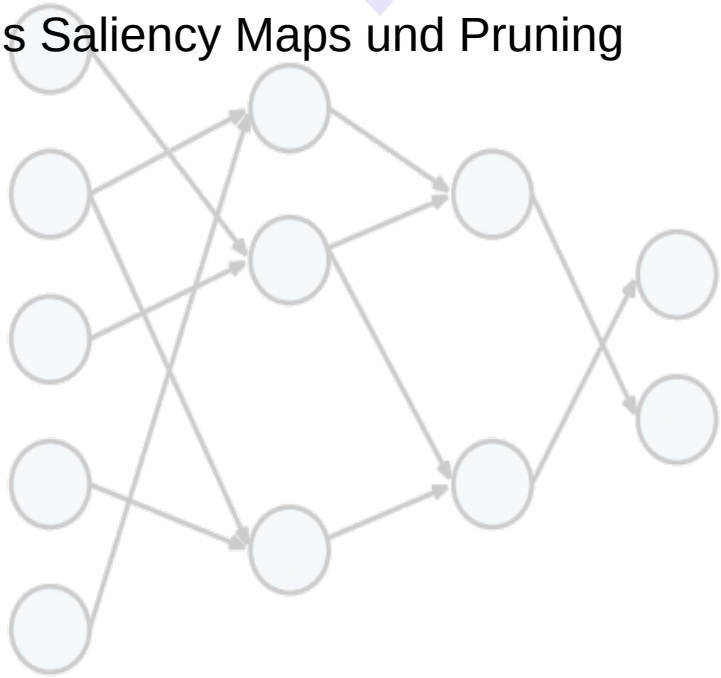
Hardware-Beschleunigung von Edge-AI

Erklärbarkeit und Optimierung von Netzarchitekturen mittels Saliency Maps und Pruning



Hardware-Beschleunigung von Edge-AI

Erklärbarkeit und Optimierung von Netzarchitekturen mittels Saliency Maps und Pruning

Verfasser: Béla H. Böhnke

Matrikelnummer: 64048

Betreuer Betrieb: Norbert Link

Betreuer Hochschule: Astrid Laubenheimer

Arbeit: Master Projektarbeit 2

Zusammenfassung

In dieser Arbeit wurden Saliency Maps, eine Methode aus dem Bereich der Erklärbarkeit neuronaler Netze, genutzt, um eine Optimierung einer vorhandenen Netzarchitektur in Bezug auf Inferenzgeschwindigkeit und Speicherverbrauch durch die Anwendung von Pruning durchzuführen.
Durch das Nutzen von Gradient Saliency Maps kann bestimmt werden wie wichtig einzelne Neuronen und Gewichte für die Netzentscheidung sind. Ohne größere Verluste in der Accuracy, kann so die Anzahl an Parametern durch weglassen unwichtiger Parameter stark reduziert werden.

Abstract

In this work saliency maps, a method from the filed of network explainability, was used to optimize a given network architecture in respect of inference speed and memory usage thru the usage of pruning.
With the help of gradient saliency maps can be determined how important single neurons and weights are for the network decision. Without major loss in accuracy, the parameter count can be greatly reduced by removing unimportant parameters.

Inhaltsverzeichnis

Abbildungsverzeichnis

Formelverzeichnis

1 Einleitung

Im Teil *2 Grundlagen* werden die Begriffe Pruning und Saliency erläutert sowie Beispiele für mögliche Einsatzgebiete gegeben. Es folgt dann eine ausführliche Erläuterung der Literatur in *3 Stand der Forschung* diese beiden Gebiete betreffend.

2 Grundlagen

2.1 Erklärbarkeit

Ein grundlegendes Problem der neuronalen Netze ist die Erklärbarkeit. Ein neuronales Netz bildet die Eingabedaten (X) auf eine Ausgabe (Y) ab. Das Netz ist hierbei äquivalent zu eine nichtlinearen mathematische Funktion mit vielen freien Parametern, welche während des Trainings des Netzes mit bekannten X und Y im Rahmen eines Optimierungsproblems optimiert werden. Sind die freien Parameter, die mit Anlehnung an natürliche neuronale Netze als Gewichte / Weights (W) und Schwellwerte / Biases (B) bezeichnet werden, gefunden kann das neuronale Netz im Idealfall für noch unbekannte X die passenden Y berechnen. Durch die hohe Anzahl an Parametern der mathematischen Funktion, ist es für einen Menschen allerdings nur schwer nachvollziehbar, welche Muster in den Eingabedaten X zur Entscheidung des Netzes und zur Ausgabe der entsprechenden Y geführt haben.

Dies ist ein großes Problem, denn nun hat das neuronale Netz zwar einen komplexen Sachverhalt erlernt, beinhaltet in seinen W,B also Wissen. Kann dieses Wissen aber nicht so präsentieren, dass es auch zur Wissenssteigerung des menschlichen Wissens dienen kann, oder zumindest zur groben Erklärung der Entscheidungsfindung.

Ohne eine Erklärung der Entscheidungsfindung ist ein neuronales Netz allerdings in vielen Bereichen nicht einsetzbar, insbesondere in den Bereichen, in denen das Wohl von Menschen von der Entscheidung abhängt. Zudem können durch die Erklärung von Entscheidungen frühzeitig Fehler in der Entscheidungsfindung festgestellt werden.

Die Erklärbarkeit von neuronalen Netzen beschäftigt sich daher mit der Entwicklung von Algorithmen und Netzarchitekturen, die für gegebene Ausgabedaten Y beschreiben, welche Daten aus X die Entscheidung herbeigeführt haben und wenn möglich warum.

In dieser Arbeit geht es insbesondere darum festzustellen welche Bereiche im Input jeder Schicht besonders wichtig für die Ausgabe des Netzes sind, hierfür können Saliency Maps bzw. Importance Maps genutzt werden.

2.2 Pruning

Neuronale Netze haben häufig eine hohe Anzahl an Parameter, die Technik entwickelt sich dahingehend immer tiefere neuronale Netze zu verwenden womit die Parameterzahl ständig zunimmt. Mit der Parameterzahl nimmt aber auch der Speicherverbrauch und die Rechenzeit zu. Zudem besteht die Gefahr zu viele Parameter zu verwenden wodurch das Netz den Trainingsdatensatz auswendig lernt und also nicht mehr verallgemeinern kann. Neue Bilder können dann vom neuronalen Netz nicht mehr erkannt werden.

Viele der Gewichte die in einem solchen Netz vorkommen sind entweder redundant oder zu spezifisch auf eine Teilaufgabe spezialisiert. Sie sind zum lösen der Aufgabe also unwichtig.

Pruning beschäftigt sich nun damit die Rechenzeit und den Speicherverbrauch zu reduzieren indem solche unwichtigen Gewichte entfernt werden. Als positiver Nebeneffekt ist das Netz mit weniger Parametern nun auch gezwungen zu verallgemeinern.

Hierfür muss allerdings bestimmt werden, welche Gewichte als wichtig gelten und welche als unwichtig. Hierfür können verschiedene Maße zurate gezogen werden, welche auch bei der Erklärbarkeit von neuronalen Netzen zum Einsatz kommen, denn es muss nun erklärt werden welche Gewichte wichtig und welche unwichtig für die Entscheidungsfindung des Netzes sind.

2.3 Saliency und Importance

Saliency laut Duden:

das Hervortreten, Herausstechen eines Reizes; Auffälligkeit eines Ereignisses, einer Sache oder Person.

In Bezug auf neuronale Netze sind „Saliency Maps“ Karten, die anzeigen welche Bereiche eines Bildes ein neuronales Netz besonders reizen bzw. welche Bereiche das Netz als herausstechend für die Entscheidungsfindung sieht.

Im Idealfall sollten sich diese wichtigen Bereiche im Bild mit den zu erkennenden Objekten decken.

Es gibt eine Vielzahl anderer Verfahren um Saliency Maps zu berechnen, welche in *3 Stand der Forschung* betrachtet werden.

Im einfachsten Fall kommt die Oracle Saliency zum Einsatz, hier wird zum Erstellen einer Saliency Map nacheinander jeder einzelne Input auf 0 gesetzt (weg gelassen) und beobachtet wie sich der Loss verändert. Inputs die beim entfernen eine große Änderung am Loss bewirken sind offensichtlich wichtiger für die Erfüllung der Aufgabe. Der Nachteil an dieser Methode ist der hohe Rechenaufwand, es muss eine Vorwärts-Berechnung des kompletten Netzes für jeden Input durchgeführt werden.

Saliency Maps (SM) besitzen sowohl positive als auch negative Werte, je nach dem in welche Richtung sie den Loss beeinflussen. Für die Bestimmung wie wichtig ein einzelner Input ist, ist allerdings die Richtung egal nur die Größe der Änderung ist hier relevant. Als Importance Map (IM) kann also einfach der Absolutwert der Saliency Map (SM) gebildet werden.

Formel I: Importance Map aus Saliency Map

$$\mathrm{IM} = |SM|$$

2.4 Verwendung von Saliency und Importance Maps

Mit der Erkenntnis, welche Bereiche der Eingabedaten für das Netz besonders relevant für die Ausgabedaten sind, können verschiedene Verwendungsszenarien abgeleitet werden.

2.4.1 Prüfung der Entscheidungsfindung

Zum Einen kann man sich ein Bild vom korrekten Arbeiten des Netzes machen. Sollte das Netz z.B. besonders viel Wert auf Bereiche in den Eingabedaten legen, die offensichtlich keine Relevanz für das Ergebnis haben sollten, kann daraus geschlossen werden, dass der Trainingsdatensatz ungewollte Korrelationen enthält. Ein bekanntes Beispiel hierfür ist ein neuronales Netz das Hunde und Wölfe auseinanderhalten soll, stattdessen unterscheidet es Bilder mit und ohne Schnee im Hintergrund, denn der Trainingsdatensatz zeigte vor allem Bilder von Wölfen im

Schnee und von Hunden ohne Schnee. Leider waren Wölfe ohne Schnee für das Netz nun Hunde.

2.4.2 Stärkere Überwachung des Trainings für Klassifikation

Eine weitere Anwendung ist bereits während der Trainingsphase die Daten zu nutzen, um die Aufmerksamkeit des Netzes direkt auf wichtige Bildbereiche zu fokussieren. Die Trainingsdaten müssen dazu semantisch annotiert sein. Die Bereiche auf die das Netz Wert legen sollte, gegenübergestellt zu den Bereichen auf die es Wert legt, können nun als Teil der Fehlerfunktion genutzt werden.

2.4.3 Schwach überwachtes Training für Segmentierung

Sehr nahe damit verwandt ist der Einsatz für die Segmentierung. So kann ein neuronales Netz zur Klassifizierung dazu genutzt werden eine Segmentierung durchzuführen, indem die Bereiche hoher Aufmerksamkeit bezogen auf eine Klasse als Segmentierung der entsprechenden Klasse verwendet werden. Es werden hier nur Annotationen der Trainingsdaten für die Klassifizierung benötigt und trotzdem kann damit ein Model zur Segmentierung trainiert werden.

2.4.4 Netz Surgery, Spezialisierung auf Teilprobleme

Ein weiterer Anwendungsfall ist der Aufbau bzw. die Optimierung der Architektur eines neuronales Netz für spezielle Aufgaben. So kann z.B für ein bereits vorhandenes Netz zur Klassifikation für alle Schichten berechnet werden, welche Neuronen besonders wichtig für die Klassifikation einer bestimmten Klasse sind, indem die Aufmerksamkeit jedes Neuron gemittelt über alle Daten der bestimmten Klasse des Trainingsdatensatzes bestimmt wird. Mit dieser Information (welche Neuronen sind wichtig zum Erkennen der Klasse) kann das Netz nun zerlegt oder verkleinert werden, sodass nur noch die Neuronen übrig bleiben die tatsächlich wichtig sind. Ein Netz das auf viele Klassen trainiert wurde, kann also zu einem kleineren Netz konvertiert werden, welches nur einen Teil der Klassen erkennen muss.

2.4.5 Pruning

Ganz analog dazu ist die Problematik des Pruning. Ein großes Netz soll verkleinert werden, um trotz begrenzter Hardware Ressourcen noch einen angemessenen

Durchsatz zu erreichen. Hierbei wird ein Verlust in der Genauigkeit hingenommen. Die Fragestellung ist nun allerdings, welche Gewichte und Neuronen vernachlässigt werden können, ohne wichtiges Wissen zu verlieren. Auch hier ist die Auswirkung jeder Schicht auf die Ausgabeschicht eine guter Richtwert dafür, welche Neuronen wichtig sind und was vernachlässigt werden kann. In diesem Fall sollte die Aufmerksamkeit über alle relevanten Daten aus dem Trainingsdatensatz gemittelt werden.

3 Stand der Forschung

Teil der Arbeit war eine ausführliche Literatur Recherche mit den Schwerpunkten der Bestimmung der Saliency bzw. der Importance in Hinsicht auf Erklärbarkeit, die Bestimmung der Importance in Hinsicht auf Pruning, sowie Verfahren zum durchführen von Pruning oder ähnlichen Techniken anhand eines der Importance Maßen. Im folgenden sind jeweils kurze Zusammenfassungen und Erklärungen der einzelnen Quellen gegeben.

3.1 Saliency für Erklärbarkeit

Die Literatur in diesem Bereich beschäftigt sich insbesondere damit die Saliency des Inputs in Bezug auf die Klassifikation-Entscheidung zu bestimmen. Die selben Methoden können allerdings auch auf andere Bereiche übertragen werden die Saliency Information benötigen, was in dieser Arbeit gemacht wird.

3.1.1 Saliency als Anteil an der Neuronen Aktivierung

In [1] wird zunächst ausführlich das Prinzip eines Multilayer Perzeptrons beschrieben. Hierbei wird gezeigt, dass bei einer Sigmoid Aktivierung ein Input mit dem Wert 0 keinen Beitrag zur Klassifizierung leisten kann.

Es wird dann die Orakle Saliency eingeführt, welche zur Bestimmung der wichtigsten Inputs einzelne Inputs entfernt und beobachten wie sich der Output verändert. Es wird darauf hingewiesen das theoretisch auch alle Kombinationen von Inputs getestet werden müssten, da der Output vom Auftreten mehrere Inputs zusammen abhängen kann. Dies ist folglich mit sehr viel Rechenaufwand verbunden.

Es wird dann eine effizientere Möglichkeit vorgestellt um die Saliency einzelner Inputs für einen Output zu bestimmen. Hierfür werden die einzelnen Terme der Summe eines Neurons, also Aktivierung*Gewicht, betrachtet und so der Anteil dieser an der Neuron Summe und somit der Aktivierung des Neurons bestimmt, siehe Formel II.

