Inhaltsverzeichnis

1 Einleitung

Hier eine kurze Einleitung in welchem Rahmen diese Arbeit entstanden ist und schonmal ganz kurz auf SETI eingehen.

2 SETI Breakthrough Listen

Die kaggle Challenge *SETI Breakthrough Listen - E.T. Signal Search* war ein öffentlicher Machine Learning Wettbewerb des *Berkeley SETI Research Centers* im Zeitraum vom 10. Mai 2021 bis 18. August 2021. Die zugrunde liegenden Daten sind noch verfügbar, sodass Interessierte sich nach wie vor mit diesem Problem beschäftigen können. Im folgenden werden wir die Challenge stets abgekürzt als *SETI* bezeichnen.

Die Herausforderung bei *SETI* besteht darin, Spektrogramme, also eine bildliche Darstellung eines Frequenzbereichs in einem bestimmten Zeitraum, die basierend auf Rohdaten des *Green Bank Telescopes* generiert worden sind, auf das Vorkommen von künstlich hinzugefügten extraterrestrischen Signalen zu untersuchen. Hierbei ist es wichtig, diese Signale von irdisches Signalen, wie etwa einem Radiosignal, zu unterscheiden. Um diese Unterscheidung vornehmen zu können, sind jeweils sechs Spektrogramme zusammengefasst, wobei die Spektrogramme eins, drei und fünf jeweils Aufnahmen des zu untersuchenden Ziels „A“ sind und die übrigen jeweils auf Aufnahmen eines anderen Himmelskörpers „B“, „C“ und „D“. Eine solche Gruppe von Spektrogrammen (ABACAD) wird bei *SETI* als *Kadenz-Ausschnitt*, im folgenden nur noch „Kadenz“, bezeichnet.

Abbildung 1 zeigt ein Beispiel für eine Kadenz mit einem extraterrestrischen Signal. Auf den drei Spektrogrammen, die auf das Ziel gerichtet sind (in der Abbildung durch „ON“ gekennzeichnet) ist jeweils ein Signal zu erkennen, welches nicht auf den anderen Spektrogrammen zu sehen ist. Offensichtlich muss ein Signal nicht auf jedem der drei *on target* Spektrogrammen zu sehen sein, da ein Signal nicht zwingend über den gesamten zeitlichen Betrachtungsraum aktiv sein muss.

Die Trainingsdaten für *SETI* enthalten 60.000 Kadenzen (*Heuhaufen*) von denen 6.000 *Nadeln* sind, also Kadenzen, die ein künstlich eingefügtes extraterrestrisches Signal enthalten. Einige dieser Signale sind bei entsprechender Visualisierung sofort mit bloßem Auge zu erkennen, andere sind in, durch irdische Signale verursachten, Rauschen versteckt. Weil die Challenge zum Zeitpunkt der Projektarbeit nicht mehr aktiv ist und die zu den Testdaten der Challenge keine Labels verfügbar sind, müssen wir die Trainingsdaten in Trainings- und Testdaten aufteilen, um ein Modell abschließend auf Testdaten laufen lassen zu können. Hierzu führen wir einen 70/30 Split der Trainingsdaten durch und erhalten danach ein Trainingsset mir 42000 Kadenzen und ein Testset mit 18000 Kadenzen. Dieser Split wird durch

