Untersuchung & Entwicklung von Ansätzen zur Detektion von Poisoning-Angriffen

Master-Arbeit

Lukas Schulth
lukas.schulth@uni.kn

1. Oktober 2021

Erstkorrektor, Zweitkorrektor, Betreuer, Fachbereich Mathematik und Statistik, Masterarbeit zur Erlangung des Titels Master of Science (M.Sc.)
vgl. mit Titelblatt Exposé(!)

Abstract

english

Zusammenfassung

deutsch

Keywords— one, two, three, four

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Algorithmenverzeichnis

Inhaltsverzeichnis

Algorithm 1 Foo bar
...

1 Einführung

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 Abschnitt 6, 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 Abschnitt 8 durchgeführt.

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

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

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

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

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

```
1    nn.Conv2d(in_channels=in_channels, out_channels=
     out_channels, **kwargs)
2    nn.BatchNorm2d(num_features=out_channels)
3    nn.ReLU()
4
```

Listing 1: python-interner Aufbau einer BatchConv Schicht@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

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

Batch Normalization[3]

[3] `https://arxiv.org/pdf/1502.03167.pdf`

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[4]

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

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.

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

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

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?

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 Projizierten 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.*

3.4.3 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. @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

3.4.4 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.

3.4.5 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

Erklärbarkeit vs. Interpretierbarkeit, youtube talk?!

4.1 Lokale Methoden

4.2 Globale Methoden

5 Layer-wise Relevance Propagation

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

[5] 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

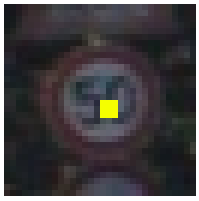
LRP in verschiedenen Anwendungsgebieten [MBL$^+$19], 10.2. In diesem Paper:LRP-0 schlechter als LRP-ε schlechter alsLRP-γ schlechter als Composite-LRP.

5.5 Verschiedene Verfahren

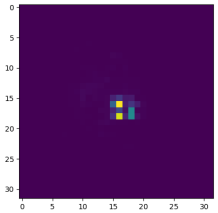
5.6 Eigenschaften

Beweise in DTD Paper

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



(a) Verkehrsschild der Klasse 'Höchstgeschwindigkeit: 50km/h' versehen mit einem 3x3 Sticker und dem Label 'Höchstgeschwindigkeit: 80km/h'



(b) Zugehörige Heatmap bezüglich der Klasse 'Höchstgeschwindigkeit: 80km/h'

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

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)[6]

- Wertebereich
- Interpretation
- Skalen
- Normalisierung

5.10 Implementierungen

5.10.1 Tensorflow

5.10.2 pytorch

Allgemeines Tutorial:[7]
pytorch-LRP für VGG16 wird vorgestellt.

GiorgioML[8]:
Alternative pytorch-Implementierung basierend auf Tensorflow paper.

moboehle[9]:
Der code entstand im Rahmen der Forschungsarbeit [BEWR19], in der eine Alzheimer-Festellung aufgrund von Bilddaten(scans?) vorgenommen wird. Framework leicht

[6] `https://bids.github.io/colormap/`
[7] `https://git.tu-berlin.de/gmontavon/lrp-tutorial`
[8] `https://giorgiomorales.github.io/Layer-wise-Relevance-Propagation-in-Pytorch/`
[9] `https://github.com/moboehle/Pytorch-LRP`

anpasspar. Benutzt pytorch hooks. Unterstützte Netzwerkschickten[10]:

```
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[11]:

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,
}


```

Listing 3: Implementierte Regeln fhvilshoj

[10] https://github.com/moboehle/Pytorch-LRP/blob/master/inverter_util.py
[11] https://github.com/fhvilshoj/TorchLRP

Zennit:[12] 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

Super Einführung in Wasserstein und OT: [AWR17]
Computing distances between probability measures on metric spaces, or more generally between point clouds, plays an increasingly preponderant role in ma- chine learning [SL11, MJ15, LG15, JSCG16, ACB17], statistics [FCCR16, PZ16, SR04, BGKL17] and computer vision [RTG00, BvdPPH11, SdGP+15]. A promi- nent example of such distances is the earth mover's distance introduced in [WPR85] (see also [RTG00]), which is a special case of Wasserstein distance, or optimal transport (OT) distance [Vil09]. While OT distances exhibit a unique ability to capture geometric features of the objects at hand, they suffer from a heavy computational cost that had been prohibitive in large scale applications until the recent introduction to the ma- chine learning community of Sinkhorn Distances by Cuturi [Cut13]. Combined with other numerical tricks, these recent advances have enabled the treatment of large point clouds in computer graphics such as triangle meshes [SdGP+15] and high-resolution neuroimaging data [GPC15]. Sinkhorn Distances rely on the idea of entropic penalization, which has been implemented in similar problems at least since Schr odinger [Sch31, Leo14]. This powerful idea has been success- fully applied to a variety of contexts not only as a statistical tool for model

How Well Do WGANs Estimate the Wasserstein Metric? [**?**]

6.1 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^2- 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 bestimmer NEtzwerkschichten hinzugenommen werden.

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]

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

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

Beispiel-Implementierung[13]

Baryzentrische Koordinaten [14]

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

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[15].

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.

[13] `https://towardsdatascience.com/k-means-implementation-in-python-and-spark-856e7eb5fe9b`
[14] `https://de.wikipedia.org/wiki/Baryzentrische_Koordinaten`
[15] `https://github.com/moboehle/Pytorch-LRP`

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 Unterunterabschnitt 6.5.2 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*

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 $[0, 1]$.

6.5.1 Histogramme & Maße

Definition 6.5.1 (Histogramm)**.** *Als Histogramm (oder: Wahrscheinlichkeitsvektor) bezeichnen wir ein Element $\boldsymbol{a} \in \Sigma_n$, das zum folgenden Wahrscheinlichkeits-Simplex gehört:*

$$\Sigma_n := \{\boldsymbol{a} \in \mathbb{R}_+^n | \sum_{i=1}^{n} \boldsymbol{a_i} = 1\}$$

Einführung W-Theorie am KIT [16]

Definition 6.5.2 (Kopplungen zwischen Histogrammen)**.** *Seien die beiden Histogramme $p \in \Sigma_{N_1}$ und $q \in \Sigma_{N_2}$ gegeben. Als Menge der Kopplungen zwischen beiden Histogrammen definieren wir*

$$\mathcal{C}_{p,q} := \{T \in (\mathbb{R}_+)^{N_1 \times N_2} | T\mathbb{1}_{N_2} = p, T^\top \mathbb{1}_{n_1} = q\},$$

wobei $\mathbb{1} := (1, ..., 1)^\top \in \mathbb{R}^N$ gilt.

Definition 6.5.3 (Distanz)**.** *Eine Distanz*

Definition 6.5.4. *Eine Metrik*

[Metrik]

6.5.2 Euklidische Distanz

Für den Fall der euklidischen Distanz schreiben wir die Relevanzwerte pro Pixel als Vektor und berechnen die Distanz zweier Heatmaps x und y wie folgt[17]:

$$d_{x,y} = \sqrt{\sum_{i=1}^{32*32} (x_i - y_i)^2}.$$

6.5.3 Gromov-Hausdorff-Distanz

Definition 6.5.5 (Hausdorff-Distanz)**.** *Sei (M, d) ein metrischer Raum. Für Seien $X, Y \subset (M, d)$ definieren wir die Hausdorff-Distanz $d_H(X, Y)$ als*

$$d_H(X, Y) = \max\{\sup_{x \in X} \inf_{y \in Y} d(x, y), \sup_{y \in Y} \inf_{x \in Y} d(x, y)\}. \tag{6.1}$$

Definition 6.5.6 (Metrischer Maßraum)**.** *Ein metrischer Maßraum ist ein Tripel (X, d_x, μ_X), wobei (X, d_X) ein metrischer Raum und μ_X ein borelsches W-Maß auf X ist.*

Definition 6.5.7 (Correspondance)**.** *content...*

Definition 6.5.8 (Kopplung)**.** *Seien*

Seien (X, d_X) und (Y, d_Y) zwei metrische Räume. Wir betrachten im Folgenden die Abbildung

$$\Gamma_{X,Y} : (X \times Y) \times (X \times Y) \to \mathbb{R}^+, \tag{6.2}$$

gegeben durch
$\Gamma_{X,Y}(x, y, x', y') := |d_X(x, x') - d_Y(y, y')|$.

Definition 6.5.9 (Gromov-Hausdorff-Distanz)**.** *Für die metrischen Räume (X, d_X) und (Y, d_Y) ist die Gromov-Hausdorff-Distanz definiert als*

$$d_{\mathcal{GH}} = \frac{1}{2} \inf_R ||\Gamma_{X,Y}||_{L^\infty(R \times R)}. \tag{6.3}$$

[16] `https://www.math.kit.edu/stoch/~henze/media/wt-ss15-henze-handout.pdf`

[17] `https://paulrohan.medium.com/euclidean-distance-and-normalization-of-a-vector-76f7a97abd9`

Definition 6.5.10 (Entropischer Optimaler Transport)**.** *Wir definieren den n-dimensionalen Zufalls-Simplex als* $\Sigma_n := \{a \in \mathbb{R}_+^n : \sum_{i=1}^n a_1 = 1$. *Ein Element* $a \in \Sigma_n$ *bezeichnen wir als Histogramm oder Zufallsvektor.*

Definition 6.5.11 (Diskretes Maß)**.**

6.5.4 Optimaler Transport (Monge Formulierung)

Bemerkung 6.5.12. *Problem zwischen diskreten Maßen*

$$\min_T \{\sum_i c(x_i, T(x_i)) : T_\# \alpha = \beta\} \tag{6.4}$$

Bemerkung 6.5.13 (Fehlende Eindeutigkeit)**.**

Definition 6.5.14 (Push-forward Operator)**.** $\beta(B) = \alpha(\{x \in \mathcal{X} : T(x) \in B\}) = \alpha(T^{-1}(B))$

Wir könne das Monge Problem wie folgt auf den Fall zweier beliebiger W-Maße (α, β) erweitern.

