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

Masterarbeit zum Thema:

Untersuchung & Entwicklung von Ansätzen zur Detektion von Poisoning-Angriffen

vorgelegt von

Lukas Schulth
lukas.schulth@uni.kn

unter der Betreuung von

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

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

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

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

1. Oktober 2021

Abstract

english

Zusammenfassung

deutsch

TODO:

- Algorithmen aufschreiben kmeans

Keywords— one, two, three, four

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Algorithmenverzeichnis

Inhaltsverzeichnis

1 Einführung

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

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

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

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

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

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

Clustering auf unterschiedlichen Repräsentationen der Bilder:

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

Progress of ML and examples.`https://people.csail.mit.edu/madry/lab/cleanlabel.pdf`

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 5 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 6 vorgestellt wird. Kern der Arbeit bildet Abschnitt 8, wo wir zu Beginn die grundlegenden Bestandteile des Algorithmus zur Detektion von Poisoning-Angriffen auf Neuronale Netzwerke erklären, bevor die experimentellen Ergebnisse in Unterabschnitt 7.4 ausführen. Ein Vergleich mit anderen Detektionsverfahren wird in Abschnitt 9 durchgeführt.

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

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

2 Neuronale Netzwerke

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

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

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

fehlende Interpretierbarkeit

ReLUs in den meisten Netzwerken

Definition Klasse

Supervised vs Unsupervised

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

Wir verwenden die Begriffe Bild und Datenpunkt äquivalent.

2.0.1 CNNS

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

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

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

Softmax am Ende für Transformation in Probabilities.