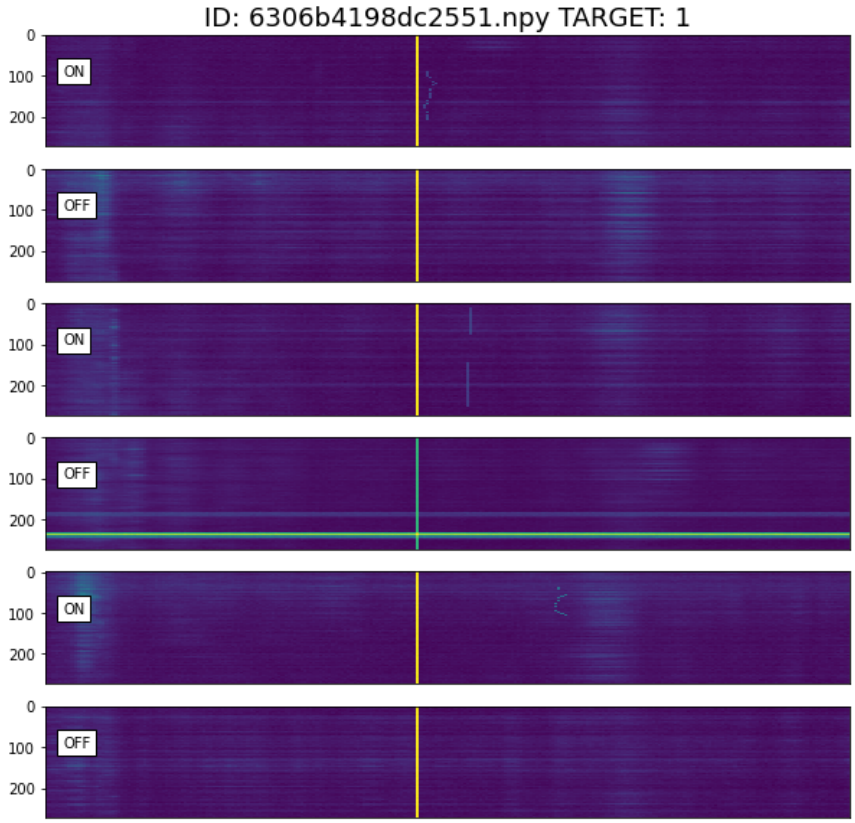


Abbildung 1: Beispiel für eine Kadenz mit Nadel

einen *Seed* reproduzierbar gemacht, um verschiedene Modelle vergleichen zu können, in dem sie auf dem jeweils selben Split der verfügbaren Daten trainiert und getestet werden.

3 Implementierung

Im folgenden beschäftigen wir uns nun mit der Problemlösung für *SETI*. Wir schauen uns erste Ansätze mit reiner Computer Vision an, die uns helfen sollen auch versteckte Signale extrahieren zu können, um sie mit bloßem Auge erkennen zu können. Wir haben uns für diesen Einstieg entschieden, um ein Gefühl für die visuelle Form der gesuchten Signale und für den Datensatz allgemein zu erhalten. Im darauf folgenden Kapitel werden wir uns mit *Convulutional Neural Networks*, kurz *CNNs*, beschäftigen.

3.1 Erste Ansätze mit Computer Vision

Wenn man über Bilderkennung spricht denkt man meistens direkt an Machine Learning und Neurale Netzwerke. Allerdings gibt es in der Bilderkennung Problematiken, die sich mithilfe der klassischen Computer Vision deutlich besser lösen lassen. Im Folgenden wird auf die grundlegenden Unterschiede zwischen der klassischen Computervision und Machine Learning eingegangen, einige Grundlagen beschrieben, Lösungen der Computervision angewandt auf die Problematik der Projektarbeit und ob die klassische Computervision für diese Projektarbeit geeignet ist.

3.1.1 Unterschiede Computervision - Machine Learning

Computervision ist nicht gleichzusetzen mit Machine Learning. Die klassische Computervision arbeitet mit rein mathematischen Ansätzen und Algorithmen, während das klassische Machine Learning eher darauf bedacht ist mithilfe von Trainingsdaten Modelle zu trainieren und dadurch Parameteranpassungen durchzuführen. Computervision ist sehr vielfältig einsetzbar. Mithilfe von Algorithmen können beispielsweise alle möglichen geometrische Primitive detektiert werden oder auch mithilfe von sogenannten Filtern das Bild entsprechend der Filter verarbeitet werden, weswegen anfangs dieser Projektarbeit mit einigen Ansätzen der Computervision experimentiert wurde. Diese Filter werden außerdem in den nachfolgenden Kapiteln noch sehr interessant für die Convolutional Neural Networks.

3.1.2 Grundlagen

Bilder sind im Prinzip nichts weiter als eine große Ansammlung von Zahlen, welche die entsprechenden Farben repräsentieren. In der klassischen Computervision arbeitet man häufig mit sogenannten Grauwertbildern. Diese sind besonders einfach zu handhaben, da diese nur einen Farbkanal besitzen im Gegensatz zu RGB Farbbildern, welche 3 Farbkanäle besitzen.

