Universität Konstanz
Fachbereich Mathematik und Statistik
&
Bundesamt für Sicherheit in der Informationstechnik

Masterarbeit zum Thema:

Untersuchung & Entwicklung von Ansätzen zur Detektion von Poisoning-Angriffen

vorgelegt von

Lukas Schulth
lukas.schulth@uni.kn

unter der Betreuung von

Erstkorrektor:
Herr Prof. Dr. Johannes Schropp
johannes.schropp@uni.kn

Herr Dr. Christian Berghoff
christian.berghoff@bsi.bund.de

Zweitkorrektor:
Herr Prof. Dipl.-Ing. Markus Ullmann
markus.ullmann@bsi.bund.de

Herr Matthias Neu
matthias.neu@bsi.bund.de

1. Oktober 2021

Abstract

english

Zusammenfassung

deutsch

TODO:

- Algorithmen aufschreiben kmeans

Keywords— one, two, three, four

Abbildungsverzeichnis

1 (Optischer) Vergleich von korrumpiertem Datenpunkt und berechnter Heatmap. 23

Tabellenverzeichnis

1 Für einen Poisoning-Angriff interessante Klassen und die zugehörige Anzahl an Bildern. 8

Listings

1 python-interner Aufbau einer BatchConv Schicht 7
2 Verfügbare Schichten und Aktivierungsfunktionen 24
3 Implementierte Regeln fhvilshoj 24
4 Augemntierung beim Einlesen der Daten 29

Algorithmenverzeichnis

Inhaltsverzeichnis

1 Einführung **5**

2 Neuronale Netzwerke **6**
2.0.1 CNNS . . . 6
2.0.2 Besondere Schichten . . . 6
2.0.3 Inception v3 . . . 7
2.0.4 VGG16 . . . 7
2.1 Datensatz . . . 7

3 Poisoning-Angriffe **8**
3.1 Standard Poisoning-Angriffe . . . 8
3.2 Label-konsistente Poisoning-Angriffe . . . 9
3.3 Bewertung von Poisoning-Angriffen . . . 10
3.4 Verteidigungen . . . 10
3.4.1 Referenzwert: kMeans(k=2) . . . 10
3.4.2 Activation Clustering . . . 11
3.5 Implementierung . . . 11
3.5.1 Methoden zur Untersuchung, ob ein Angriff vorliegt . . . 12
3.5.2 Entfernen von korrumpierten Datenpunkten . . . 13
3.5.3 Heatmap Clustering . . . 13

4 Erklärbare KI **13**
4.1 Lokale Methoden . . . 15
4.1.1 Störungs-basierte Anästze . . . 18
4.1.2 Funktions-basierte Ansätze . . . 18
4.1.3 Surrogate-/Sampling-basierte Ansätze . . . 18
4.1.4 Struktur-basierte Ansätze . . . 18
4.2 Globale Methoden . . . 18

5 Layer-wise Relevance Propagation **19**
5.1 Idee . . . 19
5.2 Behandlung von biases . . . 22
5.3 Beispiel an einem kleinen Netzwerk . . . 22
5.4 Deep Taylor Decomposition . . . 22
5.4.1 Taylor Decomposition . . . 22
5.4.2 Deep Taylor Decomposition . . . 22
5.5 Verschiedene Verfahren . . . 22
5.6 Eigenschaften . . . 22
5.7 Behandlung besonderer Schichten . . . 23
5.7.1 BatchNorm2D . . . 23
5.8 LRP für Deep Neural Nets/Composite LRP . . . 23
5.9 Verarbeitung der Heatmaps . . . 23
5.10 Implementierungen . . . 23
5.10.1 Tensorflow . . . 23
5.10.2 pytorch . . . 23

6 Detektion von Poisoning-Angriffen basierend auf LRP **25**
6.1 Idee 26
6.2 k-means / k-means++ -Clustering 26
6.3 Spektrales Clustering 26
6.4 Anwendung auf unterschiedliche Poisoning-Angriffe 27
6.5 Verwendete Distanzen & Approximationen 28

A Verwendete Netzwerke **29**

B Parameter für Training und Einlesen der Daten **29**

C Einlesen der Daten bei AC **30**

D Parameter für die ausgeführten Angriffe **30**

E Datensätze **30**

F Programmcode **30**

G Notizen **30**

1 Einführung

Allein für Deutschland wird erwartet, dass mit Dienstleistungen und Produkten, die auf dem Einsatz von Künstlicher Intelligenz (KI) basieren, im Jahr 2025 Umsätze in Höhe von 488 Milliar den Euro generiert werden – damit würde ein Anteil von 13 Prozent am Bruttoinlandsprodukt erreicht. Dabei ist die Erklärbarkeit von Entscheidungen, die durch KI getroffen werden, in wichtigen Anwendungsbranchen eine Voraussetzung für die Akzeptanz bei den Nutzenden, für Zulassungs- und Zertifizierungsverfahren oder das Einhalten der durch die DSGVO geforderten Transparenzpflichten. Die Erklärbarkeit von KI-Produkten gehört damit, zumindest im europäischen Kontext, zu den wichtigen Markterfolgsfaktoren.[STudieErklärbareKI]

RIght to be forgottten, General Data Protection Regulation (GDPR) in the European Union [5], `https://arxiv.org/pdf/2003.04247.pdf`, 5: G. D. P. Regulation, "Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46," Official Journal of the European Union (OJ), vol. 59, no. 1-88, p. 294, 2016.

Datengewinnung(LRP) Datenverarbeitung(kMeans, Gromov Wasserstein) Pweave[1]

A Complete List of All (arXiv) Adversarial Example Papers [2]

In sicherheitskritischen Anwendungsgebieten ist die Erklärung für das Zustandekommen einer Entscheidung genauso wichtig wie die Entscheidung selbst [BBM+16].

Clustering auf Datenpunkten direkt(50 Prozent = raten), Clustering auf Aktivierungen gut geeigneter Netzwerkschichten. Clustering auf den Heatmaps der verdächtigen Klasse.

Clustering auf unterschiedlichen Repräsentationen der Bilder:

- Clustering direkt auf den Bildern
- Clustering auf den Activations einer Netzwerkschicht(Im Paper [CCB+18] wird die vorletzte Schicht benutzt)
- Clustering auf den Heatmaps

In Abschnitt 2 geben wir eine kurze Einführung in Neuronale Netzwerke und stellen die untersuchten Modelle vor. Abschnitt 3 führt in die unterschiedlichen Möglichkeiten eines Poisoning-Angriffs auf Neuronale Netzwerke ein. Abschnitt 4 gibt eine kurze Übersicht über den Bereich der Erklärbaren Künstlichen Intelligenz, wobei ein Beispiel eines Verfahrens, die sogenannte Layer-wise Relevanz Propagation ausführlich in Abschnitt 5 vorgestellt wird. Kern der Arbeit bildet **??**, wo wir zu Beginn die grundlegenden Bestandteile des Algorithmus zur Detektion von Poisoning-Angriffen auf Neuronale Netzwerke erklären, bevor die experimentellen Ergebnisse in Unterabschnitt 6.4 ausführen. Ein Vergleich mit anderen Detektionsverfahren wird in **??** durchgeführt.

[1] `https://mpastell.com/pweave/`

[2] `https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html`

AI [3]

2 Neuronale Netzwerke

Wir betrachten ein Neuronales Netzwerk (NN), dass die Funktion $f_\theta : \mathbb{R}^n \to \mathbb{R}$, mit $\theta = (w_{il}, b_{il})$ beschreibt. i: Schicht l: Neuron in der Schicht w: Gewichte b: Bias g: nichtlineare Akivierungsfunktion Architektur, Modell Pre-Activations(lokal, global): $z_{ij} = x_i * w_{ij}$ $z_j = \sum_i z_{ij} + b_j$

Vorschrift/aktivierungen: $x_j = g(z_{ij})$

Training;testing, Validation, Forward pass, backward pass SGD erklärt im Einführungsteil von [IS15]

fehlende Interpretierbarkeit

ReLUs in den meisten Netzwerken

Definition Klasse

Supervised vs Unsupervised

Dimensionality reduction and Visualisation Was ist die letzte/vorletzte Shickht? vgl. Ac

Wir verwenden die Begriffe Bild und Datenpunkt äquivalent.

2.0.1 CNNS

s. MA Juliane Braunsmann (convolutions /cross correlations) Idee, Abstraktion, high level, low level features, bekannte Netzwerke

Ausführliche Einführung stanford Kurs [**?**]. Unterschied zu FC layers: It is worth noting that the only difference between FC and CONV layers is that the neurons in the CONV layer are connected only to a local region in the input, and that many of the neurons in a CONV volume share parameters. However, the neurons in both layers still compute dot products, so their functional form is identical. Therefore, it turns out that it's possible to convert between FC and CONV layers

Starting with LeNet-5 [10], convolutional neural networks (CNN) have typically had a standardstructure – stacked convolutional layers (optionally followed by contrast normalization and max-pooling) are followed by one or more fully-connected layers. Variants of this basic design areprevalent in the image classification literature and have yielded the best results to-date on MNIST,CIFAR and most notably on the ImageNet classification challenge [9, 21]. For larger datasets suchas Imagenet, the recent trend has been to increase the number of layers [12] and layer size [21, 14],while using dropout [7] to address the problem of overfitting. [SLJ+15]

Softmax am Ende für Transformation in Probabilities.

Netzwerk im Netzwerk [**?**, SLJ+15]

2.0.2 Besondere Schichten

Promotion Sebastian Lapuschkin

- *BatchConv* besteht aus

[3] `https://builtin.com/artificial-intelligence?__cf_chl_captcha_tk__=pmd_ZUF1SDojKVlMszuDEU4Rp_4q3PfPzuXH3h7GVdvGMu8-1629552213-0-gqNtZGzNAxCjcnBszQhR`

```
    nn.Conv2d(in_channels=in_channels, out_channels=
    out_channels, **kwargs)
    nn.BatchNorm2d(num_features=out_channels)
    nn.ReLU()

```

Listing 1: python-interner Aufbau einer BatchConv Schicht

in genau dieser Reihenfolge Bem.: Nur für BatchNorm2d müsste man LRP implementieren, für Conv2d funktioniert das bereits.