Formel II: Saliency als Anteil an der Neuronen Aktivierung

$$S = \frac{1}{X * W} \cdot X \circ W$$

Dieser Anteil kann Schichtweise bis zur Eingabeschicht zurück vollzogen werden, womit dann der Anteil am Output bekannt ist.

Die Orakle Saliency wird in [1] als Referenzmaß genutzt um die Vorhersage zu prüfen.

3.1.2 Saliency als Änderung des Outputs

Die Saliency als Änderung des Outputs beim weglassen eines bestimmten Inputs wurde bereits in [1] eingeführt, in [2] wird diese Idee erweitert indem einfach die Änderung des Outputs bei einer Änderung des Inputs betrachtet wird. Also kurz gesagt die Gradienten in Bezug auf den Inputs gebildet werden. Dieses Vorgehen ist sehr effizient da die Gradientenberechnung für die Backprobergation beim Training eines Netzes bereits benötigt werden.

Formel III: Saliency als Änderung des Outputs bei Änderung des Inputs

$$S = \frac{\partial J}{\partial X}$$

Als Output wird hier nicht der normale Trainings-Loss genutzt sondern ein spezieller „Class Score“ der Maximiert werden soll. Dieser „Class Score“ entspricht dem Output des Netzes der zu betrachtenden Klasse, vor einem Softmax (da bei Softmax der Score auch maximiert werden könnte indem die anderen minimiert werden).

Der Absolutwert des Gradient des „Class Score“ in Bezug auf den Input wird dann direkt als Saliency verwendet. Der „Class Score“ wird im Folgenden in dieser Arbeit als Pruning-Loss bezeichnet, die Absolutwerte der Gradienten als Importance wohingegen die Saliency direkt durch die Gradienten gegeben ist.

Die Saliency Map wird dann in Kombination mit Graph Cut dazu verwendet eine Objekt Segmentierung durchzuführen.

In [3] und [4] wird eine Anpassung des Verfahrens aus [2] vorgenommen. Anstelle die Gradient Saliency Map nur für den Input zu berechnen wird sie in einem CNN für die ersten N Layer-Inputs also die ersten N Feature Maps des CNN berechnet. Die so entstandenen Maps werden dann auf die selbe Größe gebracht (Bilineares Upsampling). Als Saliency Map jedes Layers wird wiederum das Maximum des Absolutwertes aus den Kanälen des Layers gewählt um eine 1 einkanalige Map zu erhalten. Diese einkanaligen Maps jeder Layer werden dann zu einer Saliency Map kombiniert, indem sie über den Tanh der einzelnen Maps gemittelt werden, siehe Formel IV.

Formel IV: Kombination von Saliency Maps

$$S = \frac{1}{|L|} \sum_{l \in L} \tanh(\alpha \cdot S_l)$$

Es wurde hier außerdem festgestellt das eine Klassenspezifische Map oft auch noch Bereiche anderer Klassen abdeckt. Aus diesem Grund werden die Maps aller Klassen berechnet und von der relevanten Klasse c die Maps der anderen Klassen x abgezogen, wie in Formel V.

Formel V: Saliency Klassen Differenz

$$S = \sum_{x \in C \setminus c} max(S_c - S_x, 0)$$

Für Relu Aktivierungen wird im Paper die Guided Backprobergation eingesetzt, bei der nur positive Gradienten zurück propagiert werden, dies scheint bessere Maps zu ergeben.

Die Maps können dann unter der Verwendung von Fully Connected CRF (Conditional Random Field) nachbearbeitet werden um die Objektausdehnung besser zu bestimmen.

In [5] werden die Saliency Maps in Bezug auf Objektausdehnung weiter verbessert, indem zusätzlich Bounding-Box Annotierungen verwendet werden. Insofern ist diese Veröffentlichung für diese Arbeit nicht weiter relevant, da in dieser Arbeit keine zusätzlichen Annotationen verwendet werden sollen.

Auch [6] baut auf Gradient Saliency auf. Allerdings wird hier nicht direkt der Gradient verwendet, sondern der Input des Netzes in mehreren Schritten mittels Backprobergation so optimiert, dass die Klasse danach nicht mehr erkannt wird. Hierzu wird eine spezielle Kostenfunktion verwendet auf welche Objektpixel einen stärkeren Einfluss haben als Hintergrundpixel. Die Differenz von Originalbild zum veränderten Bild entspricht dann der Saliency. Durch die höhere Anzahl an Itterationen wird die Map hier exakter als bei nur einer einfachen Gradientenberechnung wie in [2].

Kostenfunktion: Alle Outputs die nicht zur erkannten/untersuchten Klasse gehören werden konstant gehalten, also als Zielwert der Loss-Funktion der orginal Output bei unverändertem Bild gewählt. Der Output der zur zu untersuchenden Klasse gehört soll möglichst gering (Zielwert = 0) werden. Die Kosten werden dann einfach als L2 / SSE berechnet.

Es wird dabei angenommen, dass alle Outputs die nicht zur Klasse gehören auf den Hintergrund zurückzuführen sind, und nur der Output der entsprechenden Klasse auf das Objekt im Bild. Das Bild wird dann also so verändert, dass die für die Klasse relevanten Pixel entfernt werden. Von den Bildwerten werden dabei wie beim Gradientenabstieg üblich die Gradienten abgezogen. Negative Gradienten werden dabei auf 0 geklippt, damit werden die Werte wichtige Pixel immer nur kleiner und nicht größer. Das Differenzbild hat damit nur positive Werte.

Zuletzt wird eine Objektsegmentierung durchgeführt, bei der die Saliency Maps mittels SLIC (Superpixel Bildung) und Low-Lever-Featuren (Farbe, Kontrast, etc.) verbessert werden.

3.2 Importance für Pruning

Als Maß wie wichtig ein Neuron innerhalb eines Netzes ist kann betrachtet werden, wie häufig es durch Eingabedaten aktiv wird bzw. wie häufig es inaktiv bleibt. In [7] wird nach diesem Prinzip die Average Percentage of Zeros (APoZ) der Aktivierungen jeder Schicht mithilfe eines Trainings Sample bestimmt. Es wird hier bestimmt wie oft die Aktivierung eines Neurons im Durchscnitt 0 ist. Neuronen mit hoher APoZ werden hierbei als unwichtig gesehen und können entfernt werden.

Oft wird beim Training von Netzen ein Regularisierungsterm verwendet der dafür sorgt dass die Gewichte klein bleiben. Es wird in [8] davon ausgegangen, das sehr kleine Gewichtswerte unwichtiger sind und entfernt werden können.

In [9] wird eine Übersicht verschiedener Maße zur Bestimmung der Wichtigkeit einzelner Neuronen gegeben. Zunächst wird als Referenzmaß die auch in [1] vorgestellte Oracle Saliency erklärt.

Oracle Saliency: Jedes einzelne Neuron wird dabei nacheinander entfernt und die Kostendifferenz bestimmt. Neuronen die beim Entfernen eine geringe Kostendifferenz hervorrufen können bleibend entfernt werden.

Dieses Verfahren hat einen sehr hohn Rechenaufwand und kann in der Praxis deshalb nicht eingesetzt werden, es kann aber als ideales Referenzmaß verwendet werden.

Minimum Weight: Ist angelehnt an [8] aber auf Neuronen-Ebene anstelle von einzelnen Gewichten. Es wird hier die Höhe des durchschnittlichen Gewichts für jedes Neuron bestimmt und die Neuronen mit dem geringsten durchschnittlichem Gewicht entfernt.

Diese Methode ist sehr einfach und hat einen sehr geringer Rechenaufwand. Sie kann außerdem mit der L1 oder L2 Regularisierung unterstützt werden. Liefert aber auch weniger gute Ergebnisse.

Activation: Neuronen die selten Aktiviert werden sind nicht so wichtig und können entfernt werden. Als Maß kann die Mittlere Aktivierung oder die Standardabweichung der Aktivierung jedes Neurons bestimmt werden. Hier kann man auch APoZ aus [7] einordnen.

Ein Problem hierbei ist dass in frühen Schichten sehr ähnliche Werte für alle Neuronen geliefert werden.

Mutual information: Mutual Information (MI) oder für vereinfachte Berechnung Information Gain (IG) beschreibt die Abhängigkeit einer Variablen von einer anderen. Es wird die Abhängigkeit des Outputs des Netzes von der Aktivierung jedes Neurons für ein Trainings Sample bestimmt. Also wie oft eine hohe Aktivierung zusammen mit einem hohen Output auftritt. Neuronen mit geringer Abhängigkeit werden entfernt.

Taylor Expansion / Impact: Ist in [9] neu eingeführt hierbei wird Pruning als Optimierungsproblem betrachtet. Ziel ist eine bestimmte (und geringere) Anzahl an Parametern zu finden welche die absolute Differenz des Loss möglichst gering halten. Wobei der neue Loss nach entfernen eines Parameters (eines Neurons) durch ein Taylor Polynom ersten Grades geschätzt wird. Es werden also Parameter bzw. Neuronen entfernt die einen geringen Gradient in Bezug auf den Loss in Kombination mit einer geringe Aktivierung haben. Für ein Trainings Sample wird dieser Wert für jedes Neuron gemittelt. In [10] wurde parallel zu [9] die selbe Technik verwendet und als Impact bezeichnet um in CNNs die Convolutons nur an bestimmten Stellen der Feature Maps zu berechnen.

Formel VI: Saliency mittels Impact des entfernen eines Inputs

$$S = \frac{\partial J}{\partial X} \circ X$$

3.3 Pruning Vorgehen

Dropout [11] und Dropconnect [12] zählen zwar nicht zum pruning doch auch hier werden Neuronen oder einzelne Gewichte entfernt. Dies geschieht allerdings nur mit einer gewissen Wahrscheinlichkeit während des Trainings um das Netz robuster zu machen und overfitting zu vermeiden. Während der Inferenz werden dagegen wieder alle Gewichte und Neuronen genutzt, weshalb es dort keinen Geschwindigkeitsvorteil bringt.

In [8] wird ein einfaches iteratives Vorgehen verwendet bei dem das Netz nach entfernen einzelner Gewichte neu trainiert wird (fine tuning), dieser Ablauf wird wiederholt bis die Anzahl der noch verbleibenden Gewichte einen bestimmten Grenzwert unterschritten haben oder die Accuracy unter eine Grenze gefallen ist.

Pruning kann auch ganz ohne Importance Maß durchgeführt werden, wie in [13], hier wird eine Optimierung direkt auf der Adjazenzmatrix der Gewichte durchgeführt. Jedes Layer wird hierbei getrennt optimiert. Ziel dabei ist es den Output des Layers für die Trainingsdaten gleich zu belassen aber mit weniger Gewichten zu erreichen. Als Loss pro Layer wird hier der L1 Loss (Summe der Feher Beträge) verwendet. Der Vorteil an dieser Lösung ist, dass das Problem des gesamte Netz zu optimieren in kleine einfache Teilprobleme zerlegt wird.

Auch in [14] wird ähnlich vorgegangen, allerdings werden hier nicht einzelne Gewichte sondern ganze Filter (Neuronen) und somit ganze Kanäle in einer Feature Map entfernt. Es wirdvVersuche mit den verbleibenden Kanälen in der Folge Schicht einen ähnlichen Output zu erhalten, wobei der L2 Loss (Least Square Error) eingesetzt wird.

In [7] wird wiederum ein iteratives Vorgehen mit fine tuning eingesetzt, bei dem APoZ als Maß der Importance einzelner Neuronen eingesetzt wird.

In [15] wird Mutual information als Maß eingesetzt. Die Korrelation zwischen Neuronen wird hier allerdings nicht mit Testdaten bestimmt sondern mit zufälligem Rauschen. Neuronen die stark korreliert sind werden durch ein einzelnes Neuron ersetzt. Das Netz wird iterativ weiter trainiert bis die Accuracy unter einen festgelegten Grenzwert fällt. Zusätzlich ist es möglich die Loss Funktion anzupassen um Korrelation zwischen Neuronen zu fördern.

Das entfernen von Neuronen und damit Feature Maps wird in [9] durch eine Maske gelöst, welche mit dem Output eines Neurons multipliziert wird. Die Maske wird als Pruning Gate bezeichnet und kann die Werte 0 (Neuron entfernt) oder 1 (Neuron vorhande) annehmen.

Anstelle ganze Feature Maps zu entfernen, werden in [10] nur einzelne Positionen in diesen entfernt (Sparce Feature Map). Die fehlenden Werte werden mit den Nächsten Nachbarn der berechneten Positionen interpoliert. Dazu wird an zufällig Positionen evaluiert und mit den bleibenden Positionen mittels Euklidischer Distanz verglichen. Die Position wird dann durch den Wert der bleibenden Position die mit der geringsten Distanz ersetzt. Die bleibenden Positionen werden mittels einer Perforation Mask bestimmt, diese Maske kann auf verschiedene Weisen bestimmt werden:

Uniform: zufällig normalverteilte ausgewählte Positionen werden als Maske verwendet.

Grid: Ein zufälliges aber gleichmäßiges Gitter das mittels einer Pseudozufalls Zahlenfolge erstellt wird.

Pooling Structure: Beim Pooling überlappen einzelne Pooling Filter, es werden die Positionen gewählt die in möglichst vielen Filtern vorkommen, also an denen sich viele Filter überlappen.

Impact: Siehe 3.2 Importance für Pruning. Schätzen der Auswirkung des Entfernens einer Position auf den Loss und entfernen der Positionen mit geringster Auswirkung. Die Schätzung des Impacts wird mittels Tailor Polynom ersten Grades durchgeführt. Also der Gradient für diese Position multipliziert mit dem Wert der Position. Der Gradient wird dabei für die Perforierte Convolutional Layer ohne Interpolation berechnet. Für die Maske wird der Mittelwert des Impacts über ein Trainings Sample und die Summe über alle Kanäle an einer Position verwendet. Behalte werden dann nur die N Positionen mit dem höchsten Impact.

Nach dem das Netz perforiert wurde wird Fine Tuning durchgeführt.

Eines der neusten Pruningverfahren [16] adaptiert das vorgehen aus Dropout [11] und Dropconnect [12] auf Pruning. Das Pruning wird hier bereits während des Trainings durchgeführt, die Gewichte werden nicht dauerhaft sondern mit einer Pruning Wahrscheinlichkeit entfernt. Diese wird anhand einer Rangfolge der

Gewichte eines Neuron bestimmt, welche wiederum die L1 Norm (absolute Größe) der Gewichte als Importance verwendet. Hier könnten aber auch andere Maße zum erstellen der Rangfolge Anwendungfinden.

