# 

[**Abstract**](#_8ykygf5qh4qn) **3**

[**Preface**](#_rntqhseoucq8) **3**

[Acknowledgement](#_o9n9u0ogvfw5) 3

[Statutonary decoration](#_opo8rx3icfl) 3

[Contents](#_hb4wv1vc819i) **3**

[List of tables](#_2q7nn3squ4qf) 3

[List of figure](#_uzv8wugfp52l) 3

[**Acronyms and definitions**](#_svod3ldqgiw5) **3**

[**Introduction**](#_mhu3dj2egbi8) **3**

[Motivations](#_wd5j23z1lz40) 3

[Scope](#_wagtyzjmy7sy) 3

[Objectives](#_t1s9d3re1lsl) 3

[Methodology](#_xbpelrgs73df) 3

[Contributions](#_yka9hs2biuqb) 3

[**Background**](#_assls5sr1dlh) **4**

[Embedded systems](#_51g7np6qqtw6) 4

[Artificial Neural Networks](#_bbvcf8hzrhqa) 4

[Neuronale Netzwerkarchitekturen](#_qbqy15fitowk) 4

[Convolutional neural networks](#_66g3ttgaf2rb) 4

[Convolution operators](#_267f3sgzerv1) 6

[Depthwise convolution operator](#_v76k9y58b0p8) 6

[Datasets](#_me36y0b3x2u8) 6

[Skin cancer dataset](#_685m1j9vnk9q) 6

[CIFAR-10](#_n8znzhs8w3qy) 6

[Transfer learning](#_lpvsqun5omjn) 6

[Accuracy](#_bu7olh5kkyzf) 6

[Latency](#_h65mt9gpg95q) 6

[Energy consumption](#_edew21jb16lo) 6

[Memory](#_eamejj5o3ib3) 6

[Computational cost](#_34r3tnm7ht4h) 6

[Steps for machine learning](#_vnzkcoatbl6u) 6

[Deep neural network models for practical applications](#_sh917llyosre) 6

[Accuracy](#_3kcc79z1rkj3) 6

[Latency](#_6cj7zdk00h27) 6

[Energy consumption](#_cbyw3oglt7ix) 6

[Memory](#_z4dagj6ztakr) 6

[Computational cost](#_swsydassu3rh) 6

[Development environment](#_wa1jrx6xb9la) 6

[Python libraries](#_83881fc2cr44) 7

[Tensorflow](#_ez7gi67o9tte) 7

[Version control](#_xjp2o978mp60) 7

[Git](#_tw9l5mpeqdbp) 7

[**Proposed embedded system architectures**](#_mvxie510t527) **7**

[Raspberry Pi](#_m15njgg22jcf) 7

[Jetson Nano](#_lo79u8v5oqez) 7

[Google Coral](#_ze8lluugevsg) 7

[Arduino](#_m97td1oj4xoe) 7

[**Design exploration with shallow convolutional neural networks**](#_oskh79oy241j) **7**

[VGG style convolutional neural networks](#_8zboymlwldl) 7

[Accuracy](#_d9dfh74q44aj) 7

[Memory](#_9qqtp99q5z51) 7

[Computational cost](#_8vmme6w7utb9) 7

[Performance benchmarks](#_1q0zzwqdu9gh) 7

[Latency](#_jpx9m0hl8b88) 7

[Energy consumption](#_tehw4gwuwq8) 7

[Resnet style convolutional neural networks](#_bkucumzekrvc) 7

[Accuracy](#_9du5sfazpt8p) 8

[Memory](#_ivwh2de8ja3i) 8

[Computational cost](#_pbae2hfkx5gf) 8

[Performance benchmarks](#_4bqmgozhd3vz) 8

[Latency](#_a70rpkt9mucb) 8

[Energy consumption](#_xid18rmi9o1t) 8

[**Design exploration with deep convolutional neural networks**](#_npdofrgv3j3w) **8**

