Meteor Detektion mit Künstlicher Intelligenz / Machine Learning

Wilhelm Sicking

Abstract: In diesem Artikel möchte ich eine Methode vorstellen, die auf Basis von Künstlicher Intelligenz Meteorechos detektiert. Die Vorteile gegenüber einer konventionellen Objekterkennung sind bedeutend:
1) Die aktuelle Version kann Meteore der verschiedensten Formen erkennen und von Störungen und Starlink-Satelliten unterscheiden.
2) Echos, die kleiner als der Störpegel sind, können detektiert werden.
3) Weil es keinen Threshold gibt, werden mehr Echos als mit der konventionellen Methode erkannt.
4) Die Software erkennt auch Echos, die z.B. aufgrund des periodischen Umschaltens der GRAVES-Antennen unvollständig sind.
5) Ein neuronales Netzwerk lässt sich leicht erweitern oder an andere Spektrogramme oder neu auftretende Störungen anpassen.

1 Einleitung

Seit einiger Zeit entwickle ich Programme, die Meteorechos aufzeichnen und auswerten. Ursprünglich handelte es sich um eine Echtzeitversion, die auf einem Nvidia-Jetson-Nano-Computer lief. Vor einiger Zeit habe ich das Programm auf Windows 10 portiert und auf Postprozessing umgestellt (Ref. 1 und Ref. 2). Das Programm funktioniert gut, weist aber Mängel auf. Die traditionell programmierte Objekterkennung ist nie perfekt. Ihre Erkennung ist unflexibel und kompliziert, wenn eine hohe Genauigkeit gewünscht ist. Besonders wenn viele Signale im Plot sind, werden Echos übersehen. Die Daten müssen oft manuell überprüft werden. Starlink-Satelliten und Interferenzen nehmen ständig zu. Außerdem interessiere ich mich besonders für kleine Echos, um den In-Line-Peak genauer zu untersuchen. Um die Genauigkeit zu erhöhen, suchte ich nach einer Möglichkeit, den Umgang mit künstlicher Intelligenz (KI) / Machine Learning (ML) zu erlernen und stieß auf ein Tutorial und eine hervorragende Software: Das PixelLib von Ayoola Olafenwa (Ref. 3). Pixellib ist eine Programmbibliothek, die genau die Verfahren bereitstellt, die für die Objekterkennung mittels KI-Methoden benötigt werden. In diesem Artikel möchte ich zeigen, wie das Programmpaket funktioniert, wie es grundsätzlich installiert wird und welche Ergebnisse im Vergleich

zu meiner herkömmlichen Methode dabei herauskommen.

2 Setup

Zum Empfang der Meteorechos wird eine zirkular polarisierte 4-Element-Kreuz-Yagi-Antenne verwendet. Meine Antennen sind auf dem Dachboden montiert, sodass die Konfiguration leicht geändert werden kann. Direkt an die Antenne angeschlossen ist ein rauscharmer Vorverstärker mit einem Frequenzbereich von 140-150 MHz und einer Rauschzahl von 0,25 dB. Der Empfänger ist ein Icom IC-R8600. Als Aufnahmesoftware kommt Spectrum-Lab (SL) von Wolfgang Buescher (DL4YHF) zum Einsatz. SL generiert Diagramme in 20-Sekunden-Intervallen mit entsprechendem Datum und Uhrzeit im Dateinamen, die später mit der hier beschriebenen Software analysiert werden.

Zum Aufnehmen und Programmieren verwende ich Windows 11-Notebooks mit i3 oder i5-Prozessoren. Zum Trainieren des neuronalen Netzes nutze ich einen Windows 10 Gaming-PC, ebenfalls mit i5-Prozessor und einer GeForce RTX-3060 Grafikkarte (GPU). Die GPU verfügt über 3840 CUDA-Kerne. Diese GPU und die entsprechende Nvidia-Software führen zu einer deutlichen Beschleunigung, insbesondere beim Training des neuronalen Netzes. Unten im Anhang ist ein Benchtest dargestellt.

3 Setup des neuronalen Netzwerks

Pixellib ist im Internet gut dokumentiert. Daher werden hier nur die Dinge beschrieben, die für das Projekt wichtig sind.
Bevor man mit der Untersuchung von Objekten mit KI beginnen kann, muss ein neuronales Netzwerk für die zu erkennenden Objekte, ein *Modell*, erstellt werden. Typischerweise wird ein bereits vorhandenes Modell neu trainiert. Dieser Vorgang wird Transfer-Learning genannt. Die Pixellib-Autorin verwendete hierfür das Modell *mask_rcnn_coco*. Es erkennt so ziemlich alles, was uns täglich begegnet. Ein Beispiel für die Verwendung von mask_rcnn_coco ist in *Figure 2* dargestellt. In ihrem Pixellib-Tutorial hat Ayoola Olafenwa dieses Modell nun auf zwei neue Objekte umtrainiert, nämlich auf Schmetterlinge und Eichhörnchen. Diese beiden sogenannten Klassen kommen in mask_rcnn_coco nicht vor und eignen sich daher zur Demonstration von Transfer-Learning. Ich habe das

Tutorial befolgt und das Modell nun mit drei neuen Klassen neu trainiert. Meine entsprechende Zeile im Trainingsskript lautet:

segment_image.inferConfig(num_classes= 3, class_names= ["BG", "Artifical-Star", "Background", "Meteor"])

Die erste *BG* gehört zu Pixellib. Dann folgen meine drei Klassen. Die Klassen müssen in aufsteigender alphabetischer Reihenfolge vorliegen, sonst kommt es zum Durcheinander. Die Dateien in meinen Test- und Train-Ordnern beginnen ebenfalls mit A, B und der Rest ist für Meteore.

Labeln der Daten für die drei Klassen

Die wesentliche Arbeit besteht nun darin, Bilder der drei Klassen Artifical Stars, Background und Meteor zu sammeln und für das Training des Modells vorzubereiten. Hier sind es bis jetzt über 600 Plots die einzeln von Hand gelabelt werden mussten. Dieser Vorgang ist in *Figure 1* für Meteore erklärt.

