Intro (3D Object Detection – Problem Statement)

State of the art of the project (Why the project is important, what will it allow)

Tasks, concept and results (Chapter: Methodology, i.e. Machine Learning approaches from dataset to network choice to results and evaluation, include the workflow machine learning diagram with yes forward and no back to a previous step, Chapter: literature project specific information, pointnet diags) (following up the methodology with the results Chapter)

Short summary and outlook (chapter results and future)

**Chapter 1: Introduction**

**Introduction**

The problem of object detection and classification is an inherent issue in Computer Vision. IBM defines Computer Vision as “a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs, and based on those inputs, it can take action.” [IBM]

To address this problem, earlier approaches included feature extraction to identify objects, which are time consuming and highly manual. A later approach that emerged is CNNs (Convolutional Neural Networks) which is a more scalable approach that leverages methods from Linear Algebra, such as Matrix Multiplication to identify patterns in an image.

Within the standard 2D object detection, the information and the features used include edges and RGB colors [IBM] which abstracts the features (corners and edges as well as RGB channels). To position these objects, which is a task under the umbrella of object detection, a bounding box is regressed, as follows up, an image would be an input and the output if a box or a box coordinates that are around or correspond to this object (a car in an image for example) [MaxPlank].

**3D Object Detection:**

Recently, higher level applications like scene understanding and object positioning require a richer form of information such as Point Clouds, in which the regression 3D box provides not only viewpoint information, but also information about the position of the object in the 3D space [MaxPlank].

3D information is available as a set of vectors containing the x, y and z pairs representing the vertices that make up the point cloud. As the data in the pointcloud is random, and is hard to learn from, many proposals preprocess the points to voxel grids as views before network consumption which renders the data voluminous as well as introducing transformations that may change the original data [PointNet]. [VoxelNet] is an example of such approaches. Other appraches include working on the 2D images using CNNs to extract features, as the work with 2D images has been far more extensive than working with point clouds. After that, a 2D box is regressed to give the object’s position, which is further relayed to obtain the box’s 3D coordinates or the object’s z position, an example would be the works of [MobileNetSSD with Realsense Repo][MobileNetSSD repo]

Other approaches propose working directly on the pointcloud, feeding it to the network and performing the classification and/or the detection task [PointNet][PointNet++][VoteNet][BoxNet]. Though the nature of the pointcloud is random and lacks order, these approaches, almost all of them, make use of learning the pointcloud features with a modification in the network, specifically, incorporating symmetrical functions to preprocess the data for learning allowing for the learning to take place from directly consuming the pointcloud [Pointnet][Pointnet++] and they oftentimes employ the architecture of pointnet++ as a backbone to the actual network [H3DNet] [PointNet++][VoteNet][BoxNet].

**Overview of the report:**

In chapter 2 the main problem will be discussed as well as the issues related to 3D object detection in general, chapter 3 will go through project specific literature, adapted proposals and current state of the art. In chapter 4 the proposed methodology will be discussed and chapter 5 will conclude the report with the results discussion and future work.

Online REFS:

[IBM] <https://www.ibm.com/cloud/learn/convolutional-neural-networks#toc-types-of-c-yL2bT7qZ>

[MaxPlank] <https://www.mpi-inf.mpg.de/news/spotlights/understanding-images-videos/3d-object-detection/>

**Chapter 2: Problem Statement**

**Problem Description:**

The main goal is to detect and localize 3D structures (KLTs) such that the information is available to a robot manipulator equipped with a gripper and a 3D Stereo-camera. The structures are set upon one another and have different categories depending on the objects inside, in which case the objects vary from C-Parts (fittings, screws, etc.) to dampers and other automotive parts. End goal is to develop a proof of concept, utilizing tools such as deep learning and 3D object detection to confirm the validity of the approach and culminate an in-house knowledge about the approaches available and their viability.

**Setting:**

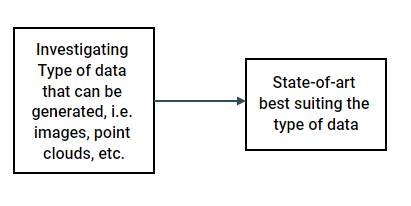
The setting incorporates a manipulator, an end-effector with a greifer and an intel RealSense D435, the information from the camera are extracted, processed by a computer which throws out data fed to the Universal Robot manipulator as commands, which after receiving the position and class information acts accordingly. Fig. () shows the robot manipulator and Fig. () shows the Intel RealSense Depth (stereo-vision) camera.