[VGG16](#_b0ge8wni42mg) 8

[Mobilenet](#_l0nbaxupbhy0) 8

[Resnet 50](#_2fc5h68siq33) 8

[**Results and discussion**](#_3upz6o8y8jvx) **8**

[Exploration results](#_lchpkgv6gl1q) 8

[**Conclusions**](#_oz5otoac0ng) **8**

[**Appendix**](#_eu2erhy0644v) **8**

[Python code](#_yd9b08xx5cnl) 8

[C code](#_ky2iyca7y95g) 8

# 

# 

# Abstract

# Preface

## Acknowledgement

## Statutonary decoration

# Contents

## List of tables

## List of figure

# Acronyms and definitions

# Introduction

## Motivations

## Scope

## Objectives

## Methodology

## Contributions

# Background

## Embedded systems

Bei eingebetteten Systemen handelt es sich um Computer, bestehend aus Hardware und Software ,deren Zweck meist nur darin besteht, eine spezifische Aufgabe zu erfüllen und in ein großes System eingebettet zu werden, deshalb besitzen viele von ihnen keine Benutzeroberfläche. Andere hingegen die auch in mobilen Geräten zu finden sind können verschiedene Benutzerschnittstellen besitzen wie zum Beispiel ein Display, Touchscreen oder auch Tasten. Häufig werden sie in Geräten in verbindung mit dem Internet der Dinge, IoT, eingesetzt für Sensoren und Echtzeit - Computing. In diesem Fall ist für den Betrieb kein Fremdeinwirken notwendig da eine Verbindung zum Internet besteht.

Embedded systems sind in vielen verschiedenen Anwendungsbereichen vorzufinden. In der Medizintechnik, der Industrie, Landwirtschaft und vielen weiteren.

## Artificial Neural Networks

ANNs zu deutsch, Künstliche neuronale Netze sind Algorithmen, deren Struktur vom menschlichen Nervensystem inspiriert sind. Auf dem Gebiet der neuronalen Netze wird viel geforscht, da sie als Grundlage für künstliche Intelligenz gelten. ANNs sind in der Lage, komplexe Aufgaben zu bewältigen; zum Beispiel haben convolutional neural networks eine bessere Leistung als Menschen bei Computer-Vision-Aufgaben gezeigt [1]. Eine ANN kann verwendet werden, um verschiedene Arten von Informationen wie Bilder, Ton, Text usw. zu verarbeiten und ihre Eigenschaften oder Merkmale zu extrahieren und sie für die weitere Verarbeitung wie Klassifizierung zu verwenden. Dies bietet die Möglichkeit, auf Basis vorhandener Daten Rückschlüsse oder Vorhersagen zu treffen.

### Neuronale Netzwerkarchitekturen

Die Architektur neuronaler Netze kann variieren, aber im Grunde sind sie wie Rechengraphen strukturiert, siehe Abb. 1. Und sie werden normalerweise durch neuronale Strukturen repräsentiert, siehe Abb. 2. Es gibt eine input layer, hidden layer und eine output layer. Hidden layers können im Gegensatz zu den anderen beiden Schichten mehr als eine Schicht sein. Wenn die ANNs sehr tiefe Netzwerkstrukturen aufweisen, handelt es sich um eine Deep-Learning-Architektur, siehe Abb. 3.

## Convolutional neural networks

Convolutional Neural Networks sind eine der beeindruckendsten arten von ANN- Architekturen. Sie besitzen eine sehr genaue und doch simple Architektur mit der sie zum Großteil dazu genutzt werden um komplizierte Bild Mustererkunnungs Aufgaben zu lösen. Dies ermöglicht es die Architektur mit bildspezifischen Merkmalen codieren, und den Fokus somit auf Bildverarbeitung zu setzen, und gleichzeitig andere Parameter für die erstellung eines Modells zu reduzieren. Hierin liegt auch der bedeutendste Unterschied zu dem ANN.

Denn das größte Problem für eine für eine herkömmliche ANN ist, das diese zu schwierigkeiten mit der Rechenkomplexität tendieren, die benötigt wird um Bilddateien zu verarbeiten.