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

2.0.2 Besondere Schichten

Promotion Sebastian Lapuschkin

- *BatchConv* besteht aus

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

```

Listing 1: python-interner Aufbau einer BatchConv Schicht

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

Batch Normalization[3]

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

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

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

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

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

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

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

Für einen Angriff auf ein Neuronales Netzwerk existieren mehrere Möglichkeiten. Eine solche Möglichkeit ist die Störung einer Eingabe aus einer bestimmten Klasse, sodass diese Eingabe anschließend falsch klassifiziert wird, während die Störung möglichst minimal und damit schwierig zu erkennen ist. Diese Art von Angriffen ist unter dem Namen *adversarial attack* bekannt. Eine andere Art von Angriff setzt, anstatt bei der Testphase, bei der Trainingsphase an. Hier ist es das Ziel, einzelne Datenpunkte im Trainingsdatensatz so zu verändern, dass die Testgenauigkeit des Netzwerkes aufgrund von falschen Klassifikationen abnimmt. In diesem Fall sprechen wir von einem *ungerichteten Poisoning-Angriff*. Dabei ist dem Angreifer egal, wie es zu einer Verminderung der Testgenauigkeit kommt. Bei sogenannten *gerichteten Poisoning-Angriffen* sollen nur Datenpunkte einer bestimmten Klasse während der Testphase falsch klassifiziert werden.

Eine weitere Möglichkeit sind sogenannte *Backdoor-Poisoning-Angriffe*. Hierbei wird während des Trainings gezielt eine Hintertür im Netzwerk implementiert, die dann bei der Verwendung des Netzwerkes im Realbetrieb ausgenutzt werden kann, d.h. dass ein Bild in Anwesenheit eines Auslösers einer falschen Klasse zugeordnet wird, ist der Auslöser jedoch nicht vorhanden, wird das Bild korrekt klassifiziert. Wir befassen uns im Folgenden ausschließlich mit Backdoor-Poisoning-Angriffen und sprechen deshalb nur von Poisoning-Angriffen.

Kenntnisse des Angreifers: Wir gehen davon aus, dass der Angreifer volle Kenntnis und Zugriff auf den Trainingsdatensatz und die Netzwerkarchitektur besitzt.

Ziel des Angreifers: Der Angreifer verfolgt also das Ziel, während des Trainings eine Hintertü im Netzwerk zu implementieren, die über den Auslöser im Realbetrieb genutzt werden kann, sodass einzelne Datenpunkte in Anwesenheit des Auslösers einer bestimmten, falschen Klasse zugeordnet werde. Außerdem möchte der Angreifer möglichst wenig am Datensatz verändern, um einen erfolgreichen Angriff zu erhalten, d.h. es sollte nur ein kleiner Anteil des Datensatzes manipuliert werden.

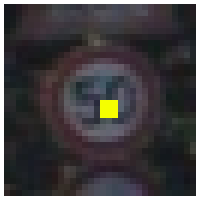

Abbildung 1: Beim Standard-Angriff wird ein Bild mit der Klasse 50km/h mit einem 3×3 Sticker und dem Label 80km/h versehen.

Den Erfolg eines Angriffs geben wir Klassen-weise als Angriffserfolgrate an. Diese ergibt sich aus dem Anteil der falsch klassifizierten Datenpunkte mit Auslöser im Testdatensatz. Wir versehen dazu jeden Datenpunkt im Testdatensatz mit einem Auslöser.

Ebenfalls wichtig für den Angreifer ist es, dass das Netzwerk in Abwesenheit eines Auslösers im Realbetrieb möglichst ähnlich gut wie dasselbe Modell, trainiert auf nicht korrumpierten Daten. Andernfalls würde sich bereits während des Trainings ein Hinweis darauf ergeben, dass der Datensatz manipuliert ist.

Somit manipuliert der Angreifer also einen Datensatz, der zum Training eine Netzwerkes verwendet wird. Die Hintertür im manipulierten Netzwerk kann der Angreifer dann im Realbetrieb ausnutzen.

Im Folgenden gehen wir auf zwei verschiedene Arten von Poisoning-Angriffen ein.

3.1 Standard Poisoning-Angriffe

Wir wollen nun eine Hintertür im Netzwerk implementieren, sodass Verkehrsschilder der Klasse 50km/h in Anwesenheit eines Auslösers als ein Verkehrsschild der Klasse 80km/h klassifiziert wird. Wir wählen diese beiden Klassen, da die Stoppschild-Klasse im Datensatz leider einen sehr kleinen Anteil ausmacht. Bei diesen Standard-Angriffen fügen wir nun bei einem bestimmen prozentualen Anteil an Schildern der Ursprungsklasse 50km/h einen gelb-grünen Sticker als Auslöser ein und ändern das Label des Bildes auf die Zielklasse 80km/h ab. Für den Sticker wählen wir den hexadezimalen Farbcode $\#f5ff00$ und eine Seitenlänge $s = 1, 2, 3$. Da wir keine Bounding-Box zur Verfügung haben, mit der wir den Sticker auf einem Schild an derselben Stelle platzieren können, fügen wir den Sticker zufällig so ein, dass die linke obere Ecke innerhalb eines festen Fensters $([x_{min}, x_{max}], [y_{min}, y_{max}])$ gesetzt wird, wobei dies die Pixel-Koordinaten in der Breite bzw. Höhe des Bildes sind. Die Pixel sind in der üblichen Lesereihenfolge durchnummeriert. Ein korrumpiertes Bild ist in Abbildung 1 dargestellt.

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 modifizierte 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 den Auslöser können wir denselben wie im Standard-Fall oder einen schwarz-weißen Sticker der Größe 3×3 zu benutzen, der in Lesereihenfolge die Farben sws-sws-wsw (s:schwarz, w:weiß) besitzt. Damit haben wir einen Angriff implementiert, bei dem Label und Bild wieder zusammenpassen.

In einem weiteren Schritt ist es möglich, die Sichtbarkeit des Auslösers zu verringern. Dabei werden nun nicht die Pixel-Werte direkt ersetzt, sondern eine Amplitude wird auf allen drei Farbkanälen addiert bzw. subtrahiert. Auf den schwarzen Feldern wird $amp \in \{16, 32, 64, 128, 256\}$ addiert auf den weißen Feldern subtrahiert. Für $amp = 256$ ergibt sich der Fall, bei dem die Pixel-Werte einfach getauscht werden. Im entsprechenden Paper werden wird auch ein Auslöser eingefügt, bei dem sich jeweils ein solcher Sticker in jeder Bildecke befindet. Um den Auslöser nicht durch die Augmentierung beim Einlesen der Trainingsdaten abzuschneiden, können wir die Sticker weiter in die Mitte verschieben.

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

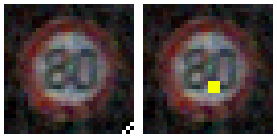

Abbildung 2: Label-konsister Poisoning-Angriff bei $\varepsilon = 300$ für zwei verschiedene Auslöser.

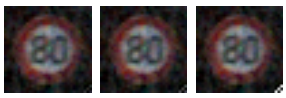

Abbildung 3: Label-konsitener Poisoning-Angriff mit Amplituden-Stickern der Amplituden $amp = 32, 64, 255$ (von links nach rechts).

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.

3.3 Bewertung von Poisoning-Angriffen

Im Trainingsdatensatz werden im Fall des Standard-Angriffs alle Bilder mit dem Sticker versehen. Die Angriffserfolgsrate(AER) beschreibt nun den Anteil an Bildern der angegriffenen 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. Wir bewerten die Qualität eines Label-konsistenten Poisoning-Angriffs also über die mittlere Angriffserfolgsrate (mAER).

4 Verteidigungen

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

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 wurde für einen Sticker mit Seitenlänge 3 und 15% korrumpierten Daten durchgeführt.

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 etwa 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 4.2.1. *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.*

4.2.1 Implementierung

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

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:

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

$$SC = \max_k \tilde{s}(k), \tag{4.1}$$

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.

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.

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 Kapitel wollen wir einen kurzen Überblick über verschiedene Methoden geben.

4.6 Bewertung von Detektionsalgorithmen

Für die Bewertung von Algorithmen/Verfahren zur Detektion von Poisoning-Angriffen betrachten wir die Anzahl an richtig-positiven (TP, englisch: true-positive), falsch-positive (FP, false-positive), falsch-negativ (FN, false-negative) und richtig-negativ(TN, true-negative) Klassifikationen. Daraus lassen sich die folgenden Raten bestimmen: Die Genauigkeit (accuracy) $acc = \frac{TP+TN}{TP+FP+FN+TN}$, die Richtig-Positiv-Rate $tpr = \frac{TP}{TP+FN}$ sowie die Richtig-Negativ-Rate $tnr = \frac{TN}{TN+FP}$ Vor allem die Richtig-Positiv-Rate ist von besonderer Bedeutung: Es ist wichtig, dass wir einen großen Anteil an korrumpierten Datenpunkten erkennen und aus dem Datensatz entfernen. Werden gleichzeitig einige saubere Datenpunkte entfernt ist diese nicht besonders problematisch, sofern dieser Anteil nicht allzu groß ist.

5 Erklärbare KI

Erklärbarkeit vs. Interpretierbarkeit, youtube talk?!

Den Kern von KI-basierten Anwendungen – womit hier im Wesentlichen Anwendungen des maschinellen Lernens gemeint sind – bilden immer die jeweils zugrundeliegenden KI-Modelle. Diese lassen sich in zwei Klassen einteilen: White- und Black-

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

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

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

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

Erklärbarkeit: Da Transparenz für diverse Modelle wie z. B. neuronale Netze nicht erreichbar ist, diese folglich nicht selbst- erklärend sind, kommt bei diesen das Konzept der „Er- klärbarkeit“ zur Anwendung. Dabei ist zwar in der Regel festgelegt, ob die Erklärbarkeit eine Entscheidung oder ein Modell betrifft. Wie eine konkrete Erklärung dabei ausgestaltet ist oder wieviel Erkenntnis sie der Zielperson gewährt, bleibt dabei jedoch zunächst unbestimmt. Beim Beispiel der Bildverarbeitung mit neuronalen Net- zen spricht man etwa bereits von einer Erklärung einer Entscheidung, wenn bestimmte Bereiche im Eingabe- bild, die zur Klassifikation eines konkreten Objektes ge- führt haben, für die anwendende Person farblich hervor- gehoben werden. In diesem Fall wird nicht jeder einzelne Schritt des Al-

gorithmus erklärt, sondern nur die für die Entscheidungsfindung bedeutsamsten Daten hervor- gehoben. Alternativ kann eine Erklärung auch durch eine textliche Beschreibung repräsentiert werden, z. B. „Auf diesem Bild ist ein Hund abgebildet, da vier Beine, eine Schnauze, Fell und ein Schwanz erkannt wurden."

Grundsätzlich wird zwischen zwei Arten von Erklärungen unterschieden:

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

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

5.1 Lokale Methoden

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

CAM / Grad-CAM / Grad-CAM++ (Gra- dient-weighted Class Activation Mapping) CAM ist eine Methode zur Visualisierung von ausschlag- gebenden Regionen für ein konkretes Klassifizierungser- gebnis eines neuronalen Netzes, insbesondere Convo- lutional Neural Network (CNN). Das Ergebnis ist eine Saliency Map, die über das ursprüngliche Bild gelegt werden kann und die betreffenden Regionen hervorhebt. Für die Erstellung der Saliency Map werden jeweils nur die letzten Schichten (Layer) des Netzes betrachtet. CAM ist nicht für jede Netzwerkarchitektur direkt an- wendbar; unter Umständen muss diese durch Hinzu- fügen weiterer Schichten zuvor angepasst und dann das Netz neu trainiert werden. Grad-CAM ist eine Generalisierung der CAM-Methode, erfordert kein erneutes Training des Modells und ist auf mehr Netzwerkarchitekturen anwendbar. Ein Nachteil von Grad-CAM ist jedoch, dass nicht mehrere Vor- kommen eines Objekts in einem Bild erkannt werden können. Grad-CAM++ löst dieses Problem, sodass das Erkennen von mehreren Objektinstanzen in einem Bild möglich wird.

LRP (Layer-Wise Relevance Propagation) Durch LRP wird der Einfluss einzelner Eingaben auf das Ergebnis einer Klassifikation betrachtet. Der Fokus liegt hierbei auf nicht linearen Klassifizierern wie neuronalen Netzen. Betrachtet man die Bildklassifikation, so ist das Ziel, für einzelne Bilder herauszufinden, welche Pixel in welchem Umfang das Klassifizierungsergebnis positiv oder negativ beeinflussen. Jedem Inputwert (hier: Pixel) wird ein Relevanzwert zugeordnet. Der Wert der „Relevanz" gibt an, wie groß der Einfluss eines Eingabewerts oder einer Unit des Netzes auf das Klassifikationsergeb- nis ist. Der Relevanzwert der Ausgabe setzt sich aus der Summe der Relevanzwerte der Eingabewerte zusam- men. Der Ausgabewert

des Netzes wird also „zerlegt“ in die jeweiligen Beiträge (bzw. den Einfluss) der Eingabe- werte (= Dekomposition). Die Berechnung der Relevanz der Eingabewerte wird iterativ von hinten (letzter Layer) nach vorn (Input-Layer) ausgeführt.

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

Sensitivitätsanalyse

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

DeepLIFT (Deep Learning Important FeaTures) DeepLIFT ist ein Erklärungswerkzeug, das zur Verbes- serung der Nachvollziehbarkeit von neuronalen Netzen genutzt wird. Bei der Methode wird einzelnen Units des neuronalen Netzes, bezogen auf einen konkreten Output (Klassifikations- oder Regressionsergebnis), ein Score zugeordnet. Wie bei der Methode Integrated Gradients wird eine Baseline genutzt: Es wird ein neutraler Input gewählt (abhängig vom konkreten Anwendungsfall), für den die Aktivierungen der einzelnen Units bzw. Neuronen des Netzes berechnet werden. Es werden also Referenzwerte bestimmt. Anschließend wird die Abweichung – der „Score“ – von diesen Referenzwerten für eine konkrete Eingabe pro Unit berechnet. Die Wahl des neutralen Inputs ist kritisch und sollte unter Nut- zung von Domänenwissen erfolgen. In einigen Fällen ist es sinnvoll, mehrere neutrale Inputs zu bestimmen und die einzelnen Scores auf Grundlage mehrerer Werte zu berechnen.

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

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

5.2 Globale Methoden

6 Layer-wise Relevance Propagation

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

6.1 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 6.1.1. *Eine Heatmap $R(x)$ heißt konservativ, falls gilt:*

$$\forall x : f(x) = \sum_p R_p(x), \tag{6.1}$$

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

Definition 6.1.2. *Eine Heatmap $R(x)$ heißt positiv, falls gilt:*

$$\forall x, p : R_p(x) \geq 0, \tag{6.2}$$

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 6.1.3. *Eine Heatmap $R(x)$ heißt konsistent, falls sie konservativ und positiv ist, d.h Definition 6.1.1 und Definition 6.1.2 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 6.1.4. *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

$$f(x) = \sum_d R_d \tag{6.3}$$

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:

$$f(x) = ... = \sum_{d \in l+1} {R_d}^{(l+1)} = \sum_{d \in l} {R_d}^{(l)} = ... = \sum_d R_d^{(1)}. \tag{6.4}$$

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:

$$\sum_i R_{i\leftarrow j}^{(l,l+1)} = R_j^{(l+1)}. \tag{6.5}$$

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:

$$R_{i\leftarrow j}^{(l,l+1)} = \frac{z_{ij}}{z_j} \cdot R_j^{(l+1)}. \tag{6.6}$$

Für diese Relevanzen $R_{i\leftarrow j}$ gilt die Erhaltungseigenschaft 6.4, denn:

$$\sum_i R_{i\leftarrow j}{}^{(l,l+1)} = R_j^{l+1} \cdot (1 - \frac{b_j}{z_j}). \tag{6.7}$$

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:

$$R_{i\leftarrow j}^{(l,l+1)} = \begin{cases} \frac{z_{ij}}{z_j+\varepsilon} \cdot R_j^{(l+1)}, & z_j \geq 0 \\ \frac{z_{ij}}{z_j-\varepsilon} \cdot R_j^{(l+1)}, & z_j < 0 \end{cases} \tag{6.8}$$

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.

6.2 Behandlung von biases

6.3 Beispiel an einem kleinen Netzwerk

6.4 Deep Taylor Decomposition

Mathematischer Hintergrund für LRP.LRP als Spezialfall von DTD

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

6.4.1 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

6.4.2 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.

6.5 Verschiedene Verfahren

6.6 Eigenschaften

Beweise in DTD Paper

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

6.7 Behandlung besonderer Schichten

6.7.1 BatchNorm2D

6.8 LRP für Deep Neural Nets/Composite LRP

6.9 Verarbeitung der Heatmaps

Aktuell benutzte defaul colormap ist Option D (Viridis)[6]

- Wertebereich
- Interpretation

[6] `https://bids.github.io/colormap/`

Abbildung 4: (Optischer) Vergleich von korrumpiertem Datenpunkt und berechnter Heatmap. Links: Verkehrsschild der Klasse 'Höchstgeschwindigkeit: 50km/h' versehen mit einem 3x3 Sticker und dem Label 'Höchstgeschwindigkeit: 80km/h'. Rechts: Zugehörige Heatmap bezüglich der Klasse 'Höchstgeschwindigkeit: 80km/h'.

- Skalen
- Normalisierung

6.10 Implementierungen

6.10.1 Tensorflow

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

```
1  torch.nn.BatchNorm1d,
2  torch.nn.BatchNorm2d
3  torch.nn.BatchNorm3d,
4  torch.nn.ReLU,
5  torch.nn.ELU,
6  Flatten,
7  torch.nn.Dropout,
8  torch.nn.Dropout2d,
9  torch.nn.Dropout3d,
```

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

```
  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

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

[11] https://github.com/fhvilshoj/TorchLRP
[12] https://github.com/chr5tphr/zennit

7 Heatmap-Clustering

In diesem Kapitel stellen wir eine neue Idee zur Detektion von Poisoning-Angriffen auf Neuronale Netzwerke vor. Dabei werden anders als beim Activation CLustering nicht die Aktivierungen der vorletzten Netzwerkschicht, sondern die durch die LRP erzeugten Heatmaps für das Clustering verwendet. Die Wahl von k für das kMeans-Clusterimng erfolgt über eine spektrale Relevanz-Analyse. Für das Clustering selbst vergleichen wir unterschiedliche Distanzbegriffe.