**Manipulator + Greifer (Pictures)**

**Intel RealSense D4 depth camera**

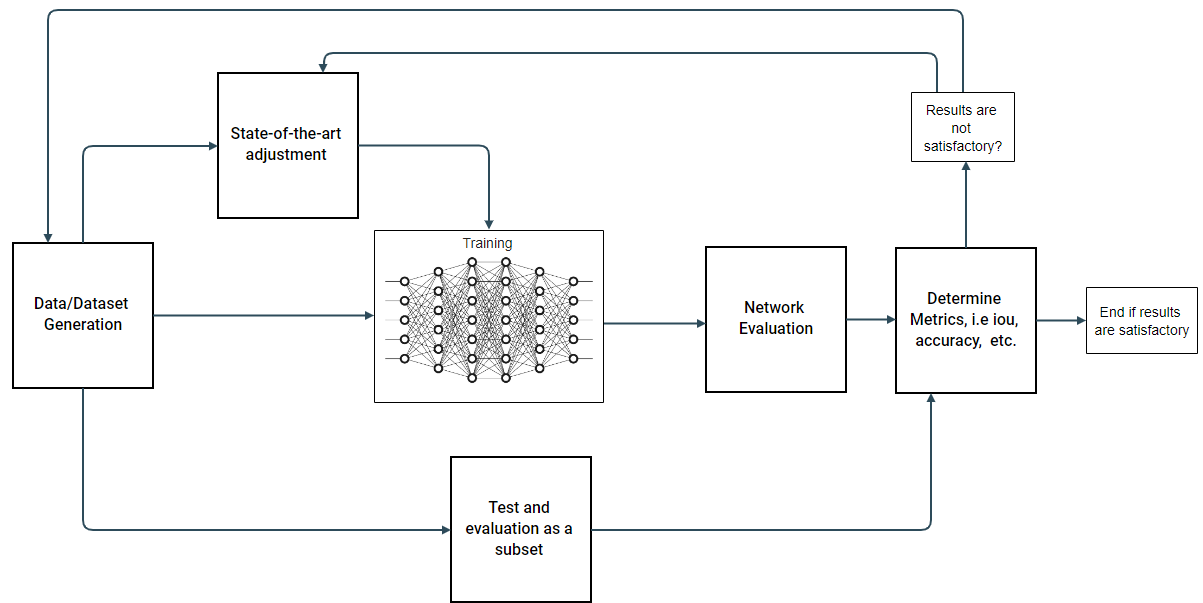
**Targeted Detection Pipeline:**

The detection pipeline includes gathering the data, investigating the state-of-the-art, passing the state of the art to the data gathered, such that the targeted Network is able to learn from this data:



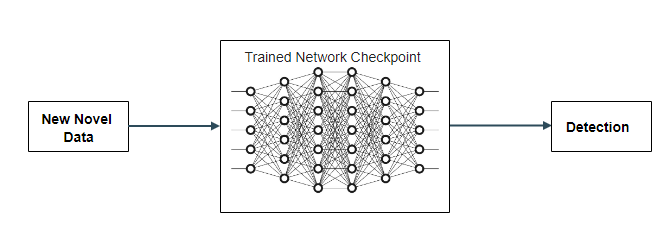
**Data à Network à Training**

After training the network, and for the purpose of testing the employed the state-of-the-art, testing data is prepared in a similar manner as the training data and the necessary code refactoring and edits to the code are carried out, proper metrics are then set, such that in the end, weakness points are determined, and necessary improvements carried out.



**Testing à Network Training Checkpoint à Inference**

After the training has been carried out, the checkpoint where the network has stopped training, i.e. the last training epoch, is saved and an inference is carried out using new novel data. Additionally the network, is the state-the-art allows it, can be tested in real-time



Chapter: Implemented States of the art: votenet – etc

**Chapter 3: State-of-the-art , deep learning and transfer learning**

**MobileNets:**

MobileNets employ a set of depth-wise separable convolutions, with the main point of reducing the computation in the first few layers [MobileNets]. The structure is mainly depthwise separable convolutions, except for the first layer which is a full convolutional network. This structure is carried out to ensure the networks ability to be employed on mobile and embedded devices carrying out inference/realtime inference on the spot, considering the limited set of computational and memory available in such devices. [MobileNets]. Fig. () is an example of vanilla convolutions.