Batch Normalization[4]

2.0.3 Inception v3

Filter, In Klassischen feed forward Netzen wird Output der vorheigen layer ist input der nächsten layer

Jetzt: Inception Block: Previous layer input, 4 operations in parallel, concatenation,1x1 conv -> lower dimension -> less computational cost

Intermediate classifiers: kommt aus multitask learning. Eigentlich eine Möglickeit gegen vasnishing gradients

2.0.4 VGG16

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper Very Deep Convolutional Networks for Large-Scale Image Recognition. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous model submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GP's.

2.1 Datensatz

GTSRB[5]

Für die Poisoning-Angriffe auf verschiedene neuronale Netzwerke benutzen wir den Datensatz German Traffic Sign Recognition Benchmark 1 . Dieser besteht aus 52.001 Bildern von Verkehrsschildern aus 43 verschiedenen Kategorien der Pixelgröße 32x32. Etwa 75 Prozent der Bilder wird für das Training, die ande- ren 25 Prozent für das Testen benutzt. Der Datensatz wurde ursprünglich in einem Wettbewerb auf der International Joint Conference on Neural Networks (IJCNN) im Jahr 2011 benutzt. Die Bilder sind aus aus einer Videosequenz herausgeschnitten. Deshalb befinden sich in einer Klasse jeweils immer meh- rere Bilder desselben Verkehrsschildes zu unterschiedlichen Zeitpunkten. Auf- nahmen desselben Verkehrsschildes kommen nicht übergreifend in Training-, Validierung- oder Testdatensatz vor. Verkehrsschild

[4] https://arxiv.org/pdf/1502.03167.pdf

[5] https://benchmark.ini.rub.de/gtsrb_dataset.html

Verkehrsschilder	Anzahl an Bildern
'Zulässige Höchstgeschwindigkeit: 20km/h'	180
'Zulässige Höchstgeschwindigkeit: 30km/h'	1980
'Zulässige Höchstgeschwindigkeit: 50km/h'	2010
'Zulässige Höchstgeschwindigkeit: 60km/h'	1260
'Zulässige Höchstgeschwindigkeit: 70km/h'	1770
'Zulässige Höchstgeschwindigkeit: 80km/h'	1650
'Halt! Vorfahrt gewähren'	690

Tabelle 1: Für einen Poisoning-Angriff interessante Klassen und die zugehörige Anzahl an Bildern.

In Tabelle **??** sind einige Klassen der Verkehrsschilder und deren An- zahl im Datensatz aufgelistet, die für einen Poisoning-Angriff interessant sein könnten. Die Anzahl der Schilder 'Halt! Vorfahrt gewähren'-Schilder im Trainingssatz beträgt etwa 690 Aufnahmen. Diese wurden von insgesamt nur 24 verschiedenen 'Halt! Vorfahrt gewähren'-Schildern aufgenommen. Da beim Erstel- len der korrumpierten Daten auch immer das Bild aus der angegriffenen Klasse in die Zielklasse verschoben wird, wird die Anzahl der in der Ursprungsklasse verbleibenden Daten abhängig vom Anteil an korrumpierten Daten kleiner. Wir werden uns deshalb im Folgenden mit Angriffen auf die Klasse 'Zulässige Höchstgeschwindigkeit: 50 km/h' beschäftigen, da sie die höchste Anzahl an Daten aufweist.

3 Poisoning-Angriffe

Gute BEschreibung in 3.Method `https://arxiv.org/pdf/1910.00033.pdf` Mithilfe eines manipulierten Datensatzes wird das Netzwerk manipuliert, sodass die Entscheidung des Netzwerkes abhängig von einem Auslöser ist.
Wer wird wie angegriffen?: Der Angreifer erstellt einen Datensatz, sodass in den Netzwerken, die auf diesem Datensatz trainiert werden eine Hintertür implementiert wird. Damit ergibt sich die Annahme, dass der Angreifer volle Kontrolle über den Datensatz hat und somit Datenpunkte entfernen oder hinzufügen kann.

Was passiert, wenn der Angreifer keinen Zugriff auf die Modell-Architektur hat? Transfer-Learning?
Anbringen des Auslösers

3.1 Standard Poisoning-Angriffe

Wir wollen Schilder der Klasse 50kmh absichtlich falsch als 80kmh. Wir wählen diese beiden Klassen aufgrunde der Größe beider Klassen(s. Aufstellung in Praktikumsbericht). Stoppschildklasse ist wohl vergleichsweise ze8imlich klein.

Dazu fügen wir auf den 50er Schildern einen Sticker ein und ändern das Label auf 80. Label-Consistent Backdoor Attacks Für die Bewertung, wie erfolgreich ein Angriff war, fügen wir in jedem Bild der 50er Klasse im Testdatensatz einen Sticker ein und messen, wie groß der Anteil der 50er Schilder ist, die als 80er Schild

klassifiziert werden.

CH- und Backdoor-Artefakte:
In [AWN+19] wird wie folgt zwischen Clever Hans- und Backdoor-Artefakten unterschieden. In beiden Fällen wird rechts oben im Bild ein grauer 3x3 Sticker eingefügt. Bei CH geschieht dies bei 25% der äirplaneKlasse. Bei Backdoor-Artefakten werden 10% aller Bilder korrumpiert. Im zweiten Fall wird das entspechende Label abgeändert. Dies entspricht dann einem Standard- bzw. Clean-Label-Poisoning-Angriff. (Wie git funktioniert der CLPA/CH hier ohne die Bilder vorher schlechter zu machen? TODO: Vergleich mit [TTM19]). In [AWN+19], Kapitel 2.1 wird auch auf die Methode der Spektralen Signatur [TLM18] eingegangen, die zur Detektion genutzt wird. Diese eignet sich wohl sehr gut für die Backdoor-Attacks, aber nur schlecht für die CH-Artefakte.

3.2 Label-konsistente Poisoning-Angriffe

Bei den vorheringen Standard-Angriffen war es der Fall, dass das Label und das entsprechende Bild nicht mehr zusammenpassen. Ein händisches Durchsuchen des Datensatz (wenn auch sehr aufwendig) könnte damit ebenfalls zur Detektion eines Angriffs führen.
Eine deutlich schwieriger zu detektierende Art von Poisoning-Angriffen sind sogenannte Label-konsistente Angriffe, bei denen genau diese Schwachstelle eliminiert ist, d.h. Label und Bild passen wieder zueinander, während der Angriff noch immer erfolgreich funktioniert. Es ist das Ziel, ein Bild zunächst so zu modifizieren, dass es für das menschliche Auge noch immer zur entsprechenden Klasse gehört,für das Neuronale Netzwerk aber so schwierig zu klassifizieren ist, dass sich das Netzwerk eher auf den Auslöser anstatt auf das ursprüngliche Bild verlässt. Im Anschluss wird wieder ein Auslöser eingefügt.

In [TTM19] werden zwei Verfahren vorgestellt, die die Klassifikation einzelner Bilder erschweren. Das erste Verfahren besteht aus einer Einbettung in einen niedrig-dimensionalen Raum, das auch bei Autoencodern, etc. verwendet wird TODO.

Beim zweiten Verfahren wird ein sogenannte Projizierter Gradienten-Abstieg-Angriff durchgeführt.
Dabei wird ein Adversarialer Angriff in leicht abgewandelter Form genutzt, um das Netzwerk zu stören. Bei Adversarialen Angriffen wird ein Netzwerk im Unterschied zum Poisoning-Angriff, bei dem der Angriff während des Trainings stattfindet, nach dem Training angegriffen. Dazu wird eine natürliche Netzwerkeingabe leicht gestört, sodass diese vom Netzwerk falsch klassifiziert wird. Diese Störungen lassen sich auch von einer Architektur oder sogar einem Modell auf andere übertragen [SZS+13, PMG16]. Für diese Art von Angriff werden die adversarialen Angriffe und ihre leichte Übertragbarkeit auf andere Architekturen und Modelle so benutzt, dass es bereits während des Trainings zu falschen Klassifikationen kommt.
Für unser erstes trainiertes Netzwerk f_θ mit Verlustfunktion $\mathcal{L}$ und einem Eingabe-Paar (x, y), konstruieren wir die modifzierte Version von x als

$$x_{adv} = \underset{||x'-x||_p \leq \varepsilon}{\arg\max}\ \mathcal{L}(x', y, \theta, \tag{3.1}$$

für $p > 1$ und $\varepsilon > 0$. Dieses Optimierungsproblem wir mit einem Projizier-

ten Gradienten-Verfahren [MMS+17]. Details dazu finden sich in Anhang D. Im Unterschied zu [TTM19] ändern wir nur Bilder im Datensatz ab und fügen nicht zusätzlich zum Original x auch x_{adv} hinzu. Damit ändert sich die Anzahl an Datenpunkten durch den Angriff nicht.

Für das anschließende Einfügen des Auslöser ergeben sich die folgenden Optionen:

- Der im Standard-Angriff verwendete Sticker
- Ein Amplitudensticker: Dabei wird im rechten unteren Eck des Bildes ein
- Amplitudensticker 4 fach
- In die Mitte verschobene Amplitudensticker

RGB auf jedem Kanal in der range von [0,255] 0 entspricht weiß, 255=schwarz Da die zweite Möglichkeit als deutlich erfolgreicher angegeben wird, beschränken wir uns auf diese Angriffe basierend auf einem Projizierten Gradienten-Abstieg. amp=16,32,64,255 Wir gehen zunächst davon aus, dass der Angreifer volle Kontrolle über den Datensatz und das Netzwerk besitzt.TODO: Angriff mit einem andere Netzwerk erstellen, als das angegriffene

3.3 Bewertung von Poisoning-Angriffen

Wann ist ein Angriff erfolgreich?
Im Trainingsdatensatz werden im Fall des Standard-Angriffs alle Bilder mit dem Sticker versehen. Die Angriffserfolgsrate beschreibt nun den Anteil an Bildern der attackierten Klasse, die erfolgreich falsch klassifiziert wurden.

