

Unreached Potentials of RGB-D Segmentation

Pascal Benschop

Version of July 4, 2024



Unreached Potentials of RGB-D Segmentation

by

Pascal Benschop

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday July 11, 2024 at 13:00.

Student number: 5052270
Project duration: November 13, 2023 – July 11, 2024
Thesis committee: Dr. J. van Gemert TU Delft, supervisor
Prof. Dr. E. Eisemann, TU Delft

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

1 Introduction to the thesis

Computer vision, the technology enabling machines to 'see' and interpret the visual world, is used in many technologies and devices that we use today. For example, face recognition for unlocking smartphones, quality evaluation of fruits and vegetables with cameras, and even combining sensors such as LiDAR with cameras for autopilot functionality in cars. Images taken by your phone or any camera are typically in an RGB format, where RGB stands for Red, Green, and Blue pixels that together determine the colors in the image. These images can be complemented with depth information from sensors that measure the distance of objects from the camera.

Depth information is beneficial for computer vision tasks because it provides spatial information and robustness to variations in color and lighting. Integrating RGB and depth information, referred to as RGB-D fusion, holds promise for enhancing the capabilities of neural networks (also referred to as models in this thesis) in computer vision tasks. However, current neural networks designed for RGB-D segmentation often underutilize the information from the RGB or depth modality when confronted with unfamiliar variations in the other modality.

This research focuses specifically on image segmentation, a task where neural networks predict the class of each pixel in an image. Effective segmentation allows both humans and computers to precisely identify which parts of an image correspond to specific classes, such as distinguishing the "Person" class in Figure 1.

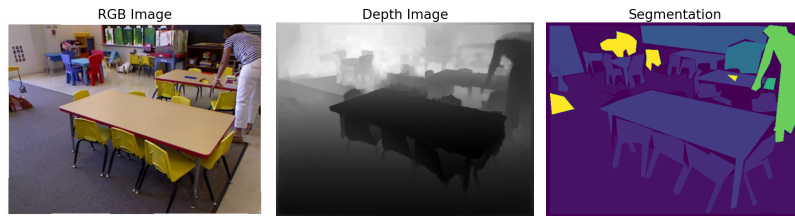


Figure 1. Sample from the NYUDepthV2 dataset showing the RGB image, depth image, and segmentation label, with the "Person" class marked in green.

The organization of this thesis is as follows.

- A research article; with the main results, in the format as acceptable for a suitable computer vision venue.
- Background material in supplementary; to introduce concepts and background for a non-expert audience.

Research article. In the research article, I investigate the following scenarios for neural networks aimed at segmentation using RGB and depth information as input:

1. The capability of neural networks to disregard either the RGB or depth modality when it provides no useful information for the segmentation task. When one modality varies unexpectedly, it can reduce the accuracy of neural networks that use both modalities.
2. The capability of neural networks to ignore irrelevant backgrounds with the use of depth information. The neural network can separate the foreground from the background based on the difference in depth, but this benefit is not always realized.

Improving robustness to unseen variations is essential for developing more reliable and versatile neural networks, particularly for real-world applications. A neural network leveraging RGB-D information should demonstrate adaptability in scenarios where using either RGB or depth alone produces superior results compared to using RGB-D. This flexibility demonstrates the network's ability to remain accurate despite encountering different data conditions.

Background material. In the supplementary materials section, background information for concepts mentioned in the research article is provided. The sections in the technical background explain the concepts and methods used in the research article, ensuring that readers can understand the technical details if they wish to delve deeper. This section covers several key topics. First, it provides an introduction to the basics of neural networks and their application in computer vision. It then delves into RGB-D segmentation, explaining in detail the workings of this technique and the benefits of combining RGB with depth data. Furthermore, the importance of robustness to unseen variations is discussed. Finally, the section explains how and why SynthDet is used to create synthetic datasets for testing.

Unreached potentials of RGB-D segmentation

Pascal Benschop

Sander Gielisse

Jan van Gemert

Delft University of Technology
The Netherlands

Delft University of Technology
The Netherlands

Delft University of Technology
The Netherlands

Abstract

It is commonly believed that image recognition based on RGB improves when using RGB-D, ie: when depth information (distance from the camera) is added. Adding depth should make models more robust to appearance variations in colors and lighting; to recognize shape and spatial relationships while allowing models to ignore irrelevant backgrounds. In this paper we investigate how robust current RGB-D models truly are to changes in appearance, depth, and background where we vary one modality (RGB or depth) and compare RGB-D to RGB-only and depth-only in a semantic segmentation setting. Experiments show that all investigated RGB-D models show some robustness to variations in color, but might severely fail for unseen variations in lighting, spatial position and backgrounds. Our results show that we need new RGB-D models that can exploit the best of both modalities while remaining robust to changes in a single modality. All code and models are publicly available¹.

Keywords: RGB-D, segmentation

1 Introduction

Automatic image recognition models can benefit by exploiting depth information, *i.e.*, the per-pixel surface distance to the camera [4, 19, 20, 24, 26], but this is often taken for granted without explaining why. Here, we make the observation that depth may not always help, see Figure 1 for example, where an RGB-only model (SegFormer [30]) outperforms an RGB-D model (DFormer [33]) that uses the same RGB image with additional depth information. In this paper we investigate information fusion from one modality (depth) to another modality (RGB) –and vice versa– to better understand the effect of adding depth to image recognition models.

The most often mentioned reason for using depth information is the inherent emphasis on geometric information [11, 16, 25, 28, 31–33]. This geometric information can indicate the shape, size, and layout of objects in 3D space. The depth differences can identify the (non)existence of object boundaries in case of occlusions or abrupt appearance changes. A model can also learn absolute depth values to exploit typical foreground-background scene configurations. In addition, depth can help make computer vision models more robust to variations in color and illumination [1, 8, 11, 18, 21]. Conceptually, this makes sense because a depth map lacks color, texture, light, and shadows, making depth invariant

¹Url: <https://github.com/pascalbenschopTU/Testing-RGBD-segmentation>

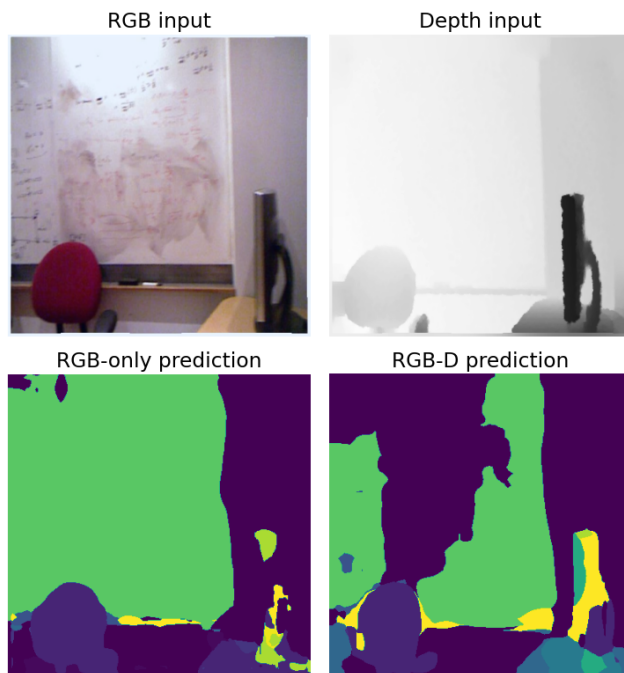


Figure 1. While depth is useful for computer vision tasks in most scenarios, there exist cases where depth has a negative effect on the prediction. In this example of a whiteboard (green label), using only RGB as input does better than using RGB together with depth through RGB-D input. In this paper we investigate why adding more information can deteriorate results in state-of-the-art deep RGB-D models.

to appearance changes. Depth is an incredibly powerful percept, and adding depth information to RGB appearance is only expected to improve accuracy. Or.. does it?

In principle, having both RGB and depth modalities available should do better for image recognition than having just one of them. Yet, current image recognition models are based on deep learning, which potentially makes them vulnerable to shortcut-learning [7] where the network does not learn true generalization, but instead exploits spurious correlations between the training data and the output. We hypothesize that such spurious correlations exist between RGB and depth modalities, which might explain why RGB-only outperforms RGB-D in Figure 1. Our hypothesis is inspired by findings in other modalities, where multi-modal neural networks often prioritize the dominant modality [9, 27], yet, the impact of

variations in less informative modalities, like depth, on overall accuracy remains unclear. To address these issues, we conduct experiments with synthetic and real-world RGB-D data, to evaluate the segmentation accuracy of neural networks across variations in RGB, depth, and backgrounds. The empirical findings from these experiments aim to answer the following research questions:

- **RQ1: How do neural networks designed for RGB-D segmentation handle unseen variations in either modality?**
- **RQ2: How does variation in the background affect the prediction of the foreground?**

The contributions of this work are as follows. 1. We give, for the first time, empirical evidence for existing claims about RGB-D robustness to variations in color and illumination compared to RGB-only models. 2. We evaluate the robustness of RGB-D compared to depth-only models. 3. We test RGB-D robustness to background changes. We evaluate 3 state-of-the-art RGB-D models and start in a fully controlled synthetic setting, followed by experiments on real data to corroborate our findings. Our conclusions are important for the field, as we call for new RGB-D models that can use the best of both modalities while remaining robust to changes in a single modality.

2 Related work

On the usefulness of depth information in computer vision. Jiawei et al. [35] question the need for depth estimation in salient object detection. Their network uses depth as supervision in the training stage. While this method can improve upon models performing salient object detection with only RGB as input, unseen variations in RGB will likely break the model without ground-truth depth as input. They mention that the errors in predicted depth typically occur in semantically ambiguous regions, highlighting the need for accurate ground-truth depth data. Yan et al. [13] highlight the usefulness of depth information in varying lighting conditions for indoor visual odometry. They demonstrate that traditional methods without depth information struggle to find matching points between images with significant lighting changes. Depth data can enhance computer vision tasks, but depth sensors present challenges. Lucia et al. [14] noted issues with detecting small objects, depth shadows from blocked IR light, and undefined depth values for objects too close or far. Martin et al. [3] highlighted noise in depth sensors without salient objects and stressed the need for careful RGB-depth calibration.

Exploiting depth for ignoring irrelevant backgrounds. Humans also benefit from depth in separating foreground from background, as Nonie et al. found [6]. In an experiment where participants need to select a target symbol which is similar to distracting symbols, separating these symbols on a different depth plane decreases the average search time.

While neural networks have no trouble with finding a target between distractions, placing the target on the background instead of the foreground could break the model. Enrique et al. [5] propose an adaptation of the Codebook algorithm which can be implemented efficiently and is robust to shadows, lighting conditions and variations in backgrounds, both in RGB and depth. While this would be an ideal baseline to improve foreground-background separation in segmentation networks, the Codebook algorithm works optimal on a sequence of frames, whereas training data is often random images with no temporal coherence. Christopher et al. [29] use two networks where the first creates initial masks from depth only, and the second network refines the masks together with RGB input for unseen instance segmentation. Separating depth from RGB in the first model ensures that noise in RGB does not hinder the model from separating foreground from background. While this approach enhances robustness to unseen backgrounds, it does not fully exploit the potential advantages offered by modality fusion techniques.

Synthetic RGB-D data. Several authors have created synthetic RGB-D datasets with different purposes. John et al. [15] created SceneNet RGB-D, a dataset providing pixel-perfect ground truth data. The RGB-D images are generated in a pipeline from physics engine to rendering engine and are customizable in terms of lighting, textures, scene layout and camera trajectory within a scene. This dataset is then used to improve computer vision models focused on indoor settings without needing (expensive) real-world data. Aakash et al. [17] generate a high-resolution RGB-D dataset by rendering scenes from the game Grand Theft Auto V. The RGB-D images are then used to improve transformer-based models that perform monocular depth estimation. Mohammad et al. [12] create a new synthetic dataset called synROD, which contains object models that closely resemble the categories defined by the real-world dataset ROD. Their dataset is used to evaluate domain adaptation methods, where the source data is labeled while the target data is not. No research has yet used synthetic RGB-D data as a benchmark to evaluate the accuracy of neural networks on datasets with variations in either modality and in backgrounds that were not present during training.

RGB-D models. We evaluate three different state-of-the-art RGB-D semantic segmentation models: TokenFusion [26], CMX [34] and DFormer [33]. TokenFusion [26] is a multi-modal transformer that adaptively fuses multiple single-modal transformers. They employ a strategy of pruning multiple single-modal transformers which can be re-utilized for multimodal fusion. CMX [34] is a cross-modal fusion transformer that is able to fuse multiple modalities for segmentation. With a Cross-Modal Feature Rectification Module bi-modal features are calibrated and better aligned for the fusion module. DFormer [33] uses an efficient architecture and employs RGB-D pretraining to get impressive segmentation

results on benchmarks with less parameters and operations per second than competitors.

RGB-only and Depth-only models. For a neural network that uses only RGB information as input, SegFormer [30] is used as a baseline. SegFormer is an RGB segmentation neural network that unifies Transformers with lightweight MLP decoders. Since no models exist that perform segmentation using only depth information, the DFormer model is adapted to create a variant that takes only depth information as input.

3 Experimental Setting

Neural networks that combine information from RGB and depth are tasked with performing semantic segmentation, which involves labeling each pixel with a class value. The objectives of the experiments to be conducted are to provide insights in how current multi-modal neural networks handle variations in either modality, and to identify unreached potentials of RGB-D segmentation. Synthetic data is used to introduce controlled variations, highlighting the limitations of current state-of-the-art RGB-D neural networks. Realistic data is then employed to validate the findings from experiments with synthetic data, ensuring the results are applicable to real-world scenarios.

Synthetic RGB-D datasets For the experiments conducted on synthetic data, datasets are created with the help of SynthDet Unity project [10]. The project contains scripts called Randomizers that can alter the scene from which dataset samples are generated. For example, these randomizers can decide how objects are placed, apply scaling, rotation, lighting and camera effects to the scene and more. The project also already contains 3D assets, namely 63 high quality models of commonly found grocery products. For more details and parameters on creating datasets see Appendix A, Table 7.