7.1 Idee

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

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

Diese werden wir im Folgenden näher betrachten.

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

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

7.2 k-means / k-means++ -Clustering

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

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

Eine Variation des k-Means-Algorithmus, der sowohl die Laufzeit als auch die Genauigkeit verbessert, ist der sogenannte $k-means++$-Algorithmus. Dabei wird die Initialisierungsphase in abgewandelter Form ausgeführt:

Sei k gegeben. Der erste Cluster-Mittelpunkt wird zufällig gleich-verteilt gewählt. Der nächste Mittelpunkt wird nun mit einer Wahrscheinlichkeit proportional zur quadratischen Distanz zwischen dem ersten Mittelpunkt und allen anderen Datenpunkten gewählt. Es wird so lange fortgefahren, bis alle k ersten Mittelpunkte mit maximaler Distanz zu den vorherigen Mittelpunkten gewählt wurden.

Wichtig sind für die Durchführung des kMeans-Clusterings also die Bestimmung von k sowie die Wahl einer "passenden"Metrik.

Baryzentrische Koordinaten [13]

7.3 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 Partitionierung 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

7.4 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[14].

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

Für eps=5e-2 liegen beide barycentren identisch weit weg. Probiere nun eps=5e-3

[13] `https://de.wikipedia.org/wiki/Baryzentrische_Koordinaten`
[14] `https://github.com/moboehle/Pytorch-LRP`

Berechnung der Distanzen und Aufstellen einer Affinitätsmatrix:

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

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

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

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

Berechnung Spektralen Einbettung:

Dimensionsreduktion vor dem Clustering ?!

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

Bemerkung 7.4.1. *In [AMN+19] Kapitel '2.3. Fisher Discriminant Analysis for Clever Hans identification' wird ein Verfahren vorgestellt, mit dem verdächtige Klassen identifiziert werden können. Für diese würde man anschließend das obige Verfahren durchführen*

7.5 Verwendete Distanzen & Approximationen

Um die Struktur innerhalb einer Klasse zu analysieren, benötigen wir eine Metrik. Anhand dieser wird abhängig von den Heatmps einer Klasse eine Affinitätsmatrix berechnet, die dann anschließend zur Berechnung der Spektralen Einbettung als wichtigster Schritt von SpRAy verwendet wird. Wir wollen dazu die im Folgenden vorgestellten Metriken verwenden.
Wie in [AMN+19] summieren wir über die Farbkanäle, um einen einzelnen Relevanzwert pro Pixelpunkt zu erhalten. Wir benötigen also eine Metrik zur Berechnung der Distanz zwischen 32x32 großen Heatmaps.
Wir normalisieren die Relevanzen zusätzlich auf das Intervall $[0, 1]$.
Die Wahl der Pixel mit 99 Prozent der Gesamtmasse und anschließende Normalisierung wird vermutlich durchgeführt, um die Bedingung Gleichung 8.1 zu erhalten.

8 Optimal Transport

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 Schrdinger [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? [MMG19] Optimal Transport is an interesting topic which connects many fields and has interesting applications Applications in image analysis Interplay between geometry, probability and PDEs Convexity, duality Numerical optimization Statistics and Machine Learning[15]

Optimal transport can deal with smooth and discrete measures and it has proved to be very useful for comparing distributions in a shared space, but with different (and even non-overlapping) supports [VCF+20]. Einführung OT Villani-OT hebt Distanzen von metrischen Räumen auf den Raum der metrischen Räume [Mém11].

In diesem Kapitel betrachten wir einige Konzepte aus dem Bereich Optimal Transport, um einen Distanzbegriff zu entwickeln, der im Unterschied zur pixelweisen euklidischen Distanz besser für das kMeans-Clustering geeignet sein könnte. Ausgehend von Gaspard Monge's ursprünglicher Transportproblme-Formulierung betrachten wir die durch Kantorobich eingeführte Relaxierung dieses Problems. Die Optimale Lösung wird für die Definition der p-Wasserstein-Distanz verwendet. Für eine Verallgemeinerung auf verschiedene Grundräume wird die Gromov-Wasserstein-Distanz definiert. Damit können wir zwei Heatmaps als sogenannte metrische Maßräume auffassen. Abschließend definieren wir sogenannte Gromov-Wasserstein-Baryzentren, die im Rahmen des kMeans-Clustering als Mittelwerte fungieren.

Stimmt Gromov-Wassertsien mit Wsserstein für dieselbe Distanfunktion(Metrik) überein? Welche Rolle spielt die Lossfunktion L bei der Definition? Ja, siehe Memoli 2011. L=KL

8.1 Einführung

8.1.1 Frage:

Was ist ein registration problem? Warum brauchen wir Gromov-Wasserstein? Warum reicht nicht Wasserstein? Weil wir Matrizen unterschiedlicher Größer vergleichen? Nein. Wir benutzen ja immer dieselbe Distanz für alle Pixel-Werte. Wenn wir eine Pixel-Wolke auswählen, sind das nicht dieselben Räume, da dann gewisse Koordinaten nicht zum Raum gehören. Wenn wir aber alle Pixel als Koordinaten im

[15] `https://indico.cern.ch/event/845380/attachments/1915103/3241592/Dvurechensky_lectures.pdf`

Raum auswählen würden, dann würden die metrischen Räume (X, d_X) und (Y, d_Y) übereinstimmen. Liegt das an der Definition der Baryzentren? Wie würde man das mit Baryzentren auf demselben Raum machen? s. [COT19] Translationen und Rotationen könnte ein wichtiger Punkt sein!!! verwendung von Grafiken aus anderen Papern?? Kann man ein anderes Netzwerk zur Detektion nutzen?

`https://arxiv.org/pdf/1705.09634.pdf` `https://arxiv.org/pdf/1412.5154.pdf` Für die Grudnlagen im Bereich des Optimalen Transports zitieren wir [**?**]illa-ni2009optimal.

Entscheidend für die Numerischen Resultate ist die Arbeit von Marco Cuturi [Cut13, com19]

[COT19] Remark 2.19 (Translations). zeigt dass die Unabhängigkeit gegenüber Translationen? Kann ich das auch für die Gromov-Wasserstein-Distanz zeigen?

8.1.2 Anwendungsgebiete

Optimal transport is also used widely in domain adaptation [CFTR16]. In domain adaptation, one has access to labeled examples from a source domain and unlabeled examples from a target domain. The goal is to predict labels for examples from the target domain. This is different from the typical train-test paradigm in machine learning, because covariate shift between the two domains is allowed.

Ein weitere Anwendung von Optimal Transport sind Wasserstein Generative Adversarial Networks (WGANs) [ACB17]. WGANs gehören zur Familie der Modelle namens Generative Adversarial Networks (GANs) [GPAM$^+$14]. Zu einem GAN gehören zwei Modelle, von denen beide typischerweise Neuronale Netzwerke sind. Das erste Netzwerk (generator) erzeugt Datenpunkte innerhalb des Datensatzes, die so realsitisch wie möglich sein sollen. Das zweite Netzwerk (discriminator network) versucht, zwischen den realen und den erzeugten Datenpunkten zu unterscheiden. Damit kann der Trainingsprozess von GANs als ein Spiel zwischen den beiden Netzwerken betrachtet werden, bei dem der Generator den Diskriminator täuschen und der Diskriminator die korrumpierten Datenpunkte aufdecken möchte.

Durch die Verwendung von WGANs ansatts GANs wurde ein stabilerer Trainingsprozess erreicht, da diese weniger anfällig für abrupte Trainingsabbrüche sind. Die ursprünglichen GANs benutzten eine Jensen-Shannon-Divergenz als Verlustfunktion, die nicht überall stetig und fast überall differenzierbar ist. Auch andere Verlustfunktion wie besipielsweise die KL-Divergenz weisen ähnliche Schwierigkeiten auf. Durch die Verwendung der Wasserstein-Verlustfunktion, basierend auf der 1-Wasserstein-Distanz, die überall stetig und fast überall differenzierbar ist, konnte der Trainingsprozess der GANs somit deutlich verbessert werden [Mau].

Definition 8.1.1 (Histogramm)**.** *Als Histogramm (oder: Wahrscheinlichkeitsvektor) bezeichnen wir ein Element $\boldsymbol{a} \in \Sigma_n$, das zum folgenden Wahrscheinlichkeits-Simplex gehört:*

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

Definition 8.1.2 ([Gro07])**.** *Ein metrischer Maßraum ist ein Tripel (X, d_X, μ_X) mit*

- *(X, d_X) ist ein kompakter metrischer Raum*

- μ_X *ist ein Borel-Maß auf* X *mit vollem Support, d.h.*

$$supp[\mu_X] = X. \tag{8.1}$$

Definition 8.1.3 (Isometrie)**.** *Seien zwei metrische Räume* (X, d_X) *und* (Y, d_Y) *gegeben. Wir nennen die Abbildung* $\phi : X \to Y$ *Isometrie, falls* ϕ *surjektiv ist und* $d_X(x, x') = d_Y(\phi(x), \phi(x'))$ *f.a.* $x, x' \in X$ *gilt. Für den Fall, dass eine solche Abbildung* ϕ *existiert, nennen wir* X *und* Y *isometrisch.*

Definition 8.1.4 (Isomorphie)**.** *Zwei MMR* (X, d_X, μ_X) *und* (Y, d_Y, μ_Y) *heißen isomorph, falls eine Maß-erhaltende Abbildung* $f : X \to Y$ *existiert, die* $f_{\#}\mu_X = \mu_Y$ *erfüllt.*

Mit $\mathcal{G}_{\mathcal{W}}$ bezeichnen wir die Menge aller metrischen Maßräume.