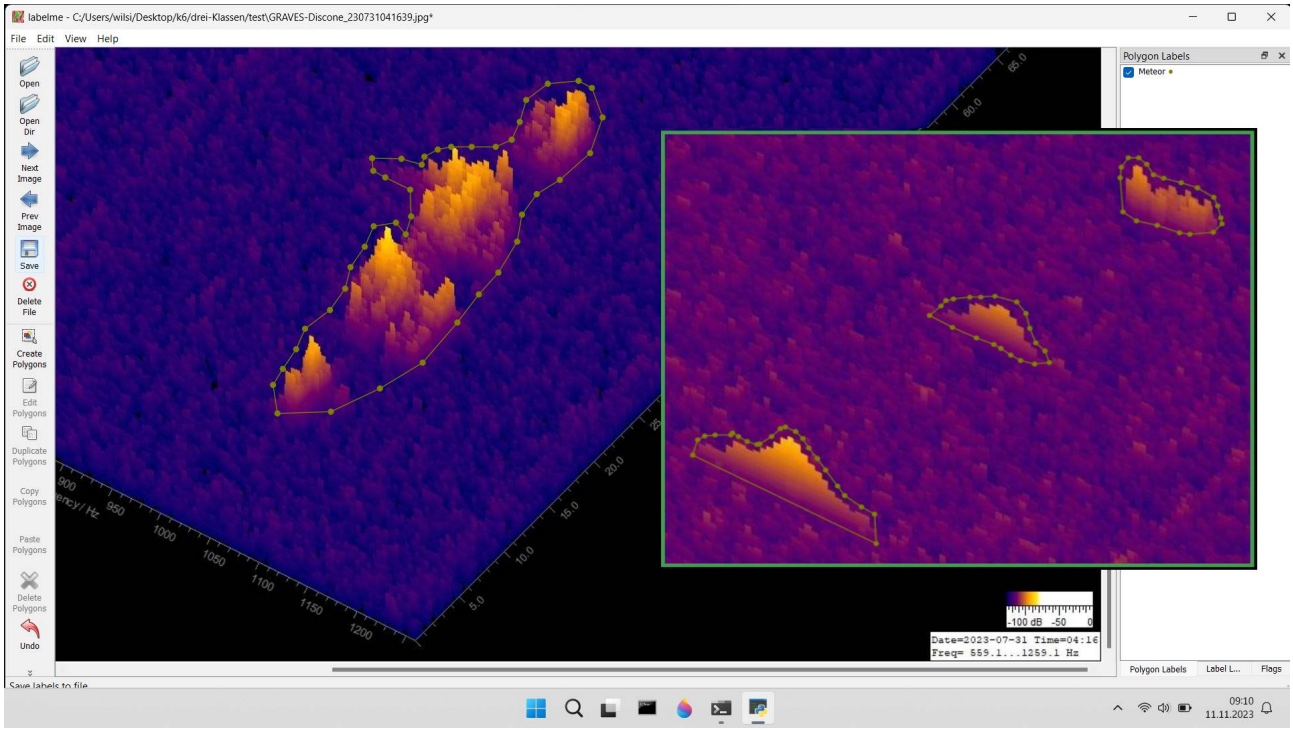

Figure 1 – Screenshot vom Programm Labelme. Der Inset zeigt, dass auch drei Echos zusammen in einem Plot gelabelt werden können. Unterschiedliche Klassen dürfen in einem Plot nicht vorkommen.

Das Labeln, also das Zeichnen der Umrandung und Benennung mit dem Klassennamen, hier *Meteor*, muss für jedes Signal durchgeführt werden.

Zu Demonstrationszwecken wurde dem Screenshot ein Einschub mit drei Echos hinzugefügt. Damit soll gezeigt werden, dass die 600 Plots deutlich mehr als nur 600 Objekte enthalten, vielleicht sogar drei- bis viermal so viele. Die Umrandung um die Fragmente des großen Echos, die durch das Umschalten der GRAVES-Antennen entstanden sind, zeigt dem Algorithmus, dass solche oder ähnliche Echos später als ein einzelner Meteor betrachtet werden müssen, wie wir ihn als Beobachter auch sehen. Die für jeden Plot generierten Labeldaten werden dann in einer .JSON-Datei gespeichert. Abschließend werden die Dateien in zwei Ordner aufgeteilt: Ein Drittel jeder Klasse wird in einen Ordner namens TEST kopiert, zwei Drittel werden in den TRAIN-Ordner kopiert. Dieser sogenannte Test-Train-Split wird verwendet, damit der Algorithmus sein Training mit Daten überprüfen kann, die nicht im Training verwendet werden. Das Schlüsselwort hierfür ist Backpropagation.
Ein paar Details zum eigentlichen Training und zu den Rechnungszeiten finden Sie im Kapitel Benchtest im Anhang.

Figure 2 – Mit dem Netzwerk *mask_rcnn_coco* werden zwei Pferde erkannt. Nur ein sehr kurzer Code wird dazu benötigt. (Das Foto wurde vom Autor aufgenommen.)

Quelle:https://pixellib.readthedocs.io/en/latest/Image_instance.html

Figure 2 verdeutlicht noch mal, wie geteilte / unterbrochene Objekte behandelt werden: Mit dem Netzwerk *mask_rcnn_coco* werden zwei Pferde analysiert, die durch den Lattenzaun *geteilt* sind. Selbstverständlich werden die Pferde als Ganzes detektiert, so wie wir sie auch wahrnehmen. In *Figure 2* ist auch der kurze Pixellib-Code gezeigt, mit dem das Bild untersucht wurde.
Wie gut die Detektion eines weichen und in Bruchstücken vorliegenden Meteors funktioniert, ist in *Figure 3* gezeigt. Folgender kurzer Code würde das Echo in *Figure 3* detektieren:

```
import pixellib
from pixellib.instance import custom_segmentation

segment_image = custom_segmentation()
segment_image.inferConfig(num_classes= 3, class_names= ["BG", "Artifical-Star", "Background", "Meteor"])
segment_image.load_model("mask_rcnn_model.075-0.393305.h5")
segment_image.segmentImage("Meteor.jpg", show_bboxes=True, output_image_name="Output_Meteor.jpg")
```

In diesem Python-Skript werden die drei neuen Klassen Artifical-Star, Background und Meteor sowie das selbst erstellte neuronale Netzwerk mask_rcnn_model.075-0.393305.h5 verwendet.

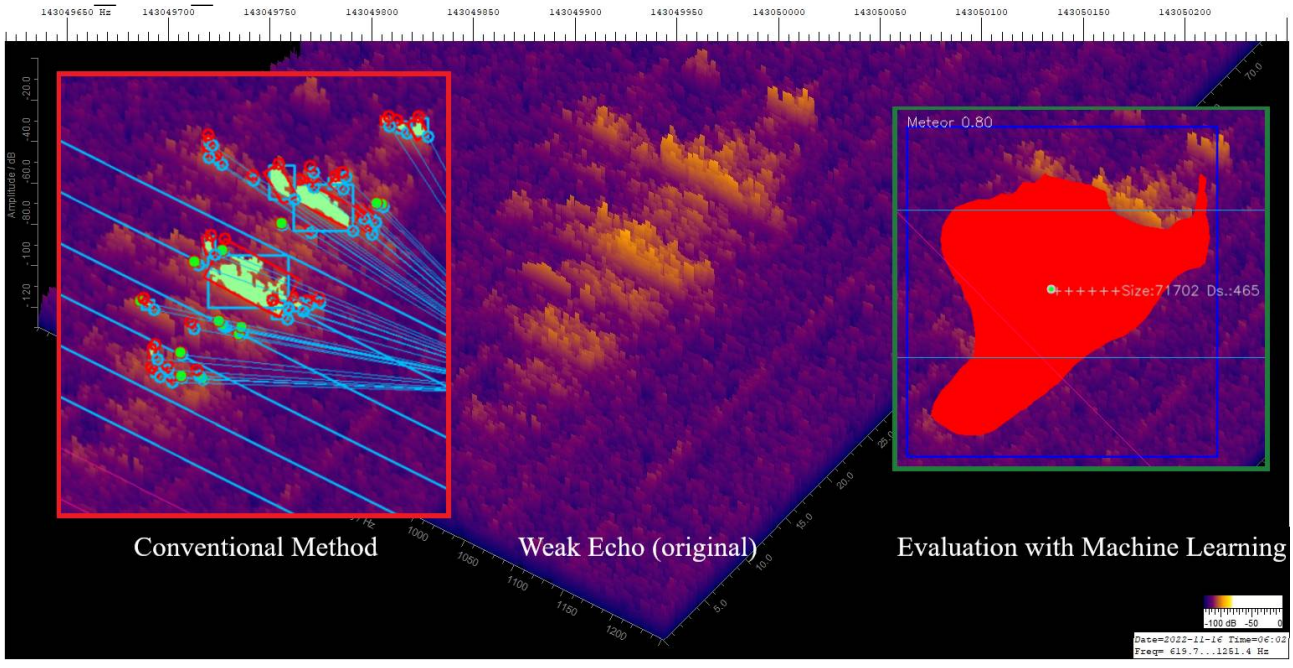

Figure 3 – Das Bild zeigt die Detektion eines sehr weichen Echos nach der konventionellen Methode verglichen mit der Machine Learning Methode. Die konventionelle Methode würde viele Echos loggen. So ein Echo hätte ich von Hand bearbeiten müssen. Im Anhang ist ein weiteres Bild (*Figure A13*) gezeigt.

