Evaluating multi-stream networks for self-supervised representation learning

by

Lars Niklas Schröder

Matriculation Number: [redacted]

A Bachelor's Thesis in Computer Science
submitted to

Technische Universität Berlin
Faculty IV - Electrical Engineering and Computer Science
Department of Computer Engineering and Microelectronics
Computer Vision & Remote Sensing Lab

March 31, 2022

First examiner:
Prof. Dr.-Ing. Olaf Hellwich

Second examiner:
Prof. Dr. Henning Sprekeler

Supervised by:
Manuel Wöllhaf

Eidestattliche Erklärung / Statutory Declaration

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

I hereby declare that the thesis submitted is my own, unaided work, completed without any unpermitted external help. Only the sources and resources listed were used.

[redacted]

Berlin, den 31. März 2022 Lars Niklas Schröder

Abstract

This thesis tries to answer the question: Can visual representation-learning neural networks profit from adding privileged information, such as semantic segmentation, instance segmentation, optical flow, or depth estimation to the RGB frames? A video-based sequence-to-sequence model was implemented, which was trained to predict future video frames. A dataset generator, which uses the *CARLA* simulator, was created. With this dataset generator, a 10-hours-long, multi-modal video dataset was created. The dataset was used to train the proposed model in a self-supervised fashion. The results suggest that the models can indeed profit from some privileged information. However, further research is necessary, as this work shows that the results are highly dependent on the model's architecture. This work also shows that a weighted binary cross-entropy (WBCE) is a better loss function than mean squared error (MSE) for evaluating binary image similarities. Furthermore, Peak Signal Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) should not be used as metrics for this task.

Zusammenfassung

In dieser Bachelorarbeit wird versucht die Frage zu klären, ob neuronale Netzwerke, welche Videos komprimieren, davon profitieren können, wenn zusätzliche Informationen zu den RGB Videodaten bereit gestellt werden z.B. Tiefeninformation, optischer Fluss, Instanzsegmentierung und semantische Segmentierung. Dafür wurde ein neuronales Netzwerk entwickelt, welches die nächsten Bilder eines Videos vorhersagen soll. Zudem wurde ein Datensatzgenerator erstellt, welcher die Simulationsumgebung *CARLA* nutzt. Mit Hilfe eines daraus generierten, 10 Stunden langen Datensatzes, der diese zusätzlichen Informationen enthält, konnten die Experimente dieser Studie mittels selbstüberwachten Lernens durchgeführt werden. Die Ergebnisse dieser Arbeit legen nahe, dass das Bereitstellen von verschiedenen der o.g. zusätzlichen Informationen zu Erfolgen führen kann, die Ergebnisse allerdings auch von der gewählten Netzwerk Architektur abhängen. Auch wurde festgestellt, dass die Metriken Peak Signal Noise Ratio (PSNR) und Structural Similarity Index Measure (SSIM), sowie die mittlere quadratische Abweichung (MSE) als Verlustfunktion nicht für die Beurteilung von Bildähnlichkeiten zwischen Binärbildern genutzt werden sollte. Daher müssen weiter Experimente durchgeführt werden, um eine abschließende Antwort auf diese Frage zu finden.

Contents

1. Introduction

The aim of this thesis is to answer the question if a representation-learning neural network, which compresses RGB videos, profits from adding a second visual input modality, such as depth estimation, segmentation information, or optical flow.

This work starts by laying out the relevance of this research question in 1.1. In 1.2 an overview is given on how this thesis tries to answer this question. In section 1.3 all relevant terms for this work are explained. Related literature is then discussed in 1.4. Last, this chapter ends by giving an outline of this thesis in 1.5.

1.1. Motivation

Agents in deep reinforcement learning (RL) perform their actions based on a learned policy. Therefore, the agents need to extract only the necessary information from the sensory input, compress the input, and map it to a policy in order to perform helpful actions. Oftentimes, vision-based RL agents are only equipped with an RGB camera and therefore have to perform their actions from this high-dimensional spatio-temporal data. During the learning process, these agents must learn which information is relevant to the task and map it into a feature representation themselves. Prior research has shown that adding other visual modalities like depth estimation and segmentation, which were learned in a supervised way, can lead to a better performance in the downstream task. But this approach has two flaws: Firstly, these visual modalities first need to be labeled, which is a tedious and costly process. Secondly, not all of these visual modalities are relevant for all tasks, and the visual modalities have a high level of redundancy in them. Selecting the best visual modalities for a certain task can e.g. be realized by a feature selection agent, which only selects the visual modalities that are most relevant for the current downstream task.

Work in the action classification literature showed that results almost always improve by adding optical flow to the RGB input video. Optical flow, however, can directly be derived from the RGB video. This means these neural networks, which only received RGB as an input, do not yet fully leverage the information that is present in their inputs.

Both findings bring up the question if a visual representation-learning system would also profit from additional visual modalities if the representations were learned in a self-supervised way.

1.2. Overview

A model was implemented, which takes a combination of different input videos (RGB, depth, optical flow, and segmentation) and learns to compress them into a representation vector. To achieve this in a self-supervised way, the model was trained to decode the representation vector and predict the input frames, as well as the next future frames in the domain of a simplified instance segmentation. The aim is to evaluate if such a representation learning system profits from adding additional visual modalities to the RGB input. The focus lies on segmentation in this thesis because it has been proven to improve the results in reinforcement learning (see 1.4.2).