Definition 6.5.15 (Monge Problem zwischen beliebigen Maßen)**.** *Seien* α *und* β *zwei W-Maße mit Support auf den Räumen* $\mathcal{X}$ *bzw.* $\mathcal{Y}$*, die durch* $T : \mathcal{X} \to \mathcal{Y}$ *verknüpft(? Fomulierung) sind. Dann ist das Monge Problem gegeben durch*

$$\min_T \{\int_{\mathcal{X}} c(x, T(x)) d\alpha(x) : T_\# \alpha = \beta\}. \tag{6.5}$$

Die Bedingung $T_\# \alpha = \beta$ bedeutet, dass die Abbildung T die gesamte Masse mithilfe des Push-forward Operators von α auf β schiebt.

6.5.5 Optimaler Transport nach Kantorovich

Das Optimal Transport Problem nach Kantorovich gehört zu den typischen Optimal Transport Problemen. Es stellt einer Relaxierung der Formulierung von Gaspard Monge[1781], bei der nun auch das Aufteilen von Masse (mass splitting) zulässig ist. In Kantorovichs Formulierung ist eine Kopplung (oder: Transportabblidung) $\boldsymbol{T}$ gesucht, die die Kosten, die bei der Verschiebung eines diskreten Maßes $\boldsymbol{a}$ auf ein anderes diskretes Maß $\boldsymbol{b}$ bezüglich der Kosten $\boldsymbol{M} \in \mathbb{R}^{n_1 \times n_2}$ entstehen, minimiert. Damit $\boldsymbol{T}$ eine Transportabbildung ist, muss $\boldsymbol{T} \in \Gamma(a, b) = \{\boldsymbol{T} \geq \boldsymbol{0}, \boldsymbol{T}\boldsymbol{1}_{n_2} = a, \boldsymbol{T}^T \boldsymbol{1}_{n_1} = \boldsymbol{b}\}$ gelten.

Für den Fall, dass die Grundkosten eine Metrik darstellen, ist auch die optimale Lösung des Optimal Transport Problems wieder eine Metrik [**?**] und definiert die *Wasserstein Distanz*. Das OT Problem ist definiert als

$$W_M(\boldsymbol{a}, \boldsymbol{b}) = \min_{T \in \Pi(\boldsymbol{a}, \boldsymbol{b})} \langle \boldsymbol{T}, \boldsymbol{M} \rangle, \tag{6.6}$$

wobei $\langle \boldsymbol{T}, \boldsymbol{M} \rangle = \sum_{ij} t_{ij} m_{ij}$ gilt.

which is a linear program. The optimization problem above is often adapted to include a regularization term for the transport plan T , such as entropic regularization (Cuturi, 2013) or squared L2. For the entropic regularized OT problem, one

may use the Sinkhorn Knopp algorithm (or variants), or stochastic optimization algorithms. POT has a simple syntax to solve these problems (see Sample 1)

Die Menge der Matrizen $\Pi(\boldsymbol{a}, \boldsymbol{b})$ ist beschränkt und durch $n + m$ Gleichungen gegeben und damit ein konvexes Polytop (die konvexe Hülle einer endlichen Menge von Matrizen). Zudem ist die Formulierung von Kantorovich im Unterschied zu Monges Formulierung immer symmetrisch in dem Sinne, dass $\boldsymbol{P} \in \Pi(\boldsymbol{a}, \boldsymbol{b})$ genau dann gilt, wenn $\boldsymbol{P}^\top \in \Pi(\boldsymbol{b}, \boldsymbol{a})$.

Kantorovichs Optimal Transport Problem lässt sich nun schreiben als

$$L_C(\boldsymbol{a}, \boldsymbol{b}) := \min_{P \in \Pi(\boldsymbol{a}, \boldsymbol{b})} \langle \boldsymbol{C}, \boldsymbol{P} \rangle := \sum_{i,j} \boldsymbol{C}_{i,j} \boldsymbol{P}_{i,j} \tag{6.7}$$

Diese Problem ist ein lineares Problem. Für diese Art von Problemen ist die optimale Lösung nicht notwendigerweise eindeutig. In fact Kantorovich is considered as the inventor of linear pro- gramming. [18]

EXISTENZ & Eindeutigkeit => vgl. Kapitel 3 in [com19]

Definition 6.5.16 (Formulierung für beliebige Maße)**.** *Im Fall beliebiger Maße betrachten wir die Kopplungen $\pi \in \mathcal{M}_+^1(\mathcal{X} \times \mathcal{Y})$, die die gemeinsame Verteilung auf dem Produktraum $\mathcal{X} \times \mathcal{Y}$ ist. Im diskreten Fall verlangen wir, dass das Produktmaß die Form $\pi = \sum_{i,j} \boldsymbol{P}_{i,j} \delta_{(x_i, y_j)}$ besitzt. Im Allgemeinen Fall wird die Massenerhaltung als Randbedingung an die gemeinsame W-Verteilung geschrieben:*

$$\Pi(\alpha, \beta) := \{\pi \in \mathcal{M}_+^1(\mathcal{X} \times \mathcal{Y}) : P_{\mathcal{X}_\#} \pi = \alpha \text{ und } P_{\mathcal{Y}_\#} \pi = \beta\}, \tag{6.8}$$

wobei $P_{\mathcal{X}_\#}$ und $P_{\mathcal{Y}_\#}$ die Push-forward Operatoren der Projektionen $P_{\mathcal{X}}(x, y) = x$ und $P_{\mathcal{Y}}(x, y) = y$ sind. Nach Theorem 6.5.14 sind diese Randbedingungen äquivalent zu den Bedingungen $\pi(A \times \mathcal{Y}) = \alpha(A)$ und $\pi(\mathcal{X} \times B) = \beta(B)$ für die Mengen $A \subset \mathcal{X}$ und $B \subset \mathcal{Y}$. Als Verallgemeinerung erhalten wir dann

$$\min_{\pi \in \Pi(\alpha, \beta)} \int_{\mathcal{X} \times \mathcal{Y}} c(x, y) d\pi(x, y). \tag{6.9}$$

Problem Gleichung 6.9 ist ein unendlich-dimensionales lineares Programm über einem Raum von Maßen. Falls $(\mathcal{X}, \mathcal{Y})$ kompakt und c stetig ist, existiert immer eine Lösung.

Beispiel 6.5.17. *Beispiele von Kopplungen*

6.5.6 Monge-Kantorovitch equivalence

The proof of Brenier theorem 1 (detailed in Section 5.3) to prove the existence of a Monge map actually studies Kantorovitch relaxation, and proves that this relaxation is tight in the sense that it has the same cost as Monge problem.[19]

6.5.7 Metrische Eigenschaften

Optimaler Transport definiert eine Distanz zwischen Histogrammen und W-Maßen, sofern die Kostenmatrix gewisse Eigenschaften erfüllt. Optimal Transport kann dabei als naheliegende Idee verstanden werden, um Distanzen zwischen Punkten auf Distanzen zwischen Histogrammen oder Maßen zu verallgemeinern.

[18] OTNotes_campride.pdf
[19] CourseOT_cuturi

Definition 6.5.18 (p-Wasserstein-Distanz auf Σ_n)**.** *Sei $n = m$ und für $p \geq$ gelte $\boldsymbol{C} = \boldsymbol{D}^p = (\boldsymbol{D}^p_{i,j})_{i,j} \in \mathbb{R}^{n\times n}$, wobei $\boldsymbol{D} \in \mathbb{R}^{n\times n}_+$ eine Distanz ist.*
Dann definiert

$$W_p(\boldsymbol{a}, \boldsymbol{b}) := L_{\boldsymbol{D}^p}(\boldsymbol{a}, \boldsymbol{b})^{1/p} \tag{6.10}$$

die p-Wasserstein-Distanz auf Σ_n

Lemma 6.5.19. *W_p ist eine Distanz, d.h. W_p ist symmetrisch, positiv, es gilt $W_p(a, b) = 0$ gdw. $a = b$ und die Dreiecksungleichung*

$$\forall \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \Sigma_n : W_p(\boldsymbol{a}, \boldsymbol{c}) \leq W_p(\boldsymbol{a}, \boldsymbol{b}) + W_p(\boldsymbol{b}, \boldsymbol{c}). \tag{6.11}$$

Beweis. Inhalt... □

Bemerkung 6.5.20 (Der Fall $0 < p \leq 1$)**.** *Für $0 < p \leq 1$ ist auch D^p eine Distanz. Damit ist für $p \leq 1$ $W_p(\boldsymbol{a}, \boldsymbol{b})^p$ eine Distanz au dem Simplex.*

6.5.8 Duale Formulierung

Kurzer Überblick hier[20]

The Kantorovich problem (2.11) is a constrained convex minimization problem, and as such, it can be naturally paired with a so-called dual problem, which is a constrained concave maximization problem. The following fundamental proposition explains the relationship between the primal and dual problems.

