**Staatliche Fachoberschule München-West**

Seminararbeit

**Thema:**

***Vergleich der Umsetzbarkeit und Performance von Objekterkennung auf Microcontrollern: Eine Untersuchung der Pixy2-Kamera mit Arduino und der Implementierung eines TensorFlow Lite Modells auf einem Raspberry Pi.***

Vorgelegt von: *Paul Nöhles*

Vorgelegt am: *14.01.2025*

Fach: *Technologie, Informatik, Physik*

Rahmenthema: *Arduino im Alltag – innovative Anwendungsmöglichkeiten*

Seminarleitung: *Herr Ciccone; Frau Meurer*

Inhalt

[Inhalt 2](#_Toc187682806)

[1 Einleitung 5](#_Toc187682807)

[2 Pixy mit Arduino 7](#_Toc187682808)

[2.1 Anschaffung 7](#_Toc187682809)

[2.2 Einrichten 7](#_Toc187682810)

[2.3 Mit dem Computer verbinden 7](#_Toc187682811)

[2.4 Pixy mit Arduino verwenden 7](#_Toc187682812)

[2.5 Arduino Test Code 8](#_Toc187682813)

[2.6 Problem des Pixys 8](#_Toc187682814)

[2.7 Weitere Anwendungen des Pixys Außer Objekt-Detektion 8](#_Toc187682815)

[3 Tflite Model auf Raspberry Pi 9](#_Toc187682816)

[3.1 Tflite auf Raspberry Pi installieren 9](#_Toc187682817)

[3.2 Tflite Modelle auf Raspberry Pi verwenden 9](#_Toc187682818)

[3.3 Eigenes Model trainieren 10](#_Toc187682819)

[3.4 Trainings Daten sammeln 10](#_Toc187682820)

[3.4.1 Trainings Daten labeln 10](#_Toc187682821)

[3.5 Model mit Google colab trainieren 10](#_Toc187682822)

[3.6 TensorFlow lokal aufsetzen 11](#_Toc187682823)

[3.7 Model trainieren 11](#_Toc187682824)

[3.8 Model zu Tflite konvertieren 12](#_Toc187682825)

[3.9 Model optimieren 12](#_Toc187682826)

[3.10 Performance des KI-Modells 12](#_Toc187682827)

[3.11 Verbesserung des Models 12](#_Toc187682828)

[4 Ergebnis 14](#_Toc187682829)

[5 Fazit 15](#_Toc187682830)

[6 Literaturverzeichnis 16](#_Toc187682831)

[7 Abbildungsverzeichnis 17](#_Toc187682832)

[8 KI-Verzeichnis 18](#_Toc187682833)

[9 Anhang 19](#_Toc187682834)

[10 Eidesstattliche Erklärung 20](#_Toc187682835)

# Einleitung

Künstliche Intelligenz hat in den letzten Jahren rapide Fortschritte gemacht und dadurch stark an Bedeutung gewonnen. Weshalb diese Technologie ein immer prominenterer Teil unseres Lebens wird und auch in den nächsten Jahren einen immer weiteren Einzug in unser Leben finden. Da KI es möglich macht viele Prozesse zu automatisieren bei denen das bisher nicht oder nur schwer möglich war.

Eine der wichtigsten und interessantesten Anwendung von KI ist die Möglichkeit Maschinen mit KI das sehen bei zu bringen, eine wichtige Art der ist Objekt Detektion. Objekt Detektion ist eine Computer-Vision-Technik die Objekte in einem Video oder Bild lokalisiert. Objekt-Detektion-Algorithmen benutzen normalerweise maschinelles Lernen um nutzbare Ergebnisse zu produzieren. Wenn Menschen Bilder oder Videos betrachten, können wir nach einem kurzen Moment interessante Objekte lokalisieren. Das Ziel von Objekte Detektion ist diese Intelligenz mit einem Computer zu replizieren (MathWorks, 2025). Diese Technologie wird in Zukunft zum Beispiel bei selbst fahrenden Autos noch eine Wichtige Rolle spielen, da diese Hindernisse im Straßenverkehr akkurat erkennen müssen um ihnen ausweichen zu können.