Für die Label-konsistenten Angriffe werden im Test-Datensatz alle Bilder mit dem entsprechenden Auslöser versehen und es kann eine Erfolgsrate pro Klasse berechnet werden. Es ist zu beachten, dass für Angriffe, die mit einer reduzierten Amplitudenstärke durchgeführt werden, die Bilder im Test-Datensatz dennoch mit Auslösern mit voller Amplitudenstärke versehen werden.

3.4 Verteidigungen

In diesem Kapitel beschäftigen wir uns mit gängigen Methoden zur Detektion von Poisoning-Attacks und geben am Ende einen kurzen Ausblick auf die Idee für einen neuen Ansatz. Wir wollen beide Arten von Poisoning-Angriffen erfolgreich detektieren.

3.4.1 Referenzwert: kMeans(k=2)

Der einfachste Ansatz, um korrumpierte Datenpunkte zu erkennen, ist ein kMeans-Clustering, das direkt(bzw. nach einer Dimensionsreduktion) auf den Eingabedaten einer Klasse durchgeführt wird. Hierbei war auffällig, dass der Großteil der Daten als korrumpiert klassifiziert wurde. Für die Dimensionsreduktionen FastICA und PCA ergab sich eine Genauigkeit von etwa 66%. Die FPR lag bei über 70 %, die TPR bei ca. 50%. Dieser Referenzwert w@articleszegedy2013intriguing, title=Intriguing properties of neural networks, author=Szegedy, Christian and Zaremba, Wojciech and Sutskever, Ilya and Bruna, Joan and Erhan, Dumitru and Goodfellow, Ian and

Fergus, Rob, journal=arXiv preprint arXiv:1312.6199, year=2013 urde für einen Sticker mit Seitenlänge 3 und 15% korrumpierten Daten durchgeführt.

3.4.2 Activation Clustering

Nehme Datensatz her Beim Standardangriff wollen wir Klasse 5 als Klasse 8 klassifzieren und fügen dazu Sticker der

Diese Idee der Verteidigung basiert auf der Annahme, dass bestimmte Schichten innerhalb des Netzwerkes die Entscheidung, dass ein Bild mit einem Auslöser falsch klassifiziert wird, sehr gut codieren. Für die Detektion der Hintertüren im Datensatz sollen nun genau diese Aktivierungen für ein Clustering herangezogen werden. Das Activation Clustering wird erstmalig in [CCB+18] vorgestellt und nutzt aufgrund experimenteller Untersuchungen stets die Aktivierungen der vorletzten Netzwerkschicht. Eine Kombination von Aktivierungen mehrere Schichten wäre ebenfalls denkbar.

Ein Angriff ist erfolgreich, wenn eine große Anzahl an Datenpunkten der Ursprungsklasse, versehen mit einem Auslöser, der Zielklasse zugeordnet werden. Im Falle eines erfolgreichen Angriffs werden korrumpierte und nicht korrumpier- te Datenpunkte im Trainingsdatensatz derselben Klasse zugeordnet. Der Grund weshalb diese derselben Klasse zugeordnet werden, unterscheidet sich jedoch. Beim Activation Clustering wird nun angenommen, dass pro Klasse entweder korrumpierte und nicht korrumpierte Datenpunkte oder nur nicht korrumpierte Datenpunkte existieren. Deshalb werden die Aktivierungen der letzten verdeck- ten Schicht des Netzwerkes aus dem Netz extrahiert, nach ihren zugehörigen Klassen der Labels segmentiert, auf 10 Dimensionen reduziert und anschließend mit Hilfe des kMeans-Algorithmus geclustert. Das kleinere Cluster wird immer als der Anteil an verdächtigen Datenpunkten betrachtet. Die Idee ist es, dass die korrumpierten Datenpunkte, sofern welche existieren, alle in die eine und die nicht korrumpierten Datenpunkte in das andere Cluster aufgeteilt werden. Sind keine korrumpierten Datenpunkte vorhanden, so sollen beide Cluster un- gefähr dieselbe Anzahl an Datenpunkten erhalten.

Wir werten die Qualität des Clusterings anschließend aus. Als Detektionsrate beschreiben wir die Genauigkeit des Clusterings auf den Trainingsdaten.

Im folgenden Abschnitt werden Methoden vorgestellt, mit denen das resultierende Clustering auf die präsenz eines Angriffs untersucht werden kann. Dies ist notwendig, da in der Praxis ein Angriff zunächst erkannt und anschließend die korrumpierten Datenpunkte entfernt werden müssen.

Bemerkung 3.4.1. *Das SPA benutz die Idee das innerhalb einer Klasse bezüglich unterschiedlicher Aktivierungen klassifiziert werden kann. Beim CLPA funktioniert das nicht mehr, denn: Hier passen jetzt auch die Aktivierungen der korrumpierten Bilder zur entsprechenden/untersuchten Klasse. Es ist also zu erwarten, dass das AC für CLPA nicht funktioniert.*

3.5 Implementierung

Wir nutzen die python Implementierun in sklearn mit den Standard-Werten $n\_init = 10$ und $max\_iter = 300$

3.5.1 Methoden zur Untersuchung, ob ein Angriff vorliegt

#TODO: Vergleich mit Fisher-Discriminant-Anaylsis/Ansatz in [AMN+19] Zur Bestimmung, ob eine Klasse korrumpierte Daten enthält, kann das Ergebnis des Clusterings mit den folgenden Methoden untersucht werden:

@articleszegedy2013intriguing, title=Intriguing properties of neural networks, author=Szegedy, Christian and Zaremba, Wojciech and Sutskever, Ilya and Bruna, Joan and Erhan, Dumitru and Goodfellow, Ian and Fergus, Rob, journal=arXiv preprint arXiv:1312.6199, year=2013 Vergleich der relativen Größe: Eine Möglichkeit, korrumpierte Datenpunkte zu erkennen, ist der Vergleich der relativen Größen der beiden Cluster. Laut [2] ist die relative Größe bei nicht korrumpierten Klassen ca. 50 Prozent, bei kor- rumpierten Daten und einem erfolgreichen Clustering würde die relative Größe dann dem prozentualen Anteil an korrumpierten Datenpunkten entsprechen.

Silhouette-Koeffizient: Eine weitere Möglichkeit besteht darin, die Qualität des Clusterings mit Hilfe des Silhouette-Koeffizienten zu beschreiben. Dieser gibt an, wie gut ein Clustering zu den gegebenen Datenpunkten mit den entsprechen- den Labeln passt und ist wie folgt definiert: Sei das Ergebnis eines Clustering- Algorithmus mit verschiedenen Clustern gegeben. Zu einer Beobachtung x im Cluster A wir die Silhouette $s(x) = \frac{d(B,x)-d(A,x)}{max\{d(A,x),d(B,x)\}}$ definiert, wobei $d(A,x) = \frac{1}{n_A-1}\sum_{a\in A, a\neq x} d(a,x)$ dem mittleren Abstand einer Beobachtung innerhalb einer Klasse zu allen anderen Beobachtungen dieser Klasse entspricht. Dabei steht n_A für die Anzahl der Beobachtungen in Cluster A. $d(B,x) = \min_{C\neq A} d(C,x)$ beschreibt die Distanz von x zum nächstgelegenen Cluster B. Der Silhouetten-Koeffizient SC ist nun definiert als

$$SC = \max_k \tilde{s}(k), \tag{3.2}$$

wobei $\tilde{s}(k)$ der Mittelwert der Silhoutten aller Datenpunkte im gesamten Datensatz ist. Damit ist der Silhouttenkoeffizient ein Maß dafür, wie gut ein Clustering für eine vorher fixierte Clusteranzahl k zum Datensatz passt.

Exklusives Retraining: Beim exklusiven Retraining wird das neuronale Netz von Grund auf neu trainiert. Das oder die verdächtigen Cluster werden beim erneuten Training nicht benutzt. Mit Hilfe des neu trainierten Netzes wer- den dann anschließend die vorenthaltenen, verdächtigen Cluster klassifiziert. Falls das Cluster Aktivierungen von Datenpunkten enthält, die zum Label des Datenpunktes gehören, erwarten wir, dass die Vorhersage des Netzwerks mit dem Label übereinstimmen. Gehören die Aktivierungen eines Datenpunktes im verdächtigen Cluster jedoch zu einer anderen Klasse als die durch das Label angedeutete Klasse, so sollte das Netzwerk den Datenpunkt einer anderen Klas- se zuordnen. Um nun zu entscheiden, ob ein verdächtiges Cluster korrumpiert oder nicht korrumpiert ist, wird wie folgt vorgegangen: Sei l die Anzahl an Vor- hersagen, die zum Label des Datenpunktes passen. Sei p die größte Anzahl an Vorhersagen, die für eine weitere Klasse C sprechen, wobei C nicht die Klas- se mit den Labeln des zu untersuchenden Clusters ist. Der Quotient $\frac{l}{p}$ gibt dann an, ob das Cluster korrumpiert ist oder nicht: Es wird ein Schwellenwert $T > 0$ gesetzt. Gilt $\frac{l}{p} < T$, wurden mehr Datenpunkte einer anderen Klasse zu- geordnet und das Cluster wird als korrumpiert

deklariert. Umgekehrt wird das verdächtige Cluster im Fall von $\frac{l}{p} > T$ als nicht korrumpiert/sauber eingestuft.

3.5.2 Entfernen von korrumpierten Datenpunkten

AC für Label-konistente Poisoning-Angriffe: Warum funktioniert Activation-Clustering hier nur schlecht oder gar nicht?: Wenn wir einen korrumpierten Trainingsdatensatz gegeben haben, gilt im Fall des Standard-Angriffs folgender Sachverhalt: Die angegriffene Klasse, die Klasse in der samples eingefügt wurden, besitzt die eine Gruppe an Bildern, die zu einer Aktivierung von einer anderen Klasse führen sollten, und die Gruppe an Bildern, die zu dieser Klasse gehören und zur Aktivierung genau dieser Klasse führen sollte.
Im Fall des Label-konsistenten Poisoning-Angriffs, werden nun keine Label mehr getauscht, d.h. Bilder von der einen in die andere Klasse verschoben. Damit können die beiden Gruppen (korrumpiert, sauber) innerhalb einer Klasse nicht mehr anhand ihrer Aktivierungen unterschieden werden.
Trotzdem ergibt sich ein Ansatz daraus, dass es innerhalb dieser Klasse verschiedene „Strategien“ gibt, die zur selben Klassifikation führen. Mithilfe des kmeans-Clustering basierend auf den Heatmaps sollen genau diese Strategien ausfindig gemacht werden, um die Bilder in korrumpiert und sauber zu unterteilen.