Implementation details All experiments are conducted with python, using the PyTorch framework for building neural networks. We make all PyTorch code, datasets and Unity scripts freely available online². All models used in the experiments are initialized with pretrained weights provided by the respective authors of the original models. Due to differences between synthetic datasets and the benchmark datasets on which the models were trained, new hyperparameters were selected through hyperparameter optimization using Optuna [2]. The hyperparameter search ranges can be seen in Table 5 in Appendix A. All results are expressed in mean Intersection over Union (mIoU) scaled to (0, 100), unless stated otherwise. This metric is commonly used for segmentation, it measures the overlap of the predicted segmentation with the ground truth.

In all experiments using synthetic data, smaller model sizes are employed compared to those used in experiments with realistic data. The simpler nature of synthetic datasets

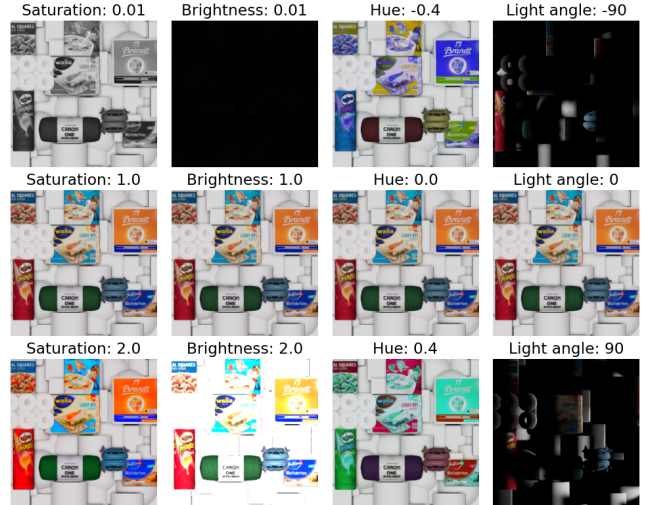


Figure 2. Example of groceries objects with changes in saturation, brightness, hue and light angle. Best viewed in color. Contrary to the other properties, the angle of the light source cannot be simulated for augmentation during training.

allow the models to achieve almost perfect accuracy with fewer parameters. For a detailed comparison of all the models and the respective configurations see Appendix A, Table 6.

4 Does adding depth enhance robustness against unseen variations of RGB?

This section addresses RQ1 for the RGB modality, focusing on how the network handles unseen RGB variations during testing. The hypothesis is that RGB-D segmentation models are more robust to appearance changes not encountered during training compared to RGB-only models. Since depth is invariant to color, texture, luminance, and shadows, it may enhance the network’s robustness to scenery changes. To test this, variations in saturation, brightness, hue, and light angles were examined, representing common and significant real-world appearance changes. This selection provides a comprehensive, though not exhaustive, evaluation of the network’s robustness to these visual variations.

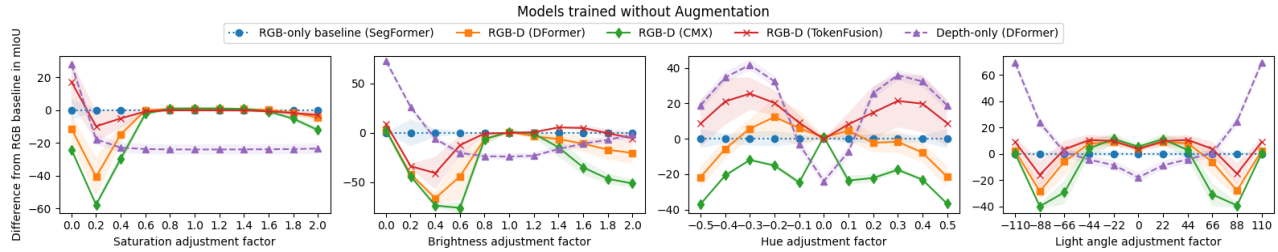
Since there is little variation in colors present in the dataset (see the middle row of Figure 2), ColorJitter augmentation [23] is applied during the training process. This technique involves modifying the colors of images in a controlled way to improve the robustness of neural networks.

4.1 Setup

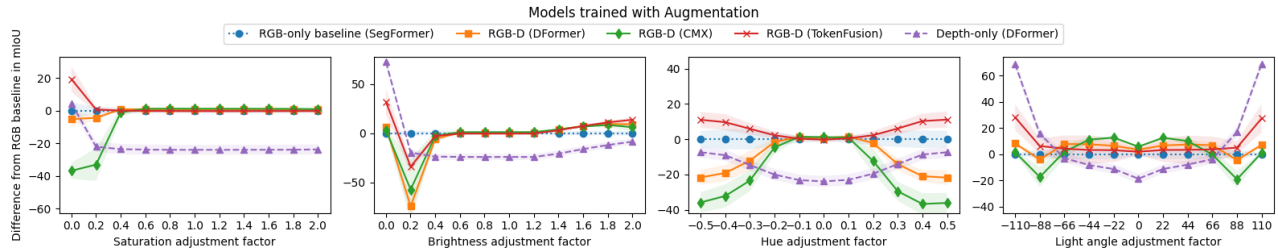
See Figure 2 for examples of variations that are applied to the dataset in the following ways:

1. Objects with color **saturation** values in the test set that are both lower than $\min(S_{\text{train}})$ and higher than $\max(S_{\text{train}})$.

²Url: <https://github.com/pascalbenschopTU/Testing-RGBD-segmentation>



(a) Models trained without ColorJitter augmentation and tested on extreme variations. For every property that is adjusted there are values at which either the RGB or depth only model outperforms the RGB-D models.



(b) Models trained with ColorJitter augmentation and tested on extreme variations. For the variations in brightness and light angle, there exist situations where RGB-D models perform worse than RGB or depth alone.

Figure 3. Variation in RGB can unnecessarily affect the models trained on RGB and depth. Standard deviations, represented by shaded regions, are computed from 3 repeats.

2. Objects where the **brightness** of the pictures in the test set is both lower than $\min(B_{\text{train}})$ and higher than $\max(B_{\text{train}})$.
3. Objects with a set of color **hues** C_{test} where $C_{\text{test}} \notin C_{\text{train}}$.
4. Simulated shadows from varying **light angles**, which cannot be simulated by augmenting the training data.

The dataset used in this experiment consists of grocery objects placed in front of a neutral background of white objects. The objects in this dataset are consistent in rotation and scale, ensuring that a model trained solely on depth can segment these adequately. With variations in rotation and scale, some box shaped objects are hard to distinguish using only depth information. The depth-only variant of Dformer is used for comparison with RGB-D models.

4.2 Results

For all experiments the RGB-D (and depth only) segmentation networks are compared to a RGB-only segmentation network. The prediction accuracy of the RGB-only baseline is set to 0, and the difference in accuracy per data point relative to the baseline is plotted for all other models.

The results from testing the models on a dataset with varying factors such as Saturation, Brightness, Hue, and Light Angle are depicted in Figure 3. Models trained both with and without ColorJitter augmentation were evaluated under extreme variations of these properties. The findings indicate that without ColorJitter augmentation, RGB-D models do

not demonstrate increased robustness to changes in saturation, brightness, or light angles compared to the RGB-only model. Even when trained with ColorJitter augmentation, only the TokenFusion model consistently outperforms the SegFormer RGB-only model.

However, a notable exception occurs when brightness is reduced to 20% of its original value. In this specific scenario, all RGB-D models perform worse than both the RGB-only and depth-only models. Figure 4 provides a detailed illustration of this shortcoming, highlighting that the depth features are negatively affected by variations in RGB features. This interaction may explain why the fusion of RGB and depth features results in lower performance compared to models trained exclusively on either depth or RGB.

Furthermore, in extreme low-light conditions, RGB-D models fail to effectively utilize depth information, leading to inaccurate predictions. This failure suggests that the depth features are not only influenced by the compromised RGB features, but also that the fusion strategy employed by the models is not robust enough to handle variations which are not encountered during training. Consequently, while RGB-D models hold potential, the current implementations require better fusion mechanisms to improve robustness under challenging conditions.

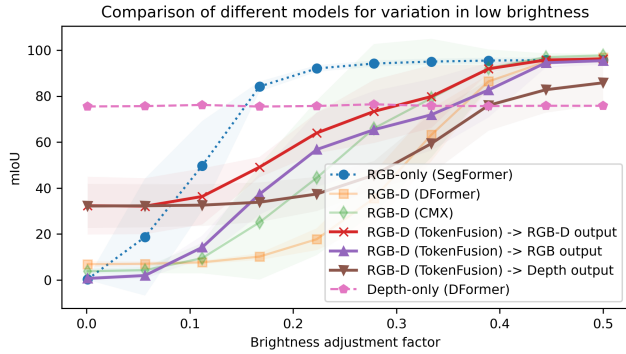


Figure 4. The figure illustrates the performance of models trained with augmentation at very low brightness values. Notably, all RGB-D models are less accurate in low-brightness scenarios compared to an RGB-only model. The different TokenFusion outputs reveal that while the depth input remains consistent across brightness levels, the predictions from the depth output branch are affected. This suggests that the interactions between the RGB and depth inputs, when faced with unseen variations in brightness, likely contribute to the observed degradation in performance. Consequently, the RGB-D models are less effective in handling the variations introduced by low brightness conditions.

5 Are networks robust to unseen variations in spatial position?

This section aims to address RQ1 for the depth modality, varying the spatial positions of objects between training and testing. A neural network should be able to adapt to different arrangements in the depth map, such as objects at varying distances from the camera. To test this hypothesis five datasets are created where objects are placed at different positions and ranges along the axis parallel to the direction of the camera.

The different dataset settings can be seen in Figure 5, objects are placed at specific ranges between the near and far clipping plane of the Unity camera’s view frustum. See Figure 6 for an example of the scene. In Unity, the camera’s view frustum defines the visible area in the 3D scene. The frustum is a pyramid-like shape with the top cut off. The camera lens is positioned at the top, the near clipping plane is at the cut-off point, and the base of the pyramid is at the far clipping plane. The planes determine the minimum and maximum distances from the camera at which objects are rendered. Objects within this range are visible in the camera view. Varying the spatial position of objects in a scene affects the depth from the camera but also the size of objects in RGB. This difference is accounted for by comparing the relative performance of an RGB-D model and a RGB model, as both models are affected by changes in scale. The challenge

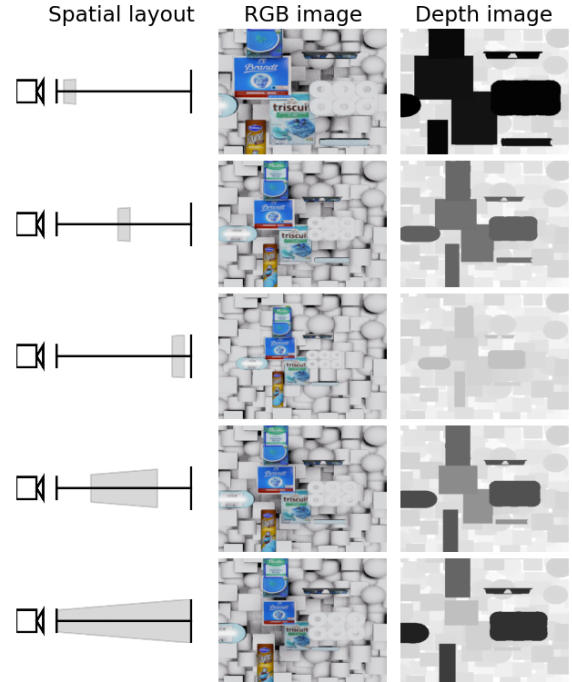


Figure 5. Visualization of foreground objects at various distances from the camera. In the depth images (second column), darker pixels indicate objects closer to the camera, while lighter pixels indicate objects further away. Objects are placed between the near plane and the far plane of the camera’s view. From left to right, objects are placed near the camera, midway, at the far plane, between the first and third quartiles of the range, and spanning the entire range.

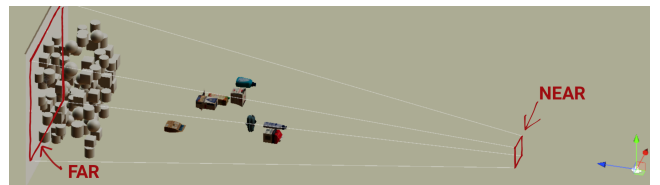







Figure 6. Example of spacing objects between the near and far clipping planes in a scene, outside these planes objects are not rendered.

is whether RGB-D networks can handle large variations in absolute depth values.

5.1 Results

Table 1 displays the results of this experiment for both RGB-only and RGB-D models combined. The RGB-only model decreases in accuracy with large variations in scale (from objects placed far away to nearby), yet it can generally handle changes in spatial position well. In contrast, RGB-D models are significantly more affected by changes in spatial position

Table 1. Results of training RGB-only and RGB-D networks on varying spatial position. Results are displayed in mean IoU score with standard deviations computed from 3 repeats. The changes in depth have little effect on an RGB-only model in contrast to RGB-D models. Except for the full range setting, there is at least one instance in all other settings that the models are trained on where the RGB-only model outperforms the RGB-D models.

Trained on	Predicted on				
					
RGB-D	91.99 ±0.99	88.63 ±6.42	10.32 ±10.7	80.57 ±12.1	74.45 ±6.86
RGB-only	91.40 ±0.27	94.69 ±0.33	89.59 ±0.58	94.72 ±0.34	93.53 ±0.33
RGB-D	78.33 ±6.99	95.77 ±1.44	16.31 ±19.8	89.52 ±5.95	76.43 ±5.34
RGB-only	87.28 ±0.32	96.79 ±0.11	93.98 ±0.29	96.57 ±0.10	94.03 ±0.14
RGB-D	43.80 ±7.09	90.07 ±2.17	94.97 ±0.86	87.37 ±2.13	73.60 ±4.37
RGB-only	61.46 ±5.18	93.29 ±1.87	94.55 ±2.07	92.15 ±2.16	83.66 ±3.47
RGB-D	85.42 ±2.91	95.96 ±1.66	24.65 ±32.4	95.96 ±1.61	86.76 ±3.53
RGB-only	88.85 ±0.33	96.56 ±0.28	94.22 ±0.56	96.51 ±0.28	94.77 ±0.29
RGB-D	89.29 ±1.14	96.60 ±0.39	93.04 ±1.06	96.64 ±0.34	94.89 ±0.34
RGB-only	90.02 ±0.85	95.10 ±0.93	92.74 ±1.33	95.11 ±0.94	93.99 ±0.98

compared to the RGB-only model. The RGB-D models experience the greatest loss in accuracy when there are large changes in absolute depth values. For results per model see Appendix C, Table 8, 9 and 10.