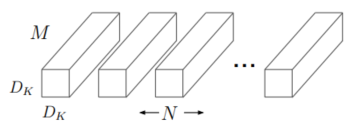


Fig. () [MobileNets]

In MobileNets the depthwise seperable convolutions are applied in 2 steps [MobileNets]:

1. A single Dk xDk x 1 input filter to each channel [MobileNets], as illustrated by Fig. (2)

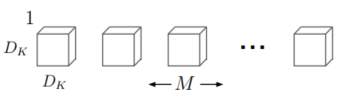


Fig. (2) [MobileNets]

1. A pointwise M x 1x 1 convolution to combine the outputs of the previous filter [MobileNets]

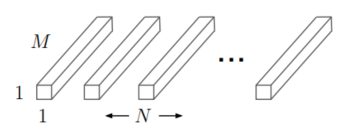


Fig. (3) [MobileNets]

MobileNets which apply 3x3 depthwise seperable convolutions use 8-9 times less computational power, comprated to regular convolutions with a slight reduction in accuracy [MobileNets].

The work of MobileNets is based on [Xception], which introduces the depthwise seperable convolutions, Fig. (4) illustrates the general structure applying both depthwise and pointwise convs.

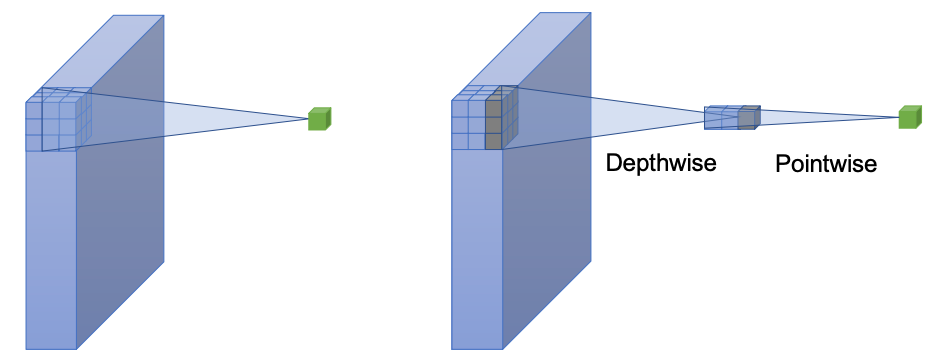


Fig. (4) [Xception]

Fig. (5) demonstrates the architecture of MobileNetV1 according to [MobileNetV1-Figure-Paper; see lit in praktikum stuffs for reference pdf]