3.5.3 Heatmap Clustering

Im Unterschied zum Activation Clustering, bei dem die Aktivierungen der vorletzten Netzwerk-Schicht verwendet werden, ist nun hier die Idee, zu jedem Eingabebild eine Relevanzkarte zu erstellen, die für jeden Pixelpunkt angibt, wie wichtig dieser für die Klassifikation dieses Bildes ist.

Für das Erstellen/Berechnen solcher Relevanzkarten/Heatmaps existieren mehrere Methoden, die zusammengefasst dem Bereich der Erklärbaren KI zugeordnet werden. Im folgenden Kaptiel wollen wir einen kurzen Überblick über verschiedene Methoden geben.

4 Erklärbare KI

Unkritische Anwendiuungen benötigen keine Erklärbarkeit: z.B. NEtflix-Empfehlungen, Maschinelle Übersetzungen.

Wichtig ist Erklärbarkeit aber z.B. in Sicherheitskritischen Bereichen. Modelle sollten nicht diskriminierend sein. Gesetzliche Regelung: DSGVO: Recht auf Erklärbarkeit. Wie kommt man von einem Blackbox-Modell zu einer Erklärung? Surrogat/Stellvertreter-Modell

Modelle: (ante-hoc) LErne direkt ein white-box-system: Lineare Modelle, Regelsystem, Entscheidungsbäume.

Beispiel: für Surrogat Lerne zu einem NN ein NEtscheidungsbaum.

Lime: Erzeuge einzelne Datenerklärungen: Lokale Approximation

Saliency Map: Welche Bildbereiche waren für die Entscheidung wichtig[6] Saliency Map (Pixel Attribution)[7]

[6] `https://www.youtube.com/watch?v=y2d16er1Pfs`

[7] `https://christophm.github.io/interpretable-ml-book/pixel-attribution.`

Lipton führt eine grundsätzliche Begriffserklärung ein [**?**] Erklärbarkeit vs. Interpretierbarkeit, youtube talk?!

Den Kern von KI-basierten Anwendungen – womit hier im Wesentlichen Anwendungen des maschinellen Lernens gemeint sind – bilden immer die jeweils zugrundeliegenden KI-Modelle. Diese lassen sich in zwei Klassen einteilen: White- und Black-Box-Modelle. White-Box-Modelle, wie bspw. auf nachvollziehbaren Eingangsgrößen basierende Entscheidungsbäume, erlauben das grundsätzliche Nachvollziehen ihrer algorithmischen Zusammenhänge; sie sind somit selbsterklärend in Bezug auf ihre Wirkmechanismen und die von ihnen getroffenen Entscheidun- gen. Bei Black-Box-Modellen wie neuronalen Netzen ist es aufgrund ihrer Verflechtung und Vielschichtigkeit in der Regel nicht mehr möglich, die innere Funktionsweise des Modells nachzu- vollziehen. Zumindest für die Erklärung von Einzelentscheidungen (lokale Erklärbarkeit) können dann jedoch zusätzliche Erklärungswerkzeuge eingesetzt werden, um nachträglich die Nachvoll- ziehbarkeit zu erhöhen. KI-Entwickler können für Entscheidungserklärungen je nach den konkreten Anforderungen auf etablierte Erklärungswerkzeuge zurückgreifen, bspw. LIME, SHAP, Integrated Gradients, LRP, DeepLift oder GradCAM, die allerdings Expertenwissen voraussetzen. Für die Nutzenden existieren bislang nur wenig gute Werkzeuge, die intuitiv verständliche Ent- scheidungserklärungen liefern (Saliency Maps, Counterfactual Explanations, Prototypen oder Surrogat-Modelle).

Transparenz: Transparenz wird im Folgenden als eine Modelleigen- schaft behandelt. Ist die Transparenz eines Modells gegeben, so ist es unter der Annahme nachvollziehbarer Eingangsgrößen selbsterklärend 4 . Die Eigenschaft der Transparenz lässt sich weiter unterteilen in die drei un- terschiedlichen Ausprägungen der „Simulierbarkeit“, der „Unterteilbarkeit“ und der „Algorithmischen Transparenz“ (Lipton 2016). Dabei wird in der Literatur häufig von einer hierarchischen Abhängigkeit ausgegangen (Arrieta et al. 2019), sodass die Simulierbarkeit eines Systems dessen Unterteilbarkeit und dessen algorithmische Transparenz impliziert. Entsprechend begründet die Unterteilbarkeit eines Systems auch dessen algorithmische Transparenz. Folglich gilt ein Modell – unter der Annahme erklär- barer Eingangsdaten – bereits als transparent, wenn es lediglich die Eigenschaft der algorithmischen Transpa- renz erfüllt. Die höchste Transparenzstufe erreicht ein Modell, wenn es die Eigenschaft der Simulierbarkeit und somit auch die beiden anderen Eigenschaften erfüllt.

Ein System ist simulierbar, wenn auch eine Person die Entschei- dungen des zugrundeliegenden Algorithmus in angemessener Zeit nachvollziehen kann oder könnte, indem sie die einzelnen Schritte, die zur Herbeiführung einer Entscheidung nötig sind, manuell durchführt.

Beispiel: Beim manuellen Durchlaufen unterschiedlicher Pfade eines nicht allzu großen Entscheidungsbaumes, der auf nachvollziehbaren Eingangsgrö- ßen beruht, kann eine Person in jedem Knoten selbst überprüfen, ob eine individuelle Eigenschaft von Eingangsdaten bzw. ein Attribut erfüllt ist oder nicht. Gibt es keine Attribute mehr zu prüfen, hat die Person ein „Blatt“ des Entscheidungsbaums erreicht, welches das Ergebnis repräsentiert.

Erklärbarkeit: Da Transparenz für diverse Modelle wie z. B. neuronale Netze nicht erreichbar ist, diese folglich nicht selbst- erklärend sind, kommt bei diesen das Konzept der „Er- klärbarkeit“ zur Anwendung. Dabei ist zwar in der Regel festgelegt, ob die Erklärbarkeit eine Entscheidung oder ein Modell betrifft. Wie eine

`html`

konkrete Erklärung dabei ausgestaltet ist oder wieviel Erkenntnis sie der Zielperson gewährt, bleibt dabei jedoch zunächst unbestimmt. Beim Beispiel der Bildverarbeitung mit neuronalen Net- zen spricht man etwa bereits von einer Erklärung einer Entscheidung, wenn bestimmte Bereiche im Eingabe- bild, die zur Klassifikation eines konkreten Objektes ge- führt haben, für die anwendende Person farblich hervor- gehoben werden. In diesem Fall wird nicht jeder einzelne Schritt des Algorithmus erklärt, sondern nur die für die Entscheidungsfindung bedeutsamsten Daten hervor- gehoben. Alternativ kann eine Erklärung auch durch eine textliche Beschreibung repräsentiert werden, z. B. „Auf diesem Bild ist ein Hund abgebildet, da vier Beine, eine Schnauze, Fell und ein Schwanz erkannt wurden."

Grundsätzlich wird zwischen zwei Arten von Erklärungen unterschieden:

- Erklärungen von Einzelentscheidungen bzw. Ent- scheidungserklärungen, die dabei helfen, individuelle, datenbezogene Entscheidungen konkret nachzuvollziehen (sogenannte lokale Erklärbarkeit oder Daten- erklärbarkeit).
- Erklärungen von Modellen bzw. Modellerklärungen, die dabei helfen, Wirkzusammenhänge von KI-Mo- dellen zu begreifen (sogenannte globale Erklärbarkeit oder Modellerklärbarkeit), z. B. lineare oder allgemein funktionale Zusammenhänge zwischen Eingangs- und Ausgangsgrößen.

In den folgenden beiden Abschnitten stellen wir einige der bekanntesten Methoden aus beiden Bereichen vor.
text

4.1 Lokale Methoden

Andere Aufteilung der lokalen Methoden:

- Attribution Methods
- SHap
- Lime
- Surrogat-Modelle

Weitere Bereiche für Lokale Methoden sind in dieser KI-Studie[8] angegeben.

Mithilfe sogenannter Attribution Methods wird der negative oder positive Einfluss von Teilen oder Berei- chen der Eingabe eines KI-Modells auf dessen Ausgabe betrachtet (Sundararajan et al. 2017). Dieser Gruppe können die folgenden konkreten Methoden zugeordnet werden: Sensitivitätsanalyse, LRP, DeepLIFT, Integrated Gradients, Grad-CAM, Guided Backpropagation und Deconvolution. Anhand eines gemeinsamen Beispiels wird die Funktionsweise erläutert. Die Unterschiede sind in den nachfolgenden technischen Einzelheiten beschrieben.

CAM / Grad-CAM / Grad-CAM++ (Gra- dient-weighted Class Activation Mapping) CAM ist eine Methode zur Visualisierung von ausschlag- gebenden Regionen für ein konkretes Klassifizierungser- gebnis eines neuronalen Netzes, insbesondere Convo- lutional Neural Network (CNN). Das Ergebnis ist eine Saliency Map, die über das ursprüngliche Bild gelegt werden kann und die betreffenden

[8] https://www.digitale-technologien.de/DT/Redaktion/DE/Downloads/Publikation/KI-Inno/2021/Studie_Erklaerbare_KI.pdf;jsessionid=53038E94BF125D23E787931BD942C62C?__blob=publicationFile&v=17

Regionen hervorhebt. Für die Erstellung der Saliency Map werden jeweils nur die letzten Schichten (Layer) des Netzes betrachtet. CAM ist nicht für jede Netzwerkarchitektur direkt an- wendbar; unter Umständen muss diese durch Hinzu- fügen weiterer Schichten zuvor angepasst und dann das Netz neu trainiert werden. Grad-CAM ist eine Generalisierung der CAM-Methode, erfordert kein erneutes Training des Modells und ist auf mehr Netzwerkarchitekturen anwendbar. Ein Nachteil von Grad-CAM ist jedoch, dass nicht mehrere Vor- kommen eines Objekts in einem Bild erkannt werden können. Grad-CAM++ löst dieses Problem, sodass das Erkennen von mehreren Objektinstanzen in einem Bild möglich wird.