Arifical-Stars

Starlink Satelliten werden in kurzen Zeitabständen in den Weltraum geschossen, so dass sie auch zunehmend im GRAVES Radar auftauchen. Daher wurde eine eigene Klasse namens Artifical-Stars erstellt. Meteore und Satelliten lassen sich mit ML sehr gut unterscheiden, da die metallene Oberfläche eine kammartige Struktur des Echos produziert, siehe *Figure 4*.

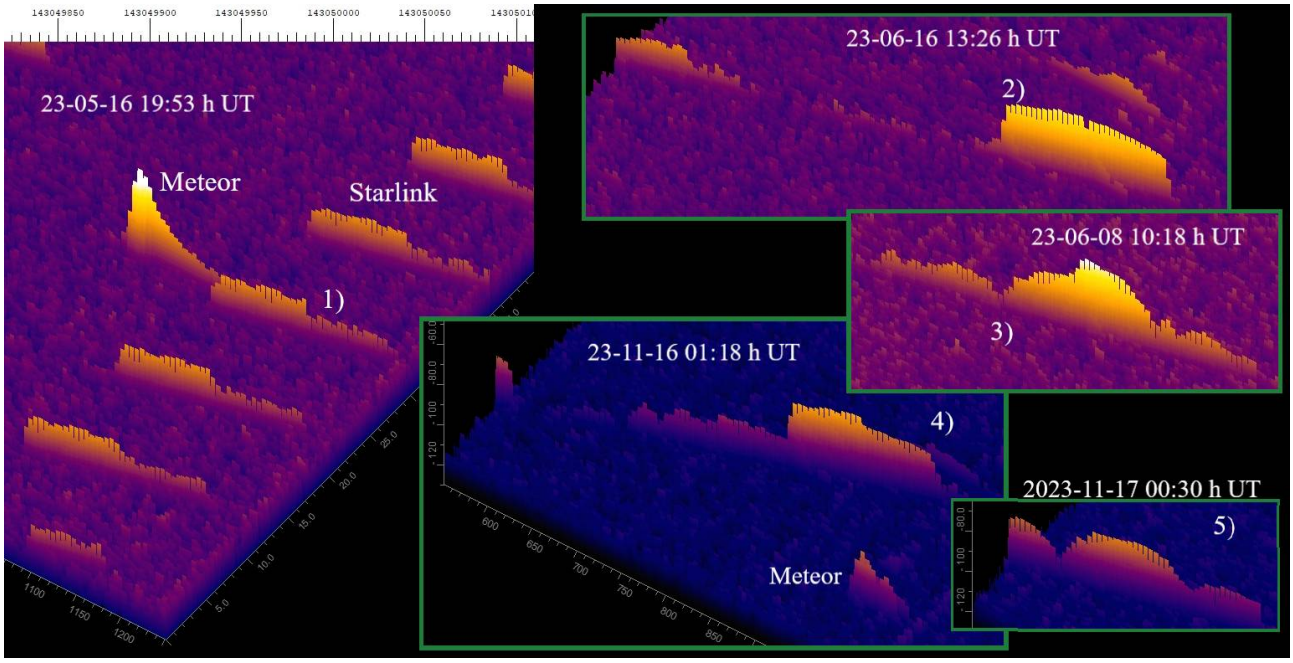

Figure 4 – Links im Bild sind ein Meteor und einige Starlink Satelliten abgebildet. Die Treppenstufen an den Echos (mit 1 markiert) stammen vom Richtungswechsel der Sendekeulen. Ferner sind rechts in den übrigen Abbildungen 4 Echos gezeigt, die vermutlich von größeren Raumschiffen stammen, siehe Text.

Die Starlink Satelliten liefern im Prinzip ein Echo mit einer geraden Oberfläche / Hüllkurve. Die Stufe rechts an den Echos (mit 1 bezeichnet) entsteht durch den Umschaltvorgang der GRAVES Antennen. Die restlichen Satelliten-Echos (2 - 5) zeigen eine mehr oder weniger gebogenen Hüllkurve, was auf eine komplexere Oberfläche hindeutet. Die Aufnahme der Starlink-Satelliten (links im Bild) zeigt sie kurz nach dem Aussetzen im ungefähr 200 km Höhe. Das würde den geringen

Größenunterschied zu den anderen Objekten erklären, wenn es größere Raumstationen wären. Die ISS zum Beispiel fliegt in 400 km Höhe.

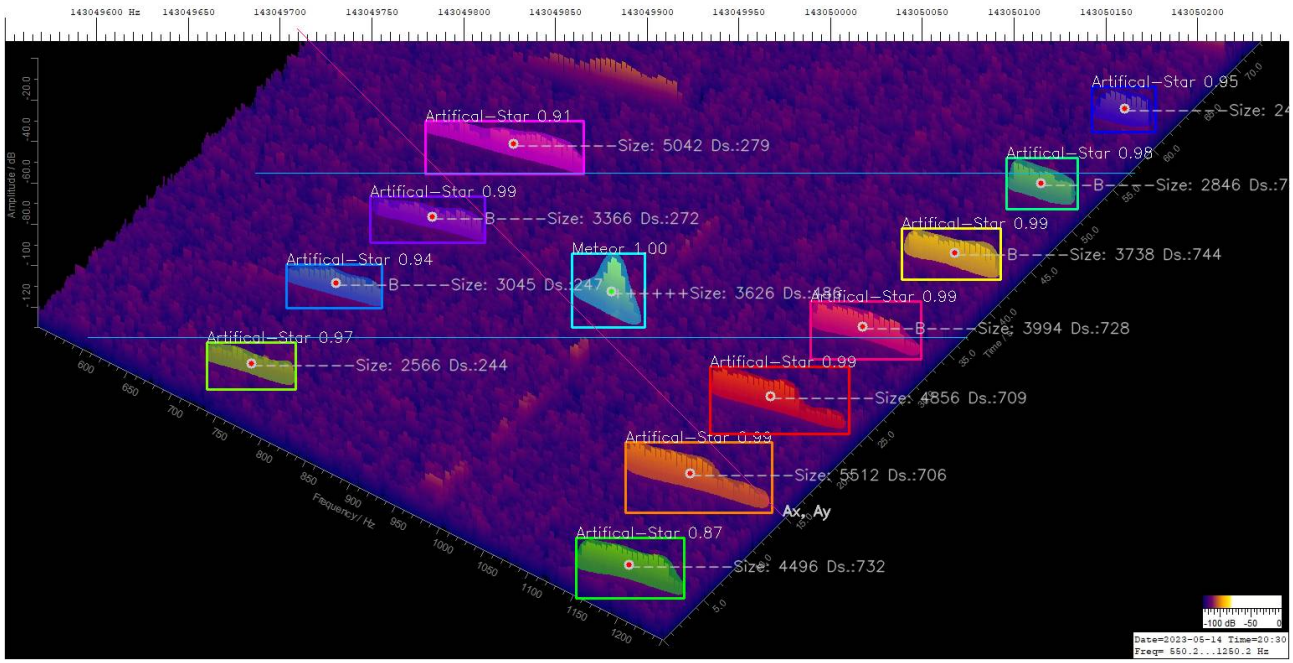

Figure 5 – Ein Meteor inmitten von Starlink Satelliten wurde mit ML detektiert. Die Echos der rechten und linken Seite gehören zum gleichen Satelliten. Durch das Umschalten der GRAVES Antennen werden sie jedoch nicht immer belichtet. Geloggt werden ein Meteor und fünf Satelliten. Das sind in diesem Fall zwei zu viel. Jedoch gibt es solche Ansammlungen eher selten.