Für alle Gewichte die in der Rangfolge unterhalb der Position liegen die durch ein Pruning Verhältnis vorgegeben ist wird die Wahrscheinlichkeit erhöht.

Pruning wird dann zufällig anhand der Wahrscheinlichkeiten durchgeführt, und das Netz weiter trainiert. In jeder Iteration wird die Pruning Maske neu anhand der Wahrscheinlichkeiten festgelegt.

4 Umsetzung

4.1 Vorüberlegungen

4.1.1 Aktivierungsfunktion

Als Aktivierungsfunktion wurden Sigmoid, Tanh und Relu getestet, es wurde schlieslich Tanh gewählt. Beim testen wurden Testbilder verarbeitet und die Accuracy sowie die Importance Maps bewertet. Trotz dem verbreiteten Einsatz von Relu fiel die Rangfolge der Accuracy unerwartet wie folgt aus:

Tanh am besten, Sigmoid in der Mitte, Relu am schlechtesten.

Es wurden auch Kombinationen aus verschiedenen Aktivierungen getestet.

Bei der Bewertung der Importance Maps wurden die Kriterien der zuvor festgelegten Kriterien siehe Abschnitt *4.1.3 Importance Map* zurate gezogen. Bei der Relu Aktivierung wurden hier die schlechtesten Maps erzeugt, was vermutlich auf die nicht Differenzierbarkeit im negativen Bereich zurückzuführen ist. Leaky Relu könnte dieses Problem lösen.

4.1.2 Datensatz

Der verwendete MNIST Datensatz liefert Bilder mit den Werten zwischen 0 und 255, zur besseren Verarbeitung werden diese Werte in der Regel auf einen Wertebereich zwischen 0 und 1 transformiert und ohne weitere Veränderung zum Training verwendet. Von diesem Vorgehen wurde in dieser Arbeit allerdings Abstand genommen und stattdessen ein mit Rauschen augmentierter und auf den Wertebereich von -1 bis 1 transformierter Datensatz verwendet. Im Folgenden sind die Gründe dazu aufgeführt.

Im nicht transformierten Bild sind nur die Zahlen selbst mit Werten von größer 0 dargestellt, der gesamte Hintergrund ist gleich 0. Es wurde als Aktivierungsfunktion Tanh gewählt welche Werte zwischen -1 und 1 liefert, wie in *4.1.1 Aktivierungsfunktion* dargestellt.

Zunächst liegt der Wertebereich des Input der ersten Schicht durch die Transformation im selben Wertebereich wie der Input aller Folgeschichten was eine einheitliche Visualisierung ermöglicht.

Ein Problem das durch nicht transformierte Bilder entsteht, ist die Berechnung der Gradienten für Positionen des Inputs mit 0 Werten, denn unabhängig von der Änderung eines Gewichts wird sich nach der Multiplikation mit dem Input immer 0 ergeben und sich so auch der Output nicht ändern [1]. Der Gradient wird für diese Stellen also 0 sein. Die Gewichte könnten also nur an den Stellen an denen eine Zahl ist angepasst werden. Durch die Transformation können Gewichte an jeder Stelle des Bildes angepasst werden.

Dieses Problem ist besonders Relevant da über die Gradienten auch die Saliency und so die Importance des Inputs bestimmt wird. Es können folglich schon allein durch die Daten nur hohe (positive und negative) Gradienten an den Stellen auftreten an denen im Input die Zahlen stehen. Da Ziel dieser Arbeit auch ist zu überprüfen ob sich die Gradienten als Vorhersage für wichtige Inputs und Gewichte eignen ist es ungünstig wenn bereits der Input dafür sorgt das große Gradienten nur an den Stellen der Zahl und somit immer nur an vermeintlich wichtigen Stellen berechnet werden können. Zur Überprüfung sollte die Berechnung der Gradienten an möglichst allen Stellen möglich sein und dennoch nur an wichtigen Stellen hohe Gradienten auftreten.

Aus diesem Grund wurde als weitere Maßnahme und um die Klassifikation zu erschweren eine Augmentierung des Datensatzes mit Gaus und Salz und Pfeffer Rauschen durchgeführt. Das Netz musste so auch erlernen zwischen Zahlen und zufälligen Aktivierungen zu unterscheiden. Und die Gradienten an Stellen mit Rauschen sollten voraussichtlich trotz hoher Aktivierungen gering bleiben.

Geht man von der (vereinfachten) Annahme aus das ein spezialisiertes Neuron der ersten Schicht jeweils Teile eine bestimmte Zahl erlernt indem es an den entsprechenden Stellen hohe (positive) Gewichte ausbildet und an allen anderen Stellen geringe (positive und negative) Gewichte hat das folgende Konsequenzen für nicht transformierte Daten.

Fall 1: Ein Neuron das gerade ein Segment einer Zahl gelernt hat die auch im Input anliegt hat an allen hohen Gewichten auch hohe Inputs anliegen, am Rest der

Gewichte 0. Es hätte eine positive Summe seiner Inputs. Es würde also feuern (+1 ausgeben), einen entsprechenden Bias größer oder ungefähr gleich 0 vorausgesetzt.

Fall 2: Ein Neuron das gerade ein Segment gelernt hat was nicht in der gezeigten Zahl vorkommt, hätte an all seinen Gewichten 0 anliegen und so wäre auch die Summe der Inputs 0. Die Gewichte an den Stellen mit hohem Input hätten durch ihre geringen Gewichte nur einen geringen negativen Einfluss. Es würde bei einer Tanh Aktivierungsfunktion also einen (hohen) negativen Bias benötigen um eine eindeutige -1 auszugeben.

Fall 3: Ein Neuron welches ein Segment erlernt hat welches nur Teilweise durch das Bild abgedeckt ist würde nur für den abgedeckten Teil ein Signal bekommen die restlichen Inputs würden 0 liefern. Es hätte in Summe also ein positives Signal, bei einem Bias größer gleich 0 würde es eine eindeutige 1 ausgeben, bei einem negativen Bias eine -1 oder etwas uneindeutiges. Wünschenswert wäre hier etwas uneindeutiges um die 0 also ein negativer Bias.

Die Summe aller Inputs tendiert hier immer zu einem positiven Wert, entsprechend wird ein ins negative verschobener Bias benötigt um diese Tendenz auszugleichen wie in Fall 3 klar wird. Fall 2 und 3 stehen allerdings im Widerspruch zu Fall 1.

Diese Auswirkung kann durch die Transformation einfach gelindert werden.

Fall 1: Ein Neuron das gerade ein Segment einer Zahl gelernt hat die auch im Input anliegt hat an allen hohen Gewichten auch hohe Inputs anliegen, am Rest der Gewichte -1. In Summe würde es einen hohen Positiven Input erhalten.

Fall 2: Ein Neuron das gerade ein Segment gelernt hat was nicht in der gezeigten Zahl vorkommt, hätte an all seinen hohen Gewichten -1 anliegen und so wäre auch die Summe der Inputs negativ.

Fall 3: Ein Neuron welches ein Segment erlernt hat welches nur Teilweise durch das Bild abgedeckt ist würde nur für den abgedeckten Teil ein positives Signal bekommen für den anderen Teil ein negatives Signal. In Summe also etwas uneindeutiges nahe 0, wie gewünscht.

Der Bias kann hier ebenfalls nahe 0 liegen.

4.1.3 Importance Map

Ziel ist es die Importance von Inputs und Gewichten des Netzes zu bestimmen. Also den Einfluss auf die Netzentscheidung. Inputs die die Netzentscheidung stark beeinflussen gelten als wichtiger als Inputs die kaum eine Auswirkung auf die Entscheidung haben.

Es wurden hier Recherchen angestellt und verschiedene mögliche Maße zusammen getragen, siehe *3.1 Saliency für Erklärbarkeit* und *3.2 Importance für Pruning*. Diese Maße mussten dann verglichen und eines ausgewählt werden. Der Vergleich wurde auf Basis der Veröffentlichungen, der Intuition hinter den Maßen sowie dem Rechenaufwand und Implementierungsaufwand getroffen. Zuletzt blieben zwei sehr ähnliche Maße, die Gradient Saliency Map und der Impact. Um diese zu vergleichen wurden beide Implementiert und die Importance Maps verglichen, siehe *5.1 Vergleich von Importance Maßen*. Dazu mussten allerdings zunächst Thesen aufgestellt werden welche Bereiche in einem Bild überhaupt wichtig sein könnten und diese Bereiche dann mit den durch das Maß bestimmten Bereiche Verglichen werden.

Folgende Bereiche wurden im Voraus als vermutlich wichtig identifiziert:

Bereich 1: Die Positionen an denen die im Input gezeigte Zahl selbst liegt.

Bereich 2: Die Positionen an denen Zahlen des Datensatzes liegen, die aber im momentanen Input nicht gezeigt sind.

Bereich 3: Positionen an den Kanten der Zahlen, da Kanten häufig zur Erkennung von Objekten verwendet werden.

Zum Pruning werden Importance Maps für ein Test Sample zusammengefasst, siehe hierzu *4.4 Pruning*. Der MNIST Datensatz besteht wie in *4.1.2 Datensatz* beschrieben aus zentrierten Zahlen, es ist also anzunehmen das die wichtigen Bereiche in der Mitte des Bildes liegen und etwa die Größe der Zahlen haben sollten. Ein weiteres Kriterium zum Vergleich stellt also die Zusammenfassung der Importance Maps dar.

Nach den Tests wurde sich dazu entschieden die Gradienten Saliency Maps zu verwenden, da diese ähnliche Regionen als wichtig erkennt aber deutlich weniger Rechenzeit benötigt.

4.1.4 Loss Funktion

a) Trainings Loss

Als Loss fürs Training wurde einfachheitshalber der MSE (Mean Square Error) verwendet. Dieser konnte direkt auf den Netzoutput angewendet werden unabhängig von der Aktivierungsfunktion der Letzten Schicht. Die GT (Grount Trouth) musste dafür nur minimal angepasst werden. Diese Eigenschaft mit beliebigen Aktivierungsfunktionen zu Arbeiten war wichtig da verschiedene Funktionen getestet werden sollten. Aus diesem Grund wurde auch auf die gängige Cross Entropie verzichtet, welche eine Summe aller Outputs von 1 vorraussetzt.

Für die Berechnung der Importance wurden zwei verschiedene Loss Funktionen getestet. Einerseits der Trainings Loss MSE, andererseits ein spezieller Pruning Loss der in [2] beschrieben ist.

b) Pruning Loss

Zum entscheiden welche Bereiche wichtig sind ist die Auswirkung dieser Bereiche auf den Netzoutput von Bedeutung. Das Verhalten des direkten Netzoutputs kann sich vom Verhalten des Loss stark unterscheiden, so kann z.b. nach einem Softmax ein Output auch maximiert werden indem alle anderen Outputs minimiert werden [2]. Der Pruning Loss sollte also direkt am rohen Netzoutput berechnet werden.

Man kann auch nur die Auswirkungen auf spezielle Teile des Outputs betrachten. Dies Ermöglicht zum Beispiel Netze die für viele Klassen trainiert wurden so zu Prunen, dass sie nur noch eine Untermenge der Klassen erkennen, dafür aber auch deutlich kleiner sind.

Es wird hierzu der Netzoutput maskiert, so hat nur noch die gewünschte, zur Klasse gehörende, Position im Output eine Auswirkung. Es wird dann direkt die vom Input abhängige Änderung an der entsprechenden Positionen betrachtet. Zum Fine Tuning nach dem Pruning müsste hier dann eine entsprechend angepasste Loss Funktion angesetzt werden, welche nur in den relevanten Bereichen der GT den Loss berechnet. Dies wurde in dieser Arbeit allerdings nicht durchgeführt. Stattdessen wurden alle Positionen führ den Pruning Loss verwendet, so konnte der unangepasste Trainings Loss weiter verwendet werden und dennoch die Funktion des Pruning Loss überprüft werden.

4.2 Aufbau Testprogramm

Im Laufe der Arbeit wurde ein Testprogramm entwickelt mit dem die Thesen überprüft werden konnten. Der interne Ablauf des Programms ist in *Abbildung 1: Ablauf des Testprogramms* dargestellt.

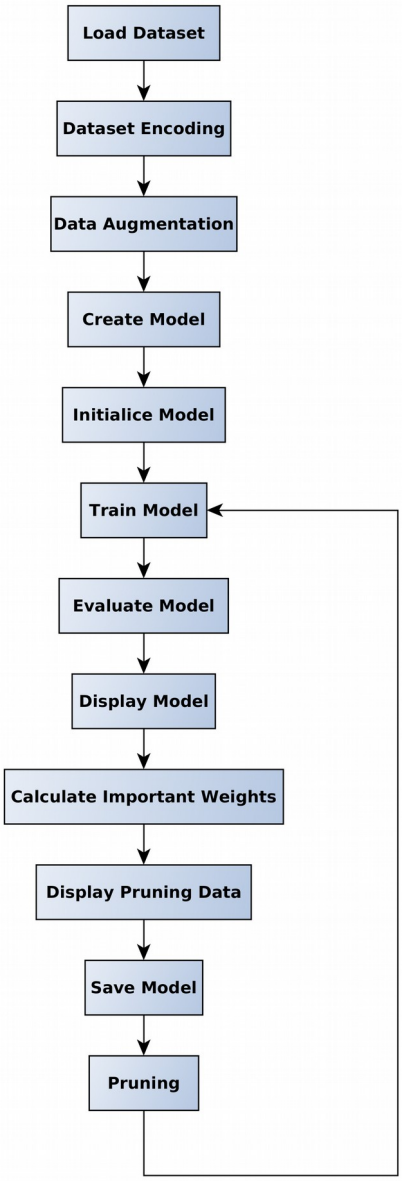


Abbildung 1: Ablauf des Testprogramms

4.2.1 Dataset

Im ersten Schritt wird der Datensatz geladen hier kommt als Datensatz der klassische MINIST Datensatz zum Einsatz. Er beinhaltet die Zahlen von 0 bis 9 mit den entsprechenden Integer Werten als Label. Es ist ein sehr einfach zu klassifizierender Datensatz bei dem alle Zahlen als Graustufenbild der selben Größe, um ihren Schwerpunkt zentriert und auf einheitliche Zahlengröße normiert, gegeben sind.