Beispiel 8.1.5. *Wir betrachten die beiden metrischen Maßräume* $(\{a, b\}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}), \{\frac{1}{2}, \frac{1}{2}\})$ *und* $(\{a', b'\}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \{\frac{1}{4}, \frac{3}{4}\})$ *Dann sind diese Räume isometrisch, jedoch nicht isomorph, vergleich dazu Abbildung 5*

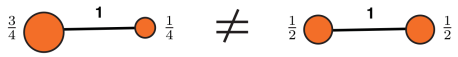

Abbildung 5: Die beiden metrischen Maßräume sind ismetrisch, aber nicht isomorph.

Quelle: [COT19]

Definition 8.1.6 (pushforward-Maß/Bildmaß)**.** *Seien* X, Y *zwei Maßräume,* $T : X \to Y$ *eine messbare Abbildung und* μ *ein Maß auf* X*. Dann ist das pushforwad-Maß* $T^{\#}\mu$ *ein Maß auf* Y*, definiert durch*

$$T^{\#}\mu(A) = \mu(T^{-1}(A)) \tag{8.2}$$

für alle $A \subset Y$*.*

Definition 8.1.7 (Kopplung)**.** *Ein Wahrscheinlichkeitsmaß* μ *auf* $X \times Y$ *heißt Kopplung zwischen* μ_x *und* μ_Y*, falls* $(\mu_1)_{\#}\mu = \mu_X$ *und* $(\mu_2)_{\#}\mu = \mu_Y$ *gilt. Wir benzeichnen mit* $\Pi(\mu_X, \mu_Y)$ *die Menge aller Kopplungen ziwschen* μ_X *und* μ_Y*.*

Einführung W-Theorie am KIT [16]

Definition 8.1.8 (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.*

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

Definition 8.1.9 ([BBB$^+$01]). *Sei X eine beliebige Menge. Eine Funktion $d : X \times X \to \mathbb{R} \cup \{\infty\}$ heißt Metrik auf X, falls die folgenden Bedingungen für alle $x, y, z \in X$ gelten:*

(1) Positive Definitheit: $d(x, y) > 0$ für $x \neq y$ und $d(x, y) = 0$ für $x = y$,

(2) Symmetrie: $d(x, y) = d(y, x)$,

(3) Dreiecksungleichung: $d(x, z) \leq d(x, y) + d(y, z)$.

Ein *metrischer Raum* ist eine Menge, versehen mit einer Metrik. Formal gesehen ist ein metrischer Raum ein Paar (X, d), wobei d eine Metrik auf X ist. Elemente von X heißen Punkte und $d(x, y)$ bezeichnet die Distanz zwischen x und y.

Geodesic Space [17]

Gromov-Hausdorff-Distanz

Definition 8.1.10 (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*

$$d_H(X, Y) = \max\{\sup_{x \in X} \inf_{y \in Y} d(x, y), \sup_{y \in Y} \inf_{x \in Y} d(x, y)\}. \tag{8.3}$$

Definition 8.1.11 (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 8.1.12 (Correspondance). *content...*

Definition 8.1.13 (Kopplung). *Seien*

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

$$\Gamma_{X,Y} : (X \times Y) \times (X \times Y) \to \mathbb{R}^+, \tag{8.4}$$

gegeben durch

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

Definition 8.1.14 (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 8.1.15 (Diskretes Maß).

8.2 Kantorovichs Optimal Transport

8.2.1 Optimaler Transport (Monge Formulierung)

Bemerkung 8.2.1. *Problem zwischen diskreten Maßen*

$$\min_T \{\sum_i c(x_i, T(x_i)) : T_\# \alpha = \beta\} \tag{8.5}$$

Bemerkung 8.2.2 (Fehlende Eindeutigkeit).

[17] `http://www.math.toronto.edu/mccann/papers/FiveLectures.pdf`

Dieses Problem wurde zuerst von Gaspard Monge im Jahr 1781 formuliert. In dieser Formulierung wurde Abbildung gesucht, die jedem Punkt in der Ausgangsverteilung einen Punkt in der Zielverteiliung zuordnet. Dabei wird die gesamte Masse eines Punktes in der Startverteilung an einen Punkt in der Zielverteilung verschoben. Eine solche Abbildung ist in Abbildung 6 dargestellt.

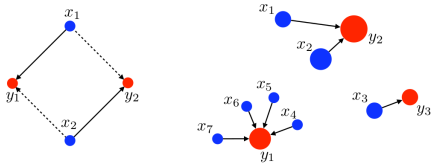

Abbildung 6: Links: Fehlende Eindutigkeit in der Zuordnung. Die beiden Punkte x_1 und x_2 können sowohl den Punkten y_1 bzw. y_2 zugeordnet werden, um eine zulässige Abbildung zu erhalten. **Rechts:** Die Monge-Abbildung assoziiert das blaue Maße $\boldsymbol{\alpha}$ mit dem roten Maß $\boldsymbol{\beta}$. Dabei ist die Masse in den jeweiligen Punkten über den Flächeninhalt der Kreise dargestellt.

Quelle: [COT19]

Für die exakte Forumlierung im diskreten Fall benötigen wir die folgende Definiton.

Definition 8.2.3 (Bildmaß)**.** *Für zwei Maßräume (X, Σ_X) und (Y, Σ_Y) und eine messbare Abbildung $f : X \to Y$ ist das Bildmaß $f_{\#}\mu$ auf (Y, Σ_Y) definiert als $f_{\#}\mu = \mu(f^{-1(B)})$ für alle $B \in \Sigma_Y$.*

Für den Fall zweier diskreter Maße

$$\alpha = \sum_{i=1}^{n} \boldsymbol{a}_i \delta_{x_i} \text{ und } \beta \sum_{j=1}^{m} \boldsymbol{b}_j \delta_{y_j} \tag{8.6}$$

können wir das Monge-Problem dann wie folgt formulieren:

Gesucht ist eine Abbildung $T : \{x_1, ..., x_n\} \to \{y_1, ..., y_m\}$, die die Bedingung

$$\forall j \in \{1, ..., m\} \boldsymbol{b}_j = \sum_{i:T(x_i)=y_i} \boldsymbol{a}_i \tag{8.7}$$

erfüllt, die wir in Kurzform als $T_{\#}\alpha = \beta$ schreiben. Aufgrund der Positivität von $\boldsymbol{b}$ ist die Abbildung notwendigerweise surjektiv. Als weitere Forderung gilt, dass die Abbildung T minimal bezüglich einer Transportkostenfuktion $c(x, y)$ für $x, y \in \mathbb{R} \times \mathbb{R}$ sein soll:

$$\min_T \sum_i c(x_i, t(x_i)) : T_{\#}\alpha = \beta. \tag{8.8}$$

Neben der fehlenden Eindeutigkeit einer solchen Abbildung ist auch die Existenz nicht immer gegeben.

8.2.2 Optimaler Transport nach Kantorovich

[San10] ist eine gute, einfache Einführung. Damit kann ich vermutlich Monge und Kantorovich beschreiben.

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.

Ein solcher Transportplan ist für drei verschiedene Ausgangssituationen in **??** dargestellt.

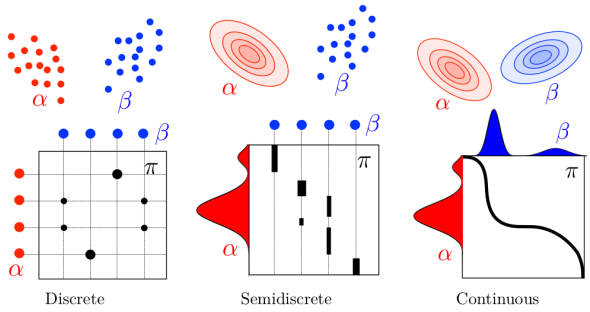

Abbildung 7: Links: Kopplung ziwschen zwei diskreten Wahrscheinlichkeitsverteilungen. **Mitte:** Kopplung zwischen einer konitnuierlichen W-Verteilung $\boldsymbol{\alpha}$ und einer diskreten W-Verteilung $\boldsymbol{\beta}$. **Rechts:** Kopplung im kontnuierlichen Fall.

Quelle: [COT19]

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

$$W_M(\boldsymbol{a}, \boldsymbol{b}) = \min_{T \in \Pi(\boldsymbol{a}, \boldsymbol{b})} \langle \boldsymbol{T}, \boldsymbol{M} \rangle, \tag{8.9}$$

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

Die Lösung des Problems ist eine Kopplung/Kopplungsmatrix, die beschreibt, wie viel Masse von einem Punkt zum anderen Punkt fließt. In einer Kostenmatrix derselne Größe sind die Kosten abgespeichert, um von einem zum anderen Punkt zu kommen.

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

$$L_C(\boldsymbol{a}, \boldsymbol{b}) := \min_{P \in \Pi(\boldsymbol{a}, \boldsymbol{b})} \langle \boldsymbol{C}, \boldsymbol{P} \rangle := \sum_{i,j} \boldsymbol{C}_{i,j} \boldsymbol{P}_{i,j} \tag{8.10}$$

Diese Problem ist ein lineares Problem. Für diese Art von Problemen ist die optimale Lösung nicht notwendigerweise eindeutig. Kantorovich gilt als Erfinder der linearen Optimierung[18].