Background

Um Interferenzen zu detektieren, wurde eine eigene Klasse eingerichtet. Es werden diverse Knackimpuse und ein Störträger gelabelt.
In einer früheren Version hatte ich auch die Streifen / Träger gelabelt, die durch Schaltnetzteile in LED-Birnen usw. verursacht werden, siehe *Figure 6.* Jedoch nicht alle Arten von Störungen müssen explizit gelabelt werden. Durch das Training mit Bildern, die sowohl gelabelte Echos und nicht gelabelte Störungen enthalten, lernt das Modell, dass diese Signale nicht zum Echo gehören. Es fließt natürlich auch Vorwissen durch das Transfer Learning ein. So kennt das Modell schon den immer wiederkehrenden Rahmen oder die Beschriftung. Bei der konventionellen Methode musste vor der Analyse alles weg-maskiert werden.

Figure 6 zeigt, dass die ML auch bei starken Interferenzen den Meteor detektiert.

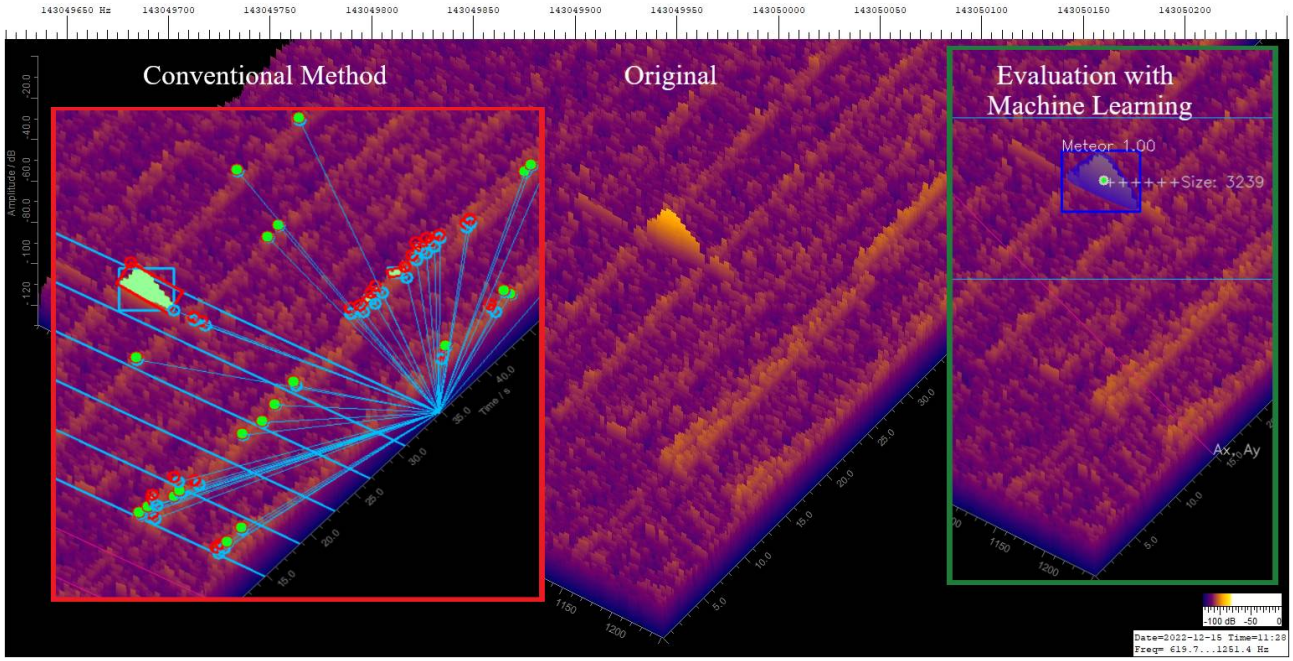

Figure 6 – Die ML-Methode kann selbst bei starken Interferenzen den Meteor detektieren. Bei der konventionellen Methode lösen die Störungen viele Trigger aus. Es war keine Auswertung mehr möglich. Mit der ML-Methode ist die Auswertung kein Problem. Im Anhang ist ein weiteres Bild (Figure A13) gezeigt, bei dem das Echo unterhalb des Störpegels liegt.

4 Die Auswertung

Die Auswertesoftware liest nacheinander alle durch Dateinamen und Wildcards adressierten Bilder, meist die eines ganzen Tages. Die oben gezeigten Pixellib-Routinen geben dann Informationen über die erkannten Objekte an das Auswerteprogramm zurück. Dies sind die Klassen, Polygone der Flächen, die Wahrscheinlichkeiten und die Box. Das Ergebnis kann im Debug-Modus angezeigt werden. Ein Beispiel ist in *Figure 7* dargestellt.