**Overfitting**

Die Überanpassung von Modellen ist ein Problem welches oft auftritt.

Es ist ein Irrtum zu glauben das man durch die erhöhung der Anzahl der Schichten in den Hidden Layers im Netzwerk eine größere Genauigkeit erreichen kann. Dies hat mehrere Gründe. Zum einen benötigt man für riesige Netzwerke auch eine größere Rechenleistung und diese steht nicht unbegrenzt zur verfügung, zum anderen dauert das Training eines ANN in diesem Fall viel zu lange. Ein weiterer Grund ist der das dies zu einem sogenannten Overfitting führen.

Overfitting bedeutet im grunde genommen das ein Netzwerk nicht dazu in der Lage ist effektiv zu lernen, dies passiert aus einer reihe von Gründen. Da dies bei sogut wie allen Algorithmen für Maschinelles Lernen passiert, ist es sehr wichtig Vorkehrungen zu treffen die Folgen des Overfittings zu reduzieren.

Anzeichen dafür dass Modelle eine Überanpassung aufweisen, ist das es zu einer verringerten Fähigkeit der Erfassung von generalisierten Eigenschaften bei den Datensätzen kommt, nicht nur beim Trainingsdatensatz, sondern auch dem Testdatensatz.

Dies ist der Hauptgrund für die Reduzierung der Komplexität von ANNs. Je weniger

Parameter für das Training erforderlich sind, desto unwahrscheinlicher ist eine Überanpassung des Netzwerks und natürlich auch die genauigkeit der Prognosen des Modells verbessern.

**Convolutional Neural Network Architektur**

Wie bereits erwähnt, sind CNNs hauptsächlich auf Eingaben bestehend aus Bildern Fokussiert. Dadurch wird die einzurichtende Architektur so ausgerichtet, dass sie den Anforderungen für den Umgang mit der jeweiligen Datenart am besten entspricht.

Einer der Hauptunterschiede besteht darin, dass die Neuronen, die die Schichten innerhalb des CNN enthalten, aus Neuronen bestehen, die in drei Dimensionen organisiert sind, die räumliche Dimension der Eingabe (Höhe und Breite) und die Tiefe. Die Tiefe bezieht sich nicht auf die Gesamtzahl der Schichten innerhalb des ANN, sondern auf die dritte Dimension eines Aktivierungsvolumens. Im Gegensatz zu Standard-ANNS verbinden sich die Neuronen innerhalb einer bestimmten Schicht nur mit einem kleinen Bereich der davorliegenden Schicht.

CNNs bestehen aus drei Arten von Schichten. Dies sind Convolutional layer (Faltungsschichten), Pooling layers (Pooling-Schichten) und fully-connected layers (vollständig verbundene Schichten). Gestapelt bilden diese Schichten eine CNN-Architektur. Eine vereinfachte CNN-Architektur für die MNIST-Klassifizierung ist in der Abbildung dargestellt

### Convolution operators

Convolutional layer

Wie der Name schon sagt, spielt die Faltungsschicht eine entscheidende Rolle dabei, wie CNNs

arbeiten. Die Layer-Parameter konzentrieren sich auf die Verwendung von lernbaren Kerneln.

Diese Kerne sind normalerweise klein in der räumlichen Dimensionalität, breiten sich aber entlang der

Gesamtheit der Tiefe der Eingabe. Wenn die Daten auf eine Faltungsschicht treffen,

die Schicht faltet jeden Filter über die räumliche Dimension der Eingabe zu

Erstellen Sie eine 2D-Aktivierungskarte.

Während wir durch die Eingabe gleiten, wird das Skalarprodukt für jeden Wert in . berechnet

Dieser Kernel. Daraus lernt das Netzwerk Kernel, die 'feuern', wenn

sie sehen ein bestimmtes Merkmal an einer bestimmten räumlichen Position der Eingabe. Diese sind

allgemein als Aktivierungen bekannt.