These results highlight potential improvements for RGB-D segmentation models to handle variations in depth that are not present in the training data. While the changes to depth are artificial and might not occur in real-world scenarios, a miscalibration of a depth sensor or a software error could result in similar, unnecessary reductions in accuracy.

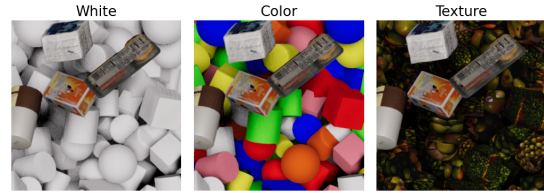


Figure 7. Datasets where the foreground is exactly the same but the background is changed, best viewed in color. From left to right, the objects in the backgrounds are white, randomly colored, and textured.

6 How do neural networks use depth for separating foreground from background?

Depth provides extra information about the spatial position of objects, so it can be used to separate foreground from background. The hypothesis for RQ2 is that current neural networks designed for RGB-D segmentation rely on background features to segment foreground objects. This dependency can cause a reduction in accuracy when the network faces unseen backgrounds.

6.1 Evaluation method

To test the hypothesis that neural networks rely on irrelevant backgrounds, three datasets are generated with different backgrounds only observed in the RGB modality. For all datasets, the foreground segmentation objects are kept identical (see Figure 7). These foreground objects are placed with variations in spatial position, rotation, and scale. This setup forces an RGB-D network to rely on RGB input for foreground segmentation, while it could use depth information to distinguish the foreground from the background. The objective for a neural network with RGB and depth inputs is to accurately segment the foreground objects, even when the backgrounds differ from the backgrounds in the training dataset.

6.2 Results

Models trained on RGB-D inputs are still influenced by unseen backgrounds that provide no useful information for the foreground. Optimal neural networks should score the same mean IoU score for the different backgrounds, but this is not the case as can be seen in Table 2. The models trained on the dataset with the most noisy background, colored objects, are more robust to variations in background. Nonetheless, there is still a performance drop when testing on out-of-distribution backgrounds, which indicates room for improvement.

6.3 Potential of RGB-D segmentation

To highlight the limitations of existing RGB-D models, the models are compared to a two-stage pipeline based on adaptations of the DFormer [33] model, which can take a single

Table 2. Combined results of training RGB-D neural networks on datasets with varying backgrounds. Standard deviations are computed from 3 repeats per model. Ideally, the models should achieve consistent results regardless of the background.

Background trained on	Background predicted on		
	White BG	Color BG	Texture BG
White BG	91.75 ±4.77	76.68 ±11.7	80.43 ±8.04
Color BG	90.52 ±1.65	91.07 ±1.51	85.64 ±4.36
Texture BG	80.39 ±8.01	75.96 ±9.54	91.56 ±2.01

Table 3. Demonstrating the limitations of current RGB-D models using a two-stage adaptation of DFormer: first segmenting foreground from background using depth-only, then predicting foreground using RGB-only. This two-stage adaptation shows superior handling of background variations.

Method for all datasets	Background predicted on		
	White BG	Color BG	Texture BG
Depth → RGB	90.20 ±1.59	89.43 ±1.52	89.97 ±1.70

input modality. Initially, the adapted model uses only depth information to distinguish foreground objects from the background, effectively filtering out irrelevant RGB data. Subsequently, another adaptation of the model uses only RGB information to segment the foreground without distraction from the background. The results can be seen in Table 3. Compared to RGB-D models, this pipeline achieves higher accuracy on unseen backgrounds but lower accuracy when tested on the same background as in the training data.

7 Experiments on NYUDepthV2 dataset

While the results on synthetic datasets are interesting, models trained on real data such as from the NYUDepthV2 RGB-D dataset [22] might be more robust because there are larger variations in the appearance and shape of objects. The dataset contains 1449 densely labeled pairs of aligned RGB and depth images. The images are captured using a Microsoft Kinect

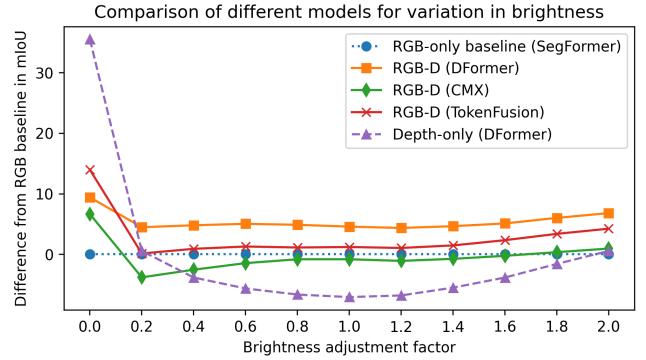


Figure 8. Varying the brightness across the NYUDepthV2 sets and testing models trained on the dataset without modifications. While the DFormer model is the most robust overall, the TokenFusion model can better handle the scenario where the image is dark.

sensor, which may introduce noise in the depth data. To mitigate this, noisy regions in the depth map are filled in with information from nearby regions. Similar experiments testing variation in appearance, spatial position and background are conducted on the NYUDepthV2 dataset.

7.1 RQ1: Variations in RGB

The RGB-D models were affected by large variations in hue, saturation, and brightness, albeit to a lesser extent than an RGB-only model. These results support the claims that depth can help make computer vision models more robust to changes in color and illumination. However, there is room for improvement in RGB-D segmentation; for instance, under very low brightness conditions, all models struggled to effectively utilize depth information. The DFormer model trained only with depth information outperforms both models trained With RGB-only and RGB-D inputs (see Figure 8). The results for variation in hue and saturation can be seen in Appendix B in figures 12 and 13.

7.2 RQ1: Variations in spatial position

The RGB-D models are more affected by variations in depth, for example when the absolute values of the depth map differ from the default range on which the model is trained on. See for example Figure 9 where different ranges of the depth map are visualized. Training the model on the original NYUDepthV2 dataset and testing on differently scaled depth reduces the accuracy of models trained with RGB-D inputs. There is room for improvement because the same model trained with RGB-only inputs performs better than the multi-modal model for unseen variations of depth.

As an ablation study, the results are also plotted at different spatial positions, for example close to the camera. The models are tested on different depth ranges at varying spatial

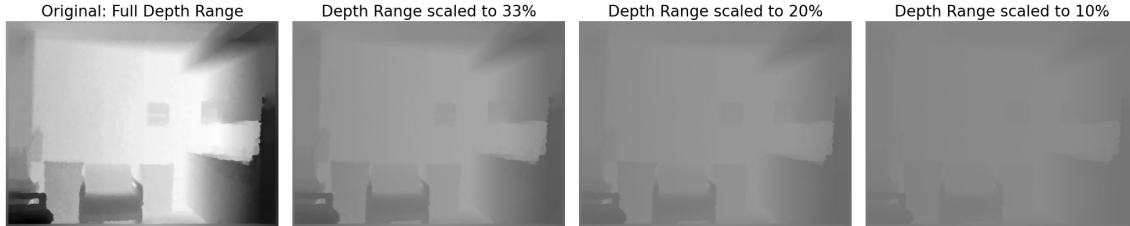


Figure 9. Example of scaling the depth to different min and max values, for these all min values are 0.

Table 4. Testing models on different scales of depth. RGB-D10 stands for RGB-Depth scaled to 10% of the original range (see Figure 9). While the spatial relationships between objects remain intact, the models trained on RGB-D are impacted by a difference in depth values.

Model	RGB-D	RGB-D33	RGB-D20	RGB-D10
DFormer	47.36	43.60	41.17	37.47
TokenFusion	44.43	40.88	38.82	35.83
CMX	42.41	38.38	35.88	32.17
SegFormer (RGB)			43.34	
DFormer (RGB-only)			44.16	

positions (ranging from near the camera to far from the camera). The models show some variation for different distances from the camera, but are more affected by the range of depth. The results per model can be seen in Appendix C in figures 14, 15 and 16.

While the mean Intersection over Union (IoU) score for most classes degrades when the depth is scaled to 33% of its original range, the IoU score for the whiteboard class actually improves by around 9%. Interestingly, this class is also segmented less accurately by the RGB-D model compared to the RGB-only model (see Figure 1). This finding highlights that depth can also negatively affect a multi-modal neural network.

7.3 RQ2: Variations in background

There is also room for improvement in ignoring irrelevant backgrounds with depth information. To demonstrate this potential for improvement, an experiment was conducted to assess the impact of adding noisy backgrounds behind objects belonging to a specific class. The procedure for this experiment can be seen in Algorithm 1. An example of an image with corresponding depth where the background behind class "Person" is replaced with a noisy background is visible in Figure 10. Only pixels that have a depth value larger than the furthest depth of any instance of the class of interest are replaced. The depth from the noisy background is also adjusted such that every pixel in the noisy background has a greater (further) depth value than the pixels within the region defined by the class of interest.

In Figure 11 the difference between the DFormer model tested on classes with noisy backgrounds and classes with the original background is plotted in mean accuracy per class. The results for other models, which exhibit similar reductions in accuracy, can be found in Appendix D in Figure 17. An optimal RGB-D model would predict each class with a noisy background at least as well as with the original background. Interestingly, some classes actually benefit from the added noisy backgrounds, the increased contrast from the noisy background can cause a pop-out effect for the objects from those classes.

Algorithm 1 Adapting the dataset with noisy backgrounds

```

1: Input: dataset, classes, model
2: Output: Class-wise accuracy comparison
3:  $bg \leftarrow$  background (class)
4:  $dataset\_bg, dataset\_orig \leftarrow copy(dataset)$ 
5: for class in classes do
6:   for (img, depth, label) in dataset do
7:     if label contains class then
8:        $class\_mask[i,j] \leftarrow \begin{cases} 1 & \text{if } label[i,j] = class \\ 0 & \text{else} \end{cases}$ 
9:        $label[class\_mask] \leftarrow$  class
10:       $label[\sim class\_mask] \leftarrow bg$ 
11:       $max\_depth \leftarrow \max(depth[class\_mask])$ 
12:       $bg\_mask \leftarrow depth > max\_depth$ 
13:       $bg\_img[bg\_mask] \leftarrow$  noisy_img
14:       $bg\_img[\sim bg\_mask] \leftarrow$  img
15:       $bg\_depth[bg\_mask] \leftarrow$  noisy_depth
16:       $bg\_depth[\sim bg\_mask] \leftarrow$  depth
17:     else
18:        $label \leftarrow bg$ 
19:     end if
20:     update  $dataset\_bg$  with (bg_img, bg_depth, label)
21:     update  $dataset\_orig$  with (img, depth, label)
22:   end for
23:   test model on  $dataset\_bg$ 
24:   test model on  $dataset\_orig$ 
25: end for
26: compare accuracy per class

```

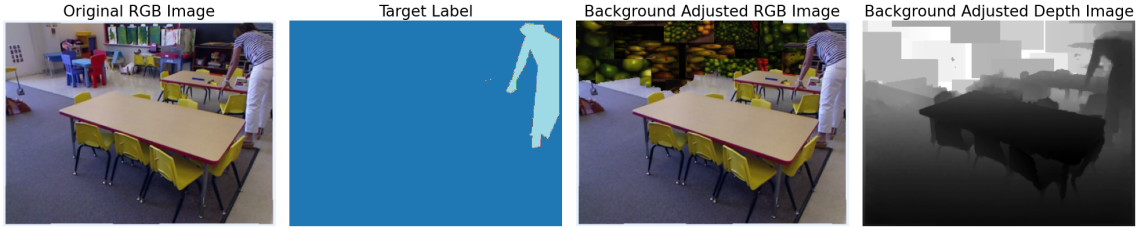


Figure 10. Example of adding a noisy background behind the class "Person" in the image based on depth.

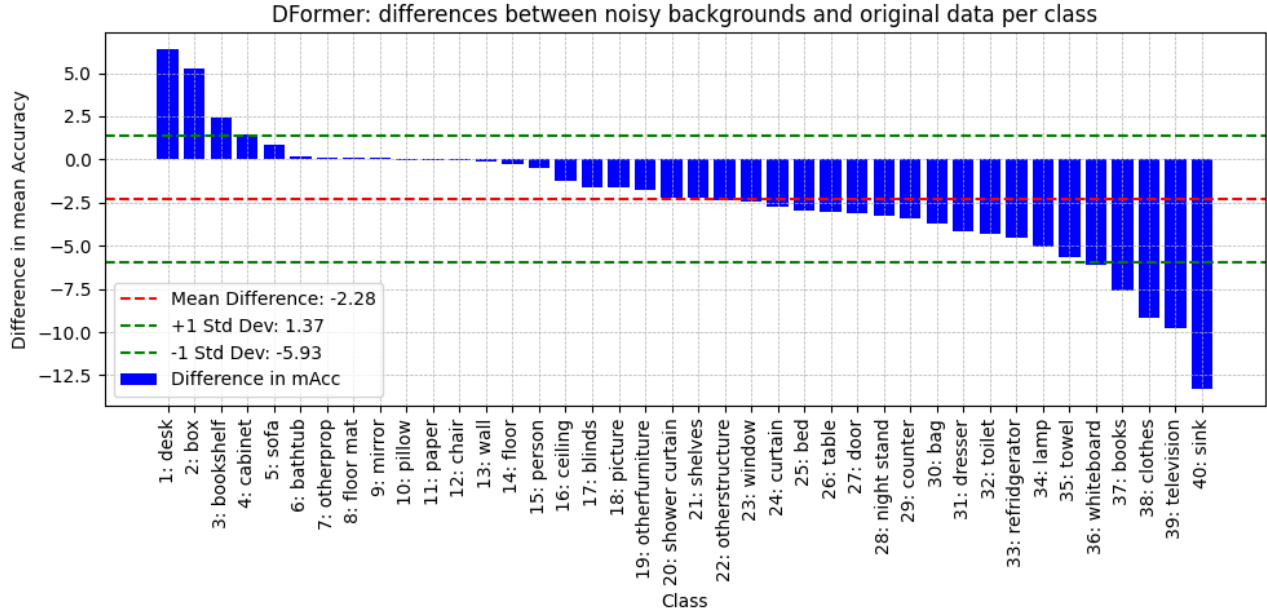


Figure 11. Experiment with the DFormer RGB-D model: adding a noisy background behind classes in the image based on depth. The network gets unnecessarily affected by the noisy background.