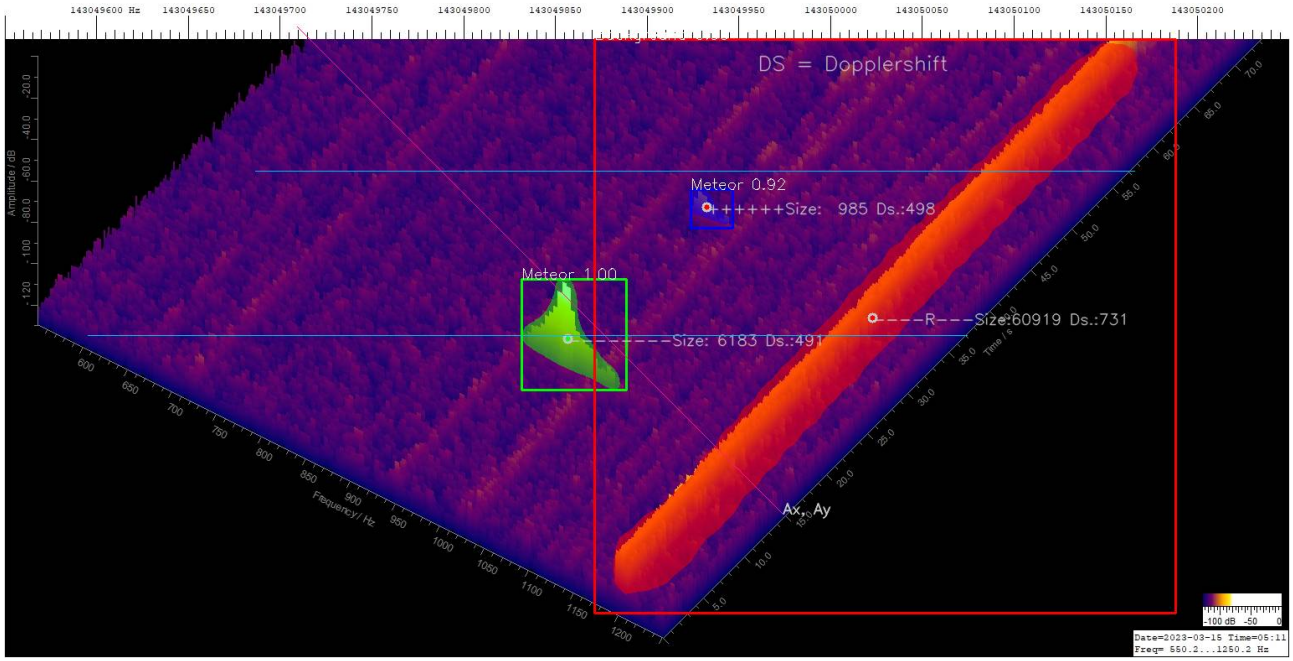

Figure 7 – Erkennung von zwei kleinen Meteoren und einer Störung. In diesem Diagramm wird nur das kleine, obere Echo geloggt, da es zwischen den beiden horizontalen blauen Linien liegt, also im 20-Sekunden-Fenster. Das untere Echo befindet sich mit dem nächsten Plot im Auswertefenster.

Figure 7 enthält zwei Echos und eine Störung, die gelegentlich und glücklicherweise nur am rechten Rand auftritt. Ich habe diesen Plot gewählt, um zu zeigen, dass solche Störungen zuverlässig erkannt werden: Das obere, kleine Echo liegt unterhalb des Pegels des Störsignals.
Diese Art der Untersuchung wird als Instanzsegmentierung bezeichnet, da mehrere Instanzen (Meteore) in einer Klasse unterschieden werden.
Die Gesamtaufzeichnungszeit pro Plot beträgt etwa eine Minute. Da die Plots jedoch alle 20 Sekunden gespeichert werden, darf nur der Inhalt eines 20-Sekunden-Zeitfensters protokolliert werden. Aufgrund der langen Aufnahmezeit werden auch große Echos, die deutlich länger als 20 Sekunden andauern, noch korrekt erfasst. Dieser Sachverhalt ist am großen Meteor in *Figure 3* zu erkennen. Im Fall von *Figure 7* wird nur der obere, kleine Meteor geloggt. Die Auswertesoftware ermittelt die Fläche und aus dem Schwerpunkt der Fläche wird eine relative Dopplerverschiebung berechnet. Der untere Meteor befindet sich beim nächsten Plot im Auswertungsfenster. Nachdem alle Plots untersucht wurden, wird eine Übersicht erstellt, siehe *Figure 8*.

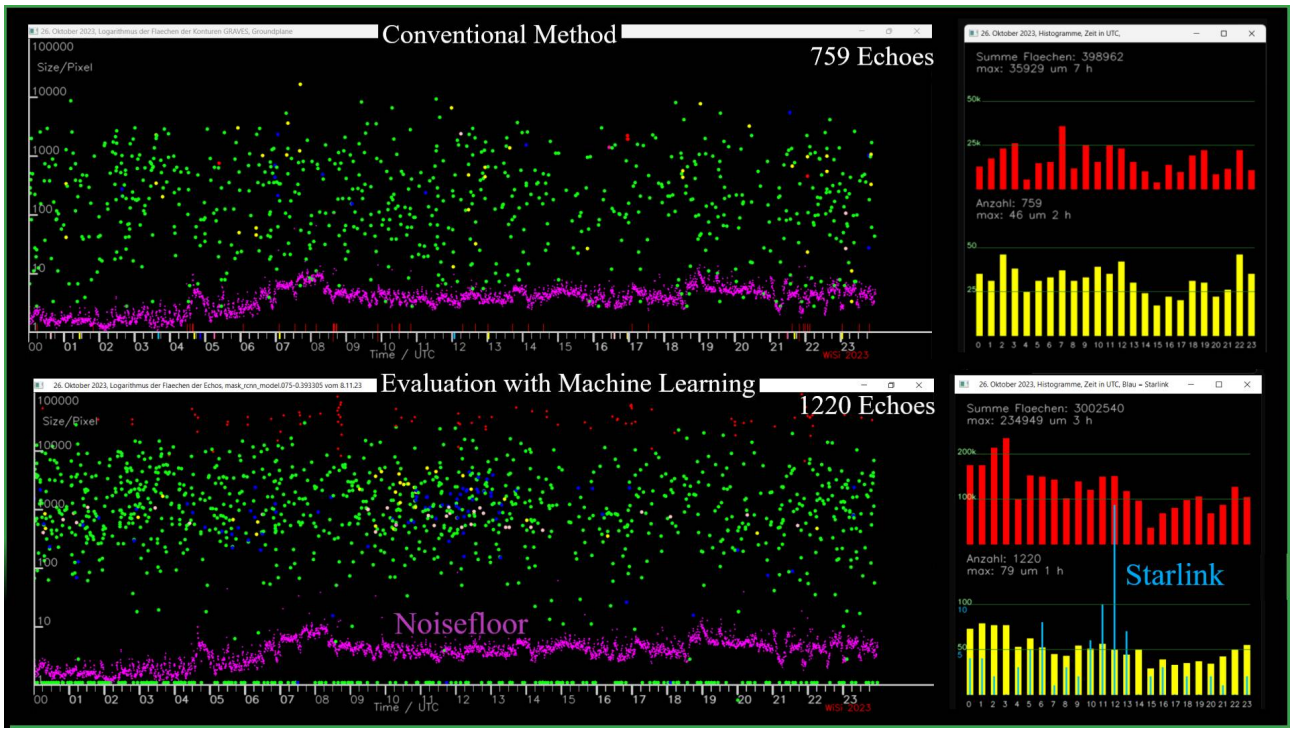

Figure 8 – Gemessene Meteorgrößen als Funktion der Zeit, aufgezeichnet am 26. Oktober 2023. Jeder grüne Punkt stellt ein Echo dar. Das untere Diagramm zeigt, dass mit der KI-Methode 1220 Echos aufgezeichnet wurden. Die blauen Punkte und das blaue Histogramm stellen die Starlink-Satelliten dar. Die roten Punkte sind die geloggten Störungen. Zum Vergleich ist im oberen Diagramm auch die Auswertung nach der konventionellen Methode dargestellt. Hier werden nur 759 Echos protokolliert, siehe Text. Der Noisefloor wird bestimmt, indem eine kleine Fläche Hintergrund aufsummiert wird.

Die Punkte in der unterer Darstellung in *Figure 8* stellen die mit ML gemessenen Echogrößen als Funktion der Zeit beispielhaft für den 26. Oktober 2023 dar. Etwa fünf Größenordnungen der Echogrößen werden aufgelöst. Es wurden 1220 Echos gezählt. Die gelben Histogramme zeigen die Rate und die roten Histogramme zeigen die Rate gewichtet mit der Größe der Meteorechos. Die blauen Punkte und die blauen Linien repräsentieren die Starlink Satelliten.