Jeder Kernel hat eine entsprechende Aktivierungskarte, die gestapelt wird

entlang der Tiefendimension, um das volle Ausgabevolumen aus der Falte zu bilden.

nationsschicht.

Wie bereits erwähnt, führt das Training von KNNs auf Eingaben wie Bilder zu

Modelle, die zu groß sind, um effektiv zu trainieren. Hier kommt es auf die voll-

verbundene Art von Standard-ANN-Neuronen, um dies zu mindern

Neuron in einer Faltungsschicht ist nur mit einem kleinen Bereich des Eingangs verbunden

Volumen. Die Dimensionalität dieser Region wird allgemein als Re-

zeptive Feldgröße des Neurons. Das Ausmaß der Konnektivität durch die

Tiefe ist fast immer gleich der Tiefe der Eingabe.

Wenn die Eingabe in das Netzwerk beispielsweise ein Bild der Größe 64 × 64 × 3 (ein RGB-

farbiges Bild mit einer Dimensionalität von 64 × 64) und wir setzen das rezeptive Feld

Größe als 6 × 6, hätten wir insgesamt 108 Gewichte auf jedem Neuron innerhalb der

Faltungsschicht. (6 × 6 × 3, wobei 3 die Größe der Konnektivität über

die Tiefe des Volumens) Um dies ins rechte Licht zu rücken, ein Standardneuron gesehen

in anderen Formen von KNN würde jede 12.288 Gewichte enthalten.

Faltungsschichten können zudem die Komplexität der

Modell durch die Optimierung seiner Ausgabe. Diese sind optimiert durch

drei Hyperparameter, die Tiefe, der Schritt und die Einstellung von Zero-Padding.Einführung in Convolutional Neural Networks

Die Tiefe des durch die Faltungsschichten erzeugten Ausgabevolumens kann

manuell durch die Anzahl der Neuronen innerhalb der Schicht auf die gleiche einstellen

Bereich der Eingabe. Dies kann bei anderen Formen von KNN beobachtet werden, bei denen die

alle Neuronen in der versteckten Schicht sind direkt mit jedem einzelnen verbunden

Neuronen vorher. Die Reduzierung dieses Hyperparameters kann erheblich minimieren

die Gesamtzahl der Neuronen des Netzwerks, kann aber auch deutlich reduziert werden

die Mustererkennungsfähigkeiten des Modells.

Wir sind auch in der Lage, die Schrittweite zu definieren, in der wir die Tiefe um den Raum herum einstellen

Dimensionalität der Eingabe, um das rezeptive Feld zu platzieren. Zum Beispiel, wenn

würden wir einen Schritt auf 1 setzen, dann hätten wir eine stark überlappende Aufnahme

Feld, das extrem große Aktivierungen erzeugt. Alternativ den Schritt auf a . einstellen

eine größere Zahl reduziert die Überlappung und erzeugt eine Ausgabe

von geringeren räumlichen Dimensionen.

Zero-Padding ist der einfache Vorgang, den Rand der Eingabe aufzufüllen, und

ist eine effektive Methode, um die Dimensionalität der

Ausgabevolumen.

Es ist wichtig zu verstehen, dass wir uns durch die Anwendung dieser Techniken verändern werden

die räumliche Dimensionalität der Ausgabe der Faltungsschichten. Um dies zu berechnen,

Dabei steht V für die Größe des Eingabevolumens (Höhe × Breite × Tiefe), R für

die rezeptive Feldgröße, Z ist die Menge des Null-Padding-Satzes und S bezieht sich auf

der Schritt. Wenn das berechnete Ergebnis dieser Gleichung nicht gleich einem Ganzen ist

ganzzahlig, dann wurde die Schrittlänge falsch eingestellt, da die Neuronen nicht in der Lage sind

passen sauber über die gegebene Eingabe.

Trotz unserer bisherigen Bemühungen werden wir feststellen, dass unsere Modelle immer noch enorm sind.

mous, wenn wir eine Bildeingabe von beliebiger realer Dimensionalität verwenden. Methoden

wurden entwickelt, um die Gesamtzahl der Parameter stark einzuschränken

innerhalb der Faltungsschicht.