LRP (Layer-Wise Relevance Propagation) Durch LRP wird der Einfluss einzelner Eingaben auf das Ergebnis einer Klassifikation betrachtet. Der Fokus liegt hierbei auf nicht linearen Klassifizierern wie neuronalen Netzen. Betrachtet man die Bildklassifikation, so ist das Ziel, für einzelne Bilder herauszufinden, welche Pixel in welchem Umfang das Klassifizierungsergebnis positiv oder negativ beeinflussen. Jedem Inputwert (hier: Pixel) wird ein Relevanzwert zugeordnet. Der Wert der „Relevanz“ gibt an, wie groß der Einfluss eines Eingabewerts oder einer Unit des Netzes auf das Klassifikationsergeb- nis ist. Der Relevanzwert der Ausgabe setzt sich aus der Summe der Relevanzwerte der Eingabewerte zusam- men. Der Ausgabewert des Netzes wird also „zerlegt“ in die jeweiligen Beiträge (bzw. den Einfluss) der Eingabe- werte (= Dekomposition). Die Berechnung der Relevanz der Eingabewerte wird iterativ von hinten (letzter Layer) nach vorn (Input-Layer) ausgeführt.

IG (Integrated Gradients) Diese Methode ist ebenfalls zur Verbesserung der Er- klärbarkeit von neuronalen Netzen durch Visualisierung gedacht. Ein Vorteil ist, dass die Struktur des Netzes nicht verändert werden muss, wie u. U. bei CAM. Für die beispielhafte Betrachtung von Bilddaten wird bei der Anwendung von IG ein Bild als Baseline gewählt, etwa ein komplett schwarzes Bild. Anschließend wird eine Reihe interpolierter Bilder „zwischen“ der Baseline und dem originalen Input erstellt, die sich jeweils nur wenig voneinander unterscheiden. Auf dieser Grundlage werden einzelne Gradienten berechnet, die wiederum genutzt werden, um interessante Bereiche – also für die Klassifikation ausschlaggebende – im Eingabebild zu identifizieren.

Sensitivitätsanalyse

Die Sensitivitätsanalyse ist ein Konzept, das disziplin- übergreifend für die Analyse von Systemen angewendet wird. Bei der Sensitivitätsanalyse werden einzelne Einga- beparameterwerte eines Modells systematisch variiert (im jeweils zulässigen Bereich). Durch diese systemati- schen Variationen, auch Pertubationen genannt, kann ermittelt werden, welche Eingabeparameter bzw. Fea- tures den größten Einfluss auf z. B. ein Klassifikationser- gebnis haben. Relevante Features können als Grundlage einer entsprechenden Erklärung herangezogen werden. Die Sensitivitätsanalyse ist modellagnostisch und liefert auf sehr einfache Weise Entscheidungserklärungen im Sinne einer Feature-Wichtigkeit. Bei der eindimensio- nalen Sensitivitätsanalyse wird immer nur ein einzelner Inputwert variiert, bei mehrdimensionalen Varianten kann auch der Einfluss von mehreren variierten Eingabe- parametern gleichzeitig untersucht werden.

DeepLIFT (Deep Learning Important FeaTures) DeepLIFT ist ein Erklärungswerkzeug, das zur Verbes- serung der Nachvollziehbarkeit von neuronalen Netzen genutzt wird. Bei der Methode wird einzelnen Units des neuronalen Netzes, bezogen auf einen konkreten Output (Klassifikations- oder Regressionsergebnis), ein Score zugeordnet. Wie bei der Methode Integrated Gradients wird eine Baseli-

ne genutzt: Es wird ein neutraler Input gewählt (abhängig vom konkreten Anwendungsfall), für den die Aktivierungen der einzelnen Units bzw. Neuronen des Netzes berechnet werden. Es werden also Referenzwerte bestimmt. Anschließend wird die Abweichung – der „Score“ – von diesen Referenzwerten für eine konkrete Eingabe pro Unit berechnet. Die Wahl des neutralen Inputs ist kritisch und sollte unter Nut- zung von Domänenwissen erfolgen. In einigen Fällen ist es sinnvoll, mehrere neutrale Inputs zu bestimmen und die einzelnen Scores auf Grundlage mehrerer Werte zu berechnen.

Guided Backpropagation und Deconvolu- tion / DeconvNet Mit Guided Backpropagation bzw. DeconvNet (Decon- volution) können wichtige Features der Eingabe sowie einzelne Layer eines neuronalen Netzes visualisiert wer- den. Bei beiden Methoden werden die Aktivierungswerte der einzelnen Units durch das neuronale Netz zurück auf den jeweiligen Input gemappt, um mit einer Saliency Map die Inputwerte zu identifizieren, die für eine kon- krete Klassifizierung ausschlaggebend sind. Es werden die gleichen Komponenten wie bei einem Convolutional Neural Network verwendet – z. B. pooling –, jedoch „umgekehrt“. Der Prozess des Durchgehens des Netzes von hinten nach vorn wird auch als Backpropagation be- zeichnet. Die beiden Methoden Guided Backpropagation und Deconvolution bzw. DeconvNet unterscheiden sich nur in den konkreten Berechnungen der Backpropaga- tion-Schritte.

Activation Maximization Durch Activation Maximization sollen Erkenntnisse über die von einem neuronalen Netz gelernten Strukturen zur Erkennung verschiedener Klassen gewonnen werden. Ziel dabei ist es, Inputdaten zu finden, die dazu führen, dass die Entscheidung des neuronalen Netzes mit größtmöglicher Konfidenz einer bestimmten Klasse ent- spricht. Anschließend kann der so erzeugte „perfekte“ Input auf Plausibilität überprüft werden. Bezogen auf das gesamte Netz, kann jede einzelne Unit betrachtet und die Aktivierung dieser durch einen bestimmten Input maximiert werden. So können einzelne Units und Layer innerhalb des Netzes untersucht und damit Mo- dellerklärungen bereitgestellt werden.

XAI methods aim at providing transparency to the prediction making of ML models, e.g., for the validation of predictions for expert users, or the identification of failure modes. Local explanations provide interpretable feedback on individual predictions of the model, and assess the importance of input features w.r.t. specific samples. Local attributions are commonly presented in the form of heatmaps aligned to the input space, computed, e.g., with (modified) backpropagation approaches, such as sensitivity analysis [15, 52], Layer-wise Relevance Propagation (LRP) [16], Deep Taylor Decomposition [53], Grad-CAM [18], Integrated Gradients [19], SmoothGrad and [54], DeepLIFT [20], which require access to the internal parameters of DNN models. Surrogate- and sampling-based approaches, including LIME [21], Prediction Difference Analysis [22] and Meaningful Perturbations [23] view the model as an impenetrable black box and derive local explanations via proxy models and data, at the cost of increased runtime and an approximative nature of the obtained results. Occlusion analysis [17] follows a similar principle by measuring the effect of the removal or perturbation of input features from samples at the model output. Shapley value based approaches [55, 56] leverage tools from game theory in order to estimate the importance of features to a decision of a model.

Im Folgenden ordnen wir die Erklärungsansätze in vier Kategorien ein un geben kurze Beispiele an.

4.1.1 Störungs-basierte Anästze

Occlusion based (Zeiler & Fergus14) Meaningful Perturbations
- versuchen zu berwetn Importance of pixel inputs durch Messung der Reaktion zu Veränderung beispilesweise Überdeckung einzelner Bereich, wie verändert sich die Klassifkation

4.1.2 Funktions-basierte Ansätze

Betrachte NN als FUnktion: Approximation Sensitivitätsanalyse Verwendung von Gradienten Gradient x Input

4.1.3 Surrogate-/Sampling-basierte Ansätze

Approximiere die Vorhersage lokal LIME(Ribeiro et. al 16) Smoothgrad (Smilkov et al 16)

4.1.4 Struktur-basierte Ansätze

Nutze die durch das Netzwerk gegebene Struktur.
LRP Bach et. al 15 Deep Taylor Decomposition (Montavon et al 17)
Excitation Backprop(Zhang et al.15) Wir erhalten einen Vortzeil gegenüber den Balckbox Anätzen

4.2 Globale Methoden

Während die bisher vorgestellten Methoden den Fokus auf die Erklärbarkeit von Einzelentscheidungen eines Modells richten, befassen sich die Methoden in diesem Abschnitt damit, das Modell als Ganzes zu verstehen. Diese Methoden versuchen allgemeine Muster aus dem Klassifizierungsverhalten des Modells zu extrahieren, indem einzelne Entscheidungen zusammengefasst und anschließend analysiert werden.[30, S. 14]
In diesem Abschnitt werden die Methoden Spectral Relevance Analysis (SpRAy), Feature Visualization, Network Dissection und Testing with Concept Activation Vectors (TCAV) vorgestellt.

Spectral Relevance Analysis (SpRAy) ist eine Weiterentwicklung von LRP. Mit dieser Methode werden zunächst Relevanzklassen für interessante Datensätze und Objektklassen über LRP bestimmt. Die Ergebnisse werden in Heatmaps dargestellt und anschließend über eine Spectralanalyse geclustert. Jedes Cluster entspricht einer durch das Modell erlernten Vorhersagestrategie. Auf diese Weise werden die verwen- deten Vorhersagestrategien für Objekte aus den erzeugten Clustern ermittelt. [LWB$^+$19][S.3ff] Mit dieser Methoden lassen sich Schwachstellen in Datensätzen und Modellen erkennen. Es wird zum Beispiel erkannt, wenn eine Klassifizierung aufgrund der Metadaten eines Bildes vorgenommen wurde.