6.5.9 Gromov-Wasserstein-Divergenz

Fast Computation of Wasserstein Barycenters[21]

Definition 6.5.21 (Divergenz)**.** *Sei S der Raum aller Wahrscheinlichkeitsverteilungen mit gemeinsamem Support. Dann bezeichnet die Divergenz auf S eine Funktion $D(\cdot||\cdot) : S \times S \to \mathbb{R}$, für die gilt:*

1. *$D(p||q) \geq 0$ f.a. $p, q \in S$*

2. *$D(p||q) = 0$ gdw. $p = q$.*

Definition 6.5.22 (Entropie)**.** *Für $T \in \mathbb{R}^{N\times N}_+$ definieren wir die Entropie als*

$$H(T) := -\sum_{i,j=1}^{N} T_{i,j}(log(T_{i,j}) - 1). \tag{6.12}$$

Definition 6.5.23 (Tensor-Matrix-Multiplikation)**.** *Für einen Tensor $\mathcal{L} = (\mathcal{L}_{i,j,k,l})_{i,j,k,l}$ und eine Matrix $(T_{i,j})_{i,j}$ definieren wir die Tensor-Matrix-Multiplikation als*

$$\mathcal{L} \otimes T := (\sum_{k,l} \mathcal{L}_{i,j,k,l} T_{k,l})_{i,j}. \tag{6.13}$$

[20] `https://arxiv.org/pdf/1609.04767.pdf`
[21] https://arxiv.org/pdf/1310.4375.pdf

6.5.10 Regularisierungen

Numerical Methods: Cuturi's Entropy Regularised Approach. Arguably the biggest de- velopment (at least in recent years) in the computation of optimal transport distances was due to Cuturi's entropy regularised approach. The idea is to use entropy to regularise the distance, then some simple rearrangements reveal this is a Kullback-Liebler divergence. Standard methods, e.g. Sinkhorns algorithm, can then be used to find minimizers of the entropy regularised distance.[22]

6.5.11 Entropisch Regularisierte Gromov-Wasserstein-Distanz

Entropic Regularization of Optimal Transport [com19] Mehrere Möglichkeiten einer Regularisierung der GW-Distanz[23]:

- Entropic regularization [Cuturi, 2013]
- Group Lasso [Courty et al., 2016a]
- KL, Itakura Saito, β-divergences, [Dessein et al., 2016]

Optimal Transport: Regularization and Applications [24] Python Optimal Transport Toolbox[25][26]

Kapitel 2.1 im Paper: Vergleich von Histogrammen auf demselben metrischen Raum.

Lemma 6.5.24. $L_C(\boldsymbol{a}, \boldsymbol{b}) := \min_{P \in \Pi(\boldsymbol{a}, \boldsymbol{b})} \langle \boldsymbol{P}, \boldsymbol{C} \rangle - \varepsilon \boldsymbol{H}(\boldsymbol{P})$
ist ein ε-streng konvexes Problem und besitzt deshalb eine eindeutige optimale Lösung test

Beweis. todo. □

Proposition 6.5.25 (Konvergenz in ε)**.** *Die eindeutige Lösung P_ε von Theorem 6.5.24 konvergiert gegen die optimale Lösung mit maximaler Entropie innerhalb der Menge aller optimalen Lösungen des Kantorovich Problems, d.h.*

$$\boldsymbol{P}_\varepsilon \xrightarrow{\varepsilon \to 0} \arg\min_P \{-\boldsymbol{H}(\boldsymbol{P}) : \boldsymbol{P} \in \Pi(\boldsymbol{a}, \boldsymbol{b}), \langle \boldsymbol{P}, \boldsymbol{C} \rangle = L_{\boldsymbol{C}}(\boldsymbol{a}, \boldsymbol{b})\}, \tag{6.14}$$

d.h. es gilt

$$L^\varepsilon_{\boldsymbol{C}}(\boldsymbol{a}, \boldsymbol{b}) \xrightarrow{\varepsilon \to 0} L_{\boldsymbol{C}}(\boldsymbol{a}, \boldsymbol{b}) \tag{6.15}$$

Zudem gilt

$$\boldsymbol{P}_\varepsilon \xrightarrow{\varepsilon \to \infty} \boldsymbol{a} \otimes \boldsymbol{b} = \boldsymbol{a}\boldsymbol{b}^\top = (\boldsymbol{a}_i \boldsymbol{b}_j)_{i,j}. \tag{6.16}$$

Beweis. todo:Abend. s. COT, S.59 □

[22] https://www.damtp.cam.ac.uk/research/cia/files/teaching/Optimal_Transport_Syllabus.pdf

[23] https://www.youtube.com/watch?v=cPVMHWF8fmE&t=2532s

[24] https://www.otra2020.com/schedule

[25] https://pythonot.github.io/quickstart.html

[26] https://pythonot.github.io/auto_examples/gromov/plot_gromov.html

Bemerkung 6.5.26. *Gleichung 6.14 zeigt, dass die Lösung für kleine ε gegen die Optimale Transport Kopplung mit maximaler Entropie konvergiert. Im Gegensatz dazu, bedeutet Gleichung 6.16, die Lösung für große Regularisierungsparameter konvergiert gegen die Kopplung mit maximaler Entropie zwischen zwei gegebenen Randverteilungen $\boldsymbol{a}$ und $\boldsymbol{b}$.*

Bemerkung 6.5.27. *A key insight is that, as ε increases, the optimal coupling becomes less and less sparse (in the sense of having entries larger than a prescribed threshold), which in turn has the effect of both accelerating computational algorithms (as we study in Abschnitt 4.2) and leading to faster statistical convergence (as shown in Abschnitt 8.5)*

Wir definieren die Kullback-Leibler-Divergenz zwischen Kopplungen als

$$\boldsymbol{KL}(\boldsymbol{P}|\boldsymbol{K}) := \sum_{i,j} \boldsymbol{P}_{i,j} \log\left(\frac{\boldsymbol{P}_{i,j}}{\boldsymbol{K}_{i,j}}\right) - \boldsymbol{P}_{i,j} + \boldsymbol{K}_{i,j}. \tag{6.17}$$

Damit ist die eindeutige Lösung $\boldsymbol{P}_\varepsilon$ von Theorem 6.5.24 eine Projektion des zur Kostenmatrix $\boldsymbol{C}$ gehörigen Gibbs-Kernels $\boldsymbol{K}_{i,j} := e^{-\frac{\boldsymbol{C}_{i,j}}{\varepsilon}}$ auf $\Pi(\boldsymbol{a}, \boldsymbol{b})$.

Mit der obigen Definition erhalten wir

$$\boldsymbol{P}_\varepsilon = Proj^{\boldsymbol{KL}}_{\boldsymbol{\Pi}(\boldsymbol{a},\boldsymbol{b})}(\boldsymbol{K}) := \underset{P \in \boldsymbol{\Pi}(\boldsymbol{a},\boldsymbol{b})}{\arg\min}\, \boldsymbol{KL}(\boldsymbol{P}|\boldsymbol{K}). \tag{6.18}$$

6.5.12 Berechnung der Lösung: Sinkhorn

In diesem Kapitel sehen wir, dass die Lösung des regularisierten Problems eine besondere Form besitzt, die wir über $m + n$ Variablen parametrisieren können.

Proposition 6.5.28. *Die Lösung des regularisierten Problems Theorem 6.5.24 besitzt die Form*

$$\forall (i,j) \in \{1,...,n\} \times \{1,...,m\} : \boldsymbol{P}_{i,j} = \boldsymbol{u}_i \boldsymbol{K}_{i,j} \boldsymbol{v}_j \tag{6.19}$$

für die beiden (unbekannten) Variablen $(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{R}^n_+ \times \mathbb{R}^m_+$.

Beweis. Wir führen für jede der beiden Nebenbedingungen die dualen Variablen $\boldsymbol{f} \in \mathbb{R}^n$ und $g \in \mathbb{R}^m$ ein. Für die Lagrange-Funktion zu Theorem 6.5.24 erhalten wir damit:

$$\mathcal{L}(\boldsymbol{P}, \boldsymbol{f}, \boldsymbol{g}) = \langle \boldsymbol{P}, \boldsymbol{C} \rangle - \varepsilon \boldsymbol{H}(\boldsymbol{P}) - \langle \boldsymbol{f}, \boldsymbol{P}\mathbf{1}_m - \boldsymbol{a} \rangle - \langle \boldsymbol{g}, \boldsymbol{P}^\top \mathbf{1}_n - \boldsymbol{a} \rangle. \tag{6.20}$$

Mit der Optimalitätsbedingung erster Ordnung ergibt sich

$$\frac{\partial \mathcal{L}(\boldsymbol{P}, \boldsymbol{f}, \boldsymbol{g})}{\partial \boldsymbol{P}_{i,j}} = \boldsymbol{C}_{i,j} + \varepsilon \log(\boldsymbol{P}_{i,j}) - \boldsymbol{f}_i - \boldsymbol{g}_j = 0, \tag{6.21}$$

womit wir für eine optimale Kopplung $\boldsymbol{P}$ für das regularisierte Problem den Ausdruck $\boldsymbol{P}_{i,j} = e^{\boldsymbol{f}_i/\varepsilon} e^{-\boldsymbol{C}_{i,j}/\varepsilon} e^{\boldsymbol{g}_j/\varepsilon}$, der in die gewünschte Form umgeschrieben werden kann. □

Bemerkung 6.5.29 (Algorithmus von Sinkhorn)**.** *Die Faktorisierung der Lösung in Gleichung 6.19 können in der folgenden Matrix-Form schreiben:* $\boldsymbol{P} = diag(\boldsymbol{u})\boldsymbol{K}diag(\boldsymbol{v})$*. Die beiden Variablen* $(\boldsymbol{u}, \boldsymbol{v})$ *müssen deshalb die folgenden nichtlinearen Gleichungen erfüllen, die aufgrund der geforderten Massenerhaltungsbedingung in* $\Pi(\boldsymbol{a}, \boldsymbol{b})$ *gelten:*

$$diag(\boldsymbol{u})\boldsymbol{K}diag(\boldsymbol{v})\mathbf{1}_m = \boldsymbol{a} \text{ und } diag(\boldsymbol{v})\boldsymbol{K}^\top diag(\boldsymbol{u})\mathbf{1}_n = \boldsymbol{b}. \tag{6.22}$$