Die Parameterfreigabe funktioniert unter der Annahme, dass, wenn eine Regionsfunktion nützlich ist

in einer bestimmten räumlichen Region zu berechnen, dann ist es wahrscheinlich in einer anderen Region nützlich.

Wenn wir jede einzelne Aktivierungskarte innerhalb des Ausgabevolumens auf die

gleiche Gewichte und Bias, dann werden wir eine massive Reduzierung der Anzahl der

Parameter, die von der Faltungsschicht erzeugt werden.

Als Ergebnis davon, wenn die Backpropagation-Stufe auftritt, wird jedes ne

Eisen in der Ausgabe stellt den Gesamtgradienten dar, der über die Tiefe summiert werden kann Dadurch wird nur ein einzelner Satz von Gewichtungen aktualisiert, im Gegensatz zu jedem einzelnen.

Pooling layer

Pooling-Layer zielen darauf ab, die Dimensionalität der Darstellung sukzessive zu reduzieren.

und reduzieren damit die Anzahl der Parameter und den rechnerischen

Komplexität des Modells.

Der Pooling-Layer funktioniert über jede Aktivierungskarte in der Eingabe und skaliert

seine Dimensionalität mit der Funktion „MAX“. In den meisten CNNs kommen diese im

Form von Max-Pooling-Schichten mit Kernen einer Dimensionalität von 2 × 2 aufgetragen

mit einem Schritt von 2 entlang der räumlichen Dimensionen der Eingabe. Dies skaliert die

Aktivierungskarte bis auf 25 % der Originalgröße - unter Beibehaltung der Tiefe

Volumen auf seine Standardgröße.

Aufgrund der destruktiven Natur der Pooling-Schicht gibt es im Allgemeinen nur zwei

beobachteten Methoden des Max-Pooling. Normalerweise sind der Schritt und die Filter des Poolings

Ebenen sind beide auf 2 × 2 eingestellt, wodurch sich die Ebene durch die

Gesamtheit der räumlichen Dimensionalität der Eingabe. Außerdem überlappend

Pooling kann verwendet werden, wobei der Schritt auf 2 mit einer Kernelgröße auf eingestellt ist

3. Aufgrund der destruktiven Natur des Poolings wird eine Kernelgröße über 3

die Leistung des Modells in der Regel stark herabsetzen.

Es ist auch wichtig zu verstehen, dass über das Max-Pooling hinaus CNN-Architekturen

kann allgemeines Pooling enthalten. Allgemeine Pooling-Layer bestehen aus Pooling

Neuronen, die in der Lage sind, eine Vielzahl allgemeiner Operationen durchzuführen, einschließlich

L1/L2-Normalisierung und durchschnittliches Pooling. Dieses Tutorial wird jedoch in erster Linie

Konzentrieren Sie sich auf die Verwendung von Max-Pooling.

Nonlinearity layer

Die Nichtlinearitätsschicht hilft, die Anpassungsfähigkeit von Neu-

rale Netzwerke. In CNN die gleichgerichtete Lineareinheit (ReLU), als

in Abb. 1(c) gezeigt, ist die am häufigsten verwendete Funktion [1].

Hyperbolische Tangensfunktion und Sigmoidfunktion sind auch

in verschiedenen neuronalen Netzen übernommen.

(Angel-Eye: A Complete Design Flow for Mapping CNN Onto Embedded FPGA)

was ist mit Flatten?