Proposition 8.2.4. *Die Lösung des regularisierten Problems* **??** *besitzt die Form*

$$\forall (i, j) \in \{1, ..., n\} \times \{1, ..., m\} : \boldsymbol{P}_{i,j} = \boldsymbol{u}_i \boldsymbol{K}_{i,j} \boldsymbol{v}_j \tag{8.11}$$

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 **??** erhalten wir damit:

$$\mathcal{L}(\boldsymbol{P}, \boldsymbol{f}, \boldsymbol{g}) = \langle \boldsymbol{P}, \boldsymbol{C} \rangle - \varepsilon \boldsymbol{H}(\boldsymbol{P}) - \langle \boldsymbol{f}, \boldsymbol{P}\mathbf{1}_m - \boldsymbol{a} \rangle - \langle \boldsymbol{g}, \boldsymbol{P}^\top \mathbf{1}_n - \boldsymbol{a} \rangle. \tag{8.12}$$

Mit der Optimalitätsbedingung erster Ordnung ergibt sich

$$\frac{\partial \mathcal{L}(\boldsymbol{P}, \boldsymbol{f}, \boldsymbol{g})}{\partial \boldsymbol{P}_{i,j}} = \boldsymbol{C}_{i,j} + \varepsilon \log(\boldsymbol{P}_{i,j}) - \boldsymbol{f}_i - \boldsymbol{g}_j = 0, \tag{8.13}$$

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 8.2.5 (Algorithmus von Sinkhorn)**.** *Die Faktorisierung der Lösung in Gleichung 8.11 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:*

$$diag(\boldsymbol{u}) \boldsymbol{K} diag(\boldsymbol{v}) \mathbf{1}_m = \boldsymbol{a} \text{ und } diag(\boldsymbol{v}) \boldsymbol{K}^\top diag(\boldsymbol{u}) \mathbf{1}_n = \boldsymbol{b}. \tag{8.14}$$

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

$$\boldsymbol{u} \odot (\boldsymbol{K}\boldsymbol{v}) = \boldsymbol{a} \text{ und } \boldsymbol{v} \odot (\boldsymbol{K}^\top \boldsymbol{v}) = \boldsymbol{a}, \tag{8.15}$$

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 8.15 erfüllt ist, und anschließend die Modifikation von* $\boldsymbol{v}$ *vorgenommen wird, sodass die*

[18] OTNotes_campride.pdf

rechte Seite in Gleichung 8.15 gilt. Mit diesen beiden Modifikationen erhalten wir den Algorithmus von Sinkhorn, der aus den beiden folgenden Updates besteht:

$$\boldsymbol{u}^{(l+1)} := \frac{\boldsymbol{a}}{\boldsymbol{K}\boldsymbol{v}^{(l)}} \text{ und } \boldsymbol{v}^{(l+1)} := \frac{\boldsymbol{b}}{\boldsymbol{K}^\top \boldsymbol{u}^{(l+1)}}, \tag{8.16}$$

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 [Sin64], hence the name of the algorithm.

Bemerkung 8.2.6. *Die Lösung zum regularisierten Problem lässt sich nach den Sinkhorn-Iterationen wie folgt angeben: $P_L = diag(U_L)Kdiag(v_L)$ Das ist der optimal Transport plan für das Regularisierte Problem.Optimal Transport Caost lässt sich berechnen als: $\langle P_L, M_{XY}\rangle = u_L^\top (K \odot M_{XY})v_L$.*

Für die globale Konvergenzanalayse der Sinkhorn-Algorithmus lässt sich ein Zusammenhang mit der Hilbertschen Projektionsmetrik benutzen, der erstmals in [FL89] vorgestellt wurde.

Diese ist wie folgt definiert:

Definition 8.2.7 (Projektive Hilbert-Metrik)**.** *Seien $u, u' \in \mathbb{R}^n_{+,\star}$. Dann ist die projektive Hilbert-Metrik $d_{\mathcal{H}}$ definiert durch*

$$d_{\mathcal{H}}(u, u') := \log \max_{i,j} \frac{u_i u'_j}{u_j u'_i}. \tag{8.17}$$

Bemerkung 8.2.8. *It can be shown to be a distance on the projective cone R n +,* / , where u u 0 means that $\exists r > 0, u = ru0$ (the vectors are equal up to rescaling, hence the name "projective"). This means that d H satisfies the triangular inequality and d H (u, u 0) = 0 if and only if u u 0 . This is a projective version of Hilbert's original distance on bounded open convex sets [Hilbert, 1895]. The projective cone R n +,* / is a complete metric space for this distance. By a logarithmic chan*

the Hilbert metric on the rays of the positive cone is isometric to the variation seminorm (it is a norm between vectors that are defined up to an additive constant) $dH(u, u_0) \log(u) - log(u_0)$

where (4.21) var def. wobei $||f||_{var} := (\max_i f_i) - (\min_i f_i)$.

This variation seminorm is closely related to the ' norm since one always has $||f||_{var} \leq ||f||_\infty$. If one imposes that f i = 0 for some fixed i, then a converse inequal- ity also holds since $||f||_\infty \leq ||f||\_var$. These bounds are especially useful to analyze Sinkhorn convergence (see Remark 4.14 below), because dual variables f = log(u) solving (4.14) are defined up to an additive constant, so that one can impose that f i = 0 for some i.

The Hilbert metric was introduced independently by [Birkhoff, 1957] and [Samelson et al., 1957]. They proved the following fundamental theorem, which shows that a positive matrix is a strict contraction on the cone of positive vectors.

Theorem 8.2.9. *Sei $K \in \mathbb{R}_{+,\star}^{n\times m}$. Dann gilt für $(v, v') \in (\mathbb{R}_{+,\star}^{m})^2$*

$$d_{\mathcal{H}}(Kv, Kv') \leq \lambda(K) d_{\mathcal{H}}(v, v'), \text{ mit } \begin{cases} \lambda(K) := \frac{\sqrt{\nu(K)}-1}{\sqrt{\nu(K)}+1} \\ \nu(K) := \max_{i,j,k,l} \frac{K_{i,k}K_{j,l}}{K_{j,k}K_{i,l}} \end{cases} \tag{8.18}$$

Visualisierung des Theorems vgl [COT19],Seite 70.
Mit diesem Resultat lässt sich die Konvergenz des Verfahrens zeigen:

Theorem 8.2.10. *Es gilt $(u^{(l)}, v^{(l)}) \to (u^\star, v^\star)$ und*

$$d_{\mathcal{H}}(u^{(l)}, u^\star) = \mathcal{O}(\lambda(K)^{2l}), \qquad d_{\mathcal{H}}(v^{(l)}, v^\star) = \mathcal{O}(\lambda(K)^{2l}) \tag{8.19}$$

Es gelten außerdem die beiden folgenden Abschätzungen

$$d_{\mathcal{H}}(u^{(l)}, u^\star) \leq \frac{d_{\mathcal{H}}(P^{(l)}\mathbf{1}_m, a)}{1-\lambda(K)^2}, \tag{8.20}$$

$$d_{\mathcal{H}}(v^{(l)}, v^\star) \leq \frac{d_{\mathcal{H}}(P^{(l)\top}\mathbf{1}_n, b)}{1-\lambda(K)^2}, \tag{8.21}$$

wobei

$$P^{(l)} := diag(u^{(l)}) K diag(v^{(l)}) \tag{8.22}$$

gilt. Desweiteren gilt die Abschätzung

$$||\log(P^{(l)}) - \log(P^\star)||_\infty \leq d_{\mathcal{H}}(u^{(l)}, u^\star) + d_{\mathcal{H}}(v^{(l)}, v^\star), \tag{8.23}$$

wobei $P^\star$ die eindeutige Lösung zu **??** *ist.*

Beweis. Für ein beliebiges Paar $(v, v') \in (\mathbb{R}_{+,\star}^{m})^2$ gilt

$$d_{\mathcal{H}}(v, v') = d_{\mathcal{H}}(v/v', \mathbf{1}_m) = d_{\mathcal{H}}(\mathbf{1}_m/v, \mathbf{1}_m/v').$$

Mit dieser Beziehung und Theorem 8.2.9 erhalten wir

$$\begin{aligned} d_{\mathcal{H}}(u^{(l+1)}, u^\star) &= d_{\mathcal{H}}\left(\frac{a}{Kv^{(l)}, \frac{a}{Kv^\star}}\right) \\ &= d_{\mathcal{H}}(Kv^{(l)}, Kv^\star) \\ &\leq \lambda(K) d_{\mathcal{H}}(v^{(l),v^\star}). \end{aligned}$$

Dies zeigt Gleichung 8.19. Mit Hilfe der Dreiecksungleichung erhalten wir

$$\begin{aligned} d_{\mathcal{H}}(u^{(l)}, u^\star) &\leq d_{\mathcal{H}}(u^{(l+1)}, u^{(l)}) + d_{\mathcal{H}}(u^{(l+1)}, u^\star) \\ &\leq d_{\mathcal{H}}\left(\frac{a}{Kv^{(l)}}, u^{(l)}\right) + \lambda(K)^2 d_{\mathcal{H}}(u^{(l),u^\star}) \\ &= d_{\mathcal{H}}\left(a, u^{(l)} \odot (Kv^{(l)})\right) + \lambda(K)^2 d_{\mathcal{H}}(u^{(l)}, u^\star) \\ &= d_{\mathcal{H}}\left(a, P^{(l)}\mathbf{1}_m\right) + \lambda(K)^2 d_{\mathcal{H}}(u^{(l)}, u^\star) \end{aligned}$$

und damit nach Division durch $1-\lambda(K)^2$ Gleichung 8.20. Die zweite Abschätzung, Gleichung 8.21, folgt analog. Die Gleichung 8.23 gilt nach Lemma 3 in [FL89]. □

Bemerkung 8.2.11.

