

Masterarbeit

Lukas Schulth
`lukas.schulth@uni.kn`

20. April 2021

Zusammenfassung

abstract

Keywords— one, two, three, four

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1	Implementierte Regeln fhvilshoj	6
2	Kleines Netzwerk	9

Inhaltsverzeichnis

1 Einführung	5
1.1 Neuronale Netzwerke	5
1.1.1 CNNS	5
1.2 Poisoning-Angriffe	5
2 Erklärbare KI	5
2.1 Lokale Methoden	5
2.2 Globale Methoden	5
3 Layer-wise Relevance Propagation	5
3.1 Idee	5
3.2 Deep Taylor Decomposition	5
3.3 Verschiedene Verfahren	5
3.4 LRP als DTD	5
3.5 Eigenschaften	6
3.6 Implementierung	6
3.6.1 Tensorflow	6
3.6.2 pytorch	6
4 Detektion von Poisoning-Angriffen basierend auf LRP	7
4.1 Idee	7
4.2 Verwendete Distanzen	7
4.2.1 Euklidische Distanz	7
4.2.2 Gromov-Wasserstein-Distanz	7
4.3 Anwendung auf unterschiedliche Poisoning-Angriffe	7
4.3.1 Standard Poisoning-Angriffe	7
4.3.2 Label-konsistente Poisoning-Angriffe	7
5 Vergleich mit anderen Verfahren	7
5.1 Activation Clustering	7
5.2 Räumliche Transformationen	7
6 Weitere mögliche Schritte	8
A Verwendete Netzwerke	9
A.1 Net	9
B Parameter für Training und Einlesen der Daten	9
C Datensätze	10
D Programmcode	10

1 Einführung

A Complete List of All (arXiv) Adversarial Example Papers ¹

In sicherheitskritischen Anwendungsgebieten ist die Interpretation einer Entscheidung genauso wichtig wie die Entscheidung selbst[1].

1.1 Neuronale Netzwerke

1.1.1 CNNs

Idee, Abstraktion, high level, low level features, bekannte Netzwerke

1.2 Poisoning-Angriffe

2 Erklärbare KI

2.1 Lokale Methoden

2.2 Globale Methoden

3 Layer-wise Relevance Propagation

3.1 Idee

In [4] wird die Layer-wise Relevance Propagation erstmalig vorgestellt. Zudem wird eine Taylor Zerlegung präsentiert, die eine Approximation der LRP darstellt.

3.2 Deep Taylor Decomposition

Laut [3] ist die in [4] vorgestellte Layer-wise Relevance Propagation eher heuristisch. In diesem Paper wird nun eine solide theoretische Grundlage geliefert.

LRP in verschiedenen Anwendungsgebieten [9], 10.2. In diesem Paper: LRP-0 schlechter als LRP- ϵ schlechter als LRP- γ schlechter als Composite-LRP.

3.3 Verschiedene Verfahren

- LRP-0
- LRP- ϵ
- LRP- γ

3.4 LRP als DTD

siehe [8], Kapitel 10.

¹<https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html>

3.5 Eigenschaften

In [1] werden die folgenden Eigenschaften für die drei verschiedenen Verfahren genannt, aber nicht wirklich untersucht?!

- Numerische Stabilität
- Konsistenz (mit Linearer Abbildung)
- Erhaltung der Relevanz

3.6 Implementierung

3.6.1 Tensorflow

3.6.2 pytorch

Allgemeines Tutorial:²

pytorch-LRP für VGG16 wird vorgestellt.

GiorgioML³:

Alternative pytorch-Implementierung basierend auf Tensorflow paper.

moboehle⁴:

Unterstützte Netzwerkschichten⁵:

allowed_pass_layers = (torch.nn.BatchNorm1d, torch.nn.BatchNorm2d, torch.nn.BatchNorm3d, torch.nn.ReLU, torch.nn.ELU, Flatten, torch.nn.Dropout, torch.nn.Dropout2d, torch.nn.Dropout3d, torch.nn.Softmax, torch.nn.LogSoftmax, torch.nn.Sigmoid)

fhvilshoj⁶:

PyTorch implementation of some of the Layer-Wise Relevance Propagation (LRP) rules, [1, 2, 3], for linear layers and convolutional layers.