Fully-connected layer

Die vollständig verbundene Schicht enthält Neuronen, die direkt mit

die Neuronen in den beiden benachbarten Schichten, ohne mit irgendwelchen Schichten verbunden zu sein

in ihnen. Dies ist analog zu der Art und Weise, wie Neuronen in traditionellen

Formen von ANN

### Depthwise convolution operator

## Datasets

### Skin cancer dataset

### CIFAR-10

Die Abkürzung CIFAR bedeutet Canadian Institute For Advanced Research. Bei CIFAR-10 handelt es sich um einen von Entwicklern des Canadian Institutes For Advanced Research geschaffenen Datensatz. Er besteht aus 60.000 jeweils 32 x 32 Pixel großen farbigen Bildern. Die Bilder sind abhängig von den gezeigten Objekten in zehn verschiedene Klassen mit jeweils 6.000 Bildern eingeteilt und mit dem jeweiligen Klassenlabel versehen.

Gegenüber normalen Fotografien sind die Bilder des Datensatzes mit ihren nur 32 x 32 Pixel wesentlich kleiner. Der Datensatz wurde entwickelt, um Algorithmen der Bildverarbeitung, Bilderkennung und des maschinellen Lernens sowie künstliche neuronale Netze zu trainieren, zu optimieren und zu testen. Es handelt sich im Umfeld der Bilderkennung um einen der weltweit meist verwendeten Datensätze für Künstliche Intelligenz ([KI](https://www.bigdata-insider.de/was-ist-kuenstliche-intelligenz-a-562354/)) und maschinelles Lernen. Aufgrund der geringen Auflösung der Bilder lassen sich Training und Test der Algorithmen sehr schnell vornehmen.

Konvolutionale neuronale Netzwerke erzielen nach einem Training mit den Bildern eine hohe CIFAR-10-Bilderkennungsrate. CIFAR-10 ist ein Subset des circa 80 Millionen Bilder umfassenden Tiny Images Datasets. CIFAR-10 und CIFAR-100 können über die Internetseiten des Canadian Institutes For Advanced Research kostenlos in verschiedenen Versionen heruntergeladen werden.

### Zusammensetzung und Klassen des CIFAR-10-Datasets

Wie bereits erwähnt, unterteilt das CIFAR-10-Dataset die 60.000 Bilder in zehn verschiedene Klassen mit jeweils 6.000 Bildern pro Klasse. Die verschiedenen Klassen sind:

* Flugzeuge
* Autos
* Vögel
* Katzen
* Hirsche
* Hunde
* Frösche
* Pferde
* Schiffe
* Lkw

Die einzelnen Klassen sind vollständig voneinander unabhängig und besitzen keine Überlappungen. Von den insgesamt 60.000 Bildern sind 50.000 Bilder Trainingsbilder und 10.000 Bilder Testbilder.

### Die verschiedenen CIFAR-10-Versionen

Abhängig vom jeweiligen Einsatzbereich und der verwendeten Methoden oder Programmiersprachen existieren drei verschiedene Versionen des CIFAR-10-Datasets. Diese Versionen sind:

* eine Python-Version von CIFAR-10 mit einer Größe von 163 Megabyte
* eine Matlab-Version von CIFAR-10 mit einer Größe von 175 Megabyte
* eine binäre Version von CIFAR-10 mit einer Größe von 162 Megabyt

## Transfer learning

Transferlernen ist die Verbesserung des Lernens in einem neuen

Aufgabe durch den Transfer von Wissen aus einer verwandten, bereits erlernten Aufgabe.

Vereinfacht ausgedrückt liegt die Idee zu Grunde, dass ein Machine Learning-System seine Erkenntnisse und sein erlerntes Datenmodell an ein anderes System weitergeben kann, das diese Daten für die Analyse einer ähnlich gearteten Fragestellung nutzen kann. Dabei lassen sich auch ähnliche Quell-Modelle und deren Daten nutzen, um wiederum Daten in einem neuen Ziel-System zu verarbeiten. (<https://www.industry-of-things.de/iot-basics-was-ist-transfer-learning-a-884674/>)

Transfer Learning ist eine Machine Learning Technik, die ein bereits auf eine Aufgabenstellung trainiertes Modell nimmt und es auf eine andere Aufgabenstellung anwendet. Insbesondere unstrukturierte Daten, wie z.B. Bild-, Video- und Audiodaten, machen einen solchen Deep Learning Ansatz interessant.