8 Conclusions and future work

The RGB-D neural networks employed in this research exhibit limited robustness to variations in either modality when trained with relatively small datasets. The networks also degrade in accuracy when confronted with irrelevant backgrounds which were not in the training data. Under conditions where the model predicts on data similar to the training distribution, combining RGB with depth information consistently improves the mean prediction accuracy. However for some classes, such as the whiteboard class in the NYUDepthV2 dataset, the addition of depth information deteriorates the accuracy of a multi-modal neural network. When a variation is applied to depth that the model has not been trained on, the predictions of the whiteboard class actually improve in accuracy. These findings highlight unreached potentials of RGB-D segmentation. Addressing these issues when developing robust multi-modal neural networks can

enhance the reliability of computer vision in real-world applications and help bridge the gap between machine learning and machine intelligence.

While this work highlights room for improvements for RGB-D neural networks performing segmentation, it does not provide concrete solutions that can help improve the networks. Increasing the amount of training data could be beneficial for most cases, however it does not guarantee optimal model behaviour in unseen scenarios. Additionally, depth-aware methods that weigh the importance of neighboring pixels based on the relative depth difference could help with ignoring irrelevant backgrounds. To conclude, future research evaluating the effectiveness of depth information in various challenging scenarios can guide the development of smarter and more robust multi-modal neural networks.

Acknowledgments

To Sander, for explaining in detail how some of my ideas could work, and others definitely not. And to Jan, for guiding

my ideas and experimental work towards systematically addressing the research questions of my thesis.

References

- [1] Andreas Aakerberg, Kamal Nasrollahi, and Thomas Heder. 2017. Improving a deep learning based RGB-D object recognition model by ensemble learning. In *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*. 1–6. <https://doi.org/10.1109/IPTA.2017.8310101>
- [2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. 2623–2631. <https://doi.org/10.1145/3292500.3330701>
- [3] Martin Brenner, Napoleon Reyes, Teo Susnjak, and Andre Barczak. 2023. RGB-D and Thermal Sensor Fusion: A Systematic Literature Review. *IEEE Access* PP (01 2023), 1–1. <https://doi.org/10.1109/ACCESS.2023.3301119>
- [4] Isa El Doori. 2019. Deep Learning Based Image Segmentation of RGB-D Data in Warehouse Automation. Available at <http://resolver.tudelft.nl/uuid:d95c85ed-f96b-4956-ad17-e8e5194a3e8c>.
- [5] Enrique J. Fernandez-Sanchez, Javier Diaz, and Eduardo Ros. 2013. Background Subtraction Based on Color and Depth Using Active Sensors. *Sensors* 13, 7 (2013), 8895–8915. <https://doi.org/10.3390/s130708895>
- [6] Nonie Finlayson, Roger Remington, James Retell, and Philip Grove. 2013. Segmentation by depth does not always facilitate visual search. *Journal of vision* 13 (07 2013). <https://doi.org/10.1167/13.8.11>
- [7] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. 2020. Shortcut learning in deep neural networks. *Nature Machine Intelligence* 2, 11 (2020), 665–673.
- [8] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, and Jianwei Wan. 2014. 3D Object Recognition in Cluttered Scenes with Local Surface Features: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 11 (2014), 2270–2287. <https://doi.org/10.1109/TPAMI.2014.2316828>
- [9] Yu Huang, Junyang Lin, Chang Zhou, Hongxia Yang, and Longbo Huang. 2022. Modality Competition: What Makes Joint Training of Multi-modal Network Fail in Deep Learning? (Provably). In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, 9226–9259. <https://proceedings.mlr.press/v162/huang22e.html>
- [10] You-Cyuan Jhang, Adam Palmar, Bowen Li, Saurav Dhakad, Sanjay Kumar Vishwakarma, Jonathan Hogins, Adam Crespi, Chris Kerr, Sharmila Chockalingam, Cesar Romero, Alex Thaman, and Sujoy Ganguly. 2020. Training a performant object detection ML model on synthetic data using Unity Perception tools. <https://blogs.unity3d.com/2020/09/17/training-a-performant-object-detection-ml-model-on-synthetic-data-using-unity-computer-vision-tools/>.
- [11] Yabei Li, Junge Zhang, Yanhua Cheng, Kaiqi Huang, and Tieniu Tan. 2018. DF2Net: A Discriminative Feature Learning and Fusion Network for RGB-D Indoor Scene Classification. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence* (New Orleans, Louisiana, USA) (AAAI’18/IAAI’18/EAAI’18). AAAI Press, Article 862, 8 pages.
- [12] Mohammad Reza Loghmani, Luca Robbiano, Mirco Planamente, Kiru Park, Barbara Caputo, and Markus Vincze. 2020. Unsupervised Domain Adaptation Through Inter-Modal Rotation for RGB-D Object Recognition. *IEEE Robotics and Automation Letters* 5, 4 (2020), 6631–6638. <https://doi.org/10.1109/LRA.2020.3007092>
- [13] Yan Lu and Dezhen Song. 2015. Robustness to lighting variations: An RGB-D indoor visual odometry using line segments. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 688–694. <https://doi.org/10.1109/IROS.2015.7353447>
- [14] Lucia Maddalena and Alfredo Petrosino. 2018. Background Subtraction for Moving Object Detection in RGBD Data: A Survey. *Journal of Imaging* 4, 5 (2018). <https://doi.org/10.3390/jimaging4050071>
- [15] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J. Davison. 2017. SceneNet RGB-D: Can 5M Synthetic Images Beat Generic ImageNet Pre-training on Indoor Segmentation?. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 2697–2706. <https://doi.org/10.1109/ICCV.2017.292>
- [16] Lingfeng Qiao, Zhongliang Jing, Han Pan, Henry Leung, and Wuji Liu. 2021. Private and common feature learning with adversarial network for RGBD object classification. *Neurocomputing* 423 (2021), 190–199. <https://doi.org/10.1016/j.neucom.2020.07.129>
- [17] Aakash Rajpal, Noshaba Cheema, Klaus Illgner-Fehns, Philipp Slusallek, and Sunil Jaiswal. 2023. High-Resolution Synthetic RGB-D Datasets for Monocular Depth Estimation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 1188–1198. <https://doi.org/10.1109/CVPRW59228.2023.00126>
- [18] Xiaofeng Ren, Liefeng Bo, and Dieter Fox. 2012. RGB-(D) scene labeling: Features and algorithms. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2759–2766. <https://doi.org/10.1109/CVPR.2012.6247999>
- [19] Lukas Schneider, Manuel Jasch, Björn Fröhlich, Thomas Weber, Uwe Franke, Marc Pollefeys, and Matthias Ratsch. 2017. Multimodal Neural Networks: RGB-D for Semantic Segmentation and Object Detection. In *Image Analysis*, Puneet Sharma and Filippo Maria Bianchi (Eds.). Springer International Publishing, Cham, 98–109.
- [20] Daniel Seichter, Mona Kohler, Benjamin Lewandowski, Tim Wengefeld, and Horst-Michael Gross. 2021. Efficient RGB-D Semantic Segmentation for Indoor Scene Analysis. 13525–13531. <https://doi.org/10.1109/ICRA48506.2021.9561675>
- [21] Ling Shao, Ziyun Cai, Li Liu, and Ke Lu. 2017. Performance evaluation of deep feature learning for RGB-D image/video classification. *Information Sciences* 385–386 (2017), 266–283. <https://doi.org/10.1016/j.ins.2017.01.013>
- [22] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. 2012. Indoor Segmentation and Support Inference from RGBD Images. In *Computer Vision – ECCV 2012*, Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 746–760.
- [23] Luke Taylor and Geoff Nitschke. 2018. Improving deep learning with generic data augmentation. In *2018 IEEE symposium series on computational intelligence (SSCI)*. IEEE, 1542–1547.
- [24] Michal Varga and Ján Jadlovský. 2019. Evaluation of Depth Modality in Convolutional Neural Network Classification of RGB-D Images. 18 (01 2019), 26–31. <https://doi.org/10.15546/aei-2018-0029>
- [25] Weiyue Wang and Ulrich Neumann. 2018. Depth-Aware CNN for RGB-D Segmentation. In *Computer Vision – ECCV 2018*, Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (Eds.). Springer International Publishing, Cham, 144–161.
- [26] Yikai Wang, Xinghao Chen, Lele Cao, Wenbing Huang, Fuchun Sun, and Yunhe Wang. 2022. Multimodal Token Fusion for Vision Transformers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 12176–12185. <https://doi.org/10.1109/CVPR52688.2022.01187>
- [27] Nan Wu, Stanislaw Jastrzebski, Kyunghyun Cho, and Krzysztof J Geras. 2022. Characterizing and Overcoming the Greedy Nature of Learning in Multi-modal Deep Neural Networks. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka,

- Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, 24043–24055. <https://proceedings.mlr.press/v162/wu22d.html>
- [28] Zongwei Wu, Guillaume Allibert, Christophe Stolz, and Cédric Demonceaux. 2021. Depth-Adapted CNN for RGB-D Cameras. In *Computer Vision – ACCV 2020*, Hiroshi Ishikawa, Cheng-Lin Liu, Tomas Pajdla, and Jianbo Shi (Eds.). Springer International Publishing, Cham, 388–404.
- [29] Christopher Xie, Yu Xiang, Arsalan Mousavian, and Dieter Fox. 2019. The Best of Both Modes: Separately Leveraging RGB and Depth for Unseen Object Instance Segmentation. *ArXiv abs/1907.13236* (2019). <https://api.semanticscholar.org/CorpusID:199000781>
- [30] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. 2024. SegFormer: simple and efficient design for semantic segmentation with transformers. In *Proceedings of the 35th International Conference on Neural Information Processing Systems (NIPS '21)*. Curran Associates Inc., Red Hook, NY, USA, Article 924, 14 pages.
- [31] Yajie Xing, Jingbo Wang, Xiaokang Chen, and Gang Zeng. 2019. 2.5D Convolution for RGB-D Semantic Segmentation. In *2019 IEEE International Conference on Image Processing (ICIP)*. 1410–1414. <https://doi.org/10.1109/ICIP.2019.8803757>
- [32] Yajie Xing, Jingbo Wang, and Gang Zeng. 2020. Malleable 2.5D Convolution: Learning Receptive Fields Along the Depth-Axis for RGB-D Scene Parsing. In *Computer Vision – ECCV 2020*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer International Publishing, Cham, 555–571.
- [33] Bowen Yin, Xuying Zhang, Zhongyu Li, Li Liu, Ming-Ming Cheng, and Qibin Hou. 2023. DFormer: Rethinking RGBD Representation Learning for Semantic Segmentation. *arXiv preprint arXiv:2309.09668* (2023).
- [34] Jiaming Zhang, Huayao Liu, Kailun Yang, Xinxin Hu, Ruiping Liu, and Rainer Stiefelhagen. 2023. CMX: Cross-Modal Fusion for RGB-X Semantic Segmentation With Transformers. *IEEE Transactions on Intelligent Transportation Systems* 24, 12 (2023), 14679–14694. <https://doi.org/10.1109/TITS.2023.3300537>
- [35] Jiawei Zhao, Yifan Zhao, Jia Li, and Xiaowu Chen. 2020. Is Depth Really Necessary for Salient Object Detection?. In *Proceedings of the 28th ACM International Conference on Multimedia (Seattle, WA, USA) (MM '20)*. Association for Computing Machinery, New York, NY, USA, 1745–1754. <https://doi.org/10.1145/3394171.3413855>

Table 5. Model training settings and hyperparameter ranges for tuning. *For large models the batch size is kept between 4 and 8.

Parameter	Synthetic	NYUDepthV2
Images for training	200	795
Epochs	40	100
Warm up epochs	10	10
Number of hyperparameter tuning trials	15	20
Tuneable parameters		
Learning Rate	(1e-5, 1e-3)	(1e-5, 1e-3)
Batch Size	[4,8,16*]	[4,8,16*]
LR Power	(0.8, 1.0)	(0.8, 1.0)
Momentum	(0.9, 0.99)	(0.9, 0.99)
Weight decay	(1e-4, 1e-2)	(1e-4, 1e-2)

Table 6. Model configurations used, each first row displays the names of the backbones and the second row the amount of parameters of the model for each backbone. M stands for million.

Model	Synthetic	NYUDepthV2
DFormer	Small	Base
parameters:	18.7M	29.5M
CMX	mit_b1	mit_b2
parameters:	44.5M	66.6M
TokenFusion	mit_b1	mit_b2
parameters:	15.0M	26.0M
SegFormer	mit_b1	mit_b2
parameters:	13.7M	27.4M

A Experiment Setting

All images are generated at a resolution of 400 by 400 pixels, this saves both time in rendering and preprocessing since the images do not have to be resized for the neural networks. Using scripts, the depth maps with depth measured in Unity meters from the camera are converted to (normalized) depth images. The format used by Unity to store the labeled data is then converted to a COCO (Common Objects in Context) format, which is subsequently transformed into a format compatible with the training and testing framework inspired by DFormer [33].