4.2.2 Encoding

Die Label des MINIST Datensatz müssen für das Netz in eine verarbeitbare Form gebracht werden. Hierzu werden die Integer Label in ein One-Hot-Encoding umgewandelt, es wird ein Tensor der Länge 10 erstellt bei dem alle werte auf 0 gesetzt sind bis auf den, dessen Index mit dem Label übereinstimmt. Zudem wird ein One-Hot-Others-Cold-Encoding erstellt, hier handelt es sich um eine Variante des One-Hot-Encoding bei dem anstelle von 0 alle anderen Werte auf -1 gesetzt werden. Dieses Encoding kann mit einer Tanh anstelle einer Sigmoid Schicht verwendet werden.

4.2.3 Data Augmentation

Eine detaillierte Erklärung zu diesem Punkt ist in *4.1.2 Datensatz* zu finden. Hier sollen noch mal die wichtigsten Punkte aufgezeigt werden.

Der MINIST Datensatz hat im Graustufenbild nur an den Stellen hohe Werte an denen die Zahlen stehen, an allen anderen Stellen ist der Wert 0. Durch die Multiplikation mit 0 sind die Aktivierungen und so auch die Gradienten an diesen Stellen immer 0, egal was für Gewichtswerte hier gelernt wurden. Bei der Berechnung der Saliency Map kann die Saliency also zwangsweise nur in den Zahlenregionen ungleich 0 sein. Tests auf diesen Daten würden also immer zeigen, dass das Netz auf die richtigen Regionen achtet.

Um die Aufgabe nicht so trivial zu machen wurden die Bilder daher mit einem Gaus und einem Salz und Pfeffer Rauschen überlagert. Die Bilder werden außerdem von einem Wertebereich von 0 bis 255 auf einen Bereich von -1 bis 1 skaliert, was den verwendeten Tanh Aktivierungen Rechnung trägt.

4.2.4 Model

Ausführlicher ist der Netzaufbau in *4.3.1 Netzaufbau* beschrieben. Einige wichtige Punkte sollen hier zum Grundverständnis dennoch genannt werden. Als Basis Model wurden 3 fuly connected Layer (FCL) verwendet. Die FCL wurden selbst implementiert um das pruning zu ermöglichen. Hierfür wurde der Gewichtstensor mit einem nicht trainierbarem variablem Tensor als Maske elementweise multipliziert. Der Output jeder Schicht wurde ebenfalls mit einem Masken Tensor verrechnet. Zu Beginn werden diese Tensoren mit 1 initialisiert. Auf diese Weise ist die Möglichkeit gegeben ganze Neuronen oder einzelne Gewichte abzuschalten indem die Maske an der entsprechenden Position auf 0 gesetzt wird.

4.2.5 Model Initialisierung

Das Model wird entweder von einer vorhandenen Gewichtsdatei initialisiert, oder wenn keine vorhanden ist zufällig.

4.2.6 Training

Das Model wird trainiert bis es nicht mehr konvergiert, alle 100 Schritte wird eine Evaluierung durchgeführt bei der die Acuracy für das aktuellen Batch berechnet wird.

4.2.7 Evaluierung

Nach dem Training wird das Model mit den Validation Daten Evaluiert, hierzu wird die Acuracy berechnet.

4.2.8 Display Model und Pruning

Die Inputs jeder Schicht sowie die Gradienten für die Gewichtswerte und Inputs und die Input Importance werden geplottet wenn das Flag „display_model“ auf True gesetzt ist. Das Flag „display_pruning“ sorgt dafür dass die Cummulierte Importance und die Pruning-Masken angezeigt werden.

4.2.9 Berechnung wichtiger Gewichte und Pruning

Diese Berechnungen sind so durchgeführt wie in *4.4 Pruning* und *4.3.3 Weight Importance* ausgeführt.

4.3 Implementierungsdetails

4.3.1 Netzaufbau

Zum testen wurde ein einfaches 3 schichtiges voll verbundenes Netz verwendet, wie in *Abbildung 2: Aufbau des Test-Netzes* zu sehen. Jedes Gewicht und jeder Output eines Neuron wurde dabei elementweise mit einer Maske multipliziert (im Bild als gestrichelte Linie dargestellt). Mit Hilfe dieser Maske kann pruning durchgeführt werden.

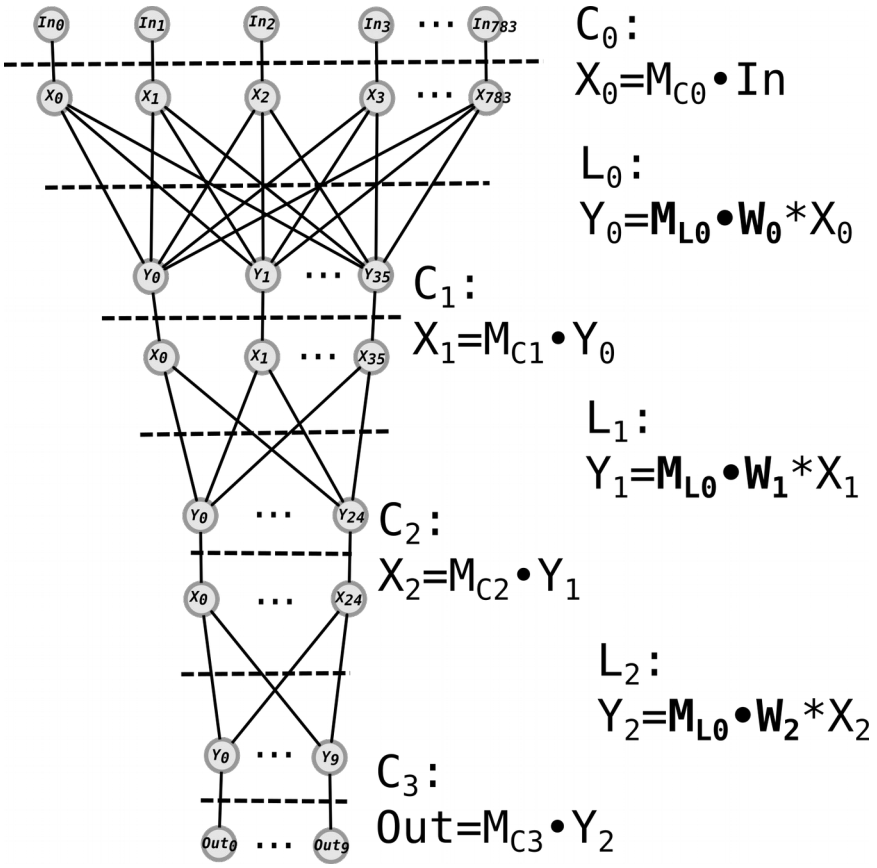


Abbildung 2: Aufbau des Test-Netzes

a) Training Model

An das Basis Model wurde fürs Training eine Loss Funktion angehängt, es wurde hier der Mean Square Error (MSE) verwendet. Und ein Gradienten Abstiegs Optimierer verwendet.

b) Evaluation Model

Zur Evaluierung wurde an das Basis Model die Berechnung für die Acuracy angehängt.

c) Pruning Model

Das Basis Model kann einen vom Trainings Loss unterschiedlichen Pruning Loss verwenden, welcher Loss hier Verwendung findet kann umgestellt werden. Der Pruning Loss entspricht hierbei einfach dem Netzoutput an der Stelle die für die im Input gegebene Zahl steht. Er wurde aus [2] übernommen, wie auch die Berechnung der Saliency.

Es werden hier außerdem zur jeder Schicht noch die nötigen Operationen hinzugefügt, um die Gradienten der Schichtinputs und der Gewichte der Schichten zu berechnen.

4.3.2 Input Importance

Für jede Schicht wird der Gradient des Losses (J) in Bezug auf den Input der Schicht (X) bestimmt, dies ergibt die Saliency Map (SM).

Formel VII: Gradient Saliency zur Bestimmung der Input Importance

$$SM = \frac{\partial J}{\partial X}$$

Große positive und negative Gradienten deuten beide auf große Auswirkungen des Inputs auf den Loss hin. Es wird daher der Absolutwert gebildet, um die Importance des Inputs (I_X) zu bestimmen.

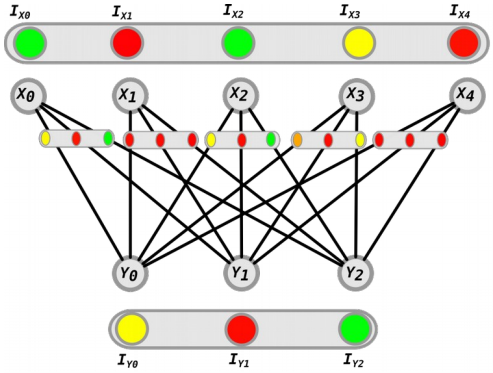
Formel VIII: Input Importance

$$I_X = |SM|$$

Die Importance wird pro Neuron noch auf den Wertebereich zwischen 0 und 1 Skaliert (Min Max Skalierung) um sie besser vergleichbar zu machen.

4.3.3 Weight Importance

Aus der Importance des Inputs einer Schicht (I_{X_l}) und der Importance des Inputs der Folgeschicht (I_{Y_l}) kann nun die Importance (I_{W_l}) der einzelnen Gewichte (W) berechnet werden. Hierbei ist zu beachten, dass der Input der Folgeschicht (X_{l+1}) immer dem Output (Y) der vorhergehenden Schicht entspricht.



$$W_{xy}:$$
$$I_{Wxy} = I_{Xx} * I_{Yy}$$

Abbildung 3: Weight Importance Berechnung

Ist der Input der Folgeschicht (Yy) unwichtig (= 0) sind folglich alle Gewichte (Wxy) die zu diesem Input führen ebenfalls unwichtig.

Ähnlich ist ein Gewicht (Wxy) welches einen unwichtigen Input (Xx) mit einem beliebigen Input der Folgeschicht (Yy) verbindet ebenfalls unwichtig.

Ist der Input (Yy) wichtig (= 1) und der Input der Vorgängerschicht (Xx) ebenfalls wichtig, muss auch das Gewicht (Wxy) welches die beiden verbindet wichtig sein.

Es muss also jeder Eintrag der Importance der beiden Schichten mit jeweils jedem anderen multipliziert werden, um die Importance der Gewichte zu ermitteln, wie in *Abbildung 3: Weight Importance Berechnung* zu sehen.

4.4 Pruning

Mit Hilfe der Weight Importance kann nun die Cummulated-Importance (CI) der Gewichte über verschiedene Inputs des Netzes hinweg bestimmt werden. Diese wird gebildet, indem für ein Trainings Sample die Importance für jedes Neuron getrennt aufaddiert wird.

Gewichte die für den größten Teil der Inputs unwichtig waren, werden hier eine geringe CI erhalten.

Auch Gewichte die nur bei sehr wenigen Inputs wichtig waren und sonst immer unwichtig erhalten eine geringe CI. Solche Gewichte entstehen durch Overfitting, da sie nur auf einen ganz bestimmten Input reagieren. Eine geringe CI ist also auch angebracht.

Anhand der CI kann nun ein Pruning-Durchlauf durchgeführt werden.

Hierzu wurde zunächst ein Pruning-Faktor (PF) verwendet der jeden Zeitschritt PF % der verbleibenden Gewichte entfernt. Dieser Exponentielle Decay (Zerfall) würde allerdings dafür sorgen, dass nach unendlich vielen Schritten alle Gewichte entfernt werden.

Deshalb wurde auf einen Pruning-Schwellwert (PT) gewechselt, alle Gewichte die unterhalb dieses liegen werden entfernt. Der Schwellwert bildet sich dabei aus dem Median der (nicht entfernten) Gewichte eines einzelnen Neurons, der Sparcity (S) des Neurons und dem Sparcification-Factor (SF), einem Hyperparameter.

Die Sparcity berechnet sich dabei aus der Anzahl der ursprünglichen Gewichte zum Zeitpunkt 0 (C0) und der Anzahl der noch verbleibenden Gewichte zum Zeitpunkt t (Ct).

Formel IX: Sparcity

$$S = \frac{C_0}{C_t}$$

Umso mehr Gewichte entfernt werden umso größer wird also die Sparcity. Beginnend bei eins läuft sie gegen unendlich wenn alle Gewichte entfernt würden.

Der Sparcification-Factor (SF) gibt an, bei wie viel Prozent des Medians der Pruning-Schwellwert liegen soll. Er wird allerdings mit der Sparcity potenziert was zu einem Decay (Abfall) führt, so dass nie alle Gewichte entfernt werden können.

Formel X: Pruning-Schwellwert

$$PT = Median * SF^{S}$$

Umso weniger Gewichte noch vorhanden sind umso weniger Gewichte werden im nächsten Schritt entfernt.

Beträgt der SF genau 1 so werden alle Gewichte die kleiner sind als der Median entfernt, da der Median in der Mitte aller Werte Liegt wird also in jedem Pruning Schrit die Hälfte aller Gewichte entfernt. Das entspricht einem Exponentiellen Zerfall der für unendlich viele Schritte gegen 0 läuft.

SF < 1 findet ebenfalls ein Exponentieller Zerfall statt der den SF gegen 0 laufen lässt. Dies bedeutet dass pro Zeitschritt immer weniger Gewichte entfernt werden. Wir erhalten also zwei Exponentielle Funktionen die gegeneinander arbeite.

Der Grenzwert gegen den sie laufen kann allerdings nicht exakt bestimmt werden, da die Verteilung der Gewichte im voraus nicht bekannt ist. SF = 50% des Median bedeutet folglich nicht 50% * 50% = 25% der Gewichte werden entfernt. Wenn man dies aber als Aproximation annimmt können folgende Gleichungen aufgestellt werden, mit $m = 0.5$ da der Median immer bei exakt 50% aller Gewichte liegt:

$$C_t = C_{t-1} - \lfloor C_{t-1} * PT_{t-1} \rfloor$$

und eingesetzt:

Formel XI: Itterative Vorschrift zum Schätzen der verbleibenden Gewichte

$$C_t = C_{t-1} - \left\lfloor C_{t-1} * m * SF^{\frac{C_0}{C_{t-1}}} \right\rfloor$$

Numerisch kann hier ein Grenzwert bestimmt werden da die Anzahl an Gewichten (C) hier nur ganzzahlige Werte annehmen kann. Wie in *Abbildung 4: Numerische berechnete Konvergenz der Anzahl der Gewichte* für SF von 0 bis 0.9 gezeigt.