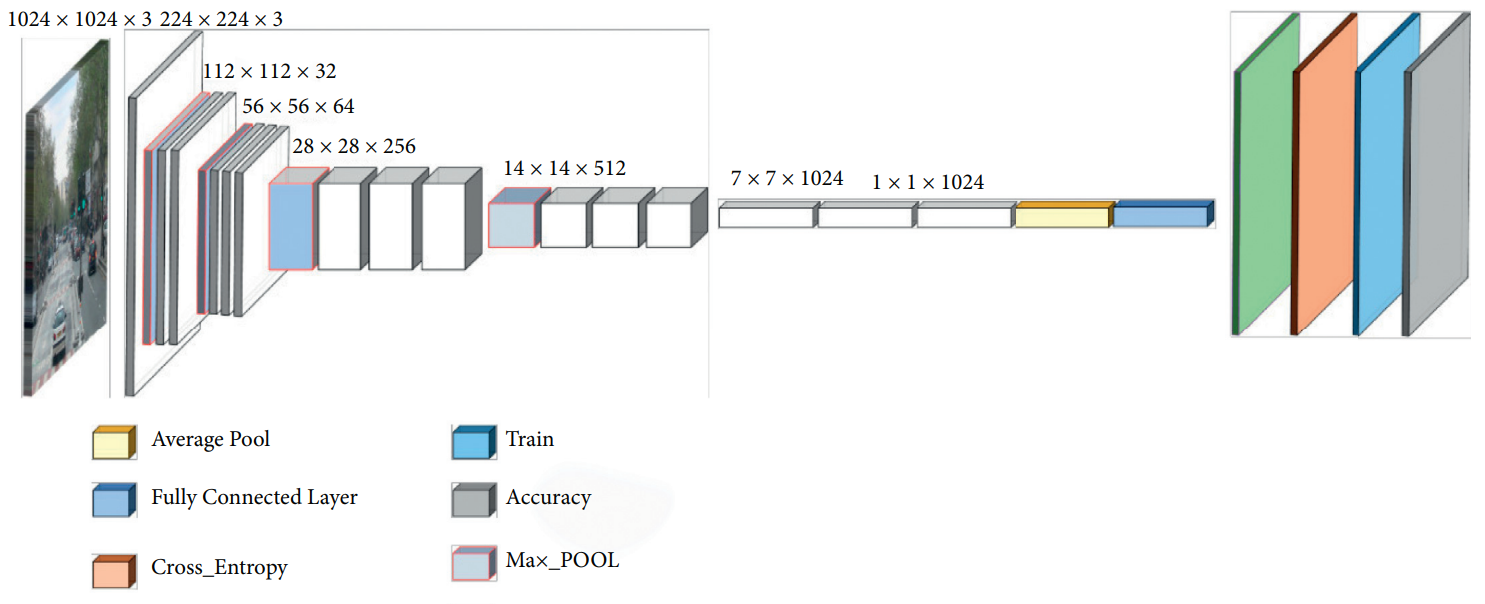


Fig. (5) [MobileNetV1-Figure-Paper; see lit in praktikum stuffs for reference pdf]

MobilenetV1:

MobilenetV2:

*is based on an inverted residual structure where the shortcut connections are between the thin bottleneck layers. The intermediate expansion layer uses lightweight depthwise convolutions to filter features as a source of non-linearity. Additionally, we find that it is important to remove non-linearities in the narrow layers in order to maintain representational power*

The structure of MobileNetV2 differs from that of MobileNetV1. It is based on the concept of inverted residuals, where a shortcut connection is established between bottleneck layers. Bottleneck layers, known as projection layers, and as the name suggests, reduce multi-layer tensors. An intermediate layer is also introduced, it uses the lightweight depthwise convolutions to filter features as a source of non-linearity. Additionally, the authors remove non-linearities in the narrow layers and the reasoning behind it is to maintain the features that are representative [MobileNetV2].

Fig. (6) demonstrated the general architecture of MobileNetV2 according to [https://machinethink.net/blog/mobilenet-v2/]

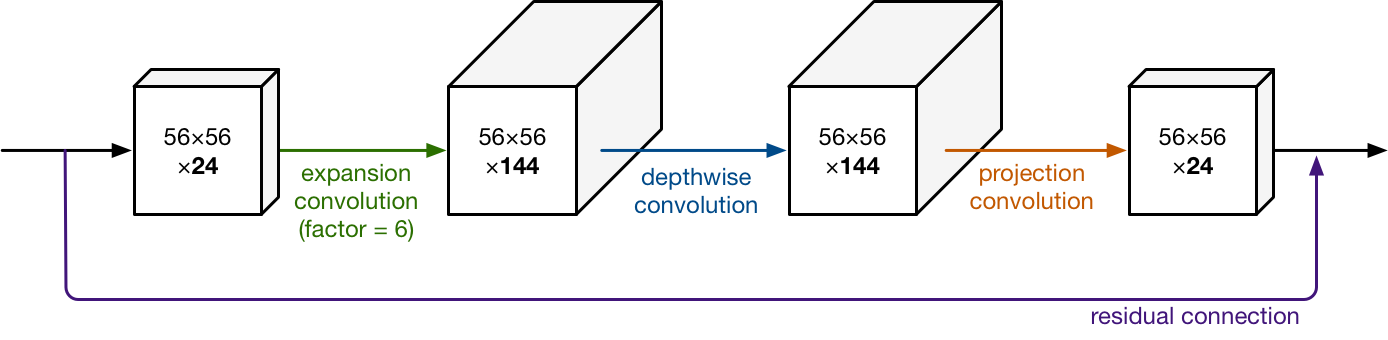


Fig. (6) MobileNetV2 general architecture according to [<https://machinethink.net/blog/mobilenet-v2/>]

Chapter 4: Implementation, tools and data type as well as preparation

Chapter 5: Results, would be short I think