For an overview of the parameters used for creating synthetic datasets see Table 7. The settings used for training the model are displayed in Table 5 and the model configurations in Table 6.

Table 7. The settings used in Unity to create the synthetic datasets. *The depth is normalized using min-max normalization in code to (0,1). "U" stands for "Uniform", "z-pos" for spatial position on the z axis.

Experiment	Near-Far plane	bg z-pos	bg color	rotation (x,y,z)	fg z-pos	fg scale
RGB OOD	90 - 110	U(98-100)*	white	(-90, 0, 0)	U(96-98)*	0.5
Spatial						
Close	50 - 110	U(100-110)	white	(-90, 0, 0)	U(50-55)	0.5
Midway	50 - 110	U(100-110)	white	(-90, 0, 0)	U(72.5-77.5)	0.5
Far	50 - 110	U(100-110)	white	(-90, 0, 0)	U(95-100)	0.5
Medium Range	50 - 110	U(100-110)	white	(-90, 0, 0)	U(65-85)	0.5
Whole Range	50 - 110	U(100-110)	white	(-90, 0, 0)	U(50-100)	0.5
FgBg Separation						
White Bg	50 - 110	U(90-110)*	white	U(0-360) all axes	U(50-90)*	U(0.4-0.7)
Color Bg	50 - 110	U(90-110)*	Colors	U(0-360) all axes	U(50-90)*	U(0.4-0.7)
Texture Bg	50 - 110	U(90-110)*	Green Textures	U(0-360) all axes	U(50-90)*	U(0.4-0.7)

B Variations in RGB on NYUDepthV2

See figures 12 and 13 for more results from RGB-D models tested on variations applied to the RGB images of the NYUDepthV2 dataset.

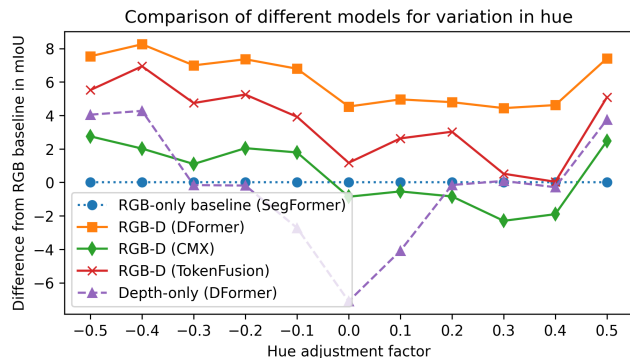


Figure 12. Testing the robustness of RGB-D models compared to a baseline RGB models on varied saturation levels applied to the NYUDepthV2 dataset. The dataset is in its original state when the saturation level is at 0. All models that use RGB-D information show increased robustness to very low and high saturation values compared to the baseline.

C Varying spatial position results per model

Tables 8, 9 and 10 show results of the variation in spatial position experiment with synthetic data for each RGB-D model.

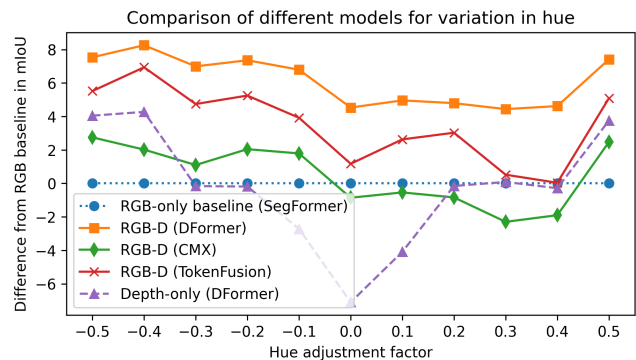


Figure 13. Testing the robustness of RGB-D models compared to a baseline RGB models on varied hue values applied to the NYUDepthV2 dataset. The dataset is in its original state when the hue value is at 0. Only DFormer shows increased robustness across all hue values compared to the baseline.

C.1 NYUDepthV2 Spatial experiments

Figures 14, 15 and 16 provide results for models trained on the NYUDepthV2 dataset and tested on the dataset with variations applied to depth.

D Results of adapting the background per class on NYUDepthV2 for different models

Figure 17 displays results for the TokenFusion, CMX and SegFormer model from varying the background of classes.

Table 8. Results of training the DFormer model and varying the spatial dimension.

Trained on	Predicted on				
	92.0 ±0.12	94.84 ±0.33	23.82 ±7.17	92.61 ±1.01	82.24 ±1.35
	71.73 ±7.70	94.03 ±0.25	43.66 ±6.94	93.87 ±0.27	82.95 ±2.06
	44.19 ±4.17	88.78 ±1.25	95.10 ±0.48	86.85 ±1.47	74.59 ±1.81
	84.66 ±0.04	94.15 ±0.25	70.41 ±1.96	94.23 ±0.21	90.1 ±0.11
	88.93 ±0.47	96.31 ±0.39	93.65 ±0.70	96.47 ±0.38	94.93 ±0.45

Table 9. Results of training the CMX model and varying the spatial dimension.

Trained on	Predicted on				
	90.81 ±0.32	85.17 ±1.82	1.40 ±0.09	68.82 ±1.23	67.45 ±0.38
	80.94 ±0.69	97.05 ±0.04	1.42 ±0.11	82.82 ±5.73	71.41 ±2.41
	38.92 ±9.19	92.84 ±0.80	95.89 ±0.04	88.94 ±2.76	70.89 ±6.52
	88.44 ±0.21	98.14 ±0.09	1.66 ±0.32	98.08 ±0.08	87.95 ±0.92
	88.23 ±0.03	97.01 ±0.06	93.38 ±0.35	96.96 ±0.06	94.86 ±0.11

Table 10. Results of training the TokenFusion model and varying the spatial dimension.

Trained on	Predicted on				
	93.17 ±0.09	85.88 ±7.87	5.74 ±3.09	80.27 ±12.5	73.66 ±5.37
	82.32 ±4.51	96.24 ±1.11	3.85 ±2.41	91.87 ±1.99	74.92 ±2.36
	48.28 ±2.22	88.6 ±0.62	93.91 ±0.01	86.33 ±0.21	75.31 ±0.45
	83.15 ±3.25	95.59 ±0.2	1.87 ±0.56	95.55 ±0.21	83.23 ±1.86
	90.7 ±0.67	96.48 ±0.16	92.09 ±1.18	96.05 ±0.19	94.87 ±0.37

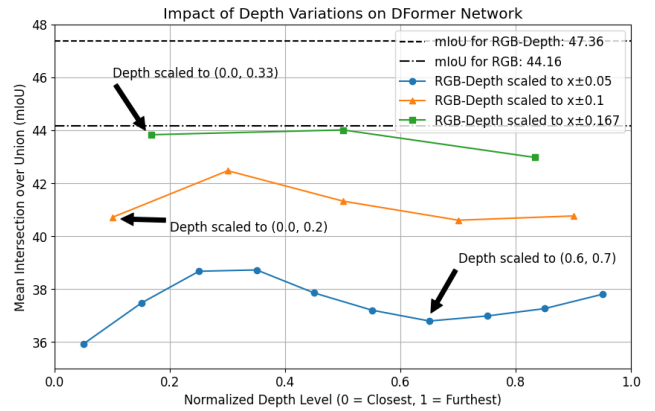


Figure 14. Varying the scale of depth on the NYUDepthV2 dataset, the DFormer model drops in accuracy for depth with a different scale than trained on.

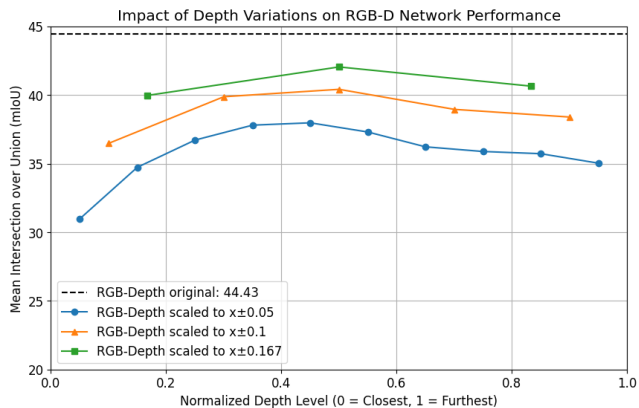


Figure 15. Varying the scale of depth on the NYUDepthV2 dataset, the TokenFusion model drops in accuracy for depth with a different scale than trained on. It is the most robust out of the three RGB-D segmentation neural networks.

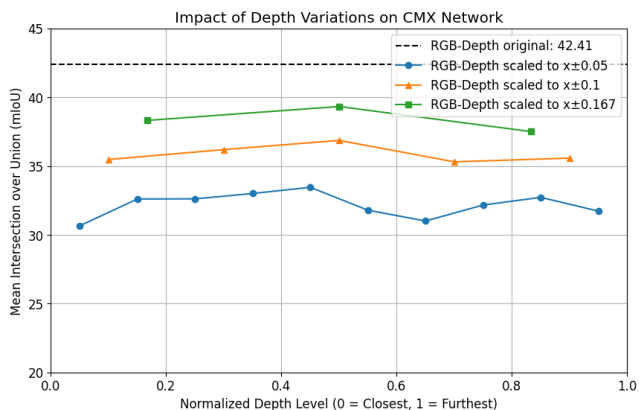
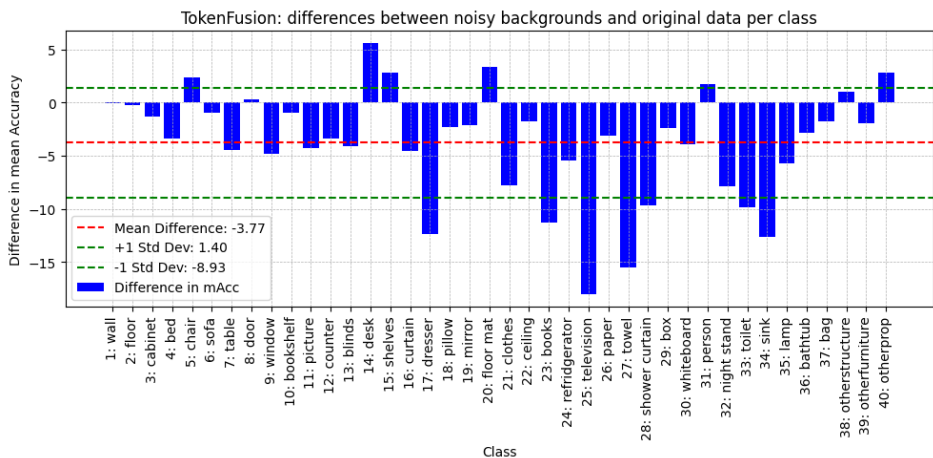
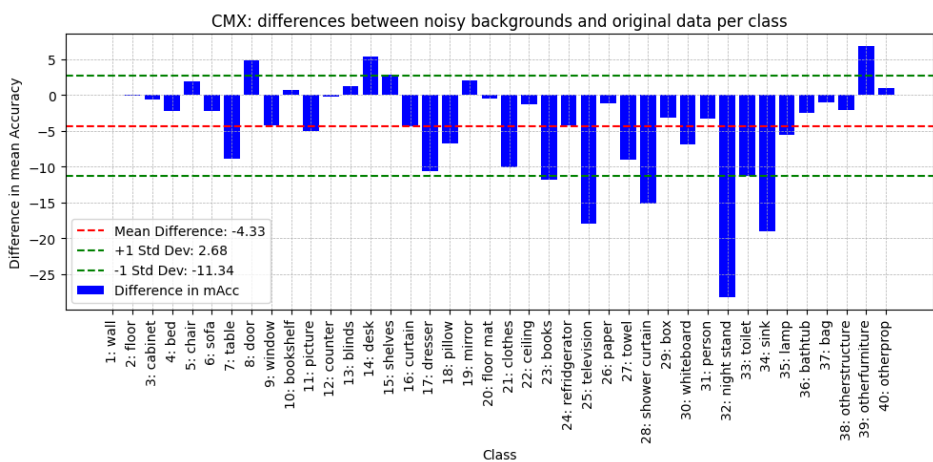


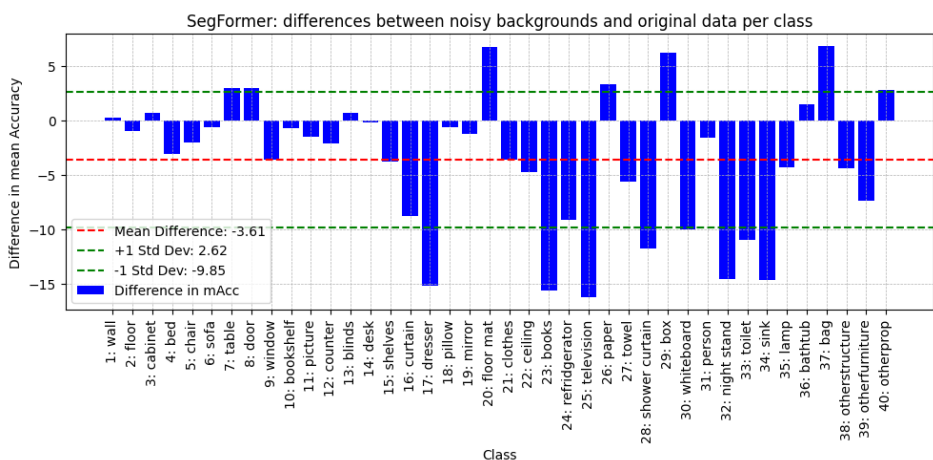
Figure 16. Varying the scale of depth on the NYUDepthV2 dataset, the CMX model drops in accuracy for depth with a different scale than trained on. It is the least robust out of the three RGB-D segmentation neural networks, with a reduction of almost 5 percent in mean Intersection over Union for depth scaled between 33% and 66% of the original values.



(a) TokenFusion model (RGB-D)



(b) CMX model (RGB-D)



(c) SegFormer model (RGB)

Figure 17. Adding a noisy background behind classes in the image based on depth. The CMX and TokenFusion models get unnecessarily affected by the noisy background, especially since the RGB SegFormer model is less affected.