3.1.3 Lokale Operatoren: Filter

Lokale Operatoren sind sogenannte Punktoperationen, also Operationen die auf jedem Pixel des Bildes angewandt werden. Bei Lokalen Operatoren ist die Besonderheit, dass benachbarte Pixel auch mit in die Berechnung der Operation einfließen und somit auch den Farbwert des betrachteten Pixels beeinflussen. Somit können zum Beispiel Grauwertübergänge, also Übergänge von dunkel nach hell, detektiert werden. Um diese besonderen Punktoperationen besser zu verstehen, muss man das Bild als eine drei-dimensionale Funktion betrachten: die x Achse steht für die Breite des Bildes, y ist die Höhe des Bildes und z die Farbe des entsprechenden Pixels an der Stelle (x,y), so erhält man eine Grauwertfunktion g(x,y)=z. An den Stellen wo

es einen schnellen Wechsel von dunkel zu hell gibt, hat die Ableitung der Grauwertfunktion einen Extrempunkt:

$$g'(x,y)=[\frac{\partial g(x,y)}{\partial x}, \frac{\partial g(x,y)}{\partial y}]^T$$

Für diskrete Bilder muss die Ableitung allerdings angenähert werden, da ein Pixel nur einen einzigen Wert beinhaltet:

$$\frac{\partial g(x,y)}{\partial x} = \frac{g(x+\Delta x,y)-g(x-\Delta x,y)}{\Delta x}$$

$$\frac{\partial g(x,y)}{\partial y} = \frac{g(x,y+\Delta y)-g(x,y-\Delta y)}{\Delta y}$$

Setzt man nun Δx beziehungsweise Δy gleich 1 erhält man folgende Approximationen:

$$\frac{\partial g(x,y)}{\partial x} = \frac{g(x+1,y)-g(x-1,y)}{1}$$

$$\frac{\partial g(x,y)}{\partial y} = \frac{g(x,y+1)-g(x,y-1)}{1}$$

Für jedes Pixel (x,y) auf welches diese Approximation angewandt wird, wird also folgendes berechnet:

$$1 \cdot g(x+1,y) + 0 \cdot g(x,y) - 1 \cdot g(x-1,y)$$

Wir erhalten also folgendes Operatorfenster:

1	0	-1

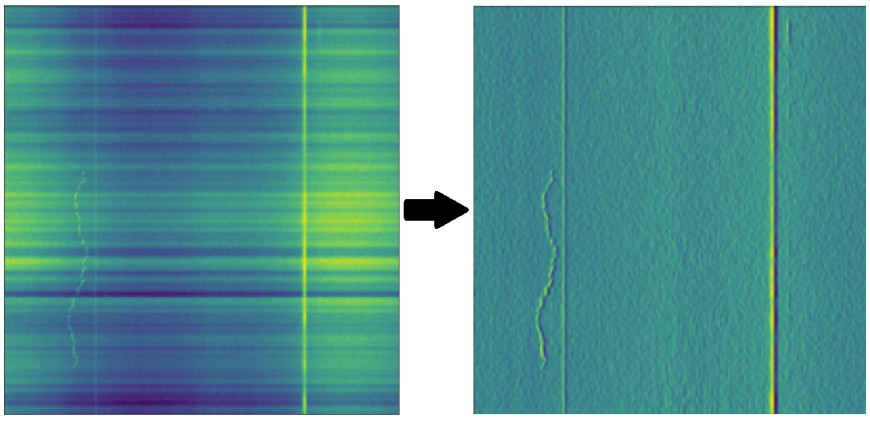
3x1 Operatorfenster

1	0	-1
1	0	-1
1	0	-1

3x3 Operatorfenster

Mit dem Operatorfenster kann auf jedem einzelnen Pixel eine sogenannte gewichtete Addition ausgeführt werden. Dafür legt man das Operatorfenster auf eine entsprechende Stelle des Bildes, multipliziert die einzelnen Komponenten des Fensters mit den darunterliegenden Werten der Pixel und summiert diese Produkte. Anstatt eines 3x1 Operatorfensters können auch 3x3 Operatorfenster genutzt werden, wodurch auch Pixel die über und unter dem zu betrachtenden Pixel mit in die Berechnung einfließen. Das Ergebnis der Anwendung des oben genannten Operatorfensters ist ein neues Bild welches insbesondere vertikale Grauwertübergänge hervorhebt und horizontale Grauwertübergänge entfernt. Der Grund dafür ist die Approximation der ersten Ableitung in x-Richtung, wodurch die Grauwertfunktion des Bildes abgeleitet wird. Dadurch kann man das resultierende Bilde sozusagen als Ableitung des Originalgrauwertbildes betrachten: Stellen die her-