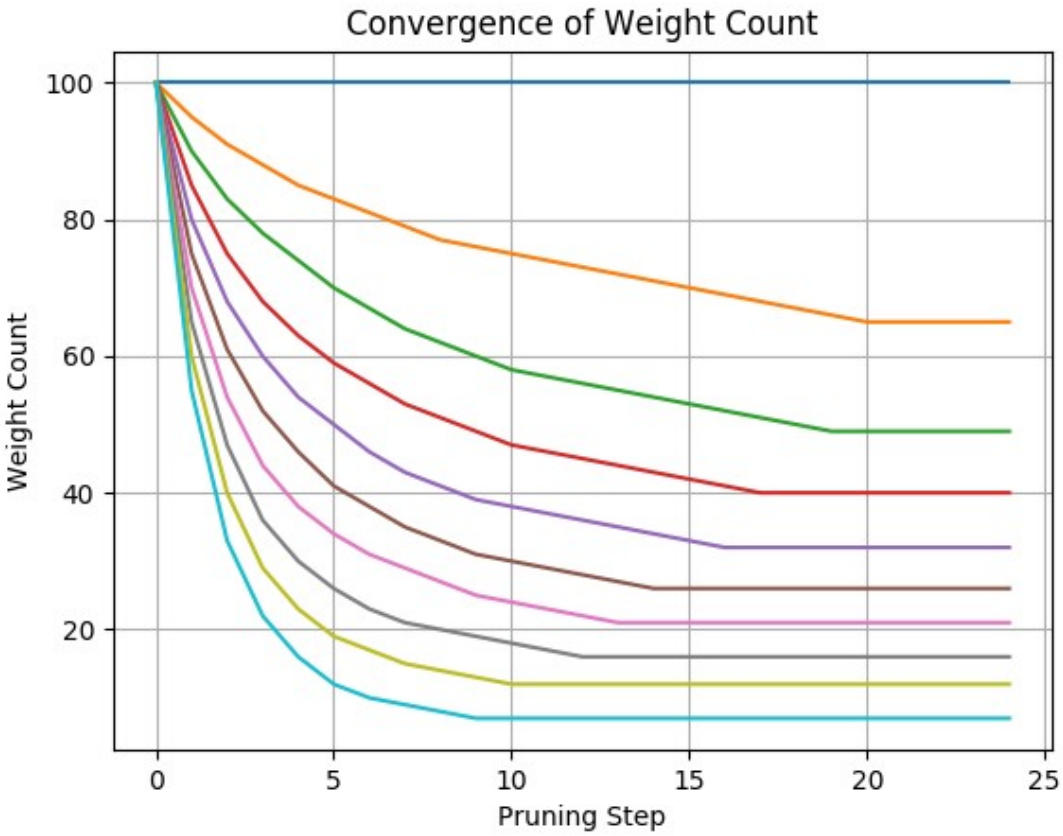


Abbildung 4: Numerische berechnete Konvergenz der Anzahl der Gewichte

Mathematisch hat es zunächst den Anschein als würde diese Folge nicht konvergieren, dadurch dass es aber immer nur eine ganzzahlige Anzahl an Gewichten geben kann, durch die Gausklammern dargestellt, kann allerdings dennoch eine Umformung vorgenommen werden und so der Grenzwert in geschlossener Form bestimmt werden. Die Herleitung ist im Folgenden zu sehen:

Für Konvergenz gilt:

$$C_t = C_{t-1}$$

Damit ergibt sich:

$$C_t = C_t - \lfloor C_t * m * SF^{\frac{C_0}{C_t}} \rfloor$$

und vereinfacht:

$$0 = \lfloor C_t * m * SF^{\frac{C_0}{C_t}} \rfloor$$

Durch weglassen der Gausklammern unter Beachtung der Rundung erhält man eine Ungleichung:

$$1 > C_t * m * SF^{\frac{C_0}{C_t}}$$

Die Umgeformt werden kann zu:

$$1 > C_t * m * e^{\frac{C_0}{C_t} * ln(SF)}$$

und mit der Umkehrfunktion von $x * e^x$ (der LambertW-Funktion W) ergibt sich:

$$C_t = -\frac{C_0 * log(SF)}{W(-C_0 * m * log(SF))}$$

Durch einsetzen von Beispielwerten erhält man ähnliche Grenzwerte wie bei der numerischen Bestimmung, siehe rote Linie in *Abbildung 4: Numerische berechnete Konvergenz der Anzahl der Gewichte*.

$$\left\{-\frac{c \log(s)}{W(-c\, m \log(s))}, c = 100, s = 0.3, m = 0.5\right\} \qquad -\frac{c \log(s)}{W(-c\, m \log(s))} \approx 40.1421$$

An dieser Stelle könnten auch andere Vorschriften zur Bestimmung der Anzahl an zu entfernenden Gewichte verwendet werden, so wird in [16] anstelle SF% des Median, eine Rangfolge gebildet und alle Gewichte unterhalb eines Rangs entfernt, so hat die statistische Verteilung der Gewichte keinen Einfluss auf die Anzahl der zu entfernenden Gewichte. Es war nicht Ziel der Arbeit hier verschiedene Vorgehen zu vergleichen weshalb Alternativen nicht getestet wurden.

5 Ergebnisse

5.1 Vergleich von Importance Maßen

Aus den bei Recherchen gefundenen Maßen für die Importance wurden zwei sehr ähnliche Maße, der Impact mittels Tailor Polynom und die Gradient Saliency, ausgewählt und Verglichen. Beide Maße sind in *3.1.2 Saliency als Änderung des Outputs* und *3.2 Importance für Pruning* im Detail beschrieben. Die Ergebnisse des

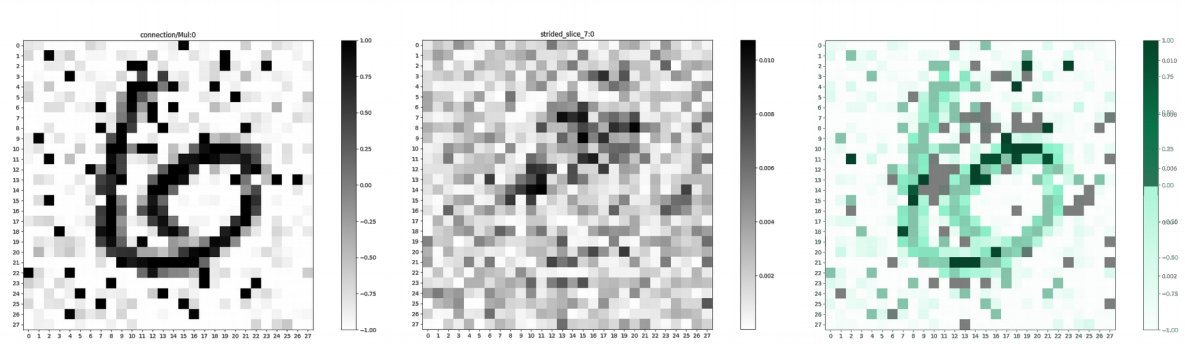


Abbildung 5: Impact der Eingabe auf den Trainings Loss

Vergleiches sind im Folgenden dargestellt.

In *Abbildung 5* ist links der Netz Input, in der Mitte der Impact des Inputs auf den Trainings Loss und rechts ein überlagertes Bild beider dargestellt. In der Überlagerung sind die dunkel grauen Pixel jeweils hohe Impact Werte die sich nicht mit der Zahl decken, die dunkel grünen Pixel entsprechend hohe Werte die sich mit der Zahl überlagern. Wie hier gut zu sehen ist tritt der höchste Impact an den Rändern der Zahl sowie auf der Zahl selbst auf. Er scheint nach den in *4.1.3 Importance Map* bestimmten Kriterien also an plausiblen Stellen hoch zu sein.

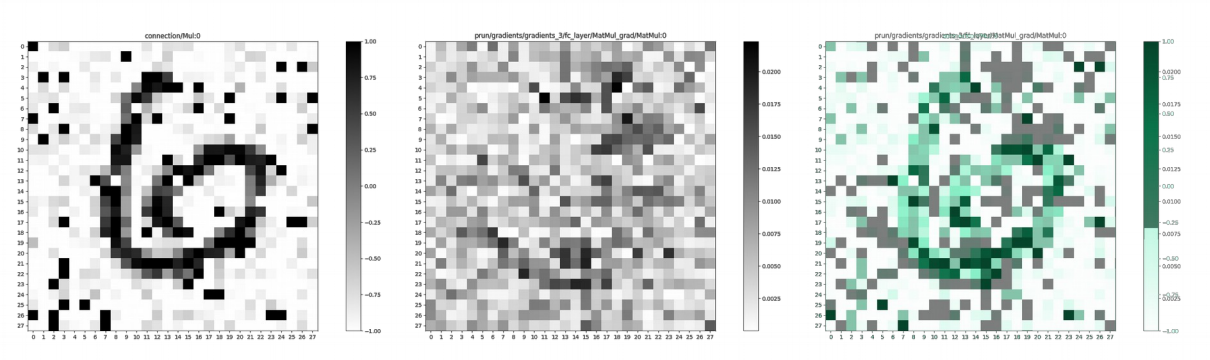


Abbildung 6: Gradient Saliency der Eingabe in Bezug auf den Trainings Loss

In *Abbildung 6* ist links der Netz Input, in der Mitte die Saliency des Inputs auf den Trainings Loss und rechts ein überlagertes Bild beider dargestellt. In der Überlagerung sind die dunkel grauen Pixel jeweils hohe Saliency Werte die sich nicht mit der Zahl decken, die dunkel grünen Pixel entsprechend hohe Werte die sich mit der Zahl überlagern. Wie hier gut zu sehen ist tritt die höchste Saliency an den Rändern der Zahl sowie auf der Zahl selbst auf. Er scheint also ebenfalls an plausiblen Stellen hoch zu sein.

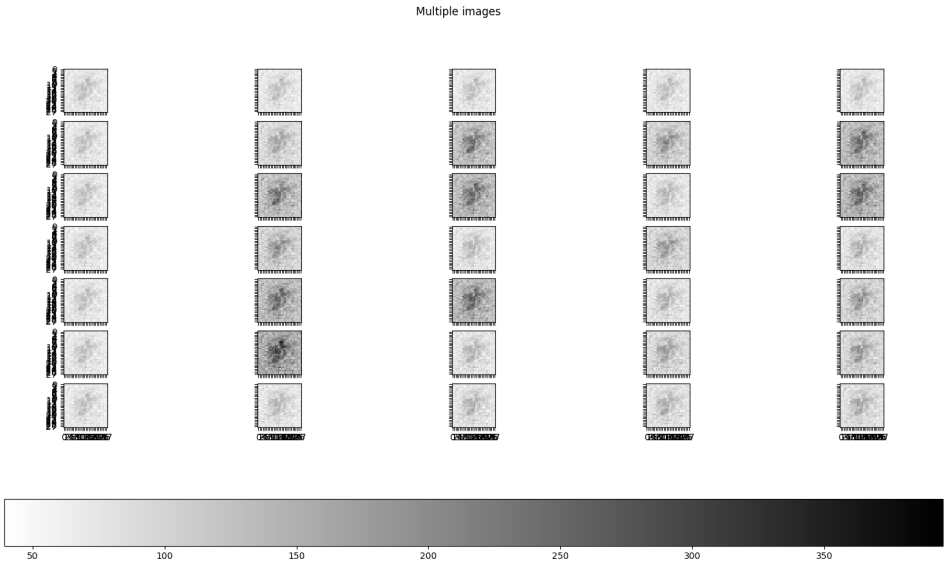


Abbildung 7: Weight Importance anhand Impact

In *Abbildung 7* ist die Weight Importance anhand des Impacts dargetellt, jedes einzelne Bild ist hierbei die Importance aller Gewichte eines Neurons. Wie zu sehen ist gibt es hier Neuronen welche generell wichtiger sind (durch die dunklere Färbung zu erkennen). Die einzelnen Gewichte sind dabei alle in der Mitte des Inputs (wo sich auch alle Zahlen des MNIST Datensatzes befinden) dunkler. Der Impact scheint sich also zu eignen um die wichtigen Gewichte zu finden.

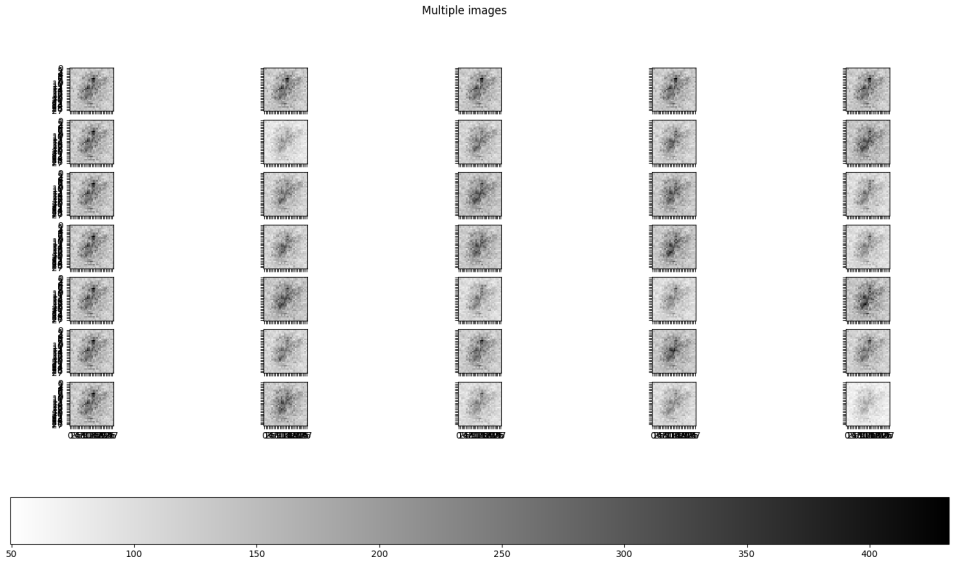


Abbildung 8: Weight Importance anhand Saliency

In *Abbildung 8* ist die Weight Importance anhand der Saliency dargetellt, jedes einzelne Bild zeigt die Importance der Gewichte eines Neurons. Die einzelnen Gewichte sind auch hier in der Mitte des Inputs dunkler. Die Saliency scheint sich also gleichermaßen zu eignen um die wichtigen Gewichte zu finden.

Die Maße scheinen also beide ähnlich gute Werte zu liefern, sie unterscheiden sich allerdings in einem wichtigen Punkt, der Rechenzeit. Die Saliency benötigt pro bild nur den Gradienten des Inputs in Bezug auf den Output, während der Impact zusätzlich noch für jeden Input eine Multiplikation mit dem Gradienten benötigt. Also eine zusätzliche Multiplikation pro Input und pro Neuron des Netzes, bereits bei dem kleinen Testnetz führte dies zu hohem Zeit und Performance Verlust, weshalb im späteren Verlauf der Arbeit bevorzugt die Importance anhand der Saliency verwendet wurde.