Nun mag Objekt Detektion sehr beindruckend wirken und sehr anspruchsvoll. Aber jeder kann heutzutage

Das Problem mit KI auf Microkontrollern ist das sie durch ihre stark begrenzte Hardware Leistung Probleme damit haben die Leistung intensiven KI-Anwendungen auszuführen. Will man Object-Detektion Modell in Echtzeit ausführen kommt man dann schnell an Grenzen. Zwar werden Prozessoren immer Leistung fähiger, kleiner und billiger [Aber]. Deshalb und weil diese Geräte Preisewert sein müssen werden sie nie die Fähigkeit haben die momentan besten KI-Modelle auszuführen, da diese zu viel Rechenleistung benötigen.

Einen Weg diese Beschränkung zu Umgehen ist es die Berechnungen in der Cloud durchzuführen anstatt auf dem Gerät selbst, diese Lösung ist aber nicht perfekt da dann der Mikrokontroller ständig mit dem Internet verbunden sein muss was nicht bei allen Anwendungen immer möglich ist. Außerdem kommen bei Cloud Anwendungen immer Latenzen dazu die vor allem bei Anwendungen wie Objekt Detektion die in Echtzeit ausgeführt werden deren Performance nach unten ziehen kann. Zusätzlich sind alle Daten die in der Cloud gespeichert werden einem deutlich höheren Risiko ausgesetzt kompromittiert zu werden als Daten die auf Endgeräten gespeichert sind, was vor allem bei Anwendungen für den Privaten Haushalt eine wichtige Rolle spielt. Deshalb wird daran geforscht KI-Modelle weniger Leistung intensiv zu machen und sie zu optimieren damit sie auch auf Edge Devices wie Microcontroller Laufen können.

Edge-Devices oder auf Deutsch Edge-Geräte sind Geräte die an der Grenze von zwei Netzwerken liegen und dienen als eingangs oder Ausgangs punkte des Netzwerks. Vor allem im Rahmen des Internets der Dinge sind Edge-Devices wichtig da sie

So soll sie in vielen Haushalts üblichen Elektrogeräten Anwendung finden. Ein Problem ist aber, dass Haushaltes übliche Geräte oft nicht die notwendige Hardware Power haben um normale KI-Modelle auszuführen.

Deshalb werde ich mir in dieser Arbeit angucken wie man Objekt Detektion auf Microcontrollern umsetzen kann. An zwei konkreten Beispielen einmal dem Pixy2 smart Vision Sensor und Tflite Modellen auf einem Raspberry Pi mit Kamera

# Pixy mit Arduino

Eine der Varianten die ich mir angeschaut habe um Objekt Detektion in DIY-Projekten umzusetzen war eine out of the Box Solution der Pixy2 Sensor in Kombination mit einem Arduino. Pixy ist ein smart Vision Sensor mit dessen Hilfe man Objekt-Detektion . Da die Berechnungen nicht auf dem Arduino stattfinden ist es egal welchen Arduino man nimmt ich habe meinen Aufbau mit dem Arduino Nano und dem Arduino uno R4 WIFI getestet.

## Anschaffung

Der Pixy smart Vision Sensor ist im Vergleich zu anderen Sensoren recht teuer da er einen eigen kleinen Prozessor besitzt und das Objekt-Detektion Programm. Ich habe den Pixy2 Vision Sensor für 80,10 € gekauft. Beim kaufe sollte man Bloß beachten, dass es zwei Version gibt eine für Microkontroller und eine Für den Lego Roboter Lego Mindstorm EV3.

## Einrichten

Als ich meinen Pixy2 erhalten habe musste ich feststellen, dass ich ausversehen die Lego Version und nicht die Arduino Version erworben hatte. Da ich die Lego-version gekauft hatte musste ich die Firmware meines Pixy2 mit der Firmware der nicht Lego-version überschreiben. Dafür konnte ich einfach die Firmware von der Webseite des Herstellers herunterladen und dann auf das Pixy aufspielen. Indem ich das Pixy über ein USB-Kabel an meinen Computer anschließe und während dessen denn Knopf am Pixy gedrückt halten und konnte dann die neue Firmware aufspielen.