Aufgrund der Beziehung $diag(\boldsymbol{v})\mathbf{1}_m = \boldsymbol{v}$ *und selbiger Beziehung für* $\boldsymbol{u}$ *erhalten wir die folgende Vereinfachung*

$$\boldsymbol{u} \odot (\boldsymbol{K}\boldsymbol{v}) = \boldsymbol{a} \text{ und } \boldsymbol{v} \odot (\boldsymbol{K}^\top \boldsymbol{v}) = \boldsymbol{a}, \tag{6.23}$$

wobei $\odot$ *für die Element-weise Multiplikation zweier Vektoren steht. Dieses Problem ist als Matrix Scaling Problem [NR99] bekannt.*
(? Sollte ich hier Bedingungen für die Lösbarkeit angeben?)
Eine Möglichkeit zur Lösung dieses Problems ist ein iteratives Vorgehen(?Verfahren), bei dem zunächst $\boldsymbol{u}$ *so modifiziert wird, dass die linke Seite in Gleichung 6.23 erfüllt ist, und anschließend die Modifikation von* $\boldsymbol{v}$ *vorgenommen wird, sodass die rechte Seite in Gleichung 6.23 gilt. Mit diesen beiden Modifikationen erhalten wir den Algorithmus von Siknhorn, der aus den beiden folgenden Updates besteht:*

$$\boldsymbol{u}^{l+1} := \frac{\boldsymbol{a}}{\boldsymbol{K}\boldsymbol{v}^{(l)}} \text{ und } \boldsymbol{v}^{l+1} := \frac{\boldsymbol{b}}{\boldsymbol{K}^\top \boldsymbol{u}^{(l+1)}}, \tag{6.24}$$

wobei zu Beginn mit einem beliebigen positiven Vektor, beispielsweise $\boldsymbol{v}^{(0)} = \mathbf{1}_m$ *initialisiert wird und* l *den aktuellen Iterationsschritt bezeichnet. Die obigen Division muss ebenfalls elementweise verstanden werden.*

The iterations (4.15) first appeared in [Yule, 1912, Kruithof, 1937]. They were later known as the iterative proportional fitting procedure (IPFP) Deming and Stephan [1940] and RAS [Bacharach, 1965] methods [Idel, 2016]. The proof of their convergence is attributed to Sinkhorn [**?**], hence the name of the algorithm.

7 TODO: SINKHORN CONVERGENCE

Bemerkung 7.0.1 (Konvergenz des Sinkhorn Algorithmus)**.** *Elementar ist [FL89] Hilbert (projective) metric test*

- *Globale Konvergenz*
- *Lokale Konvergenz*

Bemerkung 7.0.2 (Allgemeine Formulierung des entropisch regularisierten Problems für beliebige Maße)**.**

- *Benutze Relative Entropie als Verallgemeinerung der diskreten Kullback-Leibler-Divergenz*
- *Referenzmaß ist unbedeutend, lediglichd er Support hat eine Bedeutung.*

7.0.1 Verallgemeinerung

Der Vergleich zwischen Ähnlichkeits- bzw. Distanzmatrizen ist schwierig, da diese die innere Struktur eines Datensatzes beschreiben, die unabhängig von Rotationen und Translationen ist. Es existiert keine kanonische Ordnung der Reihen und Spalten. Verallgemeinerung auf beliebige Matrizen C, d.h. diese Distanzmatrizen müssen nicht notwendigerweise positiv sein und die Dreiecksungleichung erfüllen.

Definiere die verallgemeinerte Gromov-Wasserstein-Distanz (vGWD) wie folgt:

Definition 7.0.3 (Verallgemeinerte Gromov-Wasserstein-Distanz)**.** *Seien zwei gewichtete Ähnlichkeitsmatrizen $(C,p) \in \mathbb{R}^{N_1 \times N_1} \times \Sigma_{N_1}$ und $(\bar{C},q) \in \mathbb{R}^{N_2 \times N_2} \times \Sigma_{N_2}$ gegeben. Sei T eine Kopplung zwischen den beiden Räumen, auf denen die Matrizen C und $\bar{C}$ definiert sind. Sei L eine Fehlerfunktion. Dann definieren wir die verallgemeinerte Gromov-Wasserstein-Distanz als*

$$GW(C,\bar{C},p,q) := \min_{T \in \mathcal{C}_{p,q}} \varepsilon_{C,\bar{C}}(T), \tag{7.1}$$

wobei gilt $\varepsilon_{C,\bar{C}}(T) := \sum_{i,j,k,l} L(C_{i,k},\bar{C}_{j,l}) T_{i,j} T_{k,l}$.

Häufig verwendete Fehlerfunktionen sind die quadratische Fehlerfunktion $L(a,b) = L_2(a,b) := \frac{1}{2}|a-b|^2$ und die Kullback-Leibler-Divergenz $L(a,b) = KL(a|b) := a\log(a/b) - a + b$.

Diese Definition der Gromov-Wasserstein-Distanz verallgemeinert die Version in [PCS16], da nun beliebige Fehlerfunktionen betrachtet werden.

Für $L = L_2$ zeigt Memoli, 2011, dass $GW^{1/2}$ eine Distanz auf dem Raum metrischer Maßräume modulo Maß-erhaltender Isometrien (?, besser zitieren -> collständiges Resultat angeben) definiert.

Durch die Definition

$$\mathcal{L}(C,\bar{C}) := (L(C_{i,k},\bar{C}_{j,l}))_{i,j,k,l} \tag{7.2}$$

erhalten wir

$$\epsilon_{C,\bar{C}}(T) = \langle \mathcal{L}(C,\bar{C}) \otimes T, T \rangle \tag{7.3}$$

gilt.

Mit der folgenden Proposition ergibt sich eine effiziente Berechnung von $\mathcal{L}(C,\bar{C}) \otimes T$ für eine bestimme Klasse von Verlustfunktionen L:

Proposition 7.0.4. *Die Verlustfunkton L lasse sich schreiben als*

$$L(a,b) = f_1(a) + f_2(b) - h_1(a)h_2(b) \tag{7.4}$$

für $f_1, f_2, h_1, h_2 : \mathbb{R} \to \mathbb{R}$. Dann gilt für $T \in \mathcal{C}_{p,q}$:

$$\mathcal{L}(C,\bar{C}) \otimes T = c_{C,\bar{C}} - h_1(C) T h_2(\bar{C})^T, \tag{7.5}$$

wobei $c_{C,\bar{C}} := f_1(C) p \mathbf{1}_{N_2}^T + \mathbf{1}_{N_1} q^T f_2(\bar{C})^T$ unabhängig von T ist.

Beweis. Aufgrund von Gleichung 7.4 gilt nach der Tensor-Matrix-Multiplikation Gleichung 6.13 die Zerlegung $\mathcal{L}(C,\bar{C}) \otimes T = A + B + C$ mit

$$A_{i,j} = \sum_k f_1(C_{i,k}) \sum_l T_{k,l} = (f_1(C)(T\mathbf{1}))_i,$$
$$B_{i,j} = \sum_l f_2(\bar{C}_{j,l}) \sum_k T_{k,l} = (f_2(\bar{C})(T^\top \mathbf{1}))_j,$$
$$C_{i,j} = \sum_k h_1(C_{i,k}) \sum_l h_2(\bar{C}_{j,l}) T_{k,l}.$$

Dies ist äquivalent zu $(h_1(C))(h_1(\bar{C}T^\top)^\top)_{i,j}$.
(? Wie folgt daraus die Behauptung) □

Bemerkung 7.0.5 (Verbesserte Komplexität)**.** *Mit dem Resultat in Theorem 7.0.4 können wir $\mathcal{L}(C, \bar{C}) \otimes T$ effizient in der Größenordnung $\mathcal{O}(N_1^2 N_2 + N_2^2 N_1)$ mit ausschließlich Matrix/Matrix-Multiplikationen berechnen im Unterschied zur Komplexität von $\mathcal{O}(N_1^2 N_2^2)$ für die Implementierung von Gleichung 6.13.*

Bemerkung 7.0.6 (Spezialfälle)**.** *Im Fall $L = L_2$ ist die Bedingung Gleichung 7.4 für die Funktionen $f_1(a) = a^2, f_2(b) = b^2, h_1(a) = a$ und $h_2(b) = 2b$ erfüllt. Für $L = KL$ sind die Funktionen $f_1(a) = a\log(a) - a, f_2(b) = b, h_1(a) = a$ und $h_2(b) = \log(b)$ notwendig.*

Wir betrachten nun die regularisierte Version der Gromow-Wasserstein-Diskrepanz 7.1 und definieren:

Definition 7.0.7. *Für $C, \bar{C}, p, q$, wie oben, definieren wir die entropisch regularisierte Gromov-Wasserstein-Diskrepanz als*

$$GW_\varepsilon(C, \bar{C}, p, q) := \min_{T \in \mathcal{C}_{p,q}} \boldsymbol{\varepsilon}_{C,\bar{C}}(T) - \varepsilon H(T). \tag{7.6}$$

Wir erhalten damit ein nicht-konvexes Optimierungsproblem. Für dessen Lösung benutzen wir ein projiziertes Gradienten-Verfahren, bei dem sowohl die Schrittweite als auch die Projektion bezüglich der KL-Metrik berechnet werden.