- *Konvergenzrate?*
- *Berechnung der optimalen Transportpläne mit Gleichung 8.22 im Zeitschritt l und Berechnung der Transportkosten ...*
- *Abbruchkriterien (vgl. mit der POT Implementierung)*

8.2.3 Komplexitätsanalyse

[AWR17] liefert eine Komplexitätsanalyse für die Sinkhorn-Iterationen. Im Fall $n = m$ sind für die Wahl $\varepsilon = \frac{4\log(n)}{\tau}$ $\mathcal{O}(||C||_\infty^3 \log(n)\tau^{-3})$ Sinkhorn-Iterationen (inklusive eines Rundungsschrittes) notwendig, um eine zulässige Kopplung $\hat{\boldsymbol{P}} \in \Pi(\boldsymbol{a}, \boldsymbol{b})$ zu berechnen, die die Abschätzung $\langle \hat{\boldsymbol{P}}, \boldsymbol{C} \rangle \leq L_{\boldsymbol{C}}(\boldsymbol{a}, \boldsymbol{b} + \tau)$ zu berechnen. Somit liefert das Sinkhorn-Verfahren eine τ-Approximation des nicht regularisierten Optimal Transport-Problems in $\mathcal{O}(n^2 \log(n)\tau^{-3})$ Operationen. Gleichzeitig wir dort eine greedy Variante der Sinkhorn-Iterationen namens Greenkhorn präsentiert, die eine τ-Approximation in $\mathcal{O}(n^2\tau^{-3})$ Operationen liefert. [DGK18] verbessert diese auf $\mathcal{O}(n^2\tau^{-2})$ Operationen.

8.3 Gromov-Wasserstein-Divergenz

Bisher hatten wir stets Wasserstein-Distanzen auf demselben metrischen Raum betrachtet, d.h wir hatten die Verteilungen μ_X und ν_X beiden metrischen Maßräume (X, d_X, μ_X) und (X, d_X, ν_X) verglichen. In diesem Kapitel wollen wir nun einen Distanzbegriff für Verteilungen auf unterschiedlichen metrischen Maßräumen herleiten. Wir folgen dafür [Mém11], []

8.3.1 Gromov-Wasserstein-Divergenz

Fast Computation of Wasserstein Barycenters[19]

Definition 8.3.1 (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 8.3.2 (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*

$$\mathcal{L} \otimes T := \left(\sum_{k,l} \mathcal{L}_{i,j,k,l} T_{k,l} \right)_{i,j}. \tag{8.24}$$

Definition 8.3.3 (Entropie)**.** *Für $T \in \mathbb{R}_+^{N \times N}$ definieren wir die Entropie als*

$$H(T) := -\sum_{i,j=1}^{N} T_{i,j}(log(T_{i,j}) - 1). \tag{8.25}$$

[19] https://arxiv.org/pdf/1310.4375.pdf

Warum wir GW-Distanz brauchen `https://arxiv.org/pdf/1811.02834.pdf`

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. 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.

[Mém11] Durch die Definition

$$\mathcal{L}(C,\bar{C}) := (L(C_{i,k},\bar{C}_{j,l}))_{i,j,k,l} \tag{8.26}$$

erhalten wir

$$\epsilon_{C,\bar{C}}(T) = \langle \mathcal{L}(C,\bar{C}) \otimes T, T\rangle \tag{8.27}$$

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 8.3.4. *Die Verlustfunkton L lasse sich schreiben als*

$$L(a,b) = f_1(a) + f_2(b) - h_1(a)h_2(b) \tag{8.28}$$

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

$$\mathcal{L}(C,\bar{C}) \otimes T = c_{C,\bar{C}} - h_1(C)Th_2(\bar{C})^T, \tag{8.29}$$

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 8.28 gilt nach der Tensor-Matrix-Multiplikation Gleichung 8.24 die Zerlegung $\mathcal{L}(C,\bar{C}) \otimes T = A + B + C$ mit

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

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

Bemerkung 8.3.5 (Verbesserte Komplexität)**.** *Mit dem Resultat in Theorem 8.3.4 können wir $\mathcal{L}(C,\bar{C})\otimes T$ effizient in der Größenordnung $\mathcal{O}(N_1^2N_2 + N_2^2N_1)$ mit ausschließlich Matrix/Matrix-Multiplikationen berechnen im Unterschied zur Komplexität von $\mathcal{O}(N_1^2N_2^2)$ für die Implementierung von Gleichung 8.24.*

Bemerkung 8.3.6 (Spezialfälle)**.** *Im Fall $L = L_2$ ist die Bedingung Gleichung 8.28 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 **??** und definieren:

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

$$T \leftarrow Proj^{KL}_{\mathcal{C}_{p,q}}\left(T \odot e^{-\tau(\nabla \boldsymbol{\varepsilon}_{C,\bar{C}}(T) - \varepsilon H(T))}\right), \tag{8.30}$$

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

$$Proj^{KL}_{\mathcal{C}_{p,q}}(K) := \underset{T' \in \mathcal{C}_{p,q}}{\arg\min}\, KL(T'|K). \tag{8.31}$$

Proposition 8.3.7. *Für den Fall $\tau = 1/\varepsilon$ erhalten wir die Iterationsvorschrift*

$$T \leftarrow \mathcal{T}(\mathcal{L}(C, \bar{C}) \otimes T, p, q). \tag{8.32}$$

Beweis. Nach [BCC$^+$15] ist die Projektion in 8.30 gegeben durch die Lösung des regularisirten Transportproblems **??** und ist damit gegeben durch:

$$Proj^{KL}_{\mathcal{C}_{p,q}}(K) = \mathcal{T}(-\varepsilon\log(K), p, q) \tag{8.33}$$

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

$$\nabla \boldsymbol{\varepsilon}_{C,\bar{C}}(T) - \varepsilon H(T) = \mathcal{L}(C, \bar{C}) \otimes T + \varepsilon\log(T). \tag{8.34}$$

Durch Umordnen der Terme in Gleichung 8.30 erhalten wir für $\tau\varepsilon = 1$ die angegebene Vorschrift. □

Bemerkung 8.3.8. *Die Iterationsvorschrift 8.32 definiert einen einfachen Algorithmus, der in jedem Update von T eine Sinkhorn-Projektion benötigt.*

Bemerkung 8.3.9 (Konvergenz)**.** *Die Iterationen Gleichung 8.30 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 8.3.7 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 8.3.10. *Für gewöhnlich wählt man die Schrittweite τ im Gradientenverfahren nicht zu groß, um Konvergenz des Verfahrens zu erhalten. Da die Schrittweite τ hier jedoch antiproportional zum Regularisierungs-Parameter ε ist, gibt es einen Trade-off zwischen Konvergenz des Verfahrens und einer Unschärfe in der optimalen Lösung.*

Bemerkung 8.3.11 (Wahl von ε)**.** *In [Cut13] werden verschiedene Werte für ε angegeben. Diese sind $\varepsilon = 0.02, 0.1, 1.0$*

Im zugehörigen Beispiel[20] *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?

8.3.2 Gromov-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[21]

Definition 8.3.12 (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

$$\min_{C \in \mathbb{R}^{N \times N}} \sum_s \lambda_s GW_\varepsilon(C, C_s, p, p_s). \tag{8.35}$$

Existenz und Eindeutigkeit von Baryzentren: siehe [AC11].

Bemerkung 8.3.13 (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*[22]*.*

Bemerkung 8.3.14. *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].*

Wir können das Bary-Zentrum mithilfe von Kopplungen umformulieren als

$$\min_{C,(T_s)_s} \sum_s \lambda_s(\varepsilon_{C,C_s}(T_s) - \varepsilon H(T_s)), \tag{8.36}$$

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

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

[21] `https://hal.archives-ouvertes.fr/hal-00637399/document`

[22] `https://pythonot.github.io/auto_examples/gromov/plot_gromov_barycenter.html`

Minimierung bezüglich $(T_s)_s$. Anhand der Umformulierung Gleichung 8.36 sehen wir, dass das Optimierungsproblem Gleichung 8.35 bezüglich $(T_s)_s$ in S (?viele) unabhängige GW_ε-Optimierungen

$$\forall s : \min_{T_s \in \mathcal{C}_{p,p_s}} \mathcal{E}_{C,C_s}(T_s) - \varepsilon H(T_s) \tag{8.37}$$

zerfällt, die jeweils wie in Theorem 8.3.7 angegeben gelöst werden können.
Minimierung bezüglich C. Sei (T_s) gegeben. Dann lautet, die Minimierung bezüglich C:

$$\min_C \sum_s \lambda_s \langle \mathcal{L}(C, C_s) \otimes T, T \rangle. \tag{8.38}$$

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

Proposition 8.3.15. *Sei L eine Verlustfunktion, die die Bedingung Gleichung 8.28 erfüllt. Sei f_1'/h_1' invertierbar.*
Dann lässt sich die Lösung zu Gleichung 8.38 schreiben als:

$$C = \left(\frac{f_1'}{h_1'}\right)^{-1} \left(\frac{\sum_s \lambda_s T_s^\top h_2(C_s) T_s}{pp^\top}\right), \tag{8.39}$$

wobei die Normalisierung $\lambda_s = 1$ gilt.