Der Vorteil von Transfer Learning ist, dass man Teile des sehr aufwendigen Trainings überspringen kann und dadurch Zeit und Ressourcen spart, denn komplexe Deep Learning Modelle brauchen oft Wochen, bis diese trainiert sind.

Das einfachste Beispiel von transferiertem Lernen ist die Verarbeitung von Bilddaten zur Objekterkennung. Folgende Grafik soll dies anhand einer Bildklassifikation zur Objekterkennung untermalen.

Hier wird ein vortrainiertes Modell (bspw. ResNet) für Transfer Learning genutzt, was bereits rund 1000 unterschiedliche Objekte erkennen kann. Allerdings ist das vortrainierte Modell nicht auf unseren speziellen Anwendungsfall (hier am Beispiel Hunderassen) trainiert und kennt Hunderassen, wie z.B. den Chihuahua, noch nicht. Da wir die unterschiedlichen Hunderassen mit unserm Deep Learning Modell erkennen wollen, müssen wir den Algorithmus erneut trainieren.

(<https://datasolut.com/was-ist-transfer-learning/>)

## Accuracy

## Latency

## Energy consumption

## Memory

## Computational cost

## Steps for machine learning

## Deep neural network models for practical applications

### Accuracy

### Latency

### Energy consumption

### Memory

### Computational cost

## Development environment

In der Softwareentwicklung ist die Entwicklungsumgebung eine Reihe von Prozessen und Werkzeugen, die verwendet werden, um einen Quellcode oder ein Programm zu entwickeln.

Dieser Begriff wird manchmal synonym mit integrierter Entwicklungsumgebung (IDE) verwendet, bei der es sich um das Softwareentwicklungswerkzeug handelt, das zum Schreiben, Erstellen, Testen und Debuggen eines Programms verwendet wird. Sie bieten Entwicklern auch eine gemeinsame Benutzeroberfläche (UI), um in verschiedenen Modi zu entwickeln und zu debuggen. Im Allgemeinen bezieht sich der Begriff Entwicklungsumgebung auf die gesamte Umgebung, einschließlich Entwicklungs-, Staging- und Produktionsserver, während sich die IDE nur auf die lokale Anwendung bezieht, die zum Coden verwendet wird. Natürlich gibt es viele Überschneidungen, wenn Sie eine IDE zum Debuggen verwenden, genauso wie Sie einen Entwicklungsserver zum Testen verwenden.

(<https://www.techopedia.com/definition/16376/development-environment>)

### Software framework for machine learning

### Frameworks sind wichtige Werkzeuge für Entwicklungssprachen. Sie steigern die Leistung, erweitern die Funktionen und bieten Bibliotheken mit Codierungsverknüpfungen, sodass Entwickler Anwendungen nicht von Grund auf neu codieren müssen. Für die Entwicklung von Applikationen oder Websites ist ein Application Framework nicht unbedingt erforderlich. Es vereinfacht nur die Arbeit und hilft Dir Software besser und schneller zu erstellen. (<https://it-talents.de/it-wissen/application-framework/>)

python libraries

Eine Bibliothek ist eine Sammlung vorkombinierter Codes, die iterativ verwendet werden können, um die für das Programmieren erforderliche Zeit zu reduzieren. Sie sind besonders nützlich, um auf die vorgefertigten häufig verwendeten Codes zuzugreifen, anstatt sie jedes Mal von Grund auf neu zu schreiben. Ähnlich wie bei den physischen Bibliotheken handelt es sich hierbei um eine Sammlung wiederverwendbarer Ressourcen, was bedeutet, dass jede Bibliothek eine Root-Quelle hat. Dies ist die Grundlage für die zahlreichen Open-Source-Bibliotheken, die in Python verfügbar sind.