vorgehoben werden, sind starke Grauwertübergänge und horizontale Linien werden entfernt, da auf einer gleichfarbigen horizontalen Linie keine Grauwertübergänge existieren. Dasselbe gilt umgekehrt für die Approximation in y-Richtung - dafür muss das Operatorfenster einfach nur um 90 Grad gedreht werden. Mit der Approximation in y-Richtung werden vertikale Linien entfernt.



Originalbild

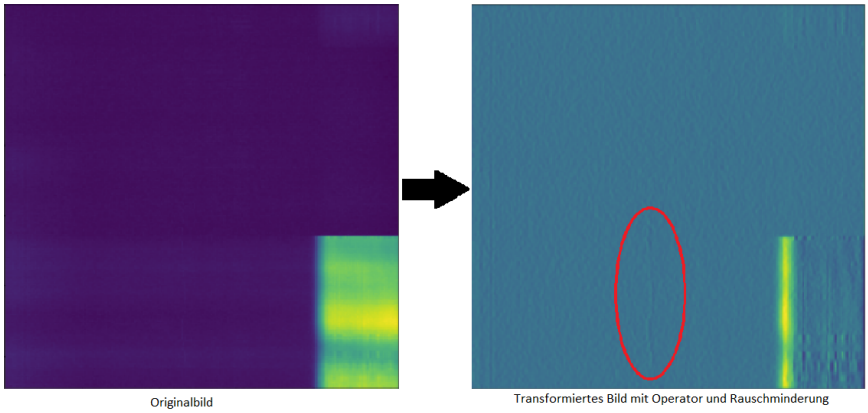
Transformiertes Bild mit Operator
(Approximation der ersten Ableitung in x-Richtung)

Die Idee hinter der Verwendung der Filter ist die, dass die Signale alle eine vertikale Richtung aufweisen. Mithilfe der Filter für die Eleminierung der horizontalen Linien kann also Rauschen recht einfach rausgefiltert werden. TODO: Referenz auf Laubenheimer Folien

3.1.4 Weitere Experimente

Das Rauschen in den Bildern ist häufig sehr ausgeprägt, weshalb Algorithmen zur Rauschminderung zum Einsatz kamen. Die Idee hinter dem ersten Algorithmus war ähnliche Pixel auf dieselbe Farbe abzubilden. Anfangs wird die Anzahl der unterschiedlichen Farben gezählt und anschließend die mittlere Differenz der Farbwerte berechnet. Anhand der mittleren Differenz und ein als Parameter festgelegter Wert, der bestimmt wie stark die mittlere Differenz den Algorithmus beeinflusst, kann ein neuer Wert berechnet werden, der bestimmt wie unterschiedlich ähnliche Pixel zueinander sein dürfen, sodass sie auf dieselbe Farbe abgebildet werden.
Nachdem nun dieser Algorithmus zur Abbildung ähnlicher Pixel auf dieselbe Farbe angewandt wurde, kann es sein dass die unterschiedlichen Farben teilweise einen geringeren, aber auch einen höheren Abstand zum jeweils nächsten Farbwert haben. Die Lösung dafür ist, dass genau diese Differenz auf dieselbe mittlere Differenz gesetzt wird. Dadurch erreicht man, dass die Farben, die bislang noch sehr nah beieinander lagen nach Anwendung dieses Algorithmus besser zu unterscheiden sind und Farben, die bislang eher weiter weg voneinander lagen, immer noch gut voneinander unterscheidbar

sind. Anfangs wird wieder die Anzahl der unterschiedlichen Farben gezählt und auch die mittlere Differenz zwischen den einzelnen Farben berechnet. Die Farben sollen nach Anwendung des Algorithmus genau diese mittlere Differenz zueinander haben, dies geschieht indem die einzelnen Farben nacheinander auf jeweils ein Vielfaches der mittleren Farbdifferenz gesetzt werden.
Führt man vor diesen beiden Algorithmen noch den lokalen Operator zur Entfernung der horizontalen Linien aus erhält man beispielsweise folgende Transformation:



Originalbild

Transformiertes Bild mit Operator und Rauschminderung

Auf dem Originalbild ist mit dem bloßen Auge kein Signal zu finden, betrachtet man allerdings das transformierte Bild ist ein schwaches Signal durchaus zu erkennen.