Beweis. Nach Theorem 8.3.4 können wir das zu minimierende Funtional schreiben als

$$\sum_s \lambda_s \rangle f_1(C) p \mathbf{1}^\top + \mathbf{1} p_s^\top f_2(C_s) - h_1(C) T_s h_2(C_s)^\top, T_s \rangle. \tag{8.40}$$

Die Optimalitätsbedingung erster Ordnung lautet folglich

$$f_1'(C) \odot pp^\top = h_1'(C) \odot \sum_s \lambda_s T_s h_2(C_s) T_s^\top. \tag{8.41}$$

Durh Umstellen der Gleichung erhalten wir die angegebene Form. □

Anhand von Gleichung 8.39 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 8.39 die folgende Vorschrift:

$$C \leftarrow \frac{1}{pp^\top} \sum_s \lambda_s T_s^\top C_s T_s. \tag{8.42}$$

Proposition 8.3.16. *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 8.42 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:

$$C \leftarrow \left(\frac{1}{pp^\top} \sum_s \lambda_s T_s^\top log(C_s) T_s \right). \tag{8.43}$$

Algorithm 1 Berechnung der GW_ε -Baryzentren

Input: $(C_s, p_s), p$.
Output: C.
Initialize C.
repeat
  // minimize over $(T_s)_s$
  for $s = 1$ **to** S **do**
    Initialize T_s.
    repeat
      // Compute $c_s = \mathcal{L}(C, C_s) \otimes T_s$ using **??**
      $c_s \leftarrow f_1(C) + f_2(C_s)^\top - h_1(C) T_s h_2(C_s)^\top$
      // Sinkhorn iterations to compute $\mathcal{T}(c_s, p, q)$
      Initialize $a \leftarrow \mathbf{1}$, set $K \leftarrow e^{-c_s/\varepsilon}$.
      repeat
        $a \leftarrow \frac{p}{Kb}, b \leftarrow \frac{q}{K^\top a}$.
      until convergence
      $T_s \leftarrow diag(a) K diag(b)$.
    until convergence
  end for
  // Minimize over C
  $C \leftarrow \left(\frac{f_1'}{h_1'}\right)^{-1} \left(\frac{\sum_s \lambda_s T_s^\top h_2(C_s) T_s}{pp^\top} \right)$
until convergence

Psuedocode.

Häufige Verwendungungen(s. Einführung hier [23]). 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.

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

8.3.3 Komplexitätsanalyse

8.3.4 Implementierung

9 (Numerische) Ergebnisse/Vergleich mit anderen Verfahren

Wir konnten beobachten, dass die Qualität des angegriffenen Netzwerkes auf nicht korrumpierten Daten für alle durchgeführten Angriffe über 96 Prozent beträgt. Deshalb werden wir diese Kennzahl als Qualität des Angriffs nicht jeweils zusätzlich angeben.

9.1 Clustering auf den Rohdaten

Für das Clustering auf den Rohdaten bei einem Anteil von 15 Prozent und einer Stickergröße von 3×3 erhalten wir die Tabelle 2 in dargestellten Werte unter Verwendung der L2-Distanz. Dazu werden die Bilder als Graustufenbild eingelesen und anschließend in der Dimension auf 10 reduziert.

	PCA	FastICA
ACC	65.49	63.13
TPR	51.54	31.99
TNR	72.36	78.48

Tabelle 2: Auswertung des Clusterings auf den rohen Bilddaten bei 33 Prozent korrumpierten Daten mit einem Sticker der Größe 3×3. Alle numerischen Werte sind in Prozent angegeben.

9.2 Standard Poisoning-Angriffe

Bemerkung 9.2.1 (Die Fälle mit einer AER uner 100 Prozent)**.** *Besonders interessant sind auch genau die Fälle, bei denen die Hintertür im NEtzwerk so implementiert ist, dass nicht alle präsentierten korrumpierten samples gewollt falsch klassifiziert werden. Was erwarten wir hier von unserem Detektionsalgorithmus? Wenn wir das Gromov-Wasserstein-Clustering auf den Heatmaps durchführen, können wir ja nur die Informationen nutzn, die das Netzwerk über die LRP liefert. Diese ist aber schon fehlerhaft/mangelhaft. Vielleicht sollte man hier dann das Clustering wieder auf den rohen Bilddaten durchführen? Wleche Pixel nimmt man hier dann? Alle? Wichtig ist hier auch der Vergleich mit dem Clustering bezüglich der L2 Distanz auf den EingabeBilder.*

Seitenlänge s des Triggers	Prozentualer Anteil	AER	DR (HC)			DR (AC,PCA)		
			ACC	TPR	TNR	ACC	TPR	TNR
s=3	33.00	100.00	99.96	99.88	100.00	94.76	90.77	96.73
	15.00	100.00	100.00	100.00	100.00	76.51	99.31	72.48
	10.00	100.00	99.29	92.9	100.00	87.62	96.17	86.17
	5.00	99.6	99.48	90.8	99.94	57.92	20.69	59.88
	2.00	100.00	52.85	0.0	53.94	52.91	0.0	54.0
s=2	33.00	1.0	100.00	100.00	100.00	99.72	99.14	100.00
	15.00	100.00	100.00	100.00	100.00	87.53	97.59	85.76
	10.00	100.00	99.72	100.00	99.69	67.21	93.99	64.24
	5.00	100.00				47.44	10.34	49.39
	2.00	100.00				23.02	4.43	32.18
	1.00	92.00						
	0.5	99.87						
	0.25	57.6						
s=1	33.0	100.0	100.00	100.00	100.00	98.78	97.66	99.33
	15.0	100.0	100.00	100.00	100.00	96.29	79.04	99.33
	10.0	100.0	100.00	100.00	100.00	71.47	87.43	69.7
	5.00	100.0	100.00	100.00	100.00	73.4	100.00	72.0
	2.00	100.0	100.00	100.00	100.00	58.31	100.00	57.45
	1.00	13.87	60.77	47.06	60.91	56.99	100.00	56.55

Tabelle 3: Qualität der Detektion unterschiedlich starker Angriffe mithilfe von LRP-Clustering(?) und Gromov-Wasserstein-Distanzen. Das LRP-Clustering für s=1. wurde mit eps=0.1 durchgeführt(seed=0).

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

Clustering(GWD):

Bemerkung 9.2.2. *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.

9.3 Label-konsistente Poisoning-Angriffe

Bemerkung 9.3.1. *Für einen einzelnen Amplitudensticker mit d=10 ist das eigentlich identisch zu dem Angriff mit dem Sticker*

In Abbildung 12 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

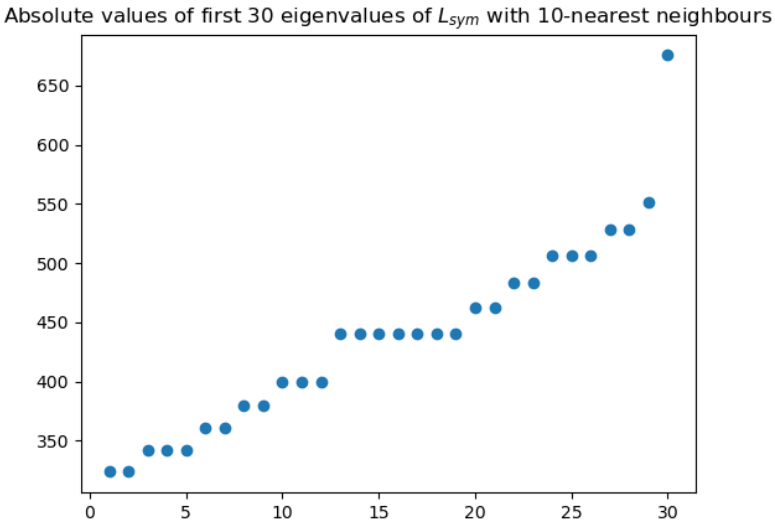

Abbildung 8: Ergebnis des spektralen Clusterings unter Verwendung der Euklidischen Distanz und k=10 Nachbarn

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:

9.4 Activation Clustering

9.5 Auswertung der CLPA

Wir werten heir die Angriffe und Verteidigungen bei CLPA aus:

Alle Rotationen: 33 Prozent korrumpierte Daten [0.9 0.85416667 0.93066667 1. 0.9969697 nan 1. 1. 1. 0.97083333 0.99393939 0.93333333 0.74057971 0.95277778 0.98888889 0.83333333 0.99333333 0.725 0.96666667 1. 0.94444444 0.97777778 0.95 0.98666667 1. 0.90625 1. 1. 0.98 1. 0.94 1. 1. 0.8 0.30833333 0.88717949 0.75833333 0.98333333 0.81449275 0.37777778 1. 0.81666667 1.]

mAER: 90.98034373809526 min: 30.833333000000003 max2 100.0

AC:(tn, fp, fn, tp): 1106 0 15 529 ACC: 99.09 TPR: 97.24 TNR: 100.0

HC: (tn, fp, fn, tp): 1106,0,7,537 acc: 99.58 tpr: 98.71 tnr: 100.0

15 Prozent: 0.85 0.77361111 0.88 1. 0.96212121 nan 1. 0.97555556 1. 0.92708333 0.98030303 0.69761905 0.68405797 0.8875 0.86666667 0.71428571 0.87333333 0.55555556 0.85897436 0.9 0.93333333 0.77777778 0.81666667 0.86666667 1. 0.78541667 0.84444444 1. 0.89333333 0.66666667 0.84666667 1. 0.88333333 0.76666667 0.275 0.76666667 0.6 0.95 0.64782609 0.33333333

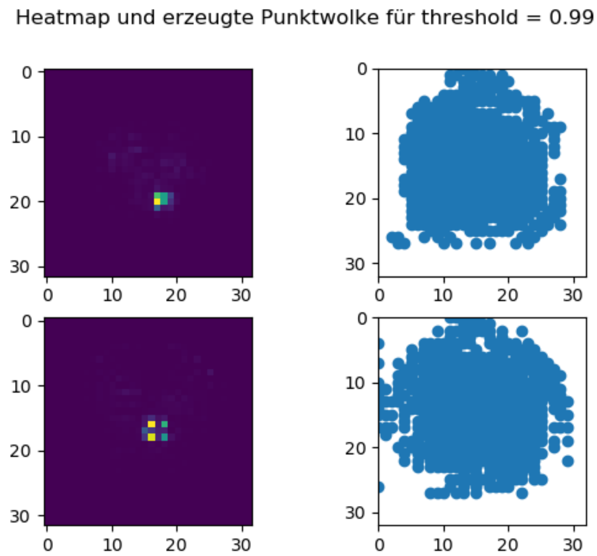

Abbildung 9: Auswahl der relevantesten Pixel (bis zu 99% der Gesamtmasse) zweier Heatmaps