Supplementary Materials

Additional Information Supporting the Thesis

A Neural networks

Neural Networks are like giant mathematical formula's with parameters that can be tuned to approximate almost any distribution of data. As we have seen in recent times, ChatGPT (based on the Generative Pre-trained Transformer model) can learn to understand basic language. But first going back to basics: the method of least squares or linear regression can be seen as the predecessor of the modern neural networks, for these methods the mean squared errors between calculated outputs and targets are minimized by adjusting weights.

In its simplest form, linear regression involves finding the best-fitting straight line through a set of points on a graph. This line is determined by the equation:

$$y = mx + b$$

where:

- y is the predicted value,
- x is the input variable,
- m is the slope of the line, and
- b is the bias term or y-intercept, which is the value of y when x is 0.

The goal is to adjust m and b such that the line minimizes the difference between the predicted values and the actual data points.

Understanding linear regression is crucial because it lays the groundwork for more complex models like neural networks. At its core, a neural network can be seen as an extension of linear regression, where instead of a single line, we have multiple layers of transformations that enable the model to capture complex, non-linear relationships in the data.

Not all problems can be solved with linear methods. Take for example the first major non-linear problem: learning the XOR function (see figure 18). Given points for class True: (0,1) and (1,0) and for class False: (0,0) and (1,1) there is no single line that can separate these classes in such a way that the classes are completely separated.

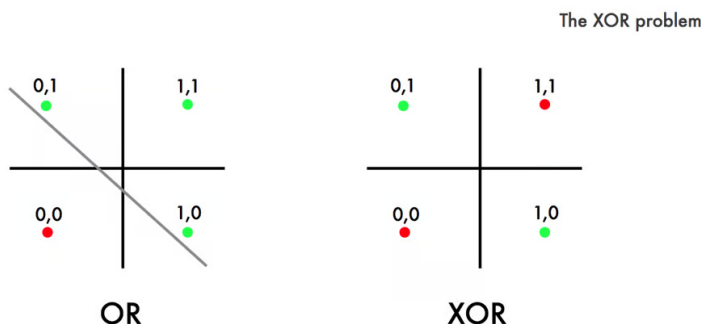


Figure 18. No linear decision boundary can separate the classes in the XOR problem. Source: [demystifying the xor problem](#)

Neural networks introduce non-linearity through an activation function, allowing it to solve more complex problems. An example of an activation function is the sigmoid, that exponentially goes to 0 when the input is smaller than 0 and exponentially to 1 when the input is larger than 0. In this research the networks mainly use Rectified Linear Unit and the Leaky Rectified Linear Unit, which are methods that change directly at the boundary (0).

The simplest neural network is the perceptron, which can be thought of as a single-layer linear classifier. A perceptron takes multiple input signals, applies weights to them, sums them up, and passes the result through an activation function to produce an output (see equation 1).

$$y = \sigma \left(\sum_{i=1}^n w_i x_i + b \right) \quad (1)$$

Where:

- σ : denotes the activation function.
- $\sum_{i=1}^n w_i x_i + b$ calculates the weighted sum of inputs and adds a bias term for a linear transformation of the input data.

The first deep learning neural network was the multilayer perceptron (MLP), in this network perceptrons are used in multiple layers that together define a pipeline from input to output. The weights in the perceptrons were first optimized by stochastic gradient descent, which is an iterative algorithm that approximates the gradient and updates the weights in the

direction of this gradient. Thereafter backpropagation was invented, an important algorithm that calculates the gradient of a loss function with respect to the weights of a network starting from the last layer. A loss function is a function that takes an input and a ground truth, and computes how far the input is from the ground truth. By starting from the last layer, the derivatives of the input with respect to the layer after the current can be cached and used in the gradient calculations of the current layer, avoiding redundant calculations. The result of backpropagation is a set of gradients for each parameter in the neural network, these gradients represent the change of the loss function with respect to each parameter. The parameters can then be optimized by performing a step, controlled by the learning rate, in the opposite direction of the gradient to reduce the loss.

A.1 Loss functions

Several loss functions exist that compute how far a prediction is from the ground truth, for segmentation a popular loss function is Cross entropy loss. Cross entropy loss is also commonly used as a loss function for classification tasks. It measures the performance of a classification model whose output is a probability value between 0 and 1.

For a single instance, the cross entropy loss is defined as:

$$L(y, \hat{y}) = - \sum_{i=1}^C y_i \log(\hat{y}_i)$$

where:

- y is the true label, often represented as a one-hot encoded vector.
- \hat{y} is the predicted probability vector.
- C is the number of classes.
- y_i is the binary indicator (0 or 1) if class label i is the correct classification for the current instance.
- \hat{y}_i is the predicted probability of class i .

For a dataset with N instances, the total cross entropy loss is the average of the individual losses:

$$L = -\frac{1}{N} \sum_{n=1}^N \sum_{i=1}^C y_{n,i} \log(\hat{y}_{n,i})$$

where:

- N is the number of instances in the dataset.
- $y_{n,i}$ is the binary indicator if class i is the correct classification for instance n .
- $\hat{y}_{n,i}$ is the predicted probability of class i for instance n .

A.2 Components of a Neural Network

Neural networks are intricate systems composed of several essential components, each contributing uniquely to their functionality. The **backbone** or **encoder** serves as the foundational framework, initiating the process by extracting fundamental features from raw (or preprocessed) input data. This initial feature extraction is pivotal in tasks like image recognition, where layers within the backbone identify patterns such as edges and textures.

Once the encoder has encoded the input data into a latent representation (see Figure 19), the **decoder** comes into play. This component decodes the learned representations, translating them into understandable outputs that align with the task requirements. For instance, the decoder outputs predictions per pixel based on the encoded features. The encoder and decoder component consist of smaller building blocks such as convolution and attention.

A.3 Building Block: Convolution

Given the basic notion of a neural network, how can this be applied to an image? It is possible to convert all pixels in the image into a giant vector which can be fed into a feed-forward network, but then all spatial relations between pixels are lost. With the introduction of the convolution operation neural networks can exploit local correlation in images. Convolution works like a sliding window that moves across the image, performing a mathematical operation at each position. Convolution in a neural network is expressed as following:

$$(I * K)(x, y) = \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} I(x+i, y+j) \cdot K(i, j) \quad (1)$$

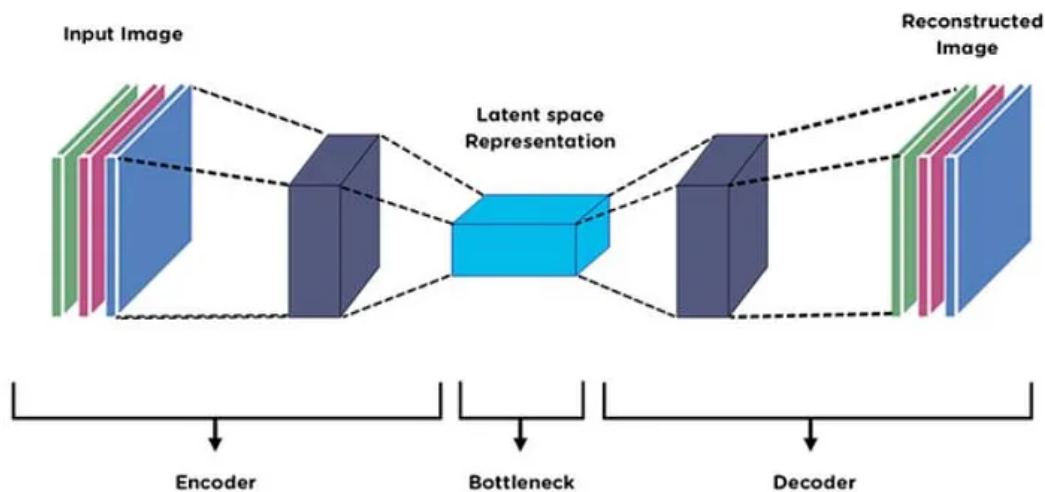


Figure 19. A visualization of an encoder and a decoder in a neural network. Source: [A Perfect guide to Understand Encoder Decoders in Depth with Visuals](#)

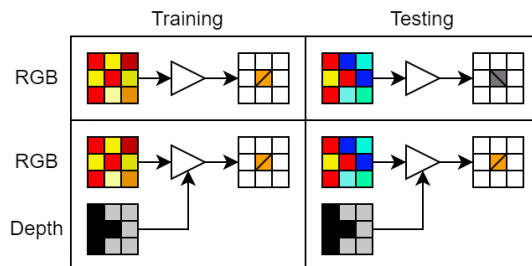


Figure 20. A visual simplified example of convolution with an averaging kernel (calculating the average color from the 3 by 3 grid). In this example depth can help neural networks ignore irrelevant backgrounds (the blue and green pixels). The diagonal line in the output is added for readers who experience trouble seeing differences between colors. A diagonal line to the top left corner indicates correct behavior, while a diagonal line to the top right corner indicates incorrect influence from the background.

Where I is the input image, K is the convolutional kernel or filter, and H and W are the height and width of the kernel. See an example of a 3 by 3 convolutional kernel (with averaging weights) in Figure 20. Another advantage of using convolution is that the weights of the filter are shared across all pixels, which reduces the complexity of the neural network.

When a convolutional kernel is applied to an image, it processes small overlapping regions of the image, computing a dot product (like in Equation 1) between the kernel and the region it covers. This operation results in a new image, often of the same size as the original (if padding is used), where each pixel value represents a feature extracted from the corresponding region of the input image. This transformed image is referred to as a feature map.

In a convolutional layer, multiple such kernels (or filters) are applied simultaneously. Each filter is designed to detect specific features or patterns within the input image. For instance, one filter might learn to detect edges in various orientations, while another might recognize textures or specific shapes. As the convolutional layer learns through training, each filter adjusts its parameters to become sensitive to different aspects of the input data.

By stacking these feature maps along the depth dimension, the convolutional layer captures a rich representation of the input image, highlighting different aspects of its content. This hierarchical extraction of features through convolution and pooling operations forms the basis of deep learning architectures like convolutional neural networks (CNNs), enabling them to automatically learn and discriminate complex patterns within images.

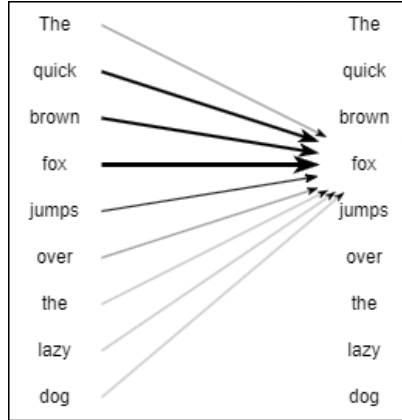


Figure 21. A visualization of how the attention operation theoretically can assign importance of each part of the input for the word "Fox".

A.4 Building Block: Attention

Attention is a sequence of operations, which are originally used for better text recognition and generation, for example in GPT models. The basic idea of attention is to look at the similarity of different parts of the input, for example the importance of the word "Fox" in Figure 21. Self-attention, introduced in the paper "Attention is All You Need", extends this concept by combining the calculation of importance with learnable weights. The self-attention operation is the basic building block of a transformer architecture, it is expressed as following:

$$\text{Attention}(X) = \text{softmax}\left(\frac{XQ(XK^T)}{\sqrt{d_k}}\right)V$$

Let's break this formula down, The input X is passed through three different linear layers (think of these as layers that perform weighted sums of the input plus a bias like a Perceptron). These layers produce the Queries (Q), Keys (K), and Values (V). This can be visualized as:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$

where W_Q , W_K , and W_V are weight matrices. By passing the input through different layers, the neural network can focus at different parts of the input and capture more information.

We then calculate the similarity between queries and keys by taking the dot product of the query matrix with the transpose of the key matrix:

$$QK^T$$

This step helps determine how much focus each word (or part) in the input should receive based on the similarity scores.

To prevent large values that can destabilize the learning process, we scale the dot product by the square root of the dimension of the keys d_k :

$$\frac{QK^T}{\sqrt{d_k}}$$

This scaling helps in maintaining stable gradients during training, making the learning process smoother. Specifically, large gradients can cause the model parameters to update too drastically, leading to issues like overshooting the optimal solution or causing numerical instability.

After scaling, softmax is applied to the scaled dot product. The softmax function converts the similarity scores into probabilities that sum to 1:

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

This step helps in highlighting the important parts of the input while suppressing the less relevant ones.

These probabilities are then used to weigh the values (V):

$$\text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

This produces the output of the attention mechanism, which is a weighted sum of the values, focusing on the important parts of the input.

Often multiple attention heads are used: the input is linearly divided into multiple sets of queries, keys, and values, allowing the model to focus on different parts of the input sequence simultaneously. Each attention operation is performed in parallel, after which the outputs are concatenated and linearly transformed to form the final output.

The main takeaway is that attention allows a model to assign weights to each part of the input, at the cost of having exponential more parameters than other operations with respect to the input size. For this reason, vision transformers scale down the input image, often into patches.

A.5 Neural Network Training

The training process begins by initializing the network's weights. These weights are typically set to small random values. The weights can also be copied from the same model trained on a different (or even the same) task. With the network initialized, a dataset is fed into the network in batches, and the network makes predictions based on its current state.

The difference between the network's predictions and the actual labels is calculated using a predefined loss function. In our case this loss function is Cross Entropy Loss. The network's weights are then adjusted to minimize this loss using an optimization algorithm, commonly stochastic gradient descent (SGD) or one of its variants like Adam.

During training, the concept of convergence is critical. **Convergence** refers to the point where further training does not significantly reduce the loss, indicating that the network has learned the underlying patterns in the data. However, training too long can lead to **overfitting**, where the network performs exceptionally well on the training data but poorly on new, unseen data. This happens because the network has learned to memorize the training data, including its noise and peculiarities, rather than generalizing from it.