Die Iterationen sind gegeben durch

$$T \leftarrow Proj^{KL}_{\mathcal{C}_{p,q}} \left(T \odot e^{-\tau(\nabla \boldsymbol{\varepsilon}_{C,\bar{C}}(T) - \varepsilon H(T))} \right), \tag{7.7}$$

wobei die Schrittweite $\tau > 0$ und die KL-Projektion einer beliebigen Matrix K gegeben ist durch:

$$Proj^{KL}_{\mathcal{C}_{p,q}}(K) := \operatorname*{arg\,min}_{T' \in \mathcal{C}_{p,q}} KL(T'|K). \tag{7.8}$$

Proposition 7.0.8. *Für den Fall $\tau = 1/\varepsilon$ erhalten wir die Iterationsvorschrift*

$$T \leftarrow \mathcal{T}(\mathcal{L}(C, \bar{C}) \otimes T, p, q). \tag{7.9}$$

Beweis. Nach [BCC$^+$15] ist die Projektion in 7.7 gegeben durch die Lösung des regularisirten Transportproblems 6.5.24 und ist damit gegeben durch:

$$Proj^{KL}_{\mathcal{C}_{p,q}}(K) = \mathcal{T}(-\varepsilon \log(K), p, q) \tag{7.10}$$

(? Wo steht das genau in dem Paper?) Es gilt außerdem

$$\nabla \boldsymbol{\varepsilon}_{C,\bar{C}}(T) - \varepsilon H(T) = blub \tag{7.11}$$

Durch Umordnen der Terme in Gleichung 7.7 erhalten wir für $\tau\varepsilon = 1$ die angegebene Vorschrift. □

Bemerkung 7.0.9. *Die Iterationsvorschrift 7.9 definiert einen einfachen Algorithmus, der in jedem Update von T eine Sinkhorn-Projektion benötigt.*

Bemerkung 7.0.10 (Konvergenz)**.** *Die Iterationen Gleichung 7.7 konvergieren nach [BCL16] für $\tau < \tau_{max}$. Im Allgemeinen gilt jedoch $1/\varepsilon < \tau_{max}$ jedoch nicht, womit die Wahl der Schrittweite $\tau = 1$ in Theorem 7.0.8 nicht durch die Theorie abgedeckt ist. Laut [PCS16] konvergiert die Iteration mit $\tau = 1/\varepsilon$ jedoch in der Praxis.*
Für den Fall $L = L_2$ ergibt sich der „Softassign quadratic assignment algorithm „ [RYM99], für welchen ein Konvergenzresultat im Fall eines konvexen Problems existiert. Da wir in unserem Fall nur positive symmetrische Matrizen C, C_s betrachten ist hier die Konvergenz gesichert.

Bemerkung 7.0.11. *content...*

Bemerkung 7.0.12 (Wahl von ε)**.** *In [Cut13] werden verschiedene Werte für ε angegeben. Diese sind $\varepsilon = 0.02, 0.1, 1.0$*
Im zugehörigen Beispiel[27] *von POT wird $\varepsilon = 0.0005$ verwendet.*
Für ε klein werden die Ergebnisse besser während der Rechenaufwand steigt.
welche Resultate existieren bezüglich der Approximationsgüte abhängig von ε?

7.0.2 Wasserstein Baryzentren

In diesem Kapitel wollen wir uns mit dem "Mittelwert"befassen, der elementar für das kmeans-Clustering ist. Auch hier soll der Mittelwert auf die abstrakte Ebene von gewichteten Distanzmatrizen angehoben werden. Dazu definieren wir im Folgenden sogenannte Wasserstein-Baryzentren und folgen [PCS16] für die Lösung des entstehenden Optimierungsproblems. Gute Einführung[28]

Definition 7.0.13 (Gromov-Wasserstein Baryzentrum)**.** *Seien die gewichteten Distanzmatrizen $C_s)_{s=1}^S$, mit $C_s \in \mathbb{R}^{N_s \times N_s}$ und den zugehörigen Histogrammen $(p_s)_s$ gegeben.*
Dann ist das Gromov-Wasserstein Baryzentrum definiert durch

$$\min_{C \in \mathbb{R}^{N \times N}} \sum_s \lambda_s GW_\varepsilon(C, C_s, p, p_s). \tag{7.12}$$

Existenz und Eindeutigkeit von Baryzentren: siehe [AC11].

Bemerkung 7.0.14 (Darstellung des Baryzentrums)**.** *Um das berechnete Bary-Zentrum C wieder zu visualisieren, kann C als Distanzmatrix wieder in den zweidimensionalen Raum eingebettet werden. Dies kann beispielsweise durch Multidimensionale Skalierung erreicht werden*[29]*.*

Bemerkung 7.0.15. *Wir gehen im Folgenden davon aus, dass sowohl die Histogramme $(p_s)_s$ als auch das Histogramm p bekannt sind. Die Größe (N, N) des gesuchten Bary-Zentrums muss ebenfalls vorher festgelegt werden. Eine Erweiterung auf den Fall, dass auch p unbekannt sein sollte und damit als Optimierungsparameter aufgefasst wird, ist leicht möglich [PCS16].*

[27] `https://pythonot.github.io/auto_examples/gromov/plot_gromov.html`
[28] `https://hal.archives-ouvertes.fr/hal-00637399/document`
[29] `https://pythonot.github.io/auto_examples/gromov/plot_gromov_barycenter.html`

Wir können das Bary-Zentrum mithilfe von Kopplungen umformulieren als

$$\min_{C,(T_s)_s} \sum_s \lambda_s(\varepsilon_{C,C_s}(T_s) - \varepsilon H(T_s)), \tag{7.13}$$

unter den Nebenbedingungen: $\forall s : T_s \in \mathcal{C}_{p,p_s} \subset \mathbb{R}_+^{N\times N_s}$. Für den Fall, dass L bezüglich der ersten Variable konvex ist, ist dieses Problem konvex bezüglich C. Bezüglich $(T_s)_s$ ist diese Problem quadratisch aber nicht notwendigerweise positiv.

Das Problem Gleichung 7.13 kann durch eine Block-Koordinaten-Relaxierung gelöst werden. Dabei wird iterativ abwechselnd bezüglich den Kopplungen $(T_s)_s$ und der Metrik C miminiert.

Minimierung bezüglich $(T_s)_s$. Anhand der Umformulierung Gleichung 7.13 sehen wir, dass das Optimierungsproblem Gleichung 7.12 bezüglich $(T_s)_s$ in S (?viele) unabhängige GW_ε-Optimierungen

$$\forall s : \min_{T_s\in\mathcal{C}_{p,p_s}} \mathcal{E}_{C,C_s}(T_s) - \varepsilon H(T_s) \tag{7.14}$$

zerfällt, die jeweils wie in Theorem 7.0.8 angegeben gelöst werden können.
Minimierung bezüglich C. Sei (T_s) gegeben. Dann lautet, die Minimierung bezüglich C:

$$\min_C \sum_s \lambda_s \langle \mathcal{L}(C, C_s) \otimes T, T\rangle. \tag{7.15}$$

Mit der folgenden Proposition erhalten wir für eine Klasse von Verlustfunktionen L eine Lösung in geschlossener Form.

Proposition 7.0.16. *Sei L eine Verlustfunktion, die die Bedingung Gleichung 7.4 erfüllt. Sei f_1'/h_1' invertierbar.*
Dann lässt sich die Lösung zu Gleichung 7.15 schreiben als:

$$C = \left(\frac{f_1'}{h_1'}\right)^{-1} \left(\frac{\sum_s \lambda_s T_s^\top h_2(C_s) T_s}{pp^\top}\right), \tag{7.16}$$

wobei die Normalisierung $\lambda_s = 1$ gilt.

Beweis. Nach Theorem 7.0.4 können wir das zu minimierende Funtional schreiben als

$$\sum_s \lambda_s \langle f_1(C) p\mathbf{1}^\top + \mathbf{1}p_s^\top f_2(C_s) - h_1(C) T_s h_2(C_s)^\top, T_s\rangle. \tag{7.17}$$

Die Optimalitätsbedingung erster Ordnung lautet folglich

$$f_1'(C) \odot pp^\top = h_1'(C) \odot \sum_s \lambda_s T_s h_2(C_s) T_s^\top. \tag{7.18}$$

Durh Umstellen der Gleichung erhalten wir die angegebene Form. □

Anhand von Gleichung 7.16 wird die folgende Interpretation deutlich/möglich: Für jedes $s \in S$ ist $T_s^\top h_2(C_s) T_s$ eine wiedereingeordnete Matrix, wobei T_s als Optimal Transport-Kopplung von Zeilen und Spalten der Distanzmatrix C_s fungiert.

Über diese Matrizen wird anschließend gemittelt, wobei die Art der Mittelung von der Verlustfunktion L abhängig ist.

Für den Fall $L = L_2$ wird aus dem Update Gleichung 7.16 die folgende Vorschrift:

$$C \leftarrow \frac{1}{pp^\top} \sum_s \lambda_s T_s^\top C_s T_s. \tag{7.19}$$

Proposition 7.0.17. *Sei $L = L_2$ und $(C_s)_s$ positiv semi-definit f.a. $s \in S$. Dann sind die Iterierten C ebenfalls positiv semi-definit.*

Beweis. Gleichung 7.19 zeigt, dass das Update aus einer Bildung des Mittelwertes der Matrizen $()diag(1/p)T_s^\top C_s T_s diag(1/p))_s$ besteht. Diese sind alle positiv semi-definit, da die $(C_s)_s$ nach Voraussetzung positiv semi-definit sind. □

Für den Fall $L = KL$ ergibt sich die folgende Verfahrensvorschrift:

$$C \leftarrow \left(\frac{1}{pp^\top} \sum_s \lambda_s T_s^\top log(C_s) T_s \right). \tag{7.20}$$

Algorithm 2 Berechnung der GW_ε Baryzentren

Input: $(C_s, p_s), p$. Initialisiere C.
Output: C.

if some condition is true **then**
  do some processing
else if some other condition is true **then**
  do some different processing
else
  do the default actions
end if

Psuedocode.