Feature Visualization arbeitet sich ebenso schrittweise durch das Neuronale Netz, um zu verstehen, wie das Modell sein Verständnis über das Ausgangsbild erstellt. Es wird erkundet, welche Eingabedaten die Ursachen für ein bestimmtes Verhalten (zum Beispiel eine Neuronen Aktivierung oder die endgültige Ausgabe)

verantwort- lich sind. Über eine Optimierungstechnik wird ein Verständnis darüber aufgebaut, wonach ein Modell sucht, dies können zum Beispiel Neuronen, Kanäle, Layer oder Klassenwahrscheinlichkeiten sein. [23]

Network Dissection ermöglicht zu verstehen, was in den einzelnen Layern eines vielschichtigen Convolutional Neural Networks (CNN) geschieht. Die Methode gleicht die Ausgabe jedes Layers mit visuellen semantischen Konzepten eines Vergleichsdatensatzes ab. Zum Vergleich wird der Broden-Datensatz eingesetzt, der viele Bilder und Konzepte enthält. Dieses Vorgehen ermöglicht die Zuordnung von Konzepten aus der realen Welt (zum Beispiel: unterschiedliche Materialien, Farben, Oberflächenstrukturen, Objekte, Szenen) zu jeder Schicht des CNN. Auf diese Weise werden die verborgenen Bereiche eines CNN interpretierbar gemacht werden und es können Erkenntnisse über den hierarchischen Aufbau des CNNs erlangt werden. [**?**][S. 1 und 12]

Für die Detektion von Poisoning-Angriffen werden die Layer-wise Relevance-Propagation sowie die Spektrale Relevanz-Analyse verwenden. Deshalb gehen wir im folgenden Kapitel nähers auf die Layer-wise Relevance Propagation ein.

5 Layer-wise Relevance Propagation

In diesem Abschnitt stellen wir die Layer-wise Relevance Propagation vor, die für einzelne Eingabebilder eine Relevanzkarte (oder: Heatmap) erstellt, die im Fall eines Bildes beispielsweise die Bereiche, die wichtiger für die resultierende Ausgabe des Netzwerkes war, farblich markiert(eher nur die Zuordnung eines Relkevanz-Wertes, die farbliche Markierung ist nur eine Methode der Visualisierung). Wie bereits oben erwähnt gehört dieses Verfahren demnach zu den Lokalen Methoden.

5.1 Idee

Die Layer-wise Relevance Propagation (LRP) wird in [BBM$^+$15] erstmalig vorgestellt. Die Idee besteht darin, einen Zusammenhang zwischen der Ausgabe eines Klassifikators $f_\theta : \mathbb{R}^d \to \mathbb{R}^+$ und der Eingabe x herzustellen. Dabei wird eine definiert, die über gewisse Eigenschaften eingeschränkt wird. Die Autoren bezeichnen die Herangehensweise hier selbst als heuristisch und liefern in **??** eine Verallgemeinerung des Konzepts, die gleichzeitig die mathematische Grundlage bildet.

Wir betrachten eine nicht-negative Funktion $f : \mathbb{R}^d \to \mathbb{R}^+$. Im Bereich der Bild-Klassifizierung ist die Eingabe $x \in \mathbb{R}^d$ ein Bild, das wir als Menge von Pixelwerten $x = \{x_p\}$ auffassen können. Dabei beschreibt der Index p einen genauen Pixelpunkt. Während für schwarz-weiß Bilder $x_p \in \mathbb{R}$ gilt, gilt im Fall von RGB-Bildern $x_p \in \mathbb{R}^3$ für die einzelnen Farbkanäle Rot, Grün und Blau. Die Funktion $f(x)$ ist ein Maß dafür, wie präsent ein oder mehrere Objekte in der Eingabe/im Eingabebild vorhanden sind. Ein Funktionswert $f(x) = 0$ beschreibt die Abwesenheit. Gilt andererseits $f(x) > 0$, so wird die Präsenz mit einem gewissen Grad an Sicherheit oder eine gewisse Menge zum Ausdruck gebracht.

Mit Hilfe der LRP soll nun jedem Pixel p im Eingabebild eine Relevanz $R_p(x)$ zugeordnet werden, die für jedes Pixel x_p angibt, mit welcher Größe es für das

Entstehen einer Entscheidung $f(x)$ verantwortlich ist. Die Relevanz eines jeden Pixels wird dabei in einer Heatmap $R(x) = \{R_p(x)\}$ zusammengefasst.

Die Heatmap besitzt dieselbe Größe wie x und kann als Bild visualisiert werden.

Wir definieren die folgenden Eigenschaften:

Definition 5.1.1. *Eine Heatmap $R(x)$ heißt konservativ, falls gilt:*

$$\forall x : f(x) = \sum_p R_p(x), \tag{5.1}$$

d.h. die Summe der im Pixelraum zugeordneten Relevanz entspricht der durch das Modell erkannten Relevanz.

Definition 5.1.2. *Eine Heatmap $R(x)$ heißt positiv, falls gilt:*

$$\forall x, p : R_p(x) \geq 0, \tag{5.2}$$

d.h. alle einzelnen Relevanzen einer Heatmap sind nicht-negativ.

Die erste Eigenschaft verlangt, dass die umverteilte Gesamtrelevanz der Relevanz entspricht, mit der ein Objekt im Eingabebild durch die Funktion $f(x)$ erkannt wurde. Die zweite EIgenschaft beschreibt, dass keine zwei Pixel eine gegensätzliche Aussage über die Existenz eines Objektes treffen können. Beide Definitionen zusammen ergeben die Definition einer *konsistenten* Heatmap:

Definition 5.1.3. *Eine Heatmap $R(x)$ heißt konsistent, falls sie konservativ und positiv ist, d.h Definition 5.1.1 und Definition 5.1.2 gelten.*

Für eine konsistente Heatmap gilt dann ($f(x) = 0 \Rightarrow R(x) = 0$), d.h. die Abwesenheit eines Objektes hat zwangsläufig auch die Abwesenheit jeglicher Relevanz in der Eingabe zur Folge, eine Kompensation durch positive und negative Relevanzen ist folglich nicht möglich.

Bemerkung 5.1.4. *Die geforderten Eigenschaften an eine Heatmap definieren diese nicht eindeutig. Es sind also mehrere Abbildungen möglich, die die genannten Forderungen erfüllen. Beispiele dafür sind eine natürliche Zerlegung und Taylor-Zerlegungen [MLB+17a].*

Die LRP liefert nun ein Konzept, mit dem eine Zerlegung

$$f(x) = \sum_d R_d \tag{5.3}$$

bestimmt werden kann.
TODO: Summenabfolge von layer zu layer einfügen

Wir gehen nun davon aus, dass die Funktion f ein NN repräsentiert, dass aus mehreren Schichten mit mehreren Neuronen pro Schicht und dazwischengeschalteten nicht-linearen Aktivierungsfunktionen aufgebaut ist. Die erste Schicht ist die Eingabe-Schicht, bestehend aus den Pixeln eines Bildes. Die letzte Schicht ist die reellwertige Ausgabe von f. Die l-te Schicht ist durch einen Vektor $z = (z_d^l)_{d=1}^{V(l)}$ der Dimension $V(l)$ dargestellt. Sei also eine Relevanz $R_d(l+1)$ für jede Dimension $z_d^{(l+1)}$ des Vektors z in der Schicht $l+1$ gegeben. Die Idee besteht nun darin, eine

Relevanz $R_d^{(l)}$ für jede Dimension $z_d^{(l)}$ des Vektors z in der Schicht l zu finden, die einen Schritt näher an der Eingabeschicht liegt, sodass die folgende Abfolge von Gleichungen gilt:

$$f(x) = ... = \sum_{d \in l+1} R_d{}^{(l+1)} = \sum_{d \in l} R_d{}^{(l)} = ... = \sum_d R_d^{(1)}. \tag{5.4}$$

Für diese Funktion benötigen wir eine Regel, mit der die Relevanz eines Neurons einer höheren Schicht $R_j^{(l+1)}$ auf ein Neuron einer benachbarten, näher an der Eingabeschicht liegendes Neuron, übertragen werden kann. Die Übertragung der Relevanz zwischen zwei solchen Neuronen wird mit $R_{i \leftarrow j}$ bezeichnet. Auch hier muss die übertragene Relevanz erhalten bleiben. Es wird also gefordert:

$$\sum_i R_{i \leftarrow j}^{(l,l+1)} = R_j^{(l+1)}. \tag{5.5}$$

D.h. die gesamte Relevanz eines Neurons der Schicht $l+1$ verteilt sich komplett auf alle Neuronen der Schicht l. Im Falle eines linearen NN $f(x) = \sum_i z_{ij}$ mit der Relevanz $R_j = f(x)$ ist eine Zerlegung gegeben durch $R_{i \leftarrow j} = z_{ij}$. Im allgemeineren Fall ist die Neuronenaktivierung x_j eine nicht-lineare Funktion abhängig von z_j.
Für die beiden Aktivierungsfunktionen $tanh(x)$ und $ReLU(x)$ - beide monoton wachsend mit $g(0) = 0$ - bieten die Vor-Aktivierungen noch immer ein sinnvolles Maß für den relativen Beitrag eines Neurons x_i zu R_j (müsste das nicht umgekehrt sein, die INdizes?!?!).

Eine erste Mögliche Relevanz-Zerlegung, basierend auf dem Verhältnis zwischen lokalen und globalen Vor-Aktivierung, ist gegeben durch:

$$R_{i \leftarrow j}^{(l,l+1)} = \frac{z_{ij}}{z_j} \cdot R_j^{(l+1)}. \tag{5.6}$$

Für diese Relevanzen $R_{i \leftarrow j}$ gilt die Erhaltungseigenschaft 5.4, denn:

$$\sum_i R_{i \leftarrow j}{}^{(l,l+1)} = R_j^{l+1} \cdot (1 - \frac{b_j}{z_j}). \tag{5.7}$$

Dabei steht der rechte Faktor für die Relevanz, die durch den Bias-Term absorbiert wird. Falls notwendig, kann die verbleibende Bias-relevanz auf jedes Neuron x_i verteilt werden(?,s.Abschnitt über Biases Promotion, S.Lapuschkin).

Diese Regel wird in der Liteartur als LRP-0 bezeichnet. Ein Nachteil dieser ist, dass die Relevanzen $R_{i \leftarrow}$ für kleine globalen Voraktivierung z_j beliebig große Werte annehmen können.