## Mit dem Computer verbinden

Um den Pixy2 Sensor an einem Computer zu benutzen kann man es einfach mit einem USB A zu Micro USB-Kabel mit dem PC verbinden und muss die Software „PixyMon v2“ vom Hersteller runterladen und kann dann, wenn der Pixy-Sensor verbunden ist in PixyMon sehen was die Kamera des Pixys sieht, sowie die Objekte die es erkennt. Dort kann man dann auch Objekte einspeichern die der Sensor erkennen soll bis zu vierzehn Stück. Wobei sieben davon einfarbige Objekte seihen müssen und die anderen sieben aus mehreren Farben bestehen müssen. Objekte kann man über mehrere Wege einspeichern entweder man zieht ihn PixyMon ein Dreieck über das Objekt das man einspeichern will oder man hält das Objekt vor das Pixy und drückt den Knopf auf dem Pixy und lässt ihn los wenn die Lampe am Pixy in der Farbe des Objekts Leuchtet.

[Bild Pixy Mon]

## Pixy mit Arduino verwenden

Da ich die Lego Version gekauft hatte, hatte ich nicht das Kabel um das Pixy mit dem Arduino zu verbinden, das bei der normalen Version dabei ist, sondern ein Kabel um es mit dem Lego Mindstorm EV3 zu verbinden. Und hab mir deshalb aus female-female Jumper Wires selbst ein Kabel gebastelt. Indem ich mir angeguckt habe welche Anschlüsse des Pixys mit welchen Anschlüssen des Arduinos verbunden werden müssen.

Um mit dem Pixy dann auch kommunizieren zu können musste ich auch die Arduino Pixy2 Library in der Arduino IDE installieren. Die einem dann die Möglichkeit gibt die Signale die der Pixy Sensor dann schickt zu verarbeiten.

Die Objekte die das Pixy erkennen soll müssen entweder vorher am Computer einmal Eingespeichert werden wie oben beschrieben oder man benutzt die Option mit dem Knopf drücken.

## Arduino Test Code

Ich habe dann getestet ob der Arduino und Pixy auch funktionieren

Das Programm ist eine angepasste Form des ccc\_hello\_world Programms das

Das zeigt, dass man mit dem Pixy

## Problem des Pixys

Der Pixy Sensor hat große Problem Objekte mit ähnlicher Farbe auseinander zu halten und kann gleiche Objekte unter unterschiedlichen Licht Bedingungen nicht auseinanderhalten. Er tut sich auch mit Farblich komplexeren Objekten schwer diese wieder zu erkennen. Am besten funktionierte er mit Objekten oder Symbolen die aus wenigen klar unterscheidbaren Farben bestehen.

## Weitere Anwendungen des Pixys Außer Objekt-Detektion

Der Pixy Vision Sensor besitzt zwei Programme einmal das Color connected components program dass wir für Object-Detektion verwenden und das Line tracking program. Das dafür da ist Linien zu erkennen

# Tflite Model auf Raspberry Pi

Die zweite Option wie man künstliche Intelligenz auf Microkontrollern umsetzen kann die ich mir angeschaut habe, ist ein TensorFlow Lite Model auf einem Raspberry Pi zu verwenden. Der große Unterschied zu Variante mit dem Pixy ist das man hier ein im Vorhinein auf einem stärkeren Computer trainiertes KI-Modell verwendet, das dann auf dem Raspberry Pi ausgeführt wird. Das auf bestimmte Objekte trainiert wurde und diese erkennt anstatt wie beim Pixy, wo Objekte nur kurz vorgehalten werden müssen. Dies ist mit dem Raspberry Pi möglich da er im Vergleich zum Arduino deutlich mehr Leistung hat, sodass er in der Lage ist selbst einfache TFLite Modelle auszuführen.

## TFLite auf Raspberry Pi installieren

Um TFLite Objekt-Detektion Modelle auf dem Raspberry Pi zu verwenden muss man erst TFLite auf dem Raspberry Pi installieren. Dafür sollte man seinen Raspberry Pi erstmal Updaten und auf die neuste Version bringen mit den Befehlen sudo apt-get update und sudo apt-get dist-upgrade, da ich meinen Raspberry Pi längere Zeit nicht verwendet hatte und er deshalb mehrere Updates runterladen musste dauerte das ganze bei mir fast eine Stunde. Der nächste Schritt ist das Tflite GitHub Project auf den Raspberry zu klonen und dann ein Python virtual Environment zu schaffen in dem man dann die von TFLite benötigten Libaries installiert. In Python ist virtuell Environment ein vom Rest der Python Installation abgetrennter Teil, sodass alle python scripts die im virtualen Environment ausgeführt werden nur auf die Libaries im Environment zugreifen können (freeCodeCamp, 2022). Das ist wichtig da man so mehrere Versionen derselben Libarie installiert haben kann so braucht TFLite bestimmte Versionen von Libaries und so konnte ich sicherstellen das die Installation der Libaries die Tflite benötigt nicht mit meinen anderen Installationen in Konflikt gerät.