Häufige Verwendungungen(s. Einführung hier [30]). zB. Shape interpolation

Wir haben in diesem Kapitel gesehen, wie mithilfe des Wasserstein-Baryzentrums eine Art Mittelwert für Wahrscheinlichkeitsverteilungen definiert werden kann. Algorithm 1 details the steps of the opti- mization technique. Note that it makes use of three nested iterations: (i) blockwise coordinate descent on (T s) s and C, (ii) projected gradient descent on each T s , and (iii) Sinkhorn iterations to compute the projection.

7.0.3 Numerische Approximationen

3

[30] `https://arxiv.org/pdf/2102.01752.pdf`

Algorithm 3 Algorithm caption

if some condition is true **then**
  do some processing
else if some other condition is true **then**
  do some different processing
else
  do the default actions
end if

8 (Numerische) Ergebnisse/Vergleich mit anderen Verfahren

8.0.1 Standard Poisoning-Angriffe

=> Did save Model - incv320epochsnormalized - at epoch: 19 => [19] loss: 0.125, accuracy: 96.514=> Accuracy of validation Dataset: 98.491

=> FINISHED TRAINING loss on test dataset: 3.284087032527311 Accuracy of test Dataset: 0.919

BDSR classwise: [nan nan 1. nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan]

Performance of poisoned net on unpoisoned training data: loss on test dataset: 0.15269652156995248 Accuracy of test Dataset: 0.961

Performance auf nicht korrumpierten Daten:
=> FINISHED TRAINING loss on test dataset: 0.15834395289583986 Accuracy of test Dataset: 0.968

Seitenlänge s des Triggers	Prozentualer Anteil	AER	DR (HC)			DR (AC,PCA)			Clustering HC
			ACC	TPR	TNR	ACC	TPR	TNR	
s=2	0.000625	0.01333333							
	0.00125	0.356							
	0.0025	0.576							
	0.005	0.99867							
	0.01	0.92							
	0.02	1.0				51.9	2.94	52.91	
true	5.00	100.00				47.44	10.34	49.39	
true	10.00	1.0				67.21	93.99	64.24	
hctrue true	15.00	1.0	100.00	100.00	100.00	87.53	97.59	85.76	(tn, fp, fn, tp): 1650,0,0,291 it3
(eps5e-4)	0.33	1.0	98.17	96.98	98.91	99.23	97.66	100.00	(tn, fp, fn, tp): 1632,18,27,786 it5
eps0.02			100.00	100.00	100.00				(tn, fp, fn, tp): 1650,0,0,813 it3
s=3	0.05	1.0							
	0.15	1.0							
s=1	0.33	1.0							

Tabelle 2: Qualität der Detektion unterschiedlich starker Angriffe mithilfe von LRP-Clustering(?) und Gromov-Wasserstein-Distanzen

AC läuft mit 33 Prozent HC läuft mit 10 Prozent

Mo, 2.August:
s=2
pp=33
eps_init: 0.0005
eps_update: 0.0005
n_samples_bary: 10
iter: 0: (tn, fp, fn, tp): 1650,0,491,322
iter: 1: (tn, fp, fn, tp): 449,1201,802,11
iter: 2: (tn, fp, fn, tp): 1650,0,32,781
iter: 3: (tn, fp, fn, tp): 1621,29,27,786
iter: 4: (tn, fp, fn, tp): 1632,18,27,786

acc = 98.17
tpr = 96.98
0.11666667 0.23055556 0.26 0.55333333 0.43333333 nan 0.23333333 0.28444444 0.11555556 0.07708333 0.15 0.0952381 0.4173913 0.41527778 0.40740741 0.45238095 0.01333333 0.35833333 0.29487179 0. 0. 0.04444444 0.05 0.01333333 0.4 0.29791667 0.13888889 0.23333333 0.11333333 0.3 0.06 0.02962963 0.11666667 0.65714286 0.475 0.71794872 0.50833333 0.43333333 0.35652174 0.45555556 0.27777778 0. 0.51111111] Performance of poisoned net on unpoisoned training data: loss on test dataset: 0.19637071904851583 Accuracy on test Dataset: 0.958 tnr = 98.91

Mi, 4.August:
s=2, pp=15
eps_init: 0.0005
eps_update: 0.0005
n_samples_bary: 10
iter: 0: (tn, fp, fn, tp): 1650,0,89,202
iter: 1: (tn, fp, fn, tp): 1648,2,3,288
iter: 2: (tn, fp, fn, tp): 1650,0,0,291

acc = 100.00
tpr = 100.00
tnr = 100.00

Do, 5.August Das ist der Vergleich zu dem Experiment am Montag mit eps=0.0005. Das Ergebnis ist deutlich besser(denn perfekt).Daraus schließen wir, dass eps=0.02 für die restlichen Experimente genügen sollte. (Stimmt so vll nicht ganz)
eps_init: 0.02
eps_update: 0.02
n_samples_bary: 10
iter: 0: (tn, fp, fn, tp): 1650,0,812,1
iter: 1: (tn, fp, fn, tp): 1650,0,0,813
iter: 2: (tn, fp, fn, tp): 1650,0,0,813

Spektralanalyse:

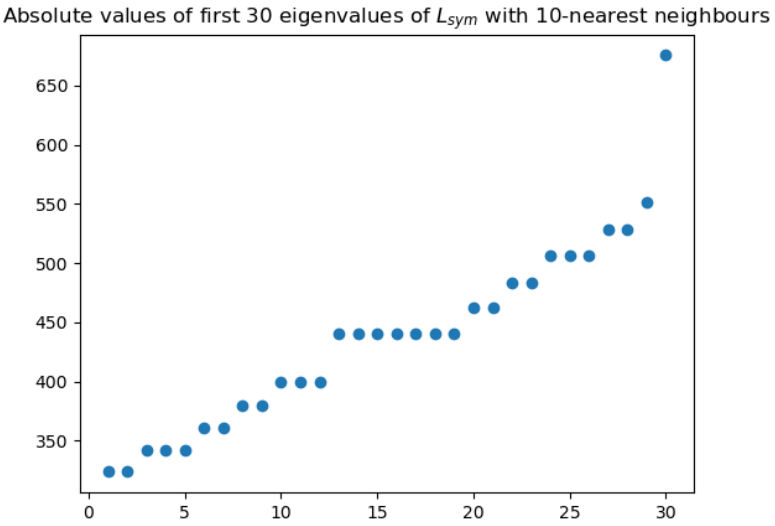


Abbildung 2: Ergebnis des spektralen Clusterings unter Verwendung der Euklidischen Distanz und k=10 Nachbarn

Clustering(euklidisch): $(tn, fp, fn, tp) = 1650, 0, 386, 427)$

Clustering(GWD):

Bemerkung 8.0.1. *Für größere Netzwerke ist es einfacher, erfolgreiche Angriffe zu implementieren, auch schon mit weniger korrumpierten Daten.*
Vermutung: Die Detektion mittels Clustering funktioniert für eine größere Anzahl an korrumpierten Daten besser. Wenn wir also nur perfekte Angriffe verteidigen wollen, d.h AER=100.00 funktioniert das bei kleineren Netzwerken besser.

8.0.2 Label-konsistente Poisoning-Angriffe

Bemerkung 8.0.2. *Für einen einzelnen Amplitudensticker mit d=10 ist das eigentlich identisch zu dem Angriff mit dem Sticker*

0.7 0.86111111 0.85066667 0.81111111 0.66666667 nan 0.42 0.72888889 0.60888889 0.77083333 0.40757576 0.46666667 0.69855072 0.88472222 0.53333333 0.74285714 0.39333333 0.67222222 0.75128205 0.11666667 0.58888889 0.32222222 0.425 0.18 0.44444444 0.44375 0.8 0.71666667 0.56 0.86666667 0.11333333 0.3 0.71666667 0.5047619 0.23333333 0.44358974 0.63333333 0.58333333 0.46666667 0.01111111 0.5 0.36666667 0.67777778 $CLP\_amplitudesticker4\_pp33\_eps1200\_d10amp32$ Jetzt dasselbe noch für nur !!!einen!!! Sticker mit derselben Amplitude am selben Ort $'CLP_amplitudesticker_pp33_eps1200_d10amp32'$ [0. 0.00833333 0.00133333 0.03555556 0.01666667 nan 0. 0. 0.00222222 0. 0. 0. 0.05362319 0.00138889 0.0037037 0. 0. 0.025 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.05 0. 0. 0. 0. 0. 0. 0. 0.00512821 0. 0. 0. 0. 0. 0. 0.]

