural network

```python
from tf.keras.layers import (Conv2D, Dense, ReLU, BatchNormalization, MaxPool2D, Flatten)

model = tf.keras.Sequential(
    Conv2D(32, 3),
    BatchNormalization(),
    ReLU(),
    Conv2D(64, 3),
    MaxPool2D(),
    ReLU(),
    Flatten(),
    Dense(10)
)
```

Lipschitz neural network

```python
from tf.keras.layers import Flatten
from deeplip.layers import (SpectralDense, SpectralConv2D, ScaledL2NormPooling2D)
from deeplip.activations import GroupSort2

model = deeplip.model.Sequential(
    SpectralConv2D(32, 3),
    # No batch normalization
    GroupSort2(),
    SpectralConv2D(64, 3),
    ScaledL2NormPooling2D(),
    GroupSort2(),
    Flatten(),
    SpectralDense(10)
)
```
Installation

You can install `deel-lip` directly from pypi:

```
pip install deel-lip
```

In order to use `deel-lip`, you also need a valid tensorflow installation. `deel-lip` supports tensorflow 2.0 and tensorflow 2.1.

Cite this work

This library has been built to support the work presented in the paper *Achieving robustness in classification using optimaltransport with Hinge regularization*.

This work can be cited as:

```
@misc{2006.06520,
    Author = {Mathieu Serrurier and Franck Mamalet and Alberto González-Sanz and Thibaut Boissin and Jean-M-},
    Title = {Achieving robustness in classification using optimal transport with hinge regularization},
    Year = {2020},
    Eprint = {arXiv:2006.06520},
}
```

## MODEL EXPORTATION

- This is done with `deel.model.Model` class
  - extends the original keras Model class
  - allows to call `model2 = model.vanilla_export()`
• Training 1-Lipshitz network
1-LIPSCHITZ TRAINING LOSS

• Problem: cross entropy not suitable for 1-Lipschitz networks

• Optimal transport loss:

$$\inf_{f \in Lip_1(\Omega)} \mathbb{E}_{x \sim P_-}[f(x)] - \mathbb{E}_{x \sim P_+}[f(x)] + \lambda \mathbb{E}_x (1 - Y f(x))_+$$

• Class = sign of f
PROPERTIES OF PROPOSED SOLUTION

• Link machine learning with optimal transportation

• Provable robustness

\[ \| x - adv(\hat{f}, x) \| \geq |\hat{f}(x)| \]

• In practice

\[ \| x - adv(\hat{f}, x) \| \approx |\hat{f}(x)| \]

• Structurally Robust to attack

• Interpretable: Adversarial examples show explicitly what you have to change to change class
• Experimentations
MNIST 0-8 ADVERSARAL ATTACKS

• Qualitative comparison of adversarial examples (50/50 score)

MLP with binary cross-entropy

1-Lipschitz MLP with binary cross-entropy

1-Lipschitz MLP with regularized OT
CELEB-A MOUSTACHE ADVERSARIAL ATTACKS

• Qualitative comparison of adversarial examples (50/50 score)

Lipschitz VGG with regularized OT

Without moustache In-between with or without moustache

(a) Fooling CelebA images classical network

(b) Fooling images 1-lipschitz network (binary crossentropy)
CELEB-A GLASS ADVERSARIAL ATTACKS

• Qualitative comparison of adversarial examples

Lipschitz VGG with regularized OT

• Raw

• Classical approaches

• Our approach
CONCLUSIONS

• Training 1-Lipschitz networks with optimal transport loss
  • New interpretation of classification problem
  • Improve robustness structurally
  • Meet certification requirement
  • Interpretable

• Deep lip : accessible and optimized library to train and use 1-Lipschitz networks

• Future works
  • Outlier detection
  • Semi-supervised/agnostic learning