5.2 Weight Importance

Die Importance der Gewichte kann entweder direkt mit einem Importance Maß bestimmt werden, oder indirekt aus der Input Importance zweier aufeinanderfolgenden Schichten berechnet werden. Die Arbeit wurde mit dem Ziel der Berechnung aus der Input Importance begonnen. Die Möglichkeit der direkten Berechnung kam dann während der Recherche auf, weshalb ein Vergleich der beiden Verfahren durchgeführt wurde. Hierzu wurden wiederum die entstehenden Importance Maps nach den Kriterien in *4.1.3 Importance Map* verglichen. Im folgenden sind die Ergebnisse für die Importance anhand der Gradient Saliency mit Trainings Loss aufgeführt, in *8 Anhänge* sind weitere Ergebnisse für den Pruning Loss, den Impact und den Fashion-MNIST Datensatz zu finden. Das eigentliche Pruning wurde nur mit der Bestimmung der Weight Importance aus der Input Importance validiert und ist in *5.3 Pruning* beschrieben.

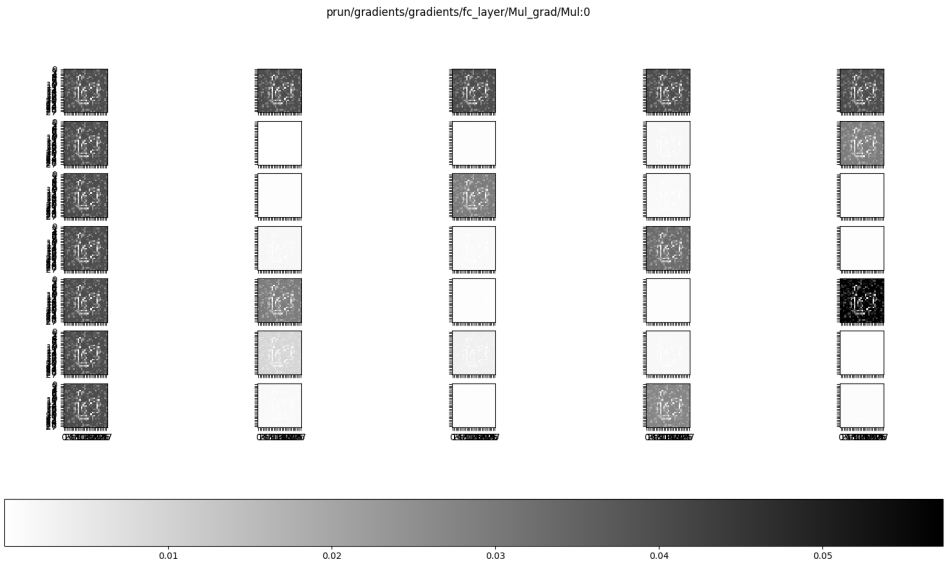


Abbildung 9: Importance direkt von Gewichten

Bei der Importance der Gewichte direkt bestimmt durch die Gradienten des Loss in Bezug auf die einzelnen Gewichte, siehe *Abbildung 9,* erhält man an den Stellen an denen sich die Zahl befindet niedrige Gradienten an den anderen Stellen hohe. Eine Änderung an wichtigen Gewichten hat damit kaum eine Auswirkung auf die Entscheidungsfindung und steht damit im Widerspruch zum gewünschten Verhalten.

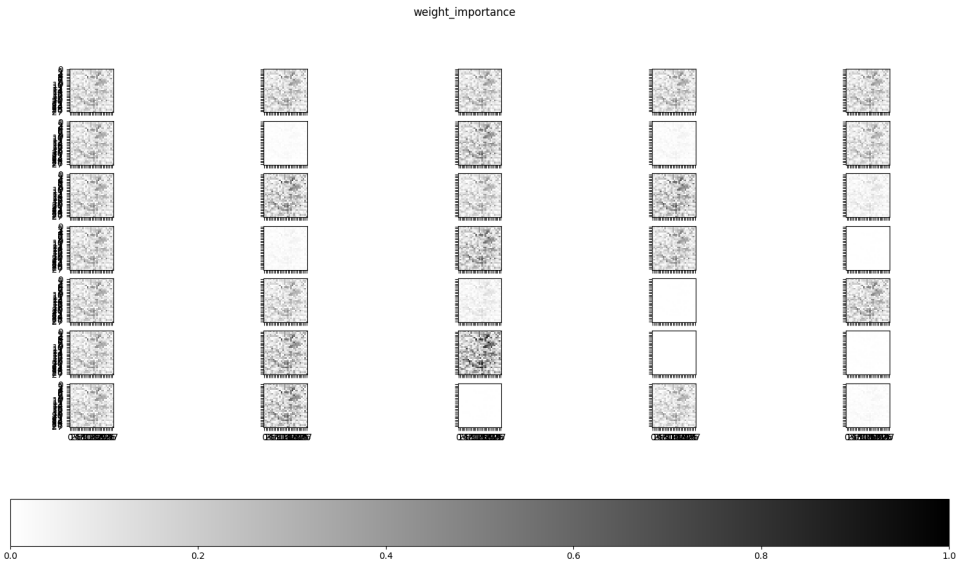


Abbildung 10: Importance der Gewichte anhand des Inputs

Die Importance der Gewichte bestimmt durch zwei aufeinanderfolgende Input Importanc Maps zeigt hingegen deutlich, dass die wichtigen Bereiche mit Bereichen hoher Importance übereinstimmen. Sie eignet sich folglich besser für die Bestimmung wichtiger Gewichte und wird im Rest der Arbeit verwendet.

5.3 Pruning

Pruning wurde auf einem initialen Netz für verschiedenen Sparcity Faktoren durchgeführt. Das Netz wurde dabei 150 Pruning Iterationen unterzogen und der Verlauf der Anzahl der Gewichte und Sparcity jeder Schicht sowie die Accuracy des Netzes gespeichert. Im folgenden ist immer nur der Gewichtsverlauf der ersten Schicht dargestellt, die Anderen Schichten verhalten sich ähnlich und sind in *8 Anhänge* zu finden. Die Sparcity ist für jede Schicht dargestellt, allerdings ohne Zeitlichen Verlauf. Dieser ist an der Dichte der Linien zu erkennen. Hier werden beispielhaft drei Sparcity Faktoren verglichen, in *8 Anhänge* sind die Diagramme weiterer Faktoren zu finden. Bei allen Diagrammen ist zur besseren Darstellung die Inverse der Sparcity aufgetragen da diese immer im Bereich zwischen 0 und 1 liegt.

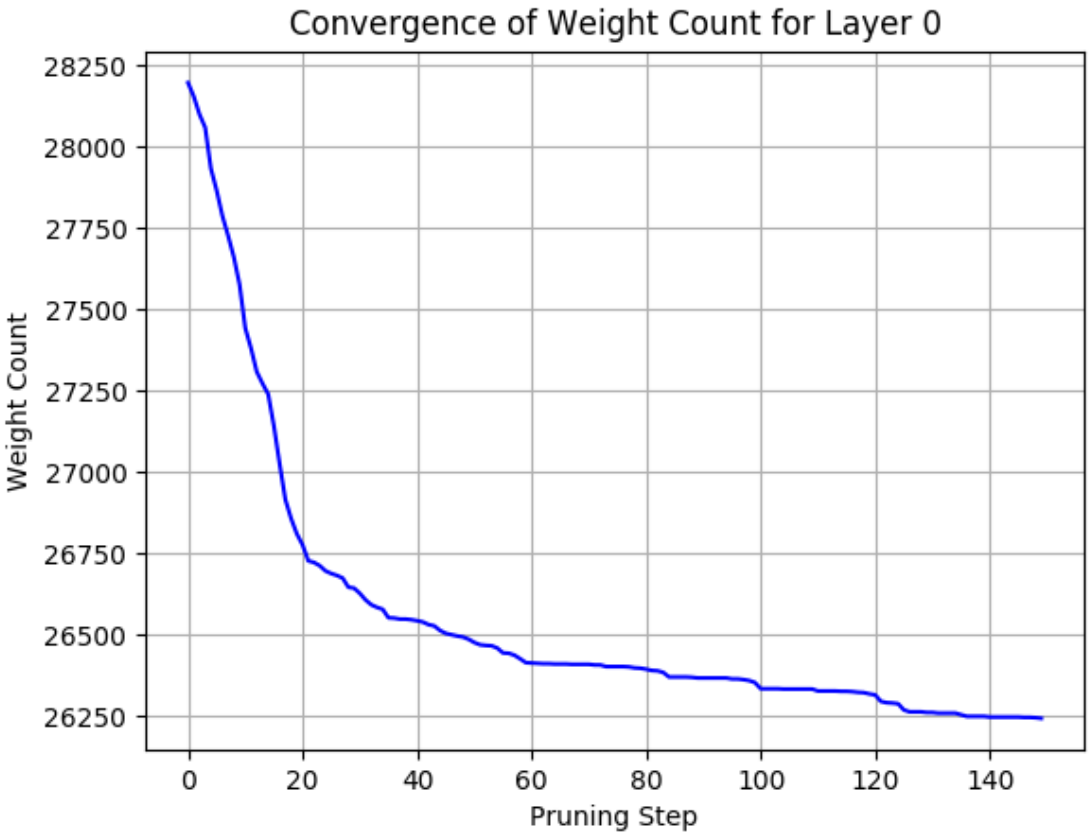


Abbildung 11: Verlauf der Anzahl der Gewichte bei SF 0.6

Gut zu erkennen ist in *Abbildung 11: Verlauf der Anzahl der Gewichte bei SF 0.6,* dass sich die Anzahl der Gewichte einem Grenzwert nähern. In *Abbildung 12* hingegen ist keine stetige Annäherung an einen Grenzwert für Layer 2 erkennbar, dies sollte eigentlich, wie bei Layer 0, durch eine zunehmende Dichte der Linien erkennbar sein. Stattdessen wirkt es eher wie ein linearer Verlauf. Dies ist auf die Imortance Verteilung der Gewichte zurückzuführen, hier wird deutlich dass SF=60% des Medians nicht automatisch heißt, dass immer 50% * 60% = 30% der Gewichte entfernt werden.

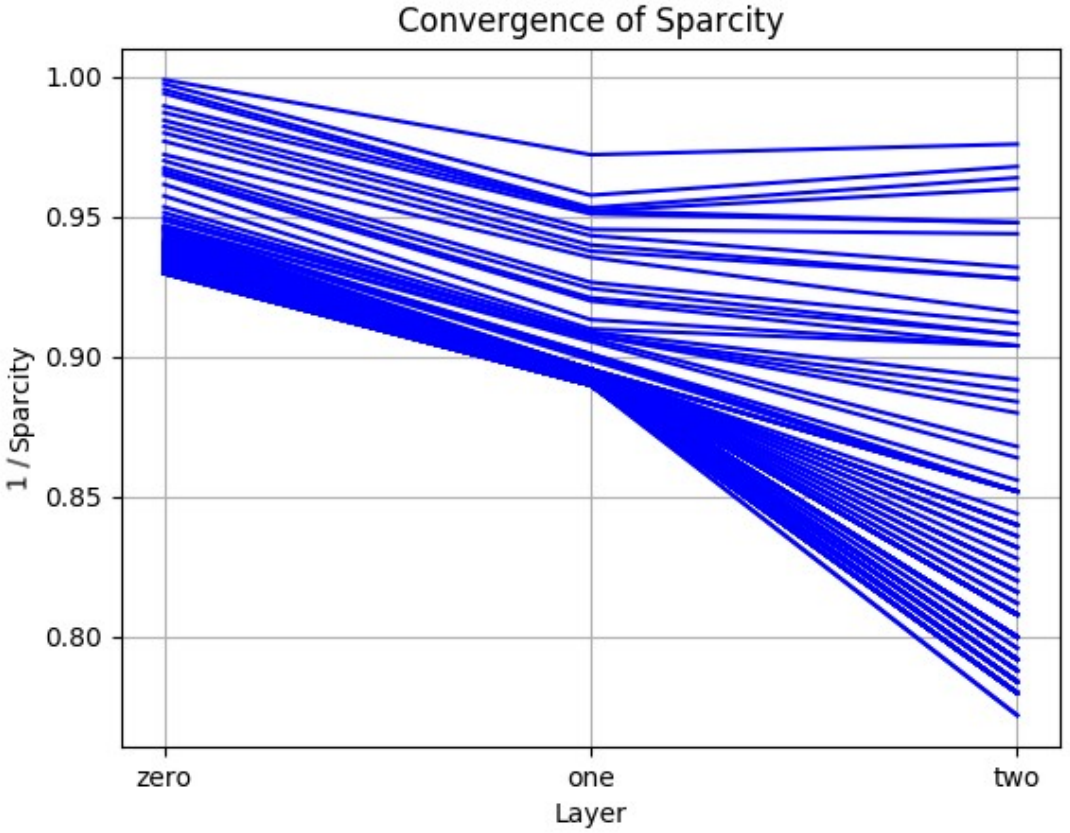


Abbildung 12: Verlauf der Sparcity bei SF 0.6

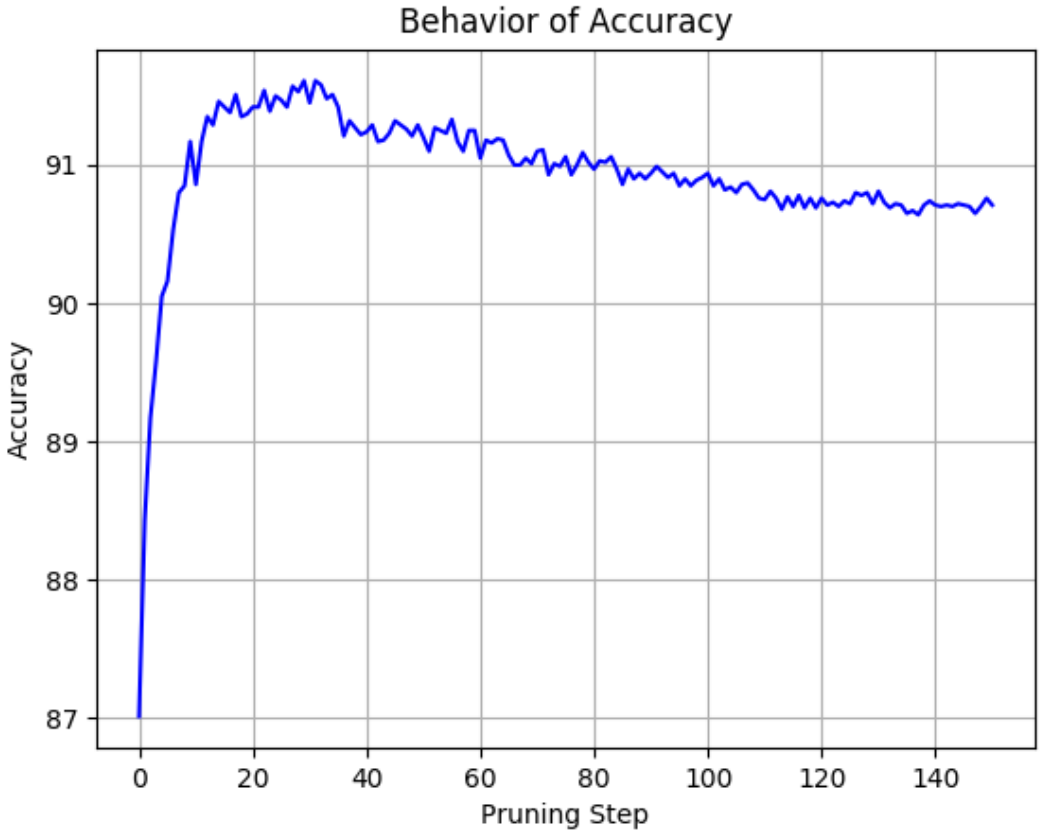


Abbildung 13: Verlauf der Accuracy bei SF 0.6

Überraschend ist zu beobachten, dass die Accuracy in *Abbildung 13* in den ersten 30 Pruning Durchläufen nicht fällt sondern steigt. Dies hat nichts mit der zusätzlichen Trainingszeit beim fine tuning zu tun, sondern ist auf das vermeiden von Overfitting zurückzuführen. Das Netz wird in jedem Schritt trainiert bis sich die Accuracy nicht mehr verändert, durch das entfernen von Gewichten die nur führ einen sehr geringe Anzahl an Trainingsdaten relevant ist wird das Netz allerdings gezwungen mit den verbleibenden Gewichten besser zu veralgemeinern.