3.1.5 Zwischenfazit: Computervision alleine bringt keinen Erfolg

Computervision ist zwar sehr vielfältig einsetzbar, allerdings für die Problematik dieser Projektarbeit nur sehr schwer anwendbar. Das liegt insbesondere daran, dass die Signale teilweise im Hintergrundrauschen sehr gut versteckt sind und teilweise nicht mal mit dem Auge zu erkennen sind. Mit reiner Algorithmik und klassischen Computervisionansätzen, wie Liniendetektion sind die Signale fast nicht zu entdecken, da einerseits die Signale nur sehr schwer vom Hintergrund separierbar sind und andererseits viele der Signale keine geraden Linien sind, sondern alle möglichen undefinierbaren Formen annehmen können. Allerdings könnten die oben beschriebenen Vorgehensweisen mit den Filtern und Algorithmen zur Rauschunterdrückung genutzt werden um ein Modelltraining mit einem Convolutional Neural Network zu unterstützen, aber dazu mehr in den nachfolgenden Kapiteln.

3.1.6 Computervision verwenden als Vorverarbeitung für das Modelltraining

TODO: Bilder vorverarbeiten mit Sobelfilter (vertikale Linien entfernen) und danach Modell trainieren (10 Epoch falls das nicht zu lange ist) mit vorverarbeiteten Bildern

4 Deep Learning

Natürlich wollen wir nicht alle Kadenzen einzeln manuell betrachten und entscheiden, ob sie eine Nadel enthalten oder nicht. Vielmehr wollen wir ein Machine Learning Model trainieren, das uns diese Arbeit abnimmt und Nadeln findet. Hierzu wollen wir ein *Convolutional Neural Network (CNN)* trainieren, das die Kadenzen in genau zwei Klassen einteilt: enthält eine Nadel oder enthält keine Nadel.

4.1 Convolutional Neural Networks

Um uns an den Umgang mit *CNNs* heranzutasten, entwerfen wir zunächst ein kleines Netz bestehend aus zwei *Hidden Layern* und einem *Fully Connected Layer*. Die beiden *Hidden Layer* bestehen aus je einem *Convolutional Layer* und einem *Pooling Layer*. Wir teilen die Trainingsdaten noch einmal mit einem Verhältnis von 90 zu 10 in Trainings- und Validierungsdaten.

Unsere erste Beobachtung ist, dass wir selbst mit einem so kleinen Netz bereits eine sehr hohe *Accuracy* erzielen und sich auch der *Loss* am Anfang deutlich verringert, bis er ein Plateau erreicht. Jedoch ist der *Recall* sehr schlecht. Diese Konstellation der Metriken lässt sich damit erklären, dass unser *CNN* nicht wirklich gelernt hat, wie die gesuchten Signale aussehen, sondern einfach die meisten Kadenzen der Klasse 0, also enthält keine Nadel, zuordnet. Aufgrund des sehr unbalanciertem Datensatzes, ist diese Klassifizierung offensichtlich sehr oft richtig und somit kommt es zu einem hohen *Accuracy Score*.

- Gewichte
- Loss Function
- Gradient
- Optimizer
- Traning und Validierung
- Splitten der Trainingsdaten
- Dataloaders
- Metriken (Accuracy, Precision, Recall, F1 Score, Roc Auc Score)

4.2 Transfer Learning

4.2.1 efficientnet

4.3 Imbalance

4.4 Scheduler

4.5 Folds

5 Tech Stack

Zusammenfassung der verwendeten Tools und Bibliotheken, die teilweise auch vorher im Text schon genannt wurden.

6 Fazit