## TFLite Modelle auf Raspberry Pi verwenden

Ist TFLite erfolgreich auf dem Raspberry Pi aufgesetzt kann man nun entweder TFLite Objekt-Detektion Modelle aus dem Internet herunterladen oder eigene Modelle verwenden . Um dann das Modell zu verwenden muss man eines der vier scripte die in dem GitHub Projekt das wir heruntergeladen haben vom Comand Promt aus ausführen lassen und mindestens ein Argument passen nämlich der Name das Directory in dem sich das KI-Modell, dass man verwenden will befindet.

python TFLite\_detection\_webcam.py --modeldir=Duck\_1000

Das TFLite\_detection\_webcam.py script öffnet, wenn an den Raspberry Pi eine Kamera angeschlossen ist ein Fenster in dem den letzten Frame über den das Programm gelaufen ist Angezeigt wird. Es ist das script

[Bild Webcam stream]

Mit den drei anderen Scripts kann man das Modell über ein oder mehrere Bilder, ein Video oder einen Stream laufen lassen und haben auch ihren Nutzen. Die Bilder in dieser Arbeit zum Beispiel auf denen Bounding boxes zu sehen sind habe ich alle mit dem TFLite\_detection\_image.py script erstellt.

## Eigenes Model trainieren

Wie oben bereits angesprochen kann man natürlich auch selbst ein Eigenes Modell trainieren. Das bietet sich vor allem an wen man eine Eigen Anwendung im Sinn hat da es sein kann das es ihm Internet kein öffentlich verfügbares Object-Detektion Model gibt das auf die Objekte trainiert ist die man für seine eigene Anwendung braucht. Ich wollte schauen wie schwer es ist ein Modell selbst zu erstellen, um zu bestimmen . Ich habe mich dafür entschieden mein Modell auf das erkennen einer Stofftier Ente zu trainieren, da es dafür sicher noch kein Modell gibt und es dadurch, dass es recht einfarbig ist für die KI kein allzu schwieriges Objekt darstellte.

## Trainings Daten sammeln

Um das KI-Model zu trainieren braucht man Trainings Daten auf denen man es trainieren kann. Um ein Objekt-Detektion Modell zu erstellen braucht man natürlich Bilder von dem Objekt das es erkennen soll in meinem Fall von der Ente. Also musste ich Bilder von der Ente machen auf denen ich das Model dann trainieren kann. Dabei gilt zu beachten das die Bilder die man macht möglichst genau die Situationen widerspiegeln in denen das Modell dann später auch verwendet wird, so dass wenn das Modell später in verschieden Situation verwendet wird diese auch all in den Bildern vertreten sind. So wollte ich das es die Ente aus verschieden Blickwinkeln, Distanzen und unter unterschiedlichen Lichtverhältnissen erkennen kann. Deswegen habe ich Bilder gemacht die alle diese verschiedenen Situationen widerspiegeln. Insgesamt 317 Stück für mein einfaches erstes Modell.

### Trainings Daten labeln

Mit Fotos allein kann das Modell nicht trainiert werden. Deshalb mussten die 317 Fotos noch gelabelt werden. Dazu habe ich die open source Software Labelimg verwendet mit der man Bilder labeln kann indem man ein Kästchen um das Objekt zieht auf das man das Modell Trainieren will und das Objekt dann Bennent.

[Bild Labelimg]

Labelimg speichert dann diese Information in XML Files in PASCAL VOC Format, das ist ein Datei Format das ursprünglich für die Visual Object Challenge erschaffen wurde und ist ein Standard Format für Objekt-Detektion-Labels das die Position eines Objektes in Form eines Vierecks, dass das Objekt eingrenzt speichert sowie dessen Namen (roboflow, kein Datum). Beim Labeln der Bilder gilt zu beachten

Diese Trainings Daten habe ich auch in das GitHub Projekt hochgeladen Anhang 1 und können im Ordner Trainings Daten gefunden werden.