The modules decorates torch.nn.Sequential, torch.nn.Linear, and torch.nn.Conv2d to be able to use autograd backprop algorithm to compute explanations.

- Ausgabe der Relevanzen von Zwischenschichten ist möglich
- : Implementierte Regeln: epsilon Regeln mit epsilon=1e-1, gamma-regel mit gamma=1e-1. alphabeta-Reagel mit alb0 und a2b1

```

1 conv2d = {
2     "gradient": F.conv2d,
3     "epsilon": Conv2DEpsilon.apply,
4     "gamma": Conv2DGamma.apply,
5     "gamma+epsilon": Conv2DGammaEpsilon.apply,
6     "alpha1beta0": Conv2DA1alpha1Beta0.apply,
7 
```

²<https://git.tu-berlin.de/gmontavon/lrp-tutorial>

³<https://giorgiomorales.github.io/Layer-wise-Relevance-Propagation-in-Pytorch/>

⁴<https://github.com/moboehle/Pytorch-LRP>

⁵https://github.com/moboehle/Pytorch-LRP/blob/master/inverter_util.py

⁶<https://github.com/fhvilshoj/TorchLRP>

```

8  "alpha2beta1":      Conv2DAlpha2Beta1.apply,
9  "patternattribution": Conv2DPatternAttribution.apply,
10 "patternnet":       Conv2DPatternNet.apply,
11 }
12

```

Listing 1: Implementierte Regeln fhvishoj

Zennit:⁷ Zennit (Zennit explains neural networks in torch) is a high-level framework in Python using PyTorch for explaining/exploring neural networks. Its design philosophy is intended to provide high customizability and integration as a standardized solution for applying LRP-based attribution methods in research.

4 Detektion von Poisoning-Angriffen basierend auf LRP

4.1 Idee

Die Idee zur Detektion von Poisoning-Angriffen besteht aus den folgenden Schritten:

- Berechnung der Heatmaps mit Hilfe der LRP
- Berechnung einer Distanzmatrix basierend auf L - oder GMW-Distanz
- Spektrale Relevanzanalyse (Bestimmung der verschiedenen Cluster innerhalb einer Klasse)

Bemerkung: Anstatt das Clustering nur auf den Heatmaps durchzuführen, könnten die LRP-Ausgaben und/oder Aktivierungen bestimmter Netzwerkschichten hinzugenommen werden.

4.2 Verwendete Distanzen

4.2.1 Euklidische Distanz

4.2.2 Gromov-Wasserstein-Distanz

4.3 Anwendung auf unterschiedliche Poisoning-Angriffe

4.3.1 Standard Poisoning-Angriffe

4.3.2 Label-konsistente Poisoning-Angriffe

5 Vergleich mit anderen Verfahren

5.1 Activation Clustering

5.2 Räumliche Transformationen

- ASR ist sehr stark vom Ort des Triggers abhängig.
- Ort des Triggers kann nicht direkt geändert werden.

⁷<https://github.com/chr5tphr/zennit>

- Benutze Transformationen(Flipping, Scaling), um den Trigger wirkungslos zu machen.
- Somit kann die ASR während der Inferenz verringert werden. Es lässt sich aber keine Aussage darüber treffen, ob ein Angriff vorliegt