To avoid overfitting, several techniques are used. One common method is to split the data into training and validation sets. The model is trained on the training set while periodically being evaluated on the validation set. This helps monitor the model's performance on unseen data, and allows for early stopping when the validation accuracy stops improving.

Regularization techniques, such as dropout and weight decay, are also employed to prevent overfitting. Dropout randomly disables a fraction of neurons during training, forcing the network to learn redundant representations and thus generalize better. Weight decay adds a penalty to the loss for large weights, discouraging the network from becoming too complex.

Lastly, it is crucial to use a diverse and representative dataset for training. This ensures that the network learns a wide range of patterns and features, improving its ability to generalize to real-world data. Proper data preprocessing (for example normalizing the input data) and augmentation (for example mirroring and randomly cropping + scaling the input data) can also enhance the network's robustness and performance.

A.6 Hyperparameters

Hyperparameters are constant parameters that are not optimized during training, these parameters are set by the user or can be found through hyperparameter searching. Take for example the learning rate, which defines how much the weights of a neural network are adjusted. Often the learning rate is changed over time, high at first and lower towards convergence. The initial learning rate can have a large impact on the outcome of training: too high and the network never converges, too low and the network never learns much outside of its initial assumptions.

By splitting the training data in a smaller set for training only and a validation set for testing, we can train the neural network multiple times with different hyperparameters and find out based on the score from testing on the validation set which hyperparameters are most effective. It is crucial that the test data is not used as the test data resembles a scenario that the network has never seen. If the network is optimized with knowledge of the test data it could overfit and not generalize to unseen data.

A.7 Modalities: RGB, Depth and RGB-D

A modality is a type of data that exhibits some structure. Take for example an RGB image, this is defined as a 3 dimensional pixel array of dimensions channels, width and height. Whereas Depth has only one channel. Other popular modalities in machine learning include: audio, video and text.

Most neural networks are defined on single modalities, these are called **unimodal**. The challenge of **multi-modal** networks, which combine input data types, is to align data from one type with another. With RGB-D data this is relatively easy since

every pixel in the RGB image has a corresponding depth, it is still possible to misalign the pixels, for example if a rock captured in an image is slightly to the left in the depth image.

B Computer Vision tasks

A simple visualization of computer vision tasks in figure 22 shows semantic segmentation, classification, object detection and instance segmentation. The tasks are explained in increasing complexity: classification, followed by object detection, then semantic segmentation, and finally instance segmentation.

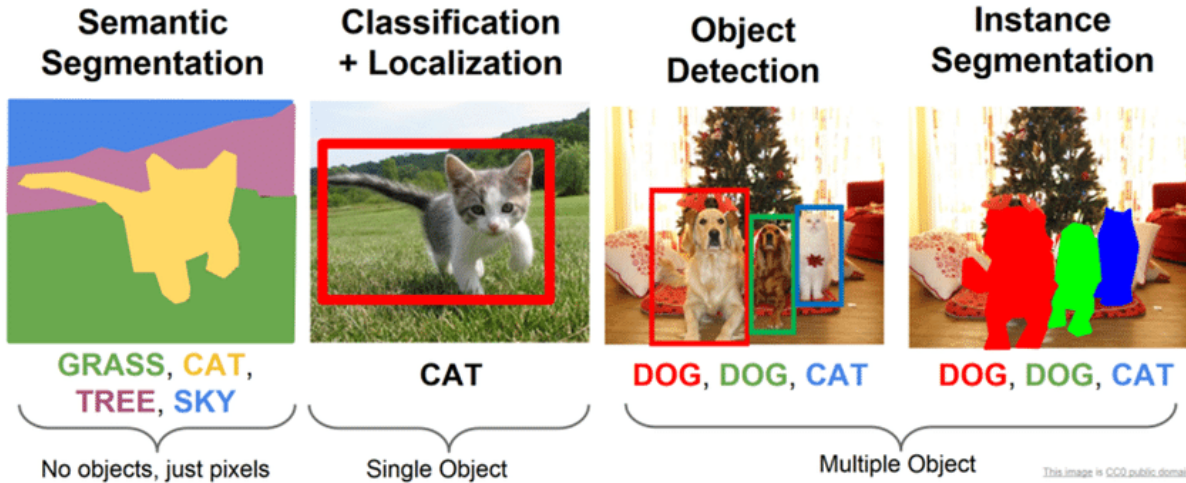


Figure 22. Comparison of semantic segmentation, classification and localization, object detection and instance segmentation (Li, Johnson and Yeung, 2017)

classification: In classification the final layer of a neural network is a list of probabilities for each class. This list can contain very high (or low) values, therefore the result is normalized with softmax. The network then returns the class with the highest probability.

Object detection A neural network that can do object detection has multiple output heads that each predict the location of an object and the class of object. Often the location of the object is denoted with a bounding box describing the coordinates of the object’s position within the image. These coordinates typically include the (x, y) position of the top-left corner of the bounding box, as well as the width and height of the bounding box. The object class is predicted as a probability distribution over a predefined set of classes.

Segmentation: In segmentation, the neural network predicts a pixel-wise classification of the input image. Each pixel in the image is assigned a label corresponding to the object class it belongs to. There are two main types of segmentation:

- **Semantic Segmentation:** This involves classifying each pixel into a predefined set of classes. The network outputs a segmentation mask where each pixel is labeled with the class it belongs to, without distinguishing between different instances of the same class. For example, in an image with multiple cats, all cat pixels will be labeled as "cat," but individual cats are not distinguished.
- **Instance Segmentation:** This extends semantic segmentation by also distinguishing between different instances of the same class. The network outputs multiple segmentation masks, one for each object instance in the image. Each mask indicates the pixels belonging to a specific instance of an object, allowing the model to differentiate between multiple objects of the same class.

C Computer Vision models used

SegFormer SegFormer [30] is a lightweight neural network that uses the Transformer framework (with attention mechanism) for semantic segmentation. While SegFormer is not a State-Of-The-Art RGB segmentation model anymore, it is still an efficient neural network that can be used as baseline.

CMX The CMX model [34] employs the same backbone as SegFormer, but for multiple modalities (e.g., RGB, depth, thermal, polarization, LiDAR). Between each layer, features (outputs) are rectified and fused, this ensures that features from different modalities such as RGB and depth actually align with each other.

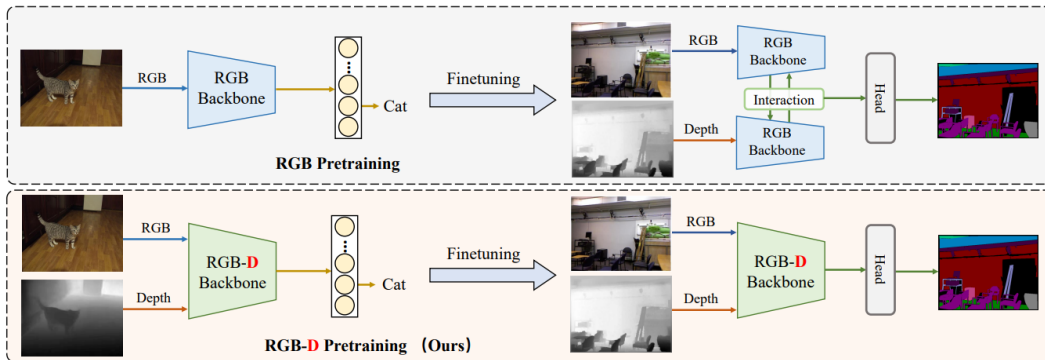


Figure 23. Taken from DFormer [33]. The backbone from DFormer can generate more expressive feature maps for depth than a backbone pretrained on RGB.

TokenFusion Similarly to CMX, TokenFusion [26] uses the SegFormer single-modal backbones. TokenFusion improves Vision Transformers by combining two techniques: merging and pruning of tokens. Tokens are small, fixed-size pieces of the image that can be processed individually and thus in parallel. For example, a token might represent a 20×20 pixel patch extracted from the image, such as from the bottom left corner. These tokens undergo a linear transformation from a linear layer in the neural network, to convert them into feature vectors.

Token pruning helps when the model is very sensitive to changes in the input images, such as resizing or scaling. It involves removing less important tokens to focus on the most relevant parts of the image. On the other hand, token merging combines similar tokens, which is useful when the model processes inputs in a straightforward manner, without being heavily affected by small changes. This approach leads to better accuracy, efficiency, and robustness, meaning the model performs well even when the input images vary.

DFormer DFormer [33] brings a new neural network architecture, but more importantly, a pretraining framework for a RGB-D backbone that learns transferable RGB-D representations, see Figure 23. The pretrained backbone uses depth more effectively than other networks that utilize a RGB pretrained backbone for depth inputs. The backbone is also more efficient as Bowen et al found that a small amount of channels is enough to encode the depth information compared to RGB features.

D Out-of-Distribution Robustness

Out-of-distribution (OOD) robustness refers to the ability of a neural network to maintain its performance when it encounters data that is significantly different from the data it was trained on. This is crucial because, in real-world applications, neural networks often face unpredictable and novel scenarios that differ from their training data. Without robustness to these variations, a model might fail in critical applications, such as autonomous driving, medical diagnosis, or security systems, leading to potentially disastrous consequences.

Neural networks are not automatically robust to OOD data, and this lack of robustness can pose significant challenges in practical applications. Neural networks often struggle with OOD data because they are trained on specific datasets that might not capture the full diversity of real-world scenarios. When the training data is biased or limited, the network fails to generalize to unseen situations. Especially deep neural networks (having many layers) are affected, since these tend to overfit on the training data more easily. Overfitting happens when a model memorizes the training data’s noise and specific details rather than learning the underlying patterns, resulting in poor performance on new data.

Furthermore, neural networks typically lack explicit mechanisms to detect when they encounter OOD data. Without these mechanisms, they treat OOD inputs as if they were in-distribution, leading to incorrect and often overconfident predictions. Neural networks can be highly sensitive to slight deviations from their training data, further undermining their robustness to OOD scenarios.

In real-world applications such as autonomous driving, medical diagnosis, or security systems, the consequences of a lack of robustness in neural networks can be severe. A model that performs well in controlled, in-distribution settings might fail catastrophically when faced with novel, unpredictable situations. This underscores the need for developing neural networks that can maintain performance across a wider range of inputs, including those from RGB and depth (RGB-D) sensors. Ensuring OOD robustness is critical for the safe and reliable deployment of neural networks in these critical applications, where failures can lead to disastrous outcomes.

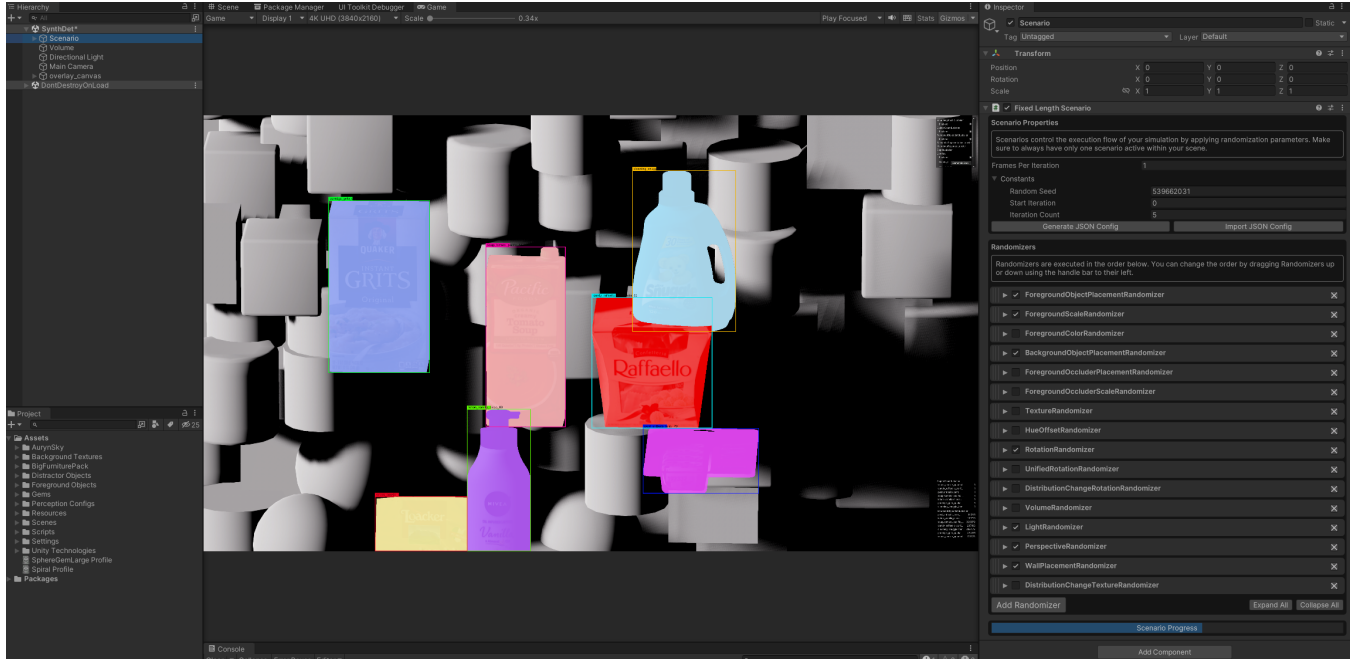


Figure 24. A view of the Unity Editor with Randomizer scripts on the right. Each tab can be opened to adjust specific parameters for that Randomizer. In the scene, grocery objects are placed in the front of a background of white objects with the light position adjusted.

E Synthetic Data Generation

For this project all synthetic data is generated with Unity, using the SynthDet project³. The project focuses on synthetic data generation for RGB object detection, but it can also output segmentation labels and depth maps. These depth maps contain depth values in Unity meters from the camera, so they need to be normalized / converted to depth images.

How does it work? The project contains scripts called Randomizers (see Figure 24), these scripts are executed sequentially. Each script plays a part in the customization of the scene that is being rendered. The first Randomizers place objects at various positions, then other Randomizers change color, light, rotations and more. The SynthDet project, by default, already includes numerous Randomizers. However, a few Randomizer scripts are adjusted for the conducted experiments. The Light Randomizer script has been modified to enable controlled changes in light direction. Additionally, a Perspective Randomizer script has been incorporated to alter the camera’s direction, followed by a translation to ensure the camera remains focused on the segmentation objects.