## Model mit Google colab trainieren

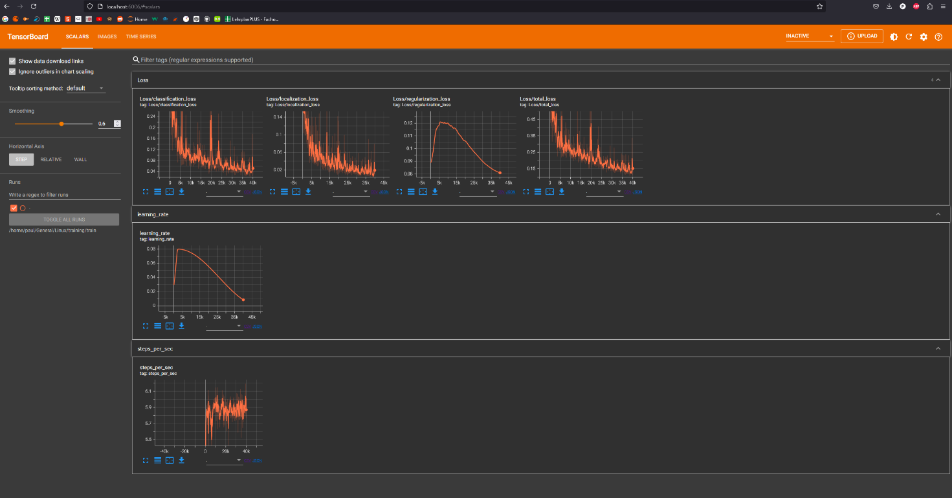
Da ich ein Tutorial gefunden hatte zu dem ein Google Colab Notebook [juypiter] gehört das ich einfach in Colab ausführen konnte und alle Python scripts enthielt die Für das Training des Modells notwendig sind. War mein ursprünglicher Plan das Objekt-Detektion Modell einfach mit Hilfe dieses Tutorials und des Notebooks in Colab auszuführen. Mit Google Colab kann man Python Notebooks im Browser auf einem Google Server mit einer Grafikkarte ausführen lassen und ist für Data Science und Maschinelles lernen ausgelegt (Google, kein Datum). Es hat auch gut funktioniert ich konnte einfach dem tutoriell folgen und musste nur meine Trainings Daten Hochladen und ein paar Variablen im Notebook an mein Modell anpassen. Als ich dann das Modell in Colab zum Schluss trainieren ließ wurde ich dann nach einer halben Stunde Training aus der Sitzung geschmissen, ohne das Mein Modell fertig Trainiert wäre und habe dabei allen Fortschritt im Training bis dahin verloren. Der Grund dafür lag darin das Colab ein Limit für die Tägliche Dauer der Nutzung von GPUs, das nicht genau spezifiziert ist, für Nutzer der gratis Version dieses Limit reicht aber nicht um mein KI-Modell zu trainieren. Der Grund, dass das Tutorial dem ich gefolgt bin dies nicht erwähnt lag daran das es von Anfang 2023 ist und Google seitdem die Täglichen Limits für die gratis Version mehrfach verringert hat.

## TensorFlow lokal aufsetzen

Da es nicht geklappt hat das KI-Modell auf Google colab zu Trainieren musst ich das KI-Modell lokal auf meiner eigenen Maschine trainieren. Um das zu bewerkstelligen musste ich TensorFlow auf meinem Computer aufsetzen. Dies erwies sich als schwieriger als erwartet. Nachdem ich mehreren längeren Tutorials gefolgt bin und immer wieder an einen Punkt kam wo ich in eine Schleife geriet bei der ich eine Fehlermeldung bekomm bei der mir eine Libary sagt das sie eine andere Version von einer Libary braucht nur um dann festzustellen das Eine andere Libary eine andere Version dieser Libary braucht. Fand ich heraus, dass jede Version von TensorFlow nur mit bestimmten Python Versionen, cuDNN und CUDA kompatible sind und ich dass beachten muss

## Model trainieren

Nach dem TensorFlow aufgesetzt war und alle Parameter eingestellt waren. Konnte ich dann mit dem Training des Modells beginne das hat dann etwas weniger als zwei Stunden gebraucht. Denn Trainings Prozess konnte ich über Tensorboard während des Trainings überwachen. Tensorboard ist ein Visualisierungstool von Google das man mit TensorFlow verwenden kann.