6 Weitere mögliche Schritte

- Automatische Platzierung des Auslösers an fest gewählter Position auf dem Verkehrsschild anstatt zufälligem Platzierem in einem Fenster mit vorher festgelegter Größe. In [7] wird Faster-RCNN (F-RCNN) zur Klassifikation des LISA-Datensatzes⁸ benutzt. Es ist die Aufgabe, die Verkehrsschilder in die 3 Superklassen Stoppschild, Geschwindigkeitsbegrenzung und Warnschild einzuteilen. Der Datensatz enthält zudem die BoundingBoxen, sodass der Auslöser genauer angebracht werden kann.
- Verbesserte Version der Layer-wise Relevance Propagation
- Untersuchung anderer Verfahren, die die Interpretierbarkeit ermöglichen, beispielsweise: VisualBackProp: efficient visualization of CNNs⁹
- Vergleich mit Cifar-10/Cifar-100 Datensatz¹⁰¹¹

⁸<http://cvrr.ucsd.edu/LISA/lisa-traffic-sign-dataset.html>

⁹<https://arxiv.org/abs/1611.05418>

¹⁰<https://www.cs.toronto.edu/~kriz/cifar.html>

¹¹<https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>

A Verwendete Netzwerke

A.1 Net

```

1
2 class Net(nn.Module):
3
4     def __init__(self, ):
5         super(Net, self).__init__()
6         self.size = 64 * 4 * 4
7         self.conv1 = nn.Conv2d(in_channels=3, out_channels=12,
8 kernel_size=5, padding=2)
9         self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
10        self.conv1_in = nn.InstanceNorm2d(12)
11        self.conv2 = nn.Conv2d(in_channels=12, out_channels=32,
12 kernel_size=5, padding=2)
13
14        self.conv2_bn = nn.BatchNorm2d(32)
15
16        self.conv3 = nn.Conv2d(in_channels=32, out_channels=64,
17 kernel_size=5, padding=2)
18
19        self.fc1 = nn.Linear(self.size, 256)
20        self.fc1_bn = nn.BatchNorm1d(256)
21        self.fc2 = nn.Linear(256, 128)
22        self.fc3 = nn.Linear(128, 43)
23
24    def forward(self, x):
25        x = self.pool(F.relu(self.conv1_in(self.conv1(x))))
26        x = self.pool(F.relu(self.conv2_bn(self.conv2(x))))
27        x = self.pool(F.relu(self.conv3(x)))
28        x = x.view(-1, self.size)
29        x = F.relu(self.fc1_bn(self.fc1(x)))
30        x = F.dropout(x)
31        xx = F.relu(self.fc2(x))
32        x = F.dropout(xx)
33        x = self.fc3(x)
34
35        return x, xx

```

Listing 2: Kleines Netzwerk

B Parameter für Training und Einlesen der Daten

Die in [2] gewählten Parameter wären ein guter Ausgangspunkt.

C Datensätze

D Programmcode

Literatur

- [1] Layer-wise Relevance Propagation for Deep Neural Network Architectures. Alexander Binder, Sebastian Bach, Gregoire Montavon, Klaus-Robert Müller und Wojciech Samek
- [2] Finding and Removing Clever Hans: Using Explanation Methods to Debug and Improve Deep Models
- [3] Explaining nonlinear classification decisions with deep Taylor decomposition
- [4] On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation
- [5] S. Bazen, X. Joutard, The Taylor decomposition: a unified generalization of the Oaxaca method to nonlinear models, Technical Report 2013-32, Aix-Marseille University, 2013.
- [6] R. Oaxaca, Male-female wage differentials in urban labor markets, *Int. Econ. Rev.* 14 (3) (1973) 693–709
- [7] BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain
- [8] Montavon G., Binder A., Lapuschkin S., Samek W., Müller KR. (2019) Layer-Wise Relevance Propagation: An Overview. In: Samek W., Montavon G., Vedaldi A., Hansen L., Müller KR. (eds) *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Lecture Notes in Computer Science, vol 11700. Springer, Cham. https://doi.org/10.1007/978-3-030-28954-6_10
- [9] Layer-Wise Relevance Propagation: An Overview Grégoire Montavon ¹ , Alexander Binder ² , Sebastian Lapuschkin ³ , Wojciech Samek ³ , and Klaus-Robert Müller ^{1,4,5}