Um dies zu verhindern, wird in der LRP-ε-Regel ein vorher festgelegter Parameter $\varepsilon > 0$ eingeführt:

$$R_{i \leftarrow j}^{(l,l+1)} = \begin{cases} \frac{z_{ij}}{z_j + \varepsilon} \cdot R_j^{(l+1)}, & z_j \geq 0 \\ \frac{z_{ij}}{z_j - \varepsilon} \cdot R_j^{(l+1)}, & z_j < 0 \end{cases} \tag{5.8}$$

In [BBM+15] wird die Layer-wise Relevance Propagation erstmalig vorgestellt. Zudem wird eine Taylor Zerlegung präsentiert, die eine Approximation der LRP darstellt.

Hier[9] werden einige Bereiche vorgestllt, in denen LRP angewendet wurde.

[9] https://towardsdatascience.com/indepth-layer-wise-relevance-propagation-340f95deb1ea

5.2 Behandlung von biases

5.3 Beispiel an einem kleinen Netzwerk

5.4 Deep Taylor Decomposition

Mathematischer Hintergrund für LRP.LRP als Spezialfall von DTD

5.4.1 Taylor Decomposition

We will assume that the function f (x) is implemented by a deep neural network, composed of multiple layers of representation, where each layer is composed of a set of neurons. Each neuron performs on its input an elementary computation consisting of a linear projection followed by a nonlinear activation function. Deep neural networks derive their high representational power from the interconnection of a large number of these neurons, each of them, realizing a small distinct subfunction.

Laut [MLB$^+$17b] ist die in [BBM$^+$15] vorgestellte Layer-wise Relevance Propagation eher heuristisch. In diesem Paper wird nun eine solide theoretische Grundlage geliefert.

DTD liefert den mathematischen Hintergrund für LRP

Simple Taylor decomposition. Finde rootpoints, sodass Erhaltungseigenschaft erhalten bleibt.

Simple Taylor in Practice: funktioniert in der Praxis nicht wirklich.Viel Noise meistens positive Relevanz

Relevanz Propagation: Heatmaps look much cleaner

Simple Taylor:- root point hard to find -gradient shattering. Gradient looses its informative structure in big layer nets

Use Taylor Decomposition to explain LRP from layer to layer

5.4.2 Deep Taylor Decomposition

LRP in verschiedenen Anwendungsgebieten [MBL$^+$19], 10.2. In diesem Paper:LRP-0 schlechter als LRP-ε schlechter alsLRP-γ schlechter als Composite-LRP.

5.5 Verschiedene Verfahren

5.6 Eigenschaften

Beweise in DTD Paper

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

Abbildung 1: (Optischer) Vergleich von korrumpiertem Datenpunkt und berechnter Heatmap. Links: Verkehrsschild der Klasse 'Höchstgeschwindigkeit: 50km/h' versehen mit einem 3x3 Sticker und dem Label 'Höchstgeschwindigkeit: 80km/h'. Rechts: Zugehörige Heatmap bezüglich der Klasse 'Höchstgeschwindigkeit: 80km/h'.

Quelle: [CCB+18]

5.7 Behandlung besonderer Schichten

5.7.1 BatchNorm2D

5.8 LRP für Deep Neural Nets/Composite LRP

5.9 Verarbeitung der Heatmaps

Aktuell benutzte defaul colormap ist Option D (Viridis)[10]

- Wertebereich
- Interpretation
- Skalen
- Normalisierung

5.10 Implementierungen

5.10.1 Tensorflow

5.10.2 pytorch

Allgemeines Tutorial:[11]
pytorch-LRP für VGG16 wird vorgestellt.

GiorgioML[12]:
Alternative pytorch-Implementierung basierend auf Tensorflow paper.

[10] `https://bids.github.io/colormap/`
[11] `https://git.tu-berlin.de/gmontavon/lrp-tutorial`
[12] `https://giorgiomorales.github.io/Layer-wise-Relevance-Propagation-in-Pytorch/`

moboehle[13]:
Der code entstand im Rahmen der Forschungsarbeit [BEWR19], in der eine Alzheimer-Festellung aufgrund von Bilddaten(scans?) vorgenommen wird. Framework leicht anpasspar. Benutzt pytorch hooks. Unterstützte Netzwerkschickten[14]:

```
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


```

Listing 2: Verfügbare Schichten und Aktivierungsfunktionen

fhvilshoj[15]:

LRP für linear und Convolutional layers

- Die Klassen
  torch.nn.Sequential, torch.nn.Linear und torch.nn.Conv2d werden erweitert, um autograd für die Berechnung der Relevanzen zu berechnen.
- Ausgabe der Relevanzen von Zwischenschichten ist möglich
- : Implementierte Regeln: epsilon Regeln mit epsilon=1e-1, gamma-regel mit gamma=1e-1. alphabeta-Reagel mit a1b0 und a2b1
- Netz muss hier umgeschrieben werden, sodass die Anwendung des Algorithmus möglich wird.

```

conv2d = {
"gradient":             F.conv2d,
"epsilon":              Conv2DEpsilon.apply,
"gamma":                Conv2DGamma.apply,
"gamma+epsilon":        Conv2DGammaEpsilon.apply,
"alpha1beta0":          Conv2DAlpha1Beta0.apply,
"alpha2beta1":          Conv2DAlpha2Beta1.apply,
"patternattribution":   Conv2DPatternAttribution.apply,
"patternnet":           Conv2DPatternNet.apply,
}

```

[13] https://github.com/moboehle/Pytorch-LRP
[14] https://github.com/moboehle/Pytorch-LRP/blob/master/inverter_util.py
[15] https://github.com/fhvilshoj/TorchLRP

```
13
```

Listing 3: Implementierte Regeln fhvilshoj

Zennit:[16] Zennit (Zennit explains neural networks in torch)

- Modell wird mithilfe eines Canonizers so aufbereitet, dass LRP möglich wird
- Backward pass wird modifiziert, um Heatmaps zu erhalten.
- VGG- und ResNet-Beispiel

6 Detektion von Poisoning-Angriffen basierend auf LRP

In diesem Kapitel stellen wir das Verfahren vor, mit dem wir die Präsenz von Poisoning-Angriffen detektieren und die korrumpierten Datenpunkte aus dem Datensatz entfernen können.

Die Idee besteht darin, ein kMeans-Clustering auf einer Klasse durchzuführen, die für verdächtig gehalten wird.

Wichtig für das Clustering sind hierbei die Repräsentation der Daten, auf denen geclustert wird, sowie die Metriken, die eine Distanz und einen Mittelwertbegriff für mehrere Datenpunkte definieren.

Als Repräsentation benutzen wir die Heatmaps, die mit der LRP, wie in Abschnitt 5 beschrieben, erzeugt werden.

Für das kMeans-Clustering müssen wir im Vorfeld eine Clusteranzahl k fixieren und die Mittelpunkte der k Cluster initialisieren.

In Lloyd's Formulierung [Llo82] wird mit einer gleich-verteilten zufälligen Initialisierung von k Cluster-Zentren begonnen. Jeder Punkt im Datensatz wird anschließend dem nächstgelegenen Cluster-Zentrum zugeordnet. Für jedes Cluster wird anschließend ein neues Zentrum als Mittelwert berechnet. Diese beiden Schritte aus Zuordnung und Mittelpunktberechnung werden so lange wiederholt, bis sich die Cluster-Zentren nicht mehr ändern oder eine maximale Iterationszahl erreicht ist. Dieses Vorgehen wird als k-Means bezeichnet.

Eine Variation des k-Means-Algorithmus, der sowohl die Laufzeit als auch die Genauigkeit verbessert, ist der sogenannte $k - means + +$-Algorithmus [AV06]. Dabei wird die Initalisierung in abgewandelter Form durchgeführt. Nach gleichverteilter, zufälliger Bestimmung des ersten Cluster-Zentrums, werden die übrigen $k - 1$ Zentren wie folgt gewählt: Die Wahl erfolgt proportional zur maximalen quadratischen Distanz zu allen vorher bestimmten Cluster-Zentren. Damit sind die Zentren so über den Datensatz verteilt, dass sie maximal weit auseinander liegen.

Anschließend werden die im kMeans-Algorithmus üblichen Schritte aus Distanzberechnung und Mittelwert-Berechnung wiederholt.

Dieses $kMeans + +$-CLustering ist nochmals in

zusammengefasst.

Für die Bestimmung der Clusteranzahl k benutzen wir die Spektrale Relevanz-Analyse.

[16] `https://github.com/chr5tphr/zennit`

6.1 Idee

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

- Berechnung der Heatmaps mithilfe der LRP
- Berechnung einer Distanzmatrix basierend auf L^2- oder GMW-Distanz
- Spektrale Relevanzanalyse (Bestimmung der verschiedenen Cluster innerhalb einer Klasse)
- kMeans-Clustering zur Bestimmung der korrumpierten Datenpunkte

Diese werden wir im Folgenden näher betrachten.

Berechnung der Heatmaps mithilfe der LRP. Für die LRP wählen wir innmodel = InnvestigateModel(model.net, lrpexponent=2, method='e-rule', beta=0.5) und normalisieren die Heatmap anschließend.

The theory of optimal transport generalizes thatintuition in the case where, instead of moving only one item at a time, one is concernedwith the problem of moving simultaneously several items (or a continuous distributionthereof) from one configuration onto another. [com19]

6.2 k-means / k-means++ -Clustering

Das k-means-Clustering gehört zu den unüberwachten Clustering-Verfahren. In der ursprünglichen Formulierung sind n Datenpunkte $x_1, ..., x_n \in \mathbb{R}^d$ und $k \in \mathbb{Z}_{>0}$ gegeben. Die Datenpunkte sollen so auf, die einzelnen Cluster verteilt werden, dass die L^2-Distanzen innerhalb eines Clusters zum Cluster-Zentrum minimal sind.

Baryzentrische Koordinaten [17]

6.3 Spektrales Clustering

Wir folgen [VL07]. Gegeben:Datenpunkte $x_i, ..., x_n$ sowie eine Größe $s = s_{ij} \in \mathbb{R}^+$, die einen paarweisen Zusammenhang der einzelnen Punkte beschreiben.