Zum Threshold

In *Figure 8* ist zu Vergleichszwecken auch die Auswertung nach der konventionellen Methode gezeigt. Mit der ML-Methode werden deutlich mehr Echos erkannt. Der Grund dafür wird nun anhand von *Figure 9* untersucht: Nach der konventionellen Methode werden zwei Echos erkannt. Das dritte mittlere Echo liegt unterhalb des Threshold. Mit der

ML-Methode werden alle drei Echos erkannt, weil es keinen Threshold gibt. Die Abwesenheit des Threshold bewirkt auch, dass die Echos nach der ML-Methode etwa um den Faktor 10 größer sind. (Es sind 10 mal mehr Pixel im Polygon.) Der Grund dafür ist, dass die konventionelle Methode nur die Spitzen der Echos detektiert, die oberhalb des Threshold liegen. Die ML-Methode detektiert auch den Teil des Echos, der im Untergrund bzw. im Rauschen liegt. Dies ist einer der großen Vorteile der Objekterkennung mit ML.

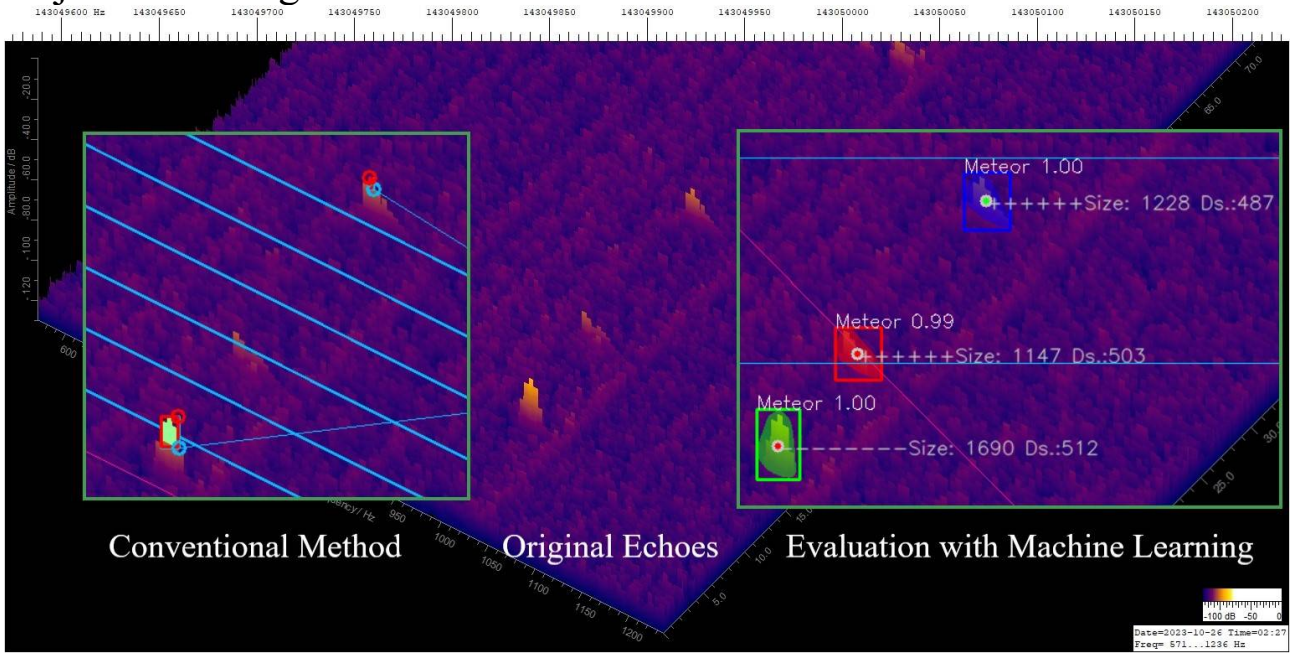

Figure 9 – Bei der herkömmlichen Methode muss die Schwelle so hoch sein, dass Störungen die Aufzeichnung nicht auslösen. Die ML-Methode detektiert auch schwache Signale.

5 Zusammenfassung und Ausblick

Die sogenannte KI, besser ML, ist keine Modeerscheinung oder etwas Mystisches, sondern ein Tool, das sehr gute Ergebnisse liefern kann. Mein Interesse gilt vor allem den sehr kleinen Echos, um den In-Line-Peak weiterhin zu untersuchen, siehe *Figure 10*, den roten Punkt. Die Partikelsortierung in Strömen auf Grund des Poynting-Robertson-Effekt möchte ich auch untersuchen. Dazu müssen vor allem kleine Echos detektiert werden. Die hier gezeigte Methode soll dabei helfen.

Wegen der Störungen in meinem Wohngebiet teste ich in Zukunft einen neuen Standort. Es ist das Clubheim des Ortsverbandes N62 Wüllen, siehe *Figure 11*. Im Moment ist es aber zu kalt und zu dunkel, so dass es

realistisch betrachtet, wohl erst bei den Perseiden richtig los geht.

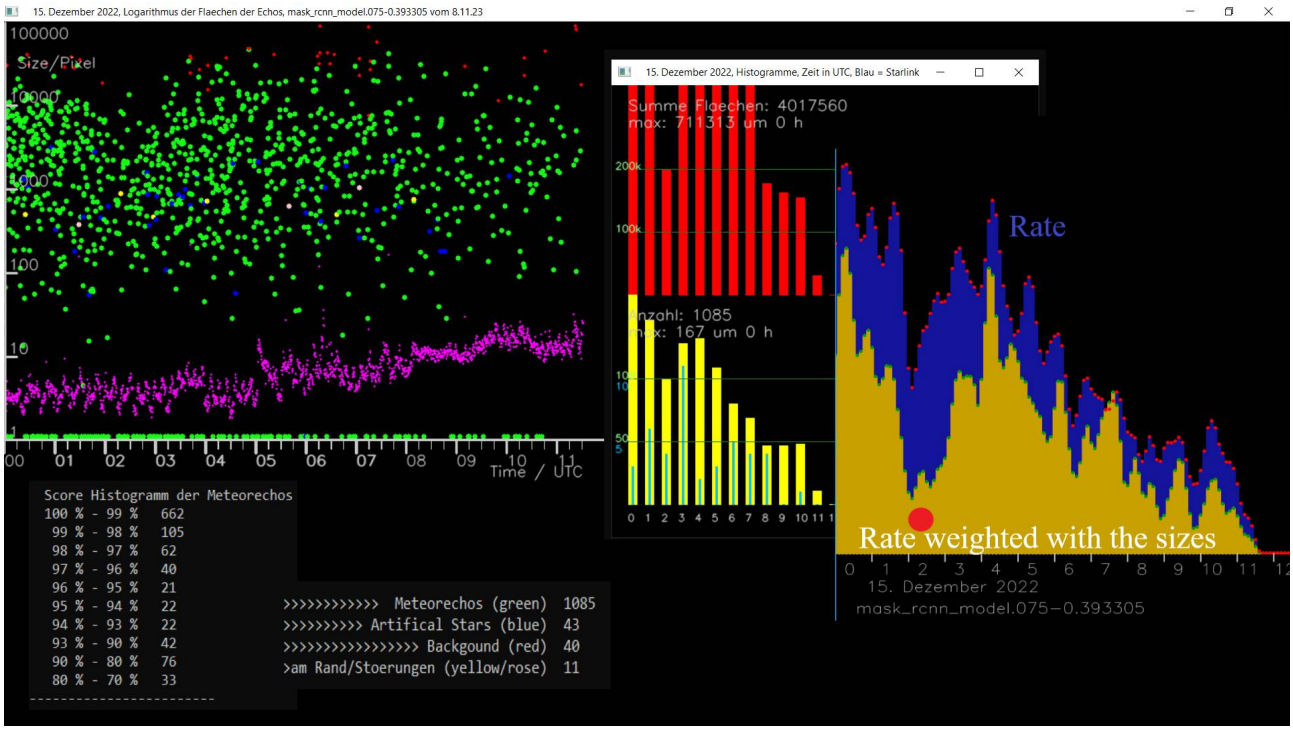

Figure 10 – Aufnahme von 15. Dezember 2022. Wegen der Störungen hatte ich die Aufnahme abgebrochen. Ein Inset zeigt deutlich den In-Line-Peak, siehe den roten Punkt.