Data generation The rendered dataset is stored in a Unity SOLO (Synthetic Optimized Labeled Objects) format, which includes JSON files for annotations and image files. While Unity provides a script to convert the SOLO format to COCO for object detection, a different format is needed for segmentation tasks. Therefore, the dataset is first converted from SOLO to COCO and then to a specific segmentation format. In the COCO format, segmentation data is stored in JSON files, but the framework for training and testing requires annotations in the form of images where each pixel represents a class value. The final format consists of folders containing RGB images, depth images, and labels.

F Extension to Salient Object Detection: HiDANet

This appendix explores the application of the Hierarchical Depth Awareness network (HiDANet) for RGB-D saliency detection to differentiate foreground and background regions in images. HiDANet aims to accurately localize salient regions by fusing and enhancing the discriminatory power of RGB and depth features through a granularity-based attention scheme. See Figure 25 for an overview of the architecture. Despite its robust design, including a cross dual-attention module for multi-modal and multi-level fusion, HiDANet’s performance can be significantly affected by variations in background appearance.

In the tests conducted, the background of images was modified while maintaining consistent depth information. These modifications included changes in color, texture, and white backgrounds. See Table 11 for the results. The variations in

³<https://github.com/Unity-Technologies/SynthDet>

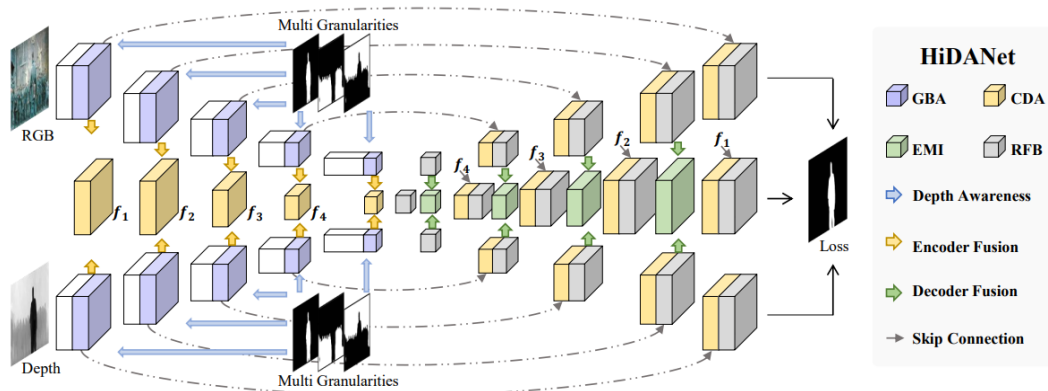


Figure 25. The architecture of the HiDANet salient object detection model, Multi Granularities are created by separating the depth map using the Multi-Otsu thresholding algorithm. The model produces outputs for each modality separate and for the fusion of both modalities. Source: HiDANet: RGB-D Salient Object Detection via Hierarchical Depth Awareness (Wu et al., 2023).

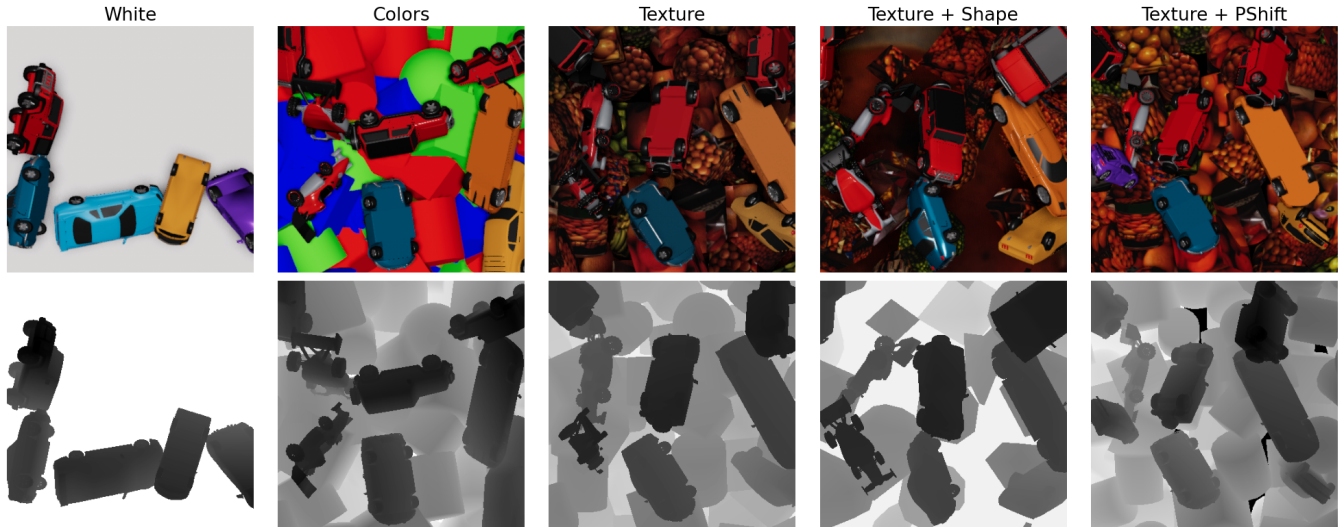
appearance (RGB), despite not affecting the depth modality, caused notable degradation in the network’s ability to accurately distinguish between foreground and background regions. This indicates a limitation in the network’s robustness to background changes, highlighting an area for potential improvement in RGB-D saliency detection models.

Table 11. Results of training the HiDANet model on datasets with varying backgrounds (see Figure 7). The model returns outputs for each modality and for the combined modalities. The output for the depth modality remains accurate, while the influence of variations in RGB also affect the RGB-D output.

Trained on	Output modality	Predicted on		
		White	Color	Texture
White	RGB	97.85	23.49	16.20
	Depth	97.51	97.51	97.51
	RGB-D	99.41	26.76	18.36
Color	RGB	96.45	97.55	16.25
	Depth	98.01	98.01	98.01
	RGB-D	99.41	99.45	20.89
Texture	RGB	16.70	18.28	97.76
	Depth	97.68	97.68	97.68
	RGB-D	17.31	19.86	99.39

G Discarded experiment: replacing the background with cars as foreground

For this experiment, the DFormer neural network has to correctly segment the cars with different backgrounds both in appearance, shape and spatial layout. The network is trained on the subset of datasets excluding a flat white background which is used as a baseline testing dataset. An ideal model would ignore unseen backgrounds when predicting the foreground classes. To simulate such a model a pipeline is created where a depth-only model predicts the background such as in figure 26b, and then another model uses this as input to ignore the background.



(a) Datasets with different backgrounds. The top row shows RGB images and the bottom row shows Depth images. Best viewed in color.



(b) Example of RGB images where the background is predicted by a network and removed.

Figure 26. Comparison of datasets with different backgrounds and the process of predicting and removing the background.

G.1 Results

The DFormer model experiment is conducted twofold: first without using pretrained weights and then with pretrained weights. For each iteration, results are compiled into tables, comparing the performance of the original model with that of the model trained on images with backgrounds removed. The results from training and testing the model without pretrained weights are detailed in Table 12, and with pretrained weights in Table 13. Interestingly, in both experiments, the mean Potential (which is the mean difference between RGB-Depth and RGB-Background_Removed) reveals that the multi-modal models underperform compared to a pipeline of unimodal models, which is an unexpected result. This indicates that the expected advantages of multi-modal integration are not realized in these experiments, highlighting a need for further investigation into the underlying causes.

H Discarded experiment: Comparing depth information with grayscale

Depth is useful because:

- It highlights shape, geometric information of the scene.
- It contains no distractions from color, texture or lighting.

For the reasons that depth is useful in addition to RGB information in computer vision tasks, experiments are carried out to test whether results from training on synthetic datasets reflect these reasons.

For this experiment, synthetic datasets were generated using the SynthDet Unity project. The project includes scripts called Randomizers that modify scene parameters to generate dataset samples. It also provides 63 high-quality 3D models of commonly found grocery products. While most experiments used these assets, objects with similar shapes were observed

Table 12. mIoU values for models trained and tested on different datasets. While fusing the RGB and Depth features leads to better In Distribution accuracy, training first on Depth and then on a background removed RGB leads to better foreground-background separation.

Training Dataset	Model	Testing Dataset				
		White	Colors	Texture	Texture Shape	Texture + PShift
Colors	RGB-D	60.263	97.28	80.269	92.973	89.303
	RGB-BR	97.362	96.903	94.25	95.456	94.554
Texture	RGB-D	64.68	64.905	96.555	97.053	95.202
	RGB-BR	97.141	94.901	95.807	96.199	93.364
Texture + Shape	RGB-D	75.845	66.913	91.495	96.884	94.855
	RGB-BR	95.839	94.912	94.278	94.936	92.347
Texture + PShift	RGB-D	72.675	73.229	85.903	97.228	97.781
	RGB-BR	97.025	96.697	95.569	96.083	96.523
<i>Potential</i>		28.476	27.154	8.81	0.161	0.302

Table 13. mIoU values for pretrained models trained and tested on different datasets. While pretraining the models does help the models separate foreground from background better, there is still room for improvement.

Training Dataset	Model	Testing Dataset				
		White	Colors	Texture	Texture Shape	Texture + PShift
Colors	RGB-D	96.169	97.803	91.859	94.5	94.646
	RGB-BR	97.78	97.505	94.196	97.443	95.561
Texture	RGB-D	96.123	90.645	97.377	98.0	92.837
	RGB-BR	98.058	97.771	97.606	97.887	95.647
Texture + Shape	RGB-D	88.981	92.501	93.202	97.891	93.904
	RGB-BR	97.839	97.571	97.299	97.772	95.562
Texture + PShift	RGB-D	97.856	91.96	95.692	97.48	97.721
	RGB-BR	98.149	97.849	97.264	97.559	97.159
<i>Potential</i>		3.174	6.028	2.669	0.970	1.794

in depth alone, posing challenges even for neural networks. For experiments prioritizing shape over appearance, additional assets representing gem structures were included.

With the synthetic datasets prepared, the multi-modal neural network DFormer was trained to compare RGB-Grayscale with RGB-Depth inputs to evaluate the impact of depth information on segmentation performance. RGB-Grayscale was chosen because grayscale encompasses the information present in the Red, Green, and Blue channels of an image. While grayscale lacks color information, which complicates testing for robustness against color in comparison with depth, different colors produce varying shades of grey. Using a black image as a depth map would unfairly bias the model parameters, providing an advantage to RGB-Depth inputs.

H.1 Depth is useful because of its emphasis on geometric information

For this hypothesis, 5 datasets are created containing the groceries objects in front of a black wall. For the first dataset, named default, the objects are placed at a random position at the same depth plane with no other modifications. Then the objects are scaled randomly to test whether depth can recognize objects of different shape better. For the dataset with variation in spatial position the objects are translated randomly along the axis aligned with depth. Another dataset tests whether depth helps with variation in rotations. In the last dataset the segmentation groceries objects are swapped with gem objects that are better distinguished by their distinctive shape. In this dataset the Hue of the objects is changed so that the network has to rely on shape, not color to differentiate between the objects. See table 14 for the results.

Table 14. Results for experiment 1, the most notable difference in mean IOU score is for the spatial and shape experiment, all experiments show a (small) improvement for using depth information

Dataset	mAcc (%)		mIOU (%)		Difference
	RGB-G	RGB-D	RGB-G	RGB-D	
Groceries (default)	99.13	98.89	96.65	95.35	1.3
· Scale varied	97.77	98.88	91.57	95.89	4.32
· Spatial position varied	96.33	98.93	89.54	96.35	6.81
· Rotation varied	95.52	96.1	88.22	89.16	0.94
Gems (Shape)	95.46	97.57	87.07	92.98	5.91

Table 15. Results for experiment 2, all experiments show a (small) improvement for using depth information

Dataset	mAcc (%)		mIOU (%)		Difference
	RGB-G	RGB-D	RGB-G	RGB-D	
Groceries + White Objects in Background	94.93	96.37	87.21	90.11	2.9
+ White Objects in Foreground	95.04	95.99	84.1	86.83	2.73
White → Color	92.93	93.83	78.0	80.1	2.1
White → Texture	93.56	94.23	80.27	81.29	1.02
White → No Texture	97.47	97.68	91.73	92.32	0.59

Table 16. Results for experiment 3, all experiments show a (small) improvement for using depth information

Dataset	mAcc (%)		mIOU (%)		Difference
	RGB-G	RGB-D	RGB-G	RGB-D	
Varied Light Angle	85.3	89.48	59.91	65.75	5.84
Varied Light Intensity	93.49	94.63	79.2	81.97	2.77
Varied Light Color	85.74	88.92	57.64	64.22	6.58
Varied Perspective	91.3	94.42	74.4	80.41	6.01
Random Noise	93.62	94.74	79.57	82.88	3.31

H.2 Depth is useful for finding accurate object boundaries

To assess whether depth has an influence on object boundaries, background objects are added to the scene containing the segmentation objects. In the first experiment, white objects are added as a distraction, which also creates a more realistic depth map (compared to 1 for objects and 0 for background). In the second, third and fourth experiments, distraction objects are also placed in front to occlude the segmentation objects. In the third and fourth dataset, colors and textures are respectively added to the background and occluder objects. Finally a control dataset is made where each segmentation object’s texture is replaced by a smooth color corresponding to the class. In this test the object’s borders should stand out and both the networks should perform equal. See Table 15 for the results.

Depth is useful because it is robust to variations in color, illumination and rotation angle Perhaps the most interesting set of experiments, assessing under what conditions adding depth makes the largest impact in model performance. All experiments here are performed on the groceries segmentation object with a white background (except for the light & color experiment) In the experiments with lighting the light is shifted and rotated, the luminance is changed and the color that the light produces is changed. See Table 16 for the results.