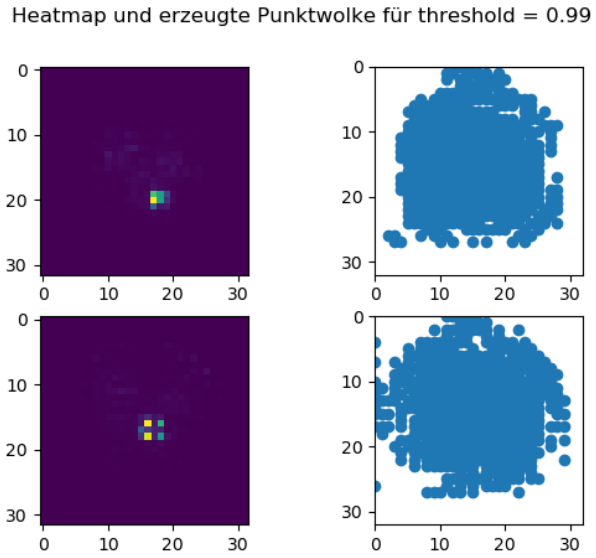


Abbildung 3: Auswahl der relevantesten Pixel (bis zu 99% der Gesamtmasse) zweier Heatmaps

Dann noch mit amp=16 wieder 4 sticker mit d=10 $CLP_amplitudesticker4_pp33_eps1200_d10amp16$ 0.11666667 0.23055556 0.26 0.55333333 0.43333333 nan 0.23333333 0.28444444 0.11555556 0.07708333 0.15 0.0952381 0.4173913 0.41527778 0.40740741 0.45238095 0.01333333 0.35833333 0.29487179 0. 0. 0.04444444 0.05 0.01333333 0.4 0.29791667 0.13888889 0.23333333 0.11333333 0.3 0.06 0.02962963 0.11666667 0.65714286 0.475 0.71794872 0.50833333 0.43333333 0.35652174 0.45555556 0.27777778 0. 0.51111111 Accuracy on test Dataset: 0.958

Jetzt: $CLP_amplitudesticker4_pp33_eps1200_d10amp255$ 0.71666667 0.78333333 0.80266667 0.99333333 0.9469697 nan 0.96 0.97555556 0.98444444 1. 1. 0.25714286 0.98985507 0.86111111 0.94444444 0.99047619 0.91333333 0.51666667 0.84871795 0.86666667 0.52222222 0.57777778 0.75 0.67333333 0.95555556 0.725 0.98888889 0.88333333 0.45333333 0.82222222 0.81333333 1. 0.98333333 0.4952381 0.26666667 0.92820513 0.63333333 0.81666667 0.77681159 0.33333333 0.74444444 0.91666667 0.83333333 Accuracy on test Dataset: 0.959

$CLP_amplitudesticker4_pp33_dist10_amp64$ eps300 [0.8 0.98888889 0.99733333 1. 1. nan 1. 1. 0.99555556 0.97916667 0.99545455 0.8047619 1. 0.99861111 0.92222222 1. 0.99333333 0.98888889 0.98717949 0.93333333 0.86666667 0.93333333 0.975 0.83333333 0.96666667 0.93958333 1. 1. 0.98 1. 0.98 0.97777778 1. 0.92380952 0.55 0.93076923 0.70833333 0.85 0.82173913 1. 0.8 1. 1.

$CLP_amplitudesticker4_pp33_dist10_amp32$ 0.7 0.86111111 0.85066667 0.81111111 0.66666667 nan 0.42 0.72888889 0.60888889 0.77083333 0.40757576 0.46666667 0.69855072 0.88472222 0.53333333 0.74285714 0.39333333 0.67222222 0.75128205 0.11666667 0.58888889 0.32222222 0.425 0.18 0.44444444 0.44375 0.8 0.71666667 0.56 0.86666667 0.11333333 0.3 0.71666667 0.5047619 0.23333333 0.44358974 0.63333333 0.58333333 0.46666667 0.01111111 0.5 0.36666667 0.67777778 Accuracy on test Dataset: 0.959

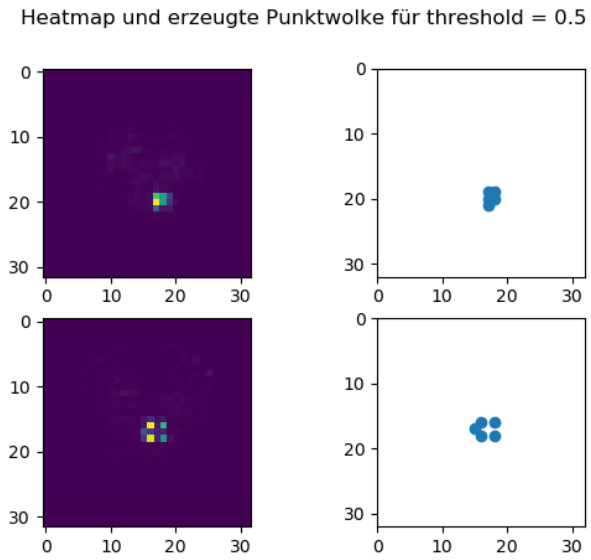


Abbildung 4: Auswahl der relevantesten Pixel (bis zu 50% der Gesamtmasse) zweier Heatmaps

$CLP_amplitudesticker4_p p33_d ist10_a mp16$ 0.11666667 0.23055556 0.26 0.55333333 0.43333333 nan 0.23333333 0.28444444 0.11555556 0.07708333 0.15 0.0952381 0.4173913 0.41527778 0.40740741 0.45238095 0.01333333 0.35833333 0.29487179 0. 0. 0.04444444 0.05 0.01333333 0.4 0.29791667 0.13888889 0.23333333 0.11333333 0.3 0.06 0.02962963 0.11666667 0.65714286 0.475 0.71794872 0.50833333 0.43333333 0.35652174 0.45555556 0.27777778 0. 0.51111111] Performance of poisoned net on unpoisoned training data: loss on test dataset: 0.19637071904851583 Accuracy on test Dataset: 0.958

In Abbildung 6 sind die Angriffserfolgsraten pro Klasse im Fall von 33% korrumpierten Daten und dem Amplitudensticker in jeder der 4 Bildecken mit dem Abstand von 10 Pixeln zum Rand dargestellt. In beiden Fällen geben wir keine AER für die Klasse 6 an, über die der Angriff stattfindet. Die mittlere Angriffserfolgsrate beträgt für $amp = 255$ 79.15%. In mehreren Klassen wird eine AER von 100% erreicht.
Für $amp = 64$ fällt der Angriff mit einer mAER von 48.17% deutlich schwächer aus. Die maximal erreichte AER beträgt 95.33% in Klasse 10. Die minimalen AER sind 25% bzw. 0.83%. Eine weitere Reduktion der Amplitude auf $amp = 32$ führt zu einer mAER von 6.55% und einer maximalen AER von 33%.

Für d=0, amplitude=64 und eps=1200 ergibt sich kein erfolgreicher Angriff, dabei war fast alles 0, die Trigger im Testdatensatz waren auch auf 64 Jetzt für d=10, auch hier gilt amp=64 im Testdatensatz:Ergebnis:

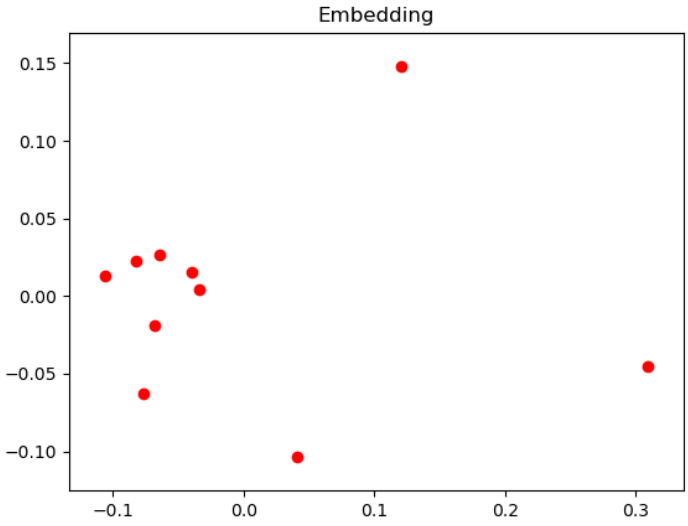


Abbildung 5: Einbettung des Baryzentrums mithilfe von Multidimensionaler Skalierung(MDS) bei der Wahl von 99% der Gesamtmasse

8.1 Activation Clustering

8.2 Räumliche Transformationen

- ASR ist sehr stark vom Ort des Triggers abhängig.
- Ort des Triggers kann nicht direkt geändert werden.
- Benutze Transformationen(Flipping, Scaling), um den Trigger wirkungslos zu machen.
- Somit kann die ASR während der Inferenz verringert werden. Es lässt sich aber keine Auussage darüber treffen, ob ein Angriff vorliegt

Seitenlänge des Triggers	Prozentualer Anteil	AER	GUD	num. korrumpierte Daten
s=2	0.000625	0.01333333	0.962	1
	0.00125	0.356	0.964	2
	0.0025	0.576	0.97	4
	0.005	0.99867	0.962	8
	0.01	0.92	0.962	17
	0.02	1.0	0.964	34
	0.10	1.0	0.968	291
	0.15	1.0	0.968	436
	0.33	1.0	0.968	813
s=3	?	?	?	?
	0.00125	34.26	96.2	2
	0.0025	85.07	96.5	4
	0.005	1.0	96.9	8
	0.01	1.0	0.962	17
	0.05	1.0	0.966	87
	0.15	1.0	0.961	
s=1	0.33	1.0	0.966	813

Tabelle 3: Qualität der Angriffe auf das Inception v3-Netz mit Stickern Seitenlänge 2 und 3 Pixel bei unterschiedlich großen Anteilen an korrumpierten Daten

9 Weitere mögliche Schritte

- Untersuchung der Detektionsqualität in Abhängigkeit von ε
- Automatische Platzierung des Auslösers an fest gewählter Position auf dem Verkehrsschild anstatt zufälligem Platzieren in einem Fenster mit vorher festgelegter Größe. In [GDGG17] wird Faster-RCNN (F-RCNN) zur Klassifikation des LISA-Datensatzes[31] 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 CNNs[32]
- Vergleich mit Cifar-10/Cifar-100 Datensatz[33][34]

[31] `http://cvrr.ucsd.edu/LISA/lisa-traffic-sign-dataset.html`
[32] `https://arxiv.org/abs/1611.05418`
[33] `https://www.cs.toronto.edu/~kriz/cifar.html`
[34] `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`

L2-Fehler				**H1-Fehler**		
	Gitter	P=1	P=2	Gitter	P=1	P=2
T=2.0	8×8	2.2e-15	1.6e-14	8×8	3.7e-14	1.7e-13
	16×16	1.4e-14	2.5e-13	16×16	1.0e-13	1.2e-12
	32×32	6.1e-15	1.1e-14	32×32	1.6e-14	6.0e-13

Tabelle 4: Fehler für Testproblem 1 zum Endzeitpunkt $T = 2.0$.