1. 0.88333333 1.

mAER: 83.15190128571427 min: 27.500000000000004 max2: 100.0

(tn, fp, fn, tp): 1106 0 59 485 ACC: 96.42 TPR: 89.15 TNR: 100.0

(tn, fp, fn, tp): 1106,0,18,526 acc: 98.91 tpr: 96.69 tnr: 100.0 ——————————————————- 10 Prozent:

[1. 0.85138889 0.96533333 1. 1. nan 1. 0.99777778 1. 0.96875 0.98636364 0.87380952 0.6826087 0.925 0.92962963 0.88571429 1. 0.69166667 0.94102564 1. 0.97777778 0.86666667 0.94166667 1. 1. 0.85833333 0.97222222 1. 0.92 1. 0.88 1. 0.93333333 0.68571429 0.425 0.82820513 0.73333333 1. 0.87391304 0.44444444 1. 0.95 1.]

mAER: 90.45161504761903 min: 42.5 max2: 100.0

(tn, fp, fn, tp): 1485 0 17 148 ACC: 98.97 TPR: 89.7 TNR: 100.0

HC: (tn, fp, fn, tp): 1485,0,3,162 acc: 99.82 tpr: 98.18 tnr: 100.0

——————————————————- 5 Prozent: [0.93333333 0.85 0.89466667 1. 0.95757576 nan 1. 0.99333333 1. 0.93541667 0.97575758 0.67380952 0.55072464 0.88472222 0.85555556 0.67142857 0.86666667 0.52222222 0.88974359 1. 0.93333333 0.86666667 0.825 0.89333333 1. 0.80416667 0.87777778 1. 0.94666667 0.9 0.9 1. 0.9 0.69047619 0.28333333 0.75384615 0.725 1. 0.81014493 0.33333333 1. 0.75 0.95555556]

mAER: 84.77045302380954 min: 28.333333 max: 100.0

(tn, fp, fn, tp): 1566 2 13 69 ACC: 99.09 TPR: 84.15 TNR: 99.87

(tn, fp, fn, tp): 1568,0,3,79 acc: 99.82 tpr: 96.34 tnr: 100.0 ——————————————————-

2 Prozent:

0.96666667 0.87222222 0.83866667 0.99777778 0.99090909 nan 1. 1. 1. 0.875 0.70151515 0.51190476 0.58550725 0.77777778 0.95555556 0.90952381 0.96666667

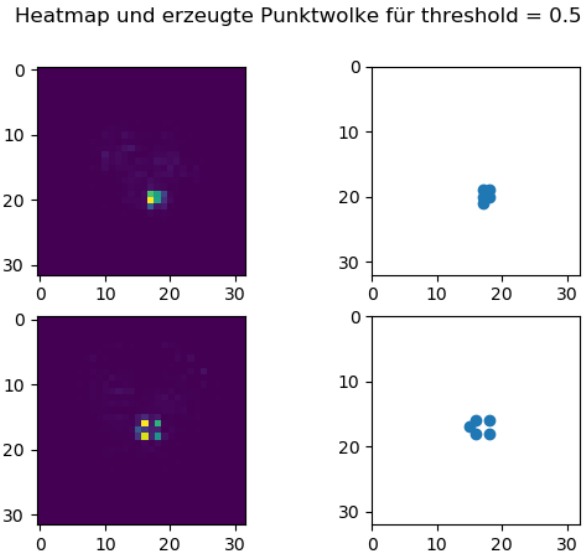

Abbildung 10: Auswahl der relevantesten Pixel (bis zu 50% der Gesamtmasse) zweier Heatmaps

0.78055556 0.90512821 0.91666667 0.8 0.68888889 0.81666667 0.76 1. 0.5875 0.94444444 1. 0.94666667 0.97777778 0.81333333 1. 0.9 0.59047619 0.4 0.85128205 0.78333333 0.98333333 0.77681159 0.33333333 1. 0.7 0.86666667

mAER5: 84.77045302380954 min: 28.333333 max 100.0
AC:(tn, fp, fn, tp): 896 721 0 33 ACC: 56.3 TPR: 100.0 TNR: 55.41
HC: (tn, fp, fn, tp): 880,737,0,33 acc: 55.33 tpr: 100.0 tnr: 54.42

9.6 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

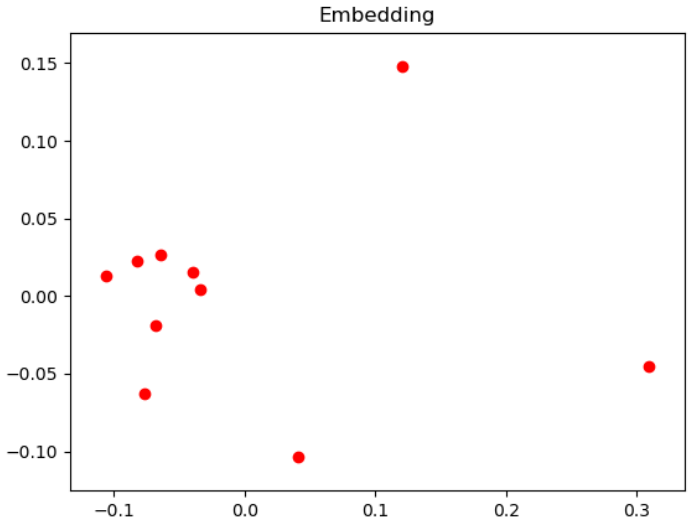

Abbildung 11: Einbettung des Baryzentrums mithilfe von Multidimensionaler Skalierung(MDS) bei der Wahl von 99% der Gesamtmasse

10 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[24] 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[25]
- Vergleich mit Cifar-10/Cifar-100 Datensatz[26][27]

11 Zusammenfassung und Ausblick

Beobachtung: Je größer die Netzwerke sind, desto leichter lassen sich Poisoning-Angriffe realisieren. Angriffe mit AER <100 Prozent sehr schwierig zu erkennen.

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

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

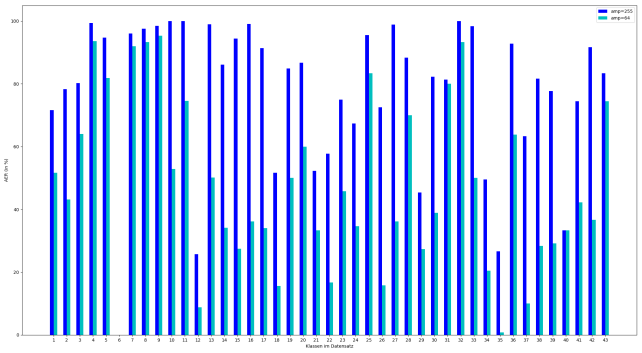

Abbildung 12: Angriffserfolgsrate pro Klasse bei Clean-Label-Poisoning-Attacks für verschiedene Werte *amp* des Amplitudenstickers in vierfacher Ausführung bei Abstand $d = 10$ zum Rand

A Verwendete Netzwerke

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

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

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

B Parameter für Training und Einlesen der Daten

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

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

]


```

```
17
18
```

Listing 4: Augemntierung beim Einlesen der Daten

Die Werte von mean und std variieren für alle ausgeführten Poisoning-Angriffe. Anstatt beide jedes Mal erneut zu berechnen, verwenden wir die von pytorch angegebenen Werte [28], die für die vor-trainierten Modelle empfohlen werden und auf dem Datensatz ImageNet[29] 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 [30], die wiederum auf PyTorch Ignite[31] 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:

E Datensätze

GTSRB[32] 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 [33] Barycenters in the Wasserstein Space[34]
Was passiert bei der Kombination von 2 verschiedenen Triggern? Einmal keine

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

[29] `https://image-net.org/`

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

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

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

[33] `https://arxiv.org/pdf/1809.06422.pdf`

[34] `https://arxiv.org/pdf/1809.06422.pdf`

Überlappung(d.h. 2verschiedene Trigger auf dem selben Bild) vs. auch beide Trigger auf einem Bild ist zulässig.

Literatur

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

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[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.

[BBB+01] Dmitri Burago, Iu D Burago, Yuri Burago, Sergei Ivanov, Sergei V Ivanov, and Sergei A Ivanov. *A course in metric geometry*, volume 33. American Mathematical Soc., 2001.

[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.

[CFTR16] Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1853–1865, 2016.

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

[COT19] 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.

[DGK18] Pavel Dvurechensky, Alexander Gasnikov, and Alexey Kroshnin. Computational optimal transport: Complexity by accelerated gradient descent is better than by sinkhorn's algorithm. In *International conference on machine learning*, pages 1367–1376. PMLR, 2018.

[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.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[Gro07] Mikhail Gromov. *Metric structures for Riemannian and non-Riemannian spaces*. Springer Science & Business Media, 2007.

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

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

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

[Mau] Abhinav Maurya. Optimal transport in statistical machine learning: Selected review and some open questions.

[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.

[Mém11] Facundo Mémoli. Gromov–wasserstein distances and the metric approach to object matching. *Foundations of computational mathematics*, 11(4):417–487, 2011.

[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.

[MMG19] Anton Mallasto, Guido Montúfar, and Augusto Gerolin. How well do wgans estimate the wasserstein metric? *arXiv preprint arXiv:1910.03875*, 2019.

[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.

[San10] Filippo Santambrogio. Introduction to optimal transport theory. *arXiv preprint arXiv:1009.3856*, 2010.

[Sin64] Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35(2):876–879, 1964.

[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.

[VCF+20] Titouan Vayer, Laetitia Chapel, Rémi Flamary, Romain Tavenard, and Nicolas Courty. Fused gromov-wasserstein distance for structured objects. *Algorithms*, 13(9):212, 2020.

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