Figure 11 – Foto vom Clubheim OV N62. Es ist ein ehemaliger NATO-Funkturm.

Das Projekt wird ständig weiterentwickelt.

Die Programme, die verwendeten Labeldaten und das Modell gebe ich gerne weiter, wenn sich jemand damit beschäftigen möchte.

Acknowledgment
Vielen Dank an Kerstin fürs gemeinsame Lesen.

References

Sicking, W. (2022). "A Notch in the Arietids Radio Data and a new so called In-Line-Effect". EMetN, 7:5, 331-335.

Sicking, W. (2022). "Radio observations on the Perseids and some other showers in August and September 2022". EMetN, 7:6, 407-410.

Ayoola Olafenwa
https://github.com/ayoolaolafenwa/PixelLib

Appendix (Installation. Benchtest und drei weitere Bilder.)

Installation Support Die wichtigen Pakete sind Python3, Pixellib und die CUDA Software für die RTX-3060 GPU.
Da Pixellib seit zwei Jahren nicht mehr gewartet wird, ist die Installation unter Windows 11 etwas trickreich.
Zunächst muss man Python 3.9.7 mit entsprechenden Abhängigkeiten installieren.
Tensorflow 2.5 ist die höchste funktionierende Version mit Pixellib.
Tensorflow 2.5 installiert eine ältere Numpy-Version, mit der ein Modul (scikit-image) in der aktuellen Version nicht kompatibel ist.
Ein Problem unter Windows11 ist auch ein Bug im Programm

labelme2coco. Mit der allerersten Version klappt es.

Diese Versionen und Reihenfolge funktionieren unter Windows 11 beim Autor:

```
pip3 install scikit-image==0.18.3
pip3 install numpy==1.19.5
pip3 install tensorflow==2.5.0
pip3 install imgaug ..............................
(ist 0.4.0)
pip3 install pixellib --upgrade
pip3 install labelme2coco==0.1.0
```

So kann man sich die installierten Versionen anzeigen lassen:

```
C:\Users\wilsi>python
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021,
20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for
more information.
>>> import numpy
>>> print (numpy.__version__)
1.19.5
>>>
```

Eine Ausnahme ist scikit-image. Das importiert man mit:

```
>>> import skimage
>>> print (skimage.__version__)
0.18.3
>>>
```

Auf dem Windows 10 Gaming-PC lief die Installation mit den Default-Einstellungen.
Wenn man CUDA Code für Nvidia GPUs installieren will, muss man die Compute Capability der GPU kennen. Meine Grafikkarte, die GeForce 3060 Laptop GPU, hat einen Compute Capability von 8.6.
Daher wurden *cuDNN 8.6 for CUDA 11.2* und das *Cuda Toolkit 11.2* installiert.
Damit rechnet Tensorflow parallel.

Benchtest zum Vergleich des Gaming-PC mit der Nvidia 3060 GPU und einem normalen i5-PC:

Zum Trainieren dient ein kurzes Script aus dem Tutorial:
https://pixellib.readthedocs.io/en/latest/custom_train.html

Für das Trainieren des aktuellen Modells habe ich 200 Epochen berechnet. Die gesamte Rechenzeit habe ich mir nicht gemerkt, aber die letzte gespeicherte Epoche war Epoche 92. Bis dahin hat es auf dem Gaming-PC eine Stunde und 20 Minuten gedauert. Eine Epoche dauert also knapp eine Minute.

Für die Analysen in diesem Paper wurde aber nicht das Modell aus Epoche 92 genommen, sondern das Modell aus Epoche 75. (mask_rcnn_model.0**75**-0.393305.h5), um Overfitting zu verhindern.

Auf dem i5 PC habe ich zu Vergleichszwecken nur eine Epoche gerechnet. Sie dauert 37 Minuten. Die 92 Epochen des Gaming-PC würden also 56.7 Stunden, also 2 Tage und fast 9 Stunden dauern. Dabei arbeitet der i5 auch mit seinen 4 Cores parallel: Der Load betrug bis 100%.

Die Analyse der 4320 Bilder von den 24 Stunden des 26.10.2023 hat auf dem Gaming-PC 17 Minuten gedauert.

Um die Analysezeiten zu vergleichen, habe ich auf beiden Rechnern nur jeweils eine Stunde Daten untersucht: Es dauert 50 Sekunden auf dem Gaming-PC und 8 Minuten auf dem normalen PC, was über 3 Stunden für die Analyse eines Tages bedeuten würde.

Beim Analysieren liegt der GPU-Load des Gaming-PC im Bereich von 30 %, während der GPU Load beim Trainieren ca. 80 % beträgt. Die Analyse profitiert also nicht so stark von der GPU, da ständig Bilder geladen werden müssen.

Dieser Vergleich sollte zeigen, dass man ohne GPU nur schwer Machine Learning betreiben kann. Es sind ja nicht nur einmalig die 2 Tage und 8 h, sondern es sind viele Tests nötig. Oft sehe ich nach 3 - 4 Epochen, wohin

die Reise geht. Das sind dann ein paar Minuten Rechenzeit. Auf dem PC wären es dann aber schon 2 Stunden.