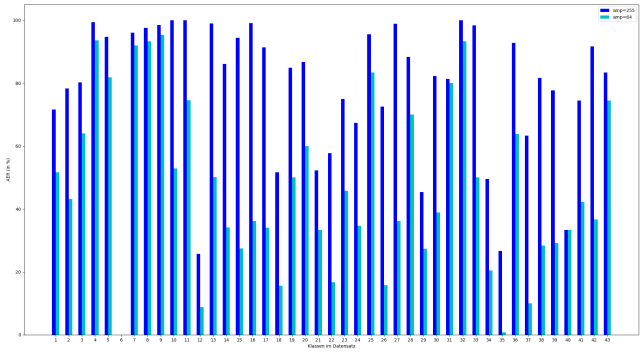


Abbildung 6: Angriffserfolgsrate pro Klasse bei Clean-Label-Poisoning-Attacks für verschiedene Werte *amp* des Amplitudenstickers in vierfacher Ausführung bei Abstand $d = 10$ zum Rand

10 Zusammenfassung

Beobachtung: Je größer die Netzwerke sind, desto leichter lassen sich Poisoning-Angriffe realisieren.

A Verwendete Netzwerke

A.1 Net

```
class Net(nn.Module):

def __init__(self, ):
super(Net, self).__init__()
self.size = 64 * 4 * 4
self.conv1 = nn.Conv2d(in_channels=3, out_channels=12,
  kernel_size=5, padding=2)
self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
self.conv1_in = nn.InstanceNorm2d(12)
self.conv2 = nn.Conv2d(in_channels=12, out_channels=32,
  kernel_size=5, padding=2)

self.conv2_bn = nn.BatchNorm2d(32)

self.conv3 = nn.Conv2d(in_channels=32, out_channels=64,
  kernel_size=5, padding=2)

self.fc1 = nn.Linear(self.size, 256)
self.fc1_bn = nn.BatchNorm1d(256)
self.fc2 = nn.Linear(256, 128)
self.fc3 = nn.Linear(128, 43)


def forward(self, x):
x = self.pool(F.relu(self.conv1_in(self.conv1(x))))
x = self.pool(F.relu(self.conv2_bn(self.conv2(x))))
x = self.pool(F.relu(self.conv3(x)))
x = x.view(-1, self.size)
x = F.relu(self.fc1_bn(self.fc1(x)))
x = F.dropout(x)
xx = F.relu(self.fc2(x))
x = F.dropout(xx)
x = self.fc3(x)

return x, xx


```

Listing 4: Kleines Netzwerk

```
InceptionNet3(
(features): Sequential(
(0): InceptionA(
(parallel_dummyA): New_parallel_chain_dummy()
(conv1x1): BatchConv(
(conv): Conv2d(3, 64, kernel_size=(1, 1), stride=(1, 1))
(bn): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
  track_running_stats=True)
(relu): ReLU()
```

```
  )
  (parallel_dummyB): New_parallel_chain_dummy()
  (conv5x5_1): BatchConv(
  (conv): Conv2d(3, 48, kernel_size=(1, 1), stride=(1, 1))
  (bn): BatchNorm2d(48, eps=1e-05, momentum=0.1, affine=True,
   track_running_stats=True)
  (relu): ReLU()
  )
  (conv5x5_2): BatchConv(
  (conv): Conv2d(48, 64, kernel_size=(5, 5), stride=(1, 1),
   padding=(2, 2))
  (bn): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
   track_running_stats=True)
  (relu): ReLU()
  )
  (parallel_dummyC): New_parallel_chain_dummy()
  (conv3x3dbl_1): BatchConv(
  (conv): Conv2d(3, 64, kernel_size=(1, 1), stride=(1, 1))
  (bn): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
   track_running_stats=True)
  (relu): ReLU()
  )
  (conv3x3dbl_2): BatchConv(
  (conv): Conv2d(64, 96, kernel_size=(3, 3), stride=(1, 1),
   padding=(1, 1))
  (bn): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True,
   track_running_stats=True)
  (relu): ReLU()
  )
  (conv3x3dbl_3): BatchConv(
  (conv): Conv2d(96, 96, kernel_size=(3, 3), stride=(1, 1),
   padding=(1, 1))
  (bn): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True,
   track_running_stats=True)
  (relu): ReLU()
  )
  (parallel_dummyD): New_parallel_chain_dummy()
  (pool): MaxPool2d(kernel_size=3, stride=1, padding=1,
   dilation=1, ceil_mode=False)
  (pool1x1): BatchConv(
  (conv): Conv2d(3, 32, kernel_size=(1, 1), stride=(1, 1))
  (bn): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
   track_running_stats=True)
  (relu): ReLU()
  )
  (parallel_dummyE): New_parallel_chain_dummy()
  (cat): Cat()
  )
  (1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation
   =1, ceil_mode=False)
  (2): BatchConv(
  (conv): Conv2d(256, 256, kernel_size=(2, 2), stride=(1, 1))
  (bn): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
   track_running_stats=True)
```

```
  (relu): ReLU()
  )
  (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation
    =1, ceil_mode=False)
  (4): BatchConv(
  (conv): Conv2d(256, 256, kernel_size=(2, 2), stride=(1, 1))
  (bn): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True)
  (relu): ReLU()
  )
  (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation
    =1, ceil_mode=False)
  (6): BatchConv(
  (conv): Conv2d(256, 256, kernel_size=(2, 2), stride=(1, 1))
  (bn): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True)
  (relu): ReLU()
  )
  (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation
    =1, ceil_mode=False)
  )
  (classifiers): Sequential(
  (0): Linear(in_features=256, out_features=256, bias=True)
  (1): ReLU(inplace=True)
  (2): Dropout(p=0.5, inplace=False)
  (3): Linear(in_features=256, out_features=128, bias=True)
  (4): ReLU(inplace=True)
  (5): Dropout(p=0.5, inplace=False)
  (6): Linear(in_features=128, out_features=43, bias=True)
  )
  )

```

Listing 5: Einfachere Version von Inception v3

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?

```

  [Linear(in_features=128, out_features=43, bias=True),
  Dropout(p=0.5, inplace=False),
  ReLU(inplace=True),
  Linear(in_features=256, out_features=128, bias=True),
  Dropout(p=0.5, inplace=False),
  ReLU(inplace=True),
  Linear(in_features=256, out_features=256, bias=True),
  MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
    ceil_mode=False),
  ReLU(),
  BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True), Conv2d(256, 256, kernel_size=(2,
```

```
    2), stride=(1, 1)),
  MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
    ceil_mode=False),
  ReLU(),
  BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True), Conv2d(256, 256, kernel_size=(2,
    2), stride=(1, 1)),
  MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
    ceil_mode=False),
  ReLU(), BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True
    , track_running_stats=True),
  Conv2d(256, 256, kernel_size=(2, 2), stride=(1, 1)),
  MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
    ceil_mode=False),

  [[Conv2d(3, 64, kernel_size=(1, 1), stride=(1, 1)),
  BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True), ReLU()],

  [Conv2d(3, 48, kernel_size=(1, 1), stride=(1, 1)),
  BatchNorm2d(48, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True), ReLU(), Conv2d(48, 64,
    kernel_size=(5, 5), stride=(1, 1), padding=(2, 2)),
    BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True), ReLU()],

  [Conv2d(3, 64, kernel_size=(1, 1), stride=(1, 1)),
  BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True), ReLU(),
  Conv2d(64, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1,
    1)),
  BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True), ReLU(),
  Conv2d(96, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1,
    1)),
  BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True), ReLU()],

  [MaxPool2d(kernel_size=3, stride=1, padding=1, dilation=1,
    ceil_mode=False), Conv2d(3, 32, kernel_size=(1, 1), stride
    =(1, 1)),
  BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True),
  ReLU()],

  [Cat()]]]

```

Listing 6: Reversed Model incv3

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 folgen-

den 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 7: 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 [35], die für die vor-trainierten Modelle empfohlen werden und auf dem Datensatz ImageNet[36] 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 [37], die wiederum auf PyTorch Ignite[38] basiert.

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:

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

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

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

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

E Datensätze

GTSRB[39] 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 [40] Barycenters in the Wasserstein Space[41]
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.

[39] `https://benchmark.ini.rub.de/gtsrb_dataset.html`
[40] `https://arxiv.org/pdf/1809.06422.pdf`
[41] `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.

[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.

[AWR17] Jason Altschuler, Jonathan Weed, and Philippe Rigollet. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. *arXiv preprint arXiv:1705.09634*, 2017.

[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.

[BCC+15] Jean-David Benamou, Guillaume Carlier, Marco Cuturi, Luca Nenna, and Gabriel Peyré. Iterative bregman projections for regularized transportation problems. *SIAM Journal on Scientific Computing*, 37(2):A1111–A1138, 2015.

[BCL16] Radu Ioan Boţ, Ernö Robert Csetnek, and Szilárd Csaba László. An inertial forward–backward algorithm for the minimization of the sum of two nonconvex functions. *EURO Journal on Computational Optimization*, 4(1):3–25, 2016.

[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.

[Cut13] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transportation distances. *arXiv preprint arXiv:1306.0895*, 2013.

[FL89] Joel Franklin and Jens Lorenz. On the scaling of multidimensional matrices. *Linear Algebra and its applications*, 114:717–735, 1989.

[GDGG17] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[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.

[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.

[NR99] Arkadi Nemirovski and Uriel Rothblum. On complexity of matrix scaling. *Linear Algebra and its Applications*, 302:435–460, 1999.

[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.

[RYM99] Anand Rangarajan, Alan Yuille, and Eric Mjolsness. Convergence properties of the softassign quadratic assignment algorithm. *Neural Computation*, 11(6):1455–1474, 1999.

[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.