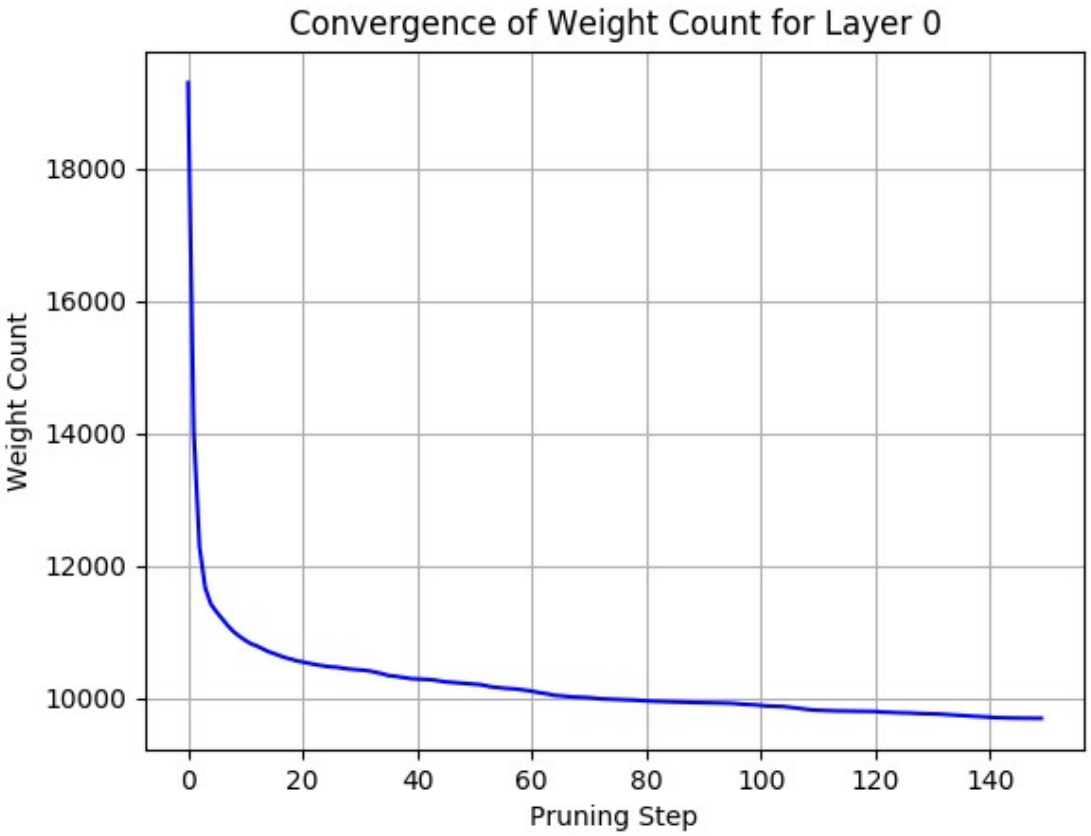


Abbildung 14: Verlauf der Anzahl der Gewichte bei SF 0.9

Auch bei einem SF von 0.9 ist ein exponentiller Verlauf gegen einen Grenzwert zu beobachten, siehe *Abbildung 13*, dieser fällt gerade im vorderen Bereich deutlich schneller und entfernt also bereits zu Beginn des Pruning deutlich mehr Gewichte.

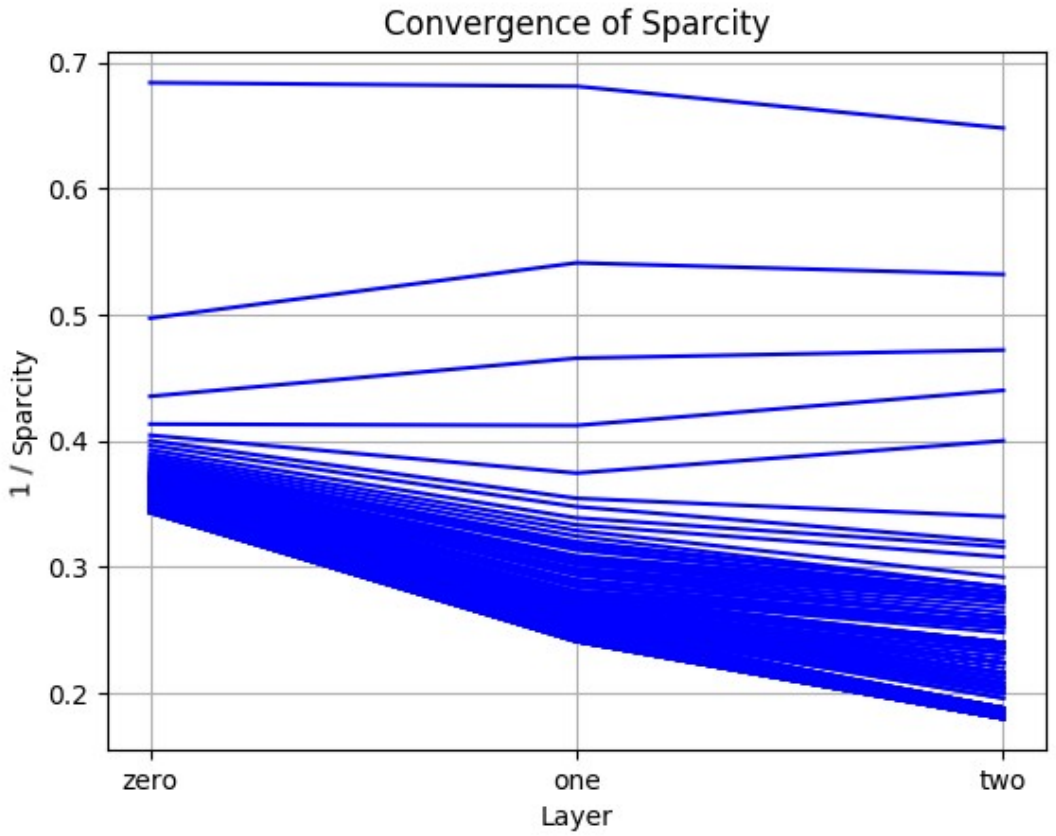


Abbildung 15: Verlauf der Sparcity bei SF 0.9

Bei einem SF der zu einem PT so nahe am Median führt spielt auch die Verteilung der Gewichte eine deutlich geringere Rolle und so kann man in *Abbildung 15* eine Annäherung der Sparcity an einen Grenzwert nun führ alle Schichten beobachten. Hier treten nur noch kleine Artefakte in der Dichte der Linien auf.

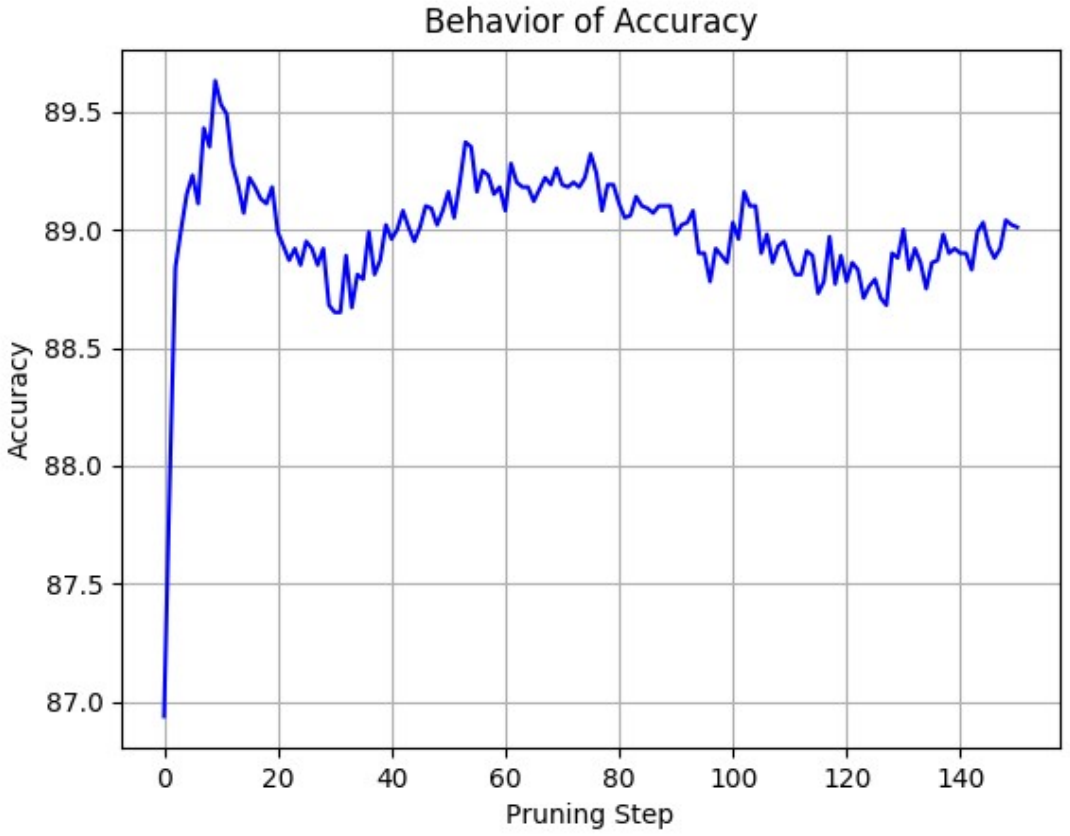


Abbildung 16: Verlauf der Accuracy bei SF 0.9

Durch das schnelle entfernen so vieler Gewichte kann das Netz nicht mehr die Accuracy erreichen die es beim langsamen entfernen der Gewichte erreicht hat, *Abbildung 16*. Dennoch ist zu beobachten dass die Accuracy in den ersten Pruning Durchläufen zunächst zunimmt, bevor sie sich dann bei einem Grenzwert einpendelt. Das Netz erreicht hier immer noch eine höhere Accuracy als das Orginal bei nur noch ca. 30% der Gewichte.

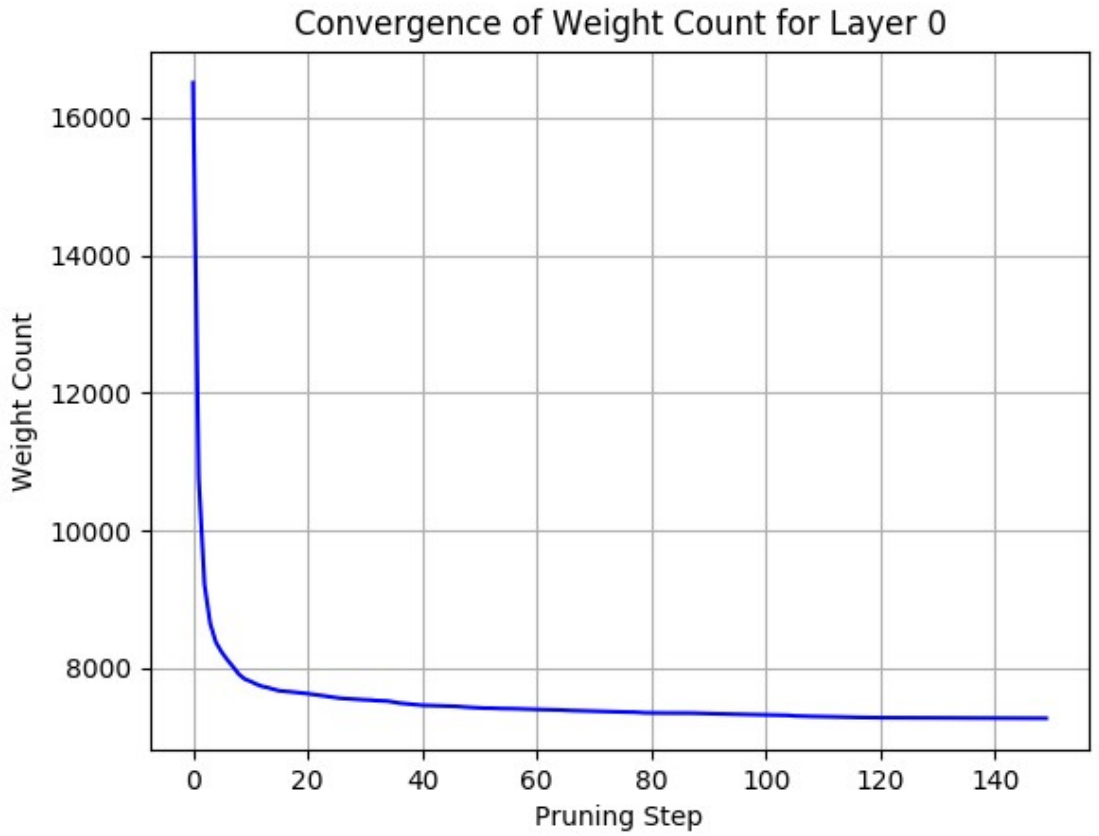


Abbildung 17: Verlauf der Anzahl der Gewichte bei SF 0.95

Sehr viele Gewichte werden bei einem SF von 0.95 entfernt, es werden jedoch immer noch nicht 100% der Gewichte entfernt wie der Verlauf der Gewichte von 28224 (nicht mehr in der Grafik) gegen den Grenzwert von 7500 zeigt, *Abbildung 17*. Tatsächlich fällt die Anzahl der Gewichte hier noch langsam, eine numerische sowie eine mathematische Schätzung liefert einen Grenzwert von ca. 300 verbleibenden Gewichten.