Bevor Sie beginnen, empfehle ich Ihnen, wenn Sie ein Neuling in der Domäne von Python sind, dieses Video zur Verwendung von Python für Datenvisualisierungs- und Datenanalysetechniken anzusehen.(<https://www.mygreatlearning.com/blog/open-source-python-libraries/#Library>)

### Tensorflow

TensorFlow ist eine End-to-End-Open-Source-Plattform für maschinelles Lernen. Es verfügt über ein umfassendes, flexibles Ökosystem aus Tools, Bibliotheken und Community-Ressourcen, mit denen Forscher den Stand der Technik im Bereich ML vorantreiben und Entwickler auf einfache Weise ML-basierte Anwendungen erstellen und bereitstellen können. TensorFlow bietet mehrere Abstraktionsebenen, sodass Sie die richtige für Ihre Anforderungen auswählen können. Erstellen und trainieren Sie Modelle mit der High-Level Keras API, die den Einstieg in TensorFlow und maschinelles Lernen erleichtert.

Wenn Sie mehr Flexibilität benötigen, ermöglicht die eifrige Ausführung eine sofortige Iteration und ein intuitives Debugging. Verwenden Sie für umfangreiche ML-Trainingsaufgaben die Verteilungsstrategie-API für verteiltes Training auf verschiedenen Hardwarekonfigurationen, ohne die Modelldefinition zu ändern.

(<https://www.tensorflow.org/>)

The most popular deep learning framework, TensorFlow is an open-source software library for high-performance numerical computation. It is an iconic math library and is also used for machine learning and deep learning algorithms. [Tensorflow](https://www.tensorflow.org/) was developed by the researchers at the Google Brain team within Google AI organisation, and today it is being used by researchers for machine learning algorithms, and by physicists for complex mathematical computations

## Version control

Die Versionskontrolle, auch bekannt als Quellcodekontrolle, ist ein Verfahren zur Verfolgung und Verwaltung von Änderungen an Softwarecode. Versionskontrollsysteme sind Softwaretools, mit deren Hilfe Softwareteams Änderungen am Quellcode im Zeitverlauf verwalten können. Da die Entwicklungsumgebungen immer schneller geworden sind, helfen Versionskontrollsysteme den Softwareteams, schneller und intelligenter zu arbeiten. Sie sind besonders nützlich für DevOps-Teams, da sie ihnen helfen, die Entwicklungszeit zu verkürzen und die Zahl erfolgreicher Deployments zu steigern.

Versionskontrollsoftware verfolgt jede Änderung am Code und speichert sie in einer speziellen Datenbank. Unterläuft Entwicklern ein Fehler, können sie einen Schritt zurückgehen und frühere Codeversionen vergleichen, um das Problem zu beheben, ohne Teammitglieder groß zu stören.

(<https://www.atlassian.com/de/git/tutorials/what-is-version-control>)

### Git

Git ist ein kostenloses und Open Source verteiltes Versionskontrollsystem, das entwickelt wurde, um alles von kleinen bis zu sehr großen Projekten schnell und effizient abzuwickeln.

Git ist einfach zu erlernen und hat einen winzigen Footprint mit blitzschneller Leistung. Es übertrifft SCM-Tools wie Subversion, CVS, Perforce und ClearCase mit Funktionen wie billiger lokaler Verzweigung, praktischen Staging-Bereichen und mehreren Workflows

(<https://git-scm.com/>)

# Proposed embedded system architectures

## Raspberry Pi

## Jetson Nano

## Google Coral

## Arduino

# Design exploration with shallow convolutional neural networks

## VGG style convolutional neural networks

### Accuracy

### Memory

### Computational cost

### Performance benchmarks

#### Latency

#### Energy consumption

## Resnet style convolutional neural networks

### Accuracy

### Memory

### Computational cost

### Performance benchmarks

#### Latency

#### Energy consumption

# Design exploration with deep convolutional neural networks

## VGG16

## Mobilenet

## Resnet 50

# Results and discussion

## Exploration results

# Conclusions

# References

[wang] Wang, Yue, et al. "Dynamic graph cnn for learning on point clouds." Acm Transactions On Graphics (tog) 38.5 (2019): 1-12.

# Appendix

## Python code

## C code