Ziel: Aufteilen der Punkte in verschiedene Cluster, sodass sich Punkte innerhalb eines Clusters ähnlich bezüglich s sind.

Alternative Repräsentation der Daten mithilfe eines Ähnlichkeitsgraphen $G = (V, E)$ möglich.

Umformulierung des Clustering-Problems mithilfe des Ähnlichkeitsgraphen: Finde Pertitionierung des Graphen, sodass die Kanten-Gewichte innerhalb einer Gruppe niedrig (niedriges Gesamtgewicht?) und außerhalb einer Gruppe große sind.

Graph-Notationen:

Verschiedene Konstruktionsmöglichkeiten von Ähnlichkeitsgraphen:

- ε-Nachbarschaft-Graph
- kNN-Graph
- fully connected graph

[17] `https://de.wikipedia.org/wiki/Baryzentrische_Koordinaten`

6.4 Anwendung auf unterschiedliche Poisoning-Angriffe

Berechnung der Relevanzen:

Wir berechnen die Relevanzen jedes einzelenen Eingabebildes klassenweise, d.h. besitzt eine Eingabe das Label y, so berechnen auf einem trainierten Netzwerk, für jeden Pixelwert der Eingabe, wie relevant dieser für die Ausgabe f(x) = y ist.
Wir summieren über die Farbaxen des Bildes, um einzelne Relevanzen pro Pixelpunkt zu erhalten.
Für die Berechnung der Relevanzen benutzen wir eine modifizierte Version des im Rahmen von [BEWR19] entstandenen Programmcodes[18].

Vorverarbeitung der Relevanzen:
In [LWB+19] wird anschließend ein Sum-Pooling auf die Relevanzen angewendet, um eine Dimensionsreduktion zu erhalten. Wie in [AMN+19] verzichten wir auf eine weitere Dimensionsreduktion, da wir nur relativ kleine Relevanzen der Größe $32x32$ verarbeiten.

Für eps=5e-2 liegen beide barycentren identisch weit weg. Probiere nun eps=5e-3
Berechnung der Distanzen und Aufstellen einer Affinitätsmatrix:

Wir berechnen zunächst eine Distannzmatrix, die die paarweisen Distanzen aller Heatmaps einer Klasse enthält.

Für die Berechnung der euklidischen Distanz betrachten wir Heatmaps x, y der Größe $32x32$ als Elemente $x, y \in \mathbb{R}^{32*32}$ Die Distanz lässt sich dann wie in **??** berechnen.

Die Gromov-Wasserstein-Distanz lässt sich wie in [PCS16] angegeben berechnen.

Barycenters Definition und Vergleich zum euklidischen Raum [AC11]
In einer Affinitätsmatrix oder Ähnlichkeitsmatrik sind die

Berechnung Spektralen Einbettung:

Dimensionsreduktion vor dem Clustering ?!

In [CCB+18] wird beispielsweise eine Dimensionsreduktion mit PCA durchgeführt.
k-Means-Clustering:

Bemerkung 6.4.1. *In [AMN+19] Kapitel '2.3. Fisher Discriminant Analysis for Clever Hans identification' wird ein Verfahren vorgestellt, mit dem verdächtige Klassen indentifiziert werden können. Für diese würde man anschließend das obige Verfahren durchführen*

[18] `https://github.com/moboehle/Pytorch-LRP`

6.5 Verwendete Distanzen & Approximationen

Um die Struktur innerhalb einer Klasse zu analysieren, benötigen wir eine Metrik. Anhand dieser wird abhängig von den Heatmps einer Klasse eine Affinitätsmatrix berechnet, die dann anschließend zur Berechnung der Spektralen Einbettung als wichtigster Schritt von SpRAy verwendet wird. Wir wollen dazu die im Folgenden vorgestellten Metriken verwenden.
Wie in [AMN+19] summieren wir über die Farbkanäle, um einen einzelnen Relevanzwert pro Pixelpunkt zu erhalten. Wir benötigen also eine Metrik zur Berechnung der Distanz zwischen 32x32 großen Heatmaps.
Wir normalisieren die Relevanzen zusätzlich auf das Intervall $[0, 1]$.
Die Wahl der Pixel mit 99 Prozent der Gesamtmasse und anschließende Normalisierung wird vermutlich durchgeführt, um die Bedingung **??** zu erhalten.

A Verwendete Netzwerke

Aufbau dieses Netzwerkes: 1. Inception-Modul 2. [pool1, batchConv1, pool2, batchConv2, pool3, batchConv3, pool4] 3. Drei Lineare Schichten mit ReLu und Dropout dazwischen

Im Unterschied zum offiziellen Inception Netz(v1v2v3) gibt es in dieser vereinfachten Version keinen ßtemäus convs, es geht direkt mit InceptionA los.

Wie ähnlich sind sich InceptionA(hier) und das offizielle InceptionA-Modul?

B Parameter für Training und Einlesen der Daten

Die in [AWN+20] gewählten Parameter wären ein guter Ausgangspunkt.
Für das Einlesen der Daten benutzen wir, sofern nicht weiter angegeben die folgenden Augmentierungen:

```
__train_transform = transforms.Compose(
[
transforms.RandomResizedCrop((image_size, image_size),
scale=(0.6, 1.0)),
transforms.RandomRotation(degrees=15),
transforms.ColorJitter(brightness=0.1, contrast=0.1,
saturation=0.1, hue=0.1),
transforms.RandomAffine(15),
transforms.RandomGrayscale(),
transforms.Normalize( mean=[0.485, 0.456, 0.406],
std=[0.229, 0.224, 0.225]),
transforms.ToTensor()

]


```

Listing 4: Augemntierung beim Einlesen der Daten

Die Werte von mean und std variieren für alle ausgeführten Poisoning-Angriffe. Anstatt beide jedes Mal erneut zu berechnen, verwenden wir die von pytorch angegebenen Werte [19], die für die vor-trainierten Modelle empfohlen werden und auf dem Datensatz ImageNet[20] basieren.
Wir trainieren die Netzwerke über maximal 100 Epochen und benutzen *early stopping* mit einer *patience* = 20. Die verwendete Implementierung ist eine modifizierte Version von Bjarte Mehus Sunde [21], die wiederum auf PyTorch Ignite[22] basiert.

[19] https://github.com/pytorch/examples/blob/97304e232807082c2e7b54c597615dc0ad8f6173/imagenet/main.py#L197-L198

[20] https://image-net.org/

[21] https://github.com/Bjarten/early-stopping-pytorch

[22] https://github.com/pytorch/ignite/blob/master/ignite/handlers/early_stopping.pyt

C Einlesen der Daten bei AC

Ohne Transformationen, wie den Testdatensatz.

D Parameter für die ausgeführten Angriffe

Für das projizierte Gradientenverfahren benutzen wir 10 Iterationen und eine Schrittweite von 0.015.

Label-konsistente Poisoning-Angriffe:

E Datensätze

GTSRB[23] Datensatz Splitting (Train; Val, test)

ImageNet besteht über 14 Millionen Bildern in 100 Klassen.

F Programmcode

Der vollständige Programmcode ist verfügbar unter `https://github.com/lukasschulth/MA-Detection-of-Poisoning-Attacks`

G Notizen

registered spaces [24] Barycenters in the Wasserstein Space[25]
Was passiert bei der Kombination von 2 verschiedenen Triggern? Einmal keine Überlappung(d.h. 2verschiedene Trigger auf dem selben Bild) vs. auch beide Trigger auf einem Bild ist zulässig.

[23] `https://benchmark.ini.rub.de/gtsrb_dataset.html`
[24] `https://arxiv.org/pdf/1809.06422.pdf`
[25] `https://arxiv.org/pdf/1809.06422.pdf`

Literatur

[AC11] Martial Agueh and Guillaume Carlier. Barycenters in the wasserstein space. *SIAM Journal on Mathematical Analysis*, 43(2):904–924, 2011.

[AMN+19] Christopher J Anders, Talmaj Marinč, David Neumann, Wojciech Samek, Klaus-Robert Müller, and Sebastian Lapuschkin. Analyzing imagenet with spectral relevance analysis: Towards imagenet un-hans' ed. *arXiv preprint arXiv:1912.11425*, 2019.

[AV06] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

[AWN+19] Christopher J. Anders, Leander Weber, David Neumann, Wojciech Samek, Klaus-Robert Müller, and Sebastian Lapuschkin. Finding and removing clever hans: Using explanation methods to debug and improve deep models, 2019.

[AWN+20] Christopher J Anders, Leander Weber, David Neumann, Wojciech Samek, Klaus-Robert Müller, and Sebastian Lapuschkin. Finding and removing clever hans: Using explanation methods to debug and improve deep models. 2020.

[BBM+15] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[BBM+16] Alexander Binder, Sebastian Bach, Gregoire Montavon, Klaus-Robert Müller, and Wojciech Samek. Layer-wise relevance propagation for deep neural network architectures. In Kuinam J. Kim and Nikolai Joukov, editors, *Information Science and Applications (ICISA) 2016*, pages 913–922, Singapore, 2016. Springer Singapore.

[BEWR19] Moritz Böhle, Fabian Eitel, Martin Weygandt, and Kerstin Ritter. Layer-wise relevance propagation for explaining deep neural network decisions in mri-based alzheimer's disease classification. *Frontiers in aging neuroscience*, 11:194, 2019.

[CCB+18] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018.

[com19] Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5-6):355–607, 2019.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[Llo82] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[LWB+19] Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Unmasking clever hans predictors and assessing what machines really learn. *Nature communications*, 10(1):1–8, 2019.

[MBL+19] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. Layer-wise relevance propagation: an overview. *Explainable AI: interpreting, explaining and visualizing deep learning*, pages 193–209, 2019.

[MLB+17a] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

[MLB+17b] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

[MMS+17] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[PCS16] Gabriel Peyré, Marco Cuturi, and Justin Solomon. Gromov-wasserstein averaging of kernel and distance matrices. In *International Conference on Machine Learning*, pages 2664–2672. PMLR, 2016.

[PMG16] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[SZS+13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[TLM18] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. *arXiv preprint arXiv:1811.00636*, 2018.

[TTM19] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks, 2019.

[VL07] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.