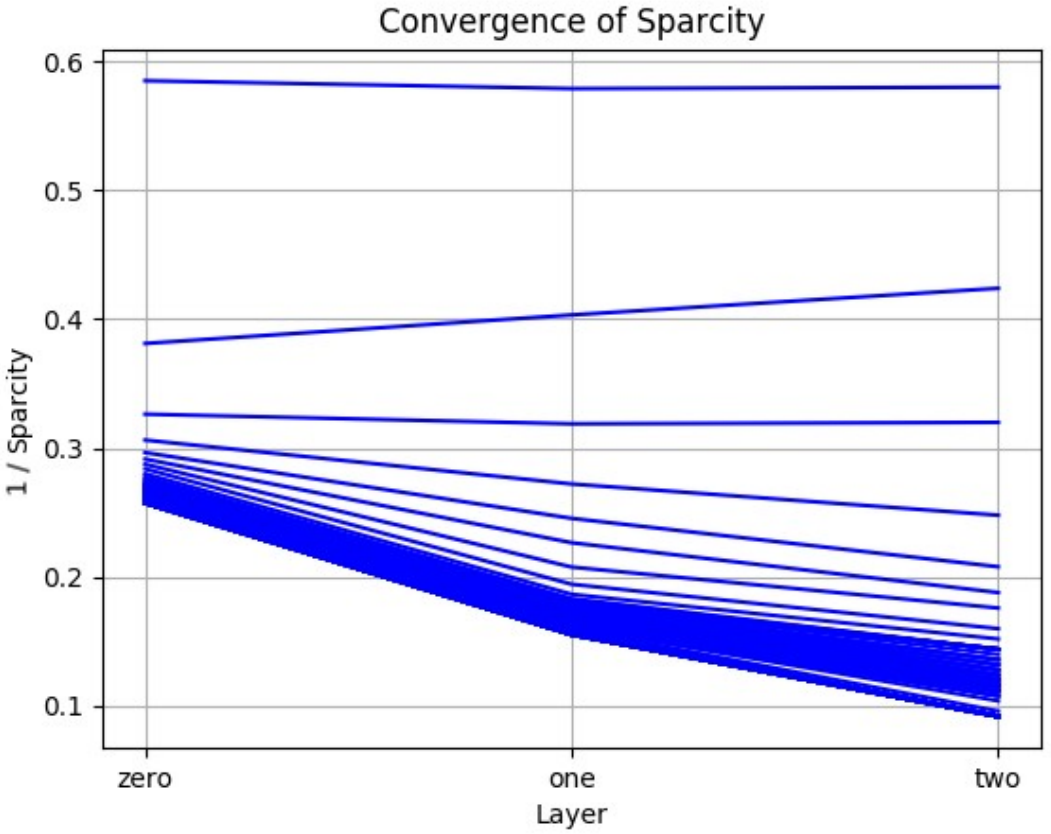


Abbildung 18: Verlauf der Sparcity bei SF 0.95

Der Verlauf der Sparcity in *Abbildung 18* ist nun noch weniger von der Importance Verteilung der Gewichte abhängig. Es bleiben hier für die verschiedenen Schichten nun nur noch 10% bis 25% aller Gewichte übrig.

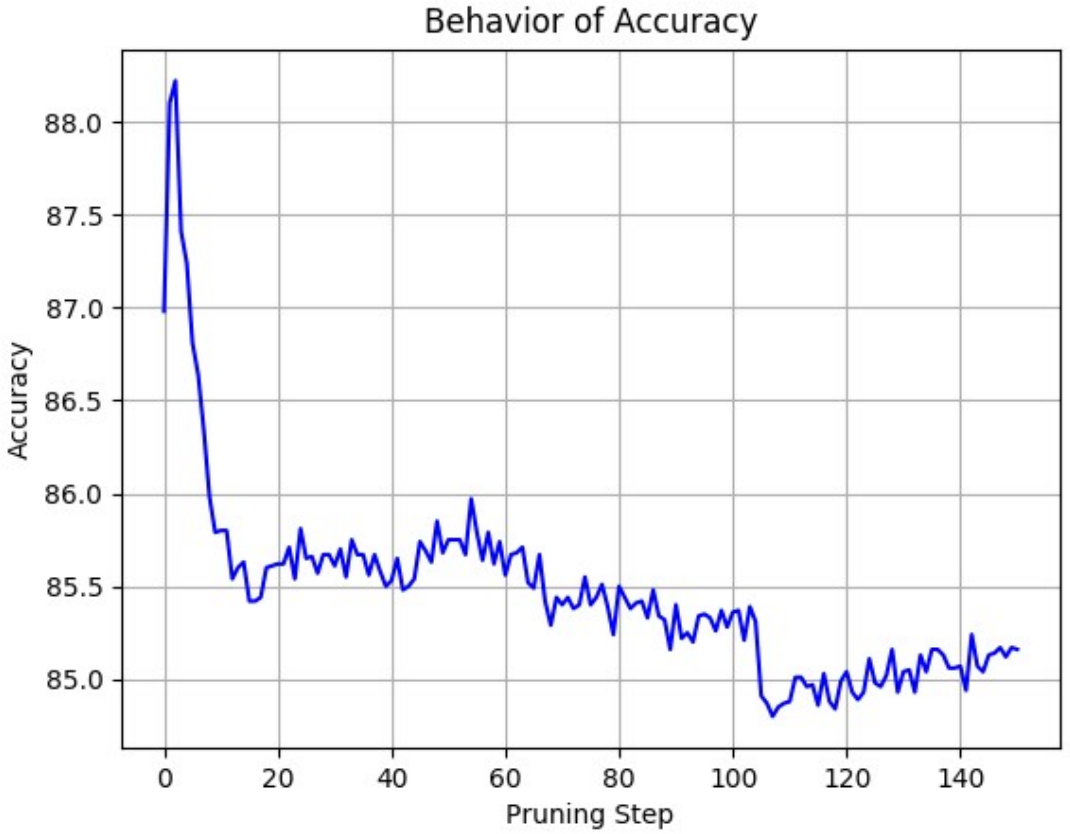


Abbildung 19: Verlauf der Accuracy bei SF 0.95

Dies ist nun auch das erste mal, dass die Accurcy unter die des original Netzes fällt, allerdings auch nur 2%, siehe *Abbildung 19*. Das ist für eine Reduzierung der Gewichte auf ca. 10% bis 25% ein beachtlich geringer Verlusst.

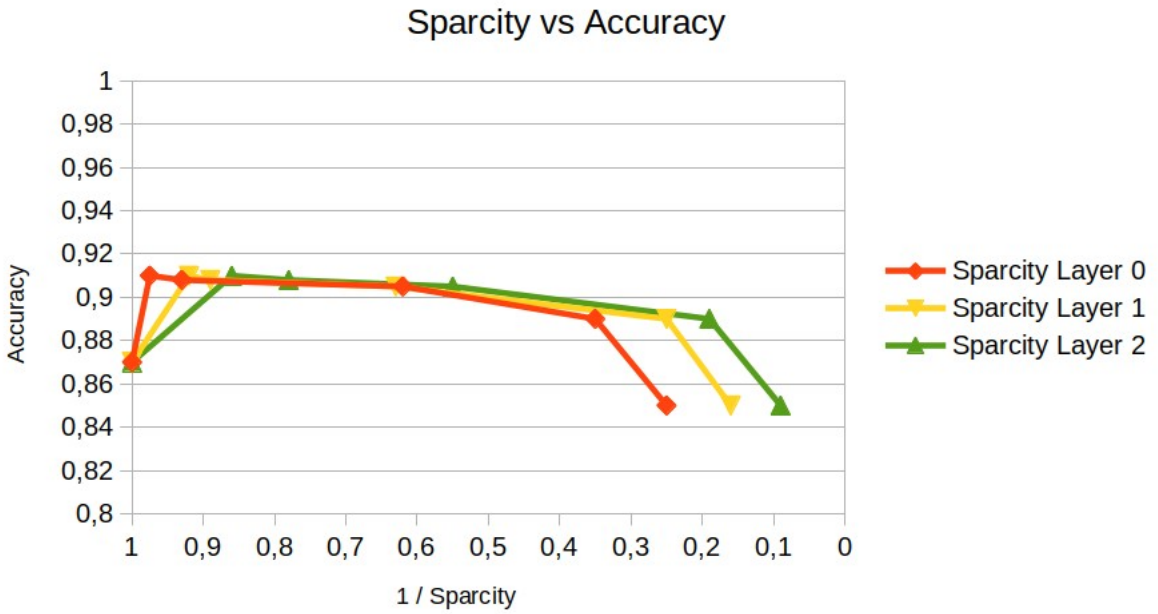


Abbildung 20: Sparcity vs Accuracy

Ein Vergleich der Sparcity gegenübergestellt zur Accuracy für alle Messwerte ist in *Abbildung 20* zu sehen, offensichtlich kann die Anzahl der Parameter im getesteten Netz auf unter 30% reduziert werden, ohne dabei einen Verlust in Accuracy verbuchen zu müssen.

Der Sparcity Faktor scheint dabei im Bereich oberhalb von 0.5 einen nahezu linearen Zusammenhang mit der Sparcity zu haben.

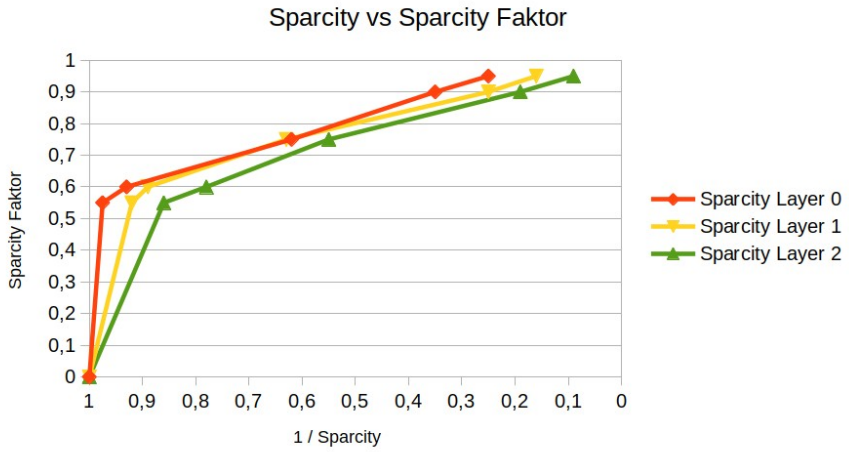


Abbildung 21: Sparcity vs Sparcity Faktor (SF)

6 Fazit

In dieser Arbeit wurden verschiedene Maße für die Wichtigkeit einzelner Gewichte, basierend auf Saliency, verglichen und das vielversprechendste, Weight Importance mithilfe Gradient Saliency der Schicht Inputs, genutzt um Netz Pruning durchzuführen. Pruning mit diesem Maß lieferte, für ein einfaches Testnetz und den MNIST Datensatz, Ergebnisse welche die Erwartungen übertrafen. So wurde die Accuracy beim Pruning zunächst nicht schlechter sondern besser, was auf die Vermeidung von Overfitting durch Pruning zurückzuführen ist. Auch bei sehr hoher Anzahl entfernter Parameter des Netzes musste nur ein geringer Verlust in der Accuracy hingenommen werden.

Eine Ausführliche Recherche und eine Darstellung der wichtigsten Veröffentlichungen in diesem Bereich ergänzt diese Arbeit.

Als Erweiterungen könnte man das in [16] beschriebene Structured Probabilistic Pruning mit dem der in dieser Arbeit verwendeten Saliency Information kombinieren, um Pruning und Overfitting bereits während des Regulären Trainings durchzuführen und zu vermeiden.

7 Quellen

Literaturverzeichnis

[1]: Marilyn Lougher Vaughn, Interpretation and Knowledge Discovery from the Multilayer Perceptron Network: Opening the Black Box., 1996
[2]: Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, 2014
[3]: Wataru Shimoda, Keiji Yanai, Distinct Class Saliency Maps For Multiple Object Images, 2016
[4]: Wataru Shimoda, Keiji Yanai, Distinct Class-specific Saliency Maps for Weakly Supervised Semantic Segmentation, 2016
[5]: Seong Joon Oh, Rodrigo Benenson, Anna Khoreva, Zeynep Akata, Mario Fritz, Bernt Schiele, Exploiting saliency for object segmentation from image level labels, 2017
[6]: Hengyue Pan, Hui Jiang, A Fast Method for Saliency Detection by Back-Propagating A Convolutional Neural Network and Clamping Its Partial Outputs, 2017
[7]: Hengyuan Hu, Rui Peng, Yu-Wing Tai, Chi-Keung Tang, Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures, 2016
[8]: S. Han, J. Pool, J. Tran, and W. Dally, Learning both weights and connections for efficient neural network, 2015
[9]: Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, Jan Kautz, Pruning Convolutional Neural Networks for Resource Efficient Inference, 2017
[10]: Michael Figurnov, Aijan Ibraimova, Dmitry Vetrov, Pushmeet Kohli, PerforatedCNNs: Acceleration through Elimination of Redundant Convolutions, 2016
[11]: N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, 2014
[12]: L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus, Regularization of neural networks using dropconnect, 2016
[13]: Alireza Aghasi, Afshin Abdi, Nam Nguyen, Justin Romberg, Net-Trim: Convex Pruning of Deep Neural Networks with Performance Guarantee, 2017
[14]: Yihui He, Xiangyu Zhang, Jian Sun, Channel Pruning for Accelerating Very Deep Neural Networks, 2017
[15]: Mohammad Babaeizadeh, Paris Smaragdis, Roy H. Campbell, NoiseOut: A Simple Way to Prune Neural Networks, 2016
[16]: Huan Wang, Qiming Zhang, Yuehai Wang, Haoji Hu, Structured Probabilistic Pruning for Convolutional Neural Network Acceleration, 2018

8 Anhänge

In Digitaler Form:

Trainierte Netze

Eingerichtete Trainings Umgebung

Hilfsskripte

Eingerichtete Validierungs Umgebung

Vorverarbeiteter Datensatz