Ein paar weitere Bilder

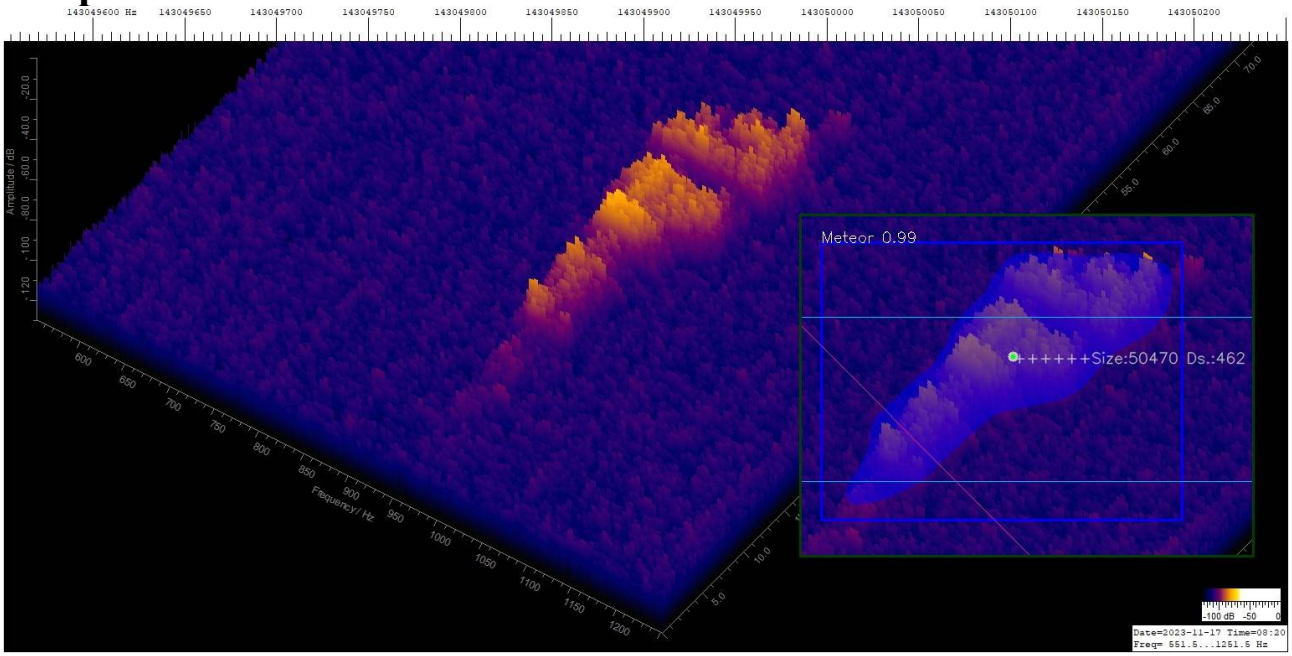

Figure A12 – Ein weiteres weiches Echo mit Auswertung.

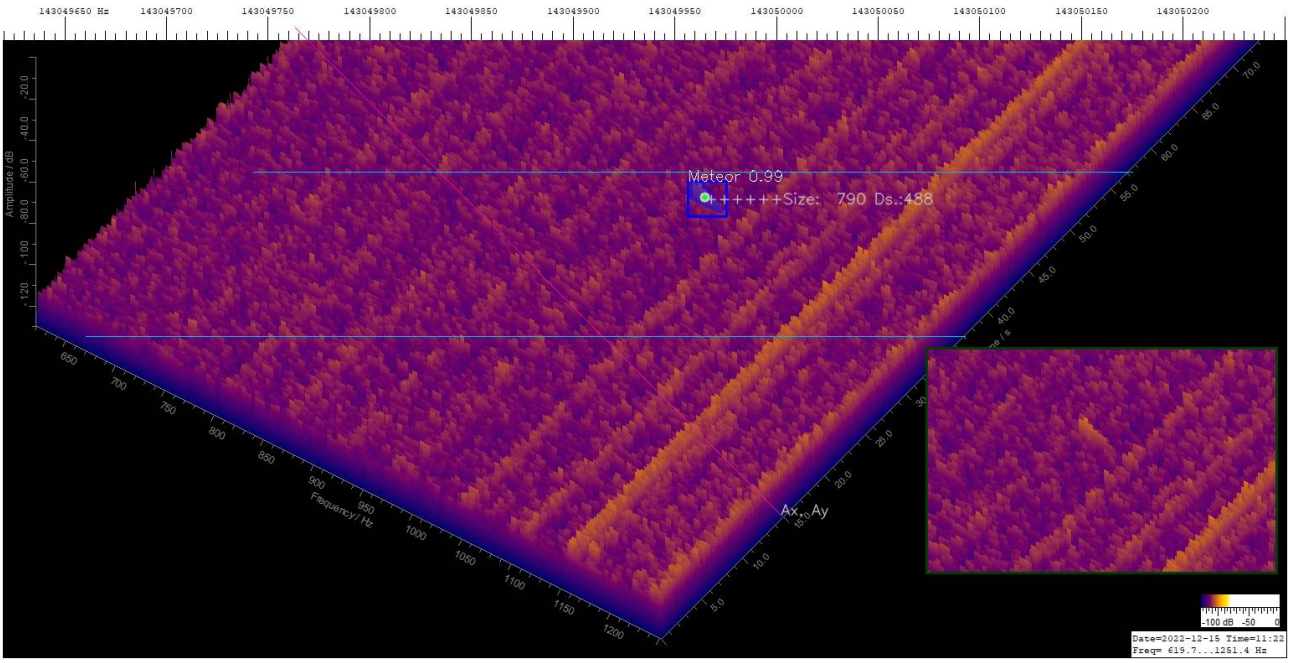

Figure A13 – Die ML Methode kann selbst bei starken Interferenzen den sehr kleinen Meteor detektieren.

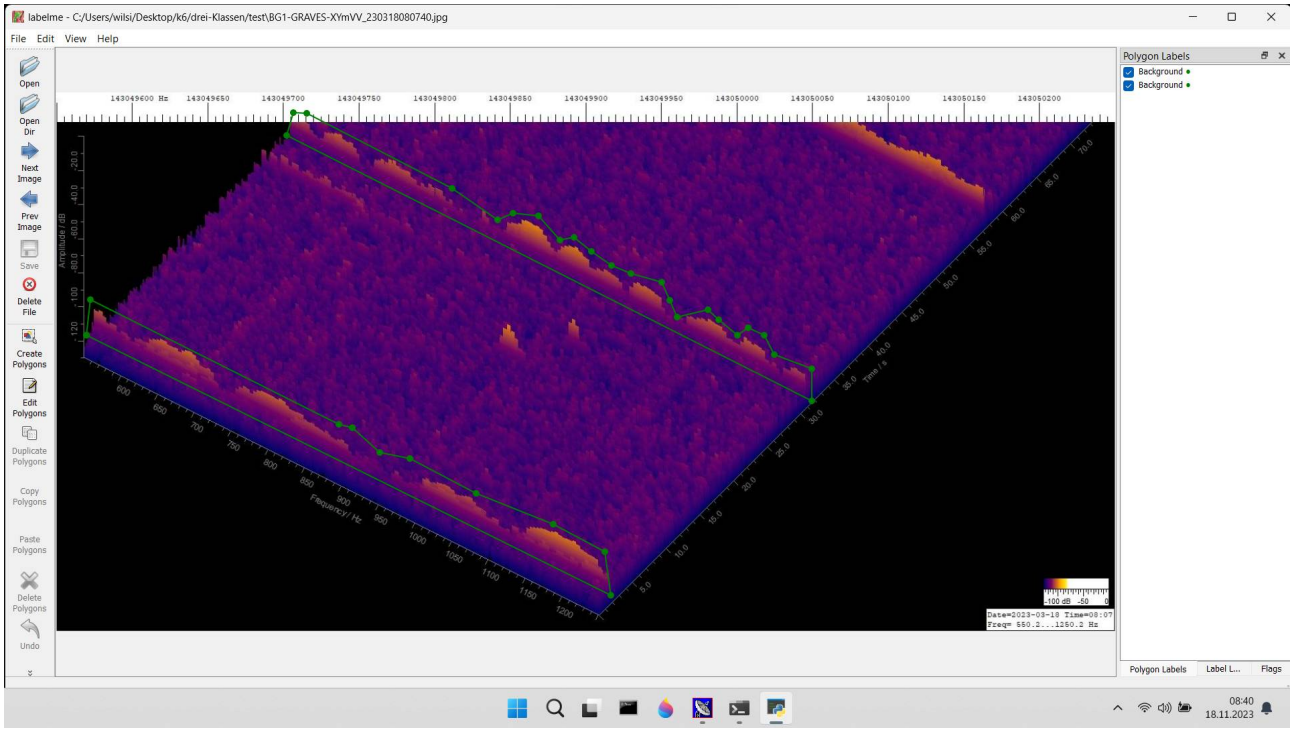

Figure A14 – Gelabelte Störungen / Knackimpulse.

End of File
WiSi 18.11.2023