Die Entwicklung des Modells während des Trainings zu sehen ist wichtig denn sollten irgendwelche Einstellungen nicht stimmen sieht man das dort direkt. So ist bei mir der Loss beim ersten Trainings Versuch nach kurzer Zeit nicht mehr weiter gesunken sondern immer größer geworden, da . Zwei weitere Trainings Versuche scheiterten daran das meine GPU nach kurzer Zeit der Grafikspeicher ausgegangen ist da ich die Batch size zu groß eingestellt hatte. Die Batch size bestimmt wie viele Bilder pro Trainings Schritt verwendet werden umso größer umso schneller wird das Modell trainiert. Deshalb musste ich meine Batch size von 16 auf 8 verringern.

## Model zu Tflite konvertieren

Das Model das ich dann trainiert hatte konnte ich nicht direkt auf den Raspberry Pi verwenden, da es bisher noch ein normales TensorFlow-Modell war. Also war der nächste schritt das Modell zu einem Tflite Model zu konvertieren.

## Model optimieren

Man kann nun nach dem Training noch eine Quantisierung durchführen. Bei Post-Training quantisieren wird ein bereits fertig trainiertes Modell quantisiert, die Variablen der Gewichte des Modells werden von einem genaueren größeren Daten typen in einem kleineren ungenaueren umgewandelt, zum Beispiel von uint16 zu uint8. Das verringert die Model Größe und verbessert dessen Geschwindigkeit es verliert dabei aber auch an Präzision (tensorflow, 2022). Ich habe mein Model auch quantifiziert da ich möglichst hohe Performance rausholen wollte und

## Performance des KI-Modells

Nachdem ich dann das KI-Modell und seine quantifizierte Version auf den Raspberry Pi übertragen hatte habe ich beide dort Ausprobiert.

Ein Problem des Modells war die Tatsache, dass es alle möglichen gelben Objekte als Ente erkannte. Besonders ein anderes Kuscheltier eine gelbe Banane hat es oft als Ente erkannt.

[Bild gelbe Objekte]

Ich habe die Genauigkeit

## Verbesserung des Models

Da ich nicht mit der Akkuratheit meines Models zufrieden war und ich schauen wollte wie sich die Perfomance des Modells verändert, wenn ich es mit mehr Bildern trainiere. Deshalb habe ich noch 683 weitere Bilder gemacht und Gelabelt und mit diesen zwei weiteren Modellen trainiert, einmal mit 650 Bildern und einmal mit 1000 Bildern.

Um das Problem das es alle gelben Objekte als Ente erkennt anzugehen habe ich bei den Bildern für die späteren Modellen bewusst Bilder von der Ente gemacht auf denen auch andere gelbe Objekte zu sehen waren, damit es lernt das Gelb nicht automatisch Ente bedeutet. Das hat auch teilweise funktioniert die späteren Modelle konnten deutlich besser zwischen der Ente und anderen gelben Objekten unterscheiden aber nicht immer.

[Bild gelbe Objekte besser]

# Ergebnis

Zusammenfassend lässt sich sagen, dass

# Fazit

# Literaturverzeichnis

MathWorks. (12. 1 2025). *What Is Object Detection?* Von https://de.mathworks.com/discovery/object-detection.html abgerufen

roboflow. (kein Datum). *Pascal VOC XML*. Abgerufen am 12. 1 2025 von https://roboflow.com/formats/pascal-voc-xml

tensorflow. (8. 3 2022). *Post-training quantization*. Abgerufen am 12. 1 2025 von https://www.tensorflow.org/model\_optimization/guide/quantization/post\_training

# Abbildungsverzeichnis

# KI-Verzeichnis

# Anhang

# Eidesstattliche Erklärung

Ich versichere, dass ich die vorliegende Seminararbeit selbständig angefertigt, nicht anderweitig für Prüfungszwecke vorgelegt, alle benutzten Quellen und Hilfsmittel, sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ort, Datum Unterschrift