1.3. Background

In the following all relevant terms and concepts used for this work are introduced briefly.

1.3.1. Visual modalities and visual cues

The proposed dataset consists of sequences of RGB images and their corresponding depth, optical flow, instance segmentation, instance edges, and semantic segmentation images. Examples of these image types can be seen in Figure 1.1.

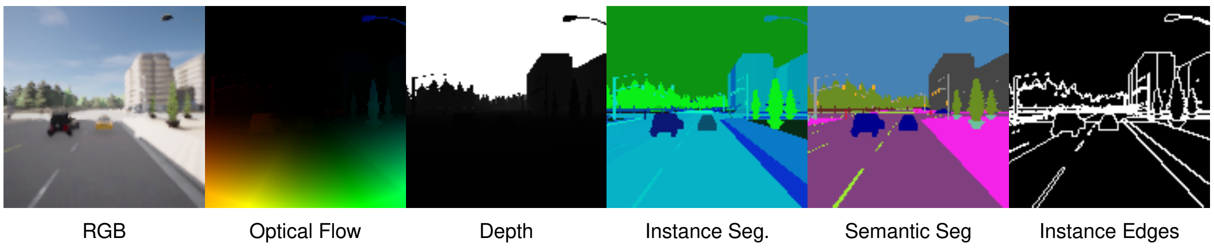

Figure 1.1.: Visual modalities extracted from the proposed dataset, which was generated in *CARLA*.

RGB

Each pixel of an RGB image consists of three values. Each value represents the brightness of the respective color red, green, and blue. All values of the same color are stored in a *channel* i.e. an RGB image has three channels, whereas a grayscale image only needs one channel.

Depth

The distance between the RGB camera sensor and an object is encoded in the depth image. The depth images in this work are grayscale images with a linear scale.

Optical flow

An optical flow image represents the movement between two consecutive RGB images. The color of a pixel indicates the relative direction and speed of all visible objects to the camera.

Instance segmentation

Instance segmentation is a mapping between an RGB pixel and the instance it belongs to. In other words, all pixels that belong to the same instance have the same color in the instance segmentation image. The instance segmentation images mentioned in this work also include a semantic identifier for the instance's class e.g. pedestrian, sky, car.

Instance edges

An image type called instance edges was extracted, which is a binary image showing only the edges of the instances from the instance segmentation image.

Semantic segmentation

An image that only includes the semantic identifier is called a semantic segmentation image.

Visual modalities and cues

In this thesis, the term "visual modalities" is used to summarize the different image types and "visual cues" to point to the information that is contained within a visual modality. The optical flow image e.g. is a visual modality and contains visual motion cues.

1.3.2. Artificial neural networks

Artificial neural networks (here: neural networks) are computational models that are inspired by biological neural networks and can be trained. Neural networks are trained by providing an input and a corresponding target that should be reached. During the training process, neural networks should minimize their loss. The loss is calculated with a loss function that takes the target and the model's output into account. The training process is – mathematically speaking – a gradient descent on a non-convex surface. If designed carefully and provided with good and enough training data, artificial neural networks can be extraordinarily good at detecting correlations among the training data and can apply the gained "knowledge" to unseen data[1].

1.3.3. Convolutional neural networks

Convolutional neural networks (CNNs) are – also biological inspired – neural networks that are specialized for data that has a grid-like structure e.g. two-dimensional data types like images. At least in one layer, these neural networks must use the mathematical convolution operation, instead of general matrix multiplication, to be considered "convolutional"[2]. CNNs have become especially ubiquitous in deep learning computer vision applications. In this work, CNNs are used to compress images into fixed-length vectors and to decode these vectors back into images.

1.3.4. Encoder-decoder architecture

An encoder-decoder network is a neural network architecture in which the encoder transforms an input, of variable length, into a fixed-size state (here: representation vector). A decoder takes this state as its input and transforms it into an output sequence of variable length. A network in which the encoder's input is the same as the decoder's target is called an "autoencoder". Thus, autoencoders effectively should compress the input[3]. An encoder-decoder architecture is sometimes also referred to as a sequence-to-sequence architecture[4]. The proposed model uses an encoder-decoder architecture e.g. to map a series of input frames of a video to a series of the next few frames.

1.3.5. Skip connections

In an ordinary feedforward neural network, the output of one layer is fed into its succeeding layer. Skip connections are connections between two layers that are not consecutive in the first place. In other words, skip connections are connections between layers that skip one or more layers. The inputs at the receiving layer of a skip connection can be added together, as in the convolutional encoder network ResNet[5], or concatenated, as in the convolutional encoder-decoder U-Net[6]. In this work, skip connections as in the ResNet encoder architectures, which only skip two or more layers within the encoder, are considered "short" skip connections. Skip connections in U-Net, which skip the representation state by connecting layers from the encoder with layers from the decoder, are considered "long" skip connections. The proposed model uses both, U-Net, and a ResNet encoder-decoder variant with long skip connections as spatial encoder and decoders.

1.3.6. Long short-term memorys

A long short-term memory (LSTM) is a neural network specialized for time-series data. Its core ability is to model dependencies within the data over a long period of time[7]. The proposed model uses LSTMs for two purposes: one is to encode a series of spatial encoding vectors into one fixed-length representation vector, and the other purpose is to decode representation vectors back into a series of spatial encodings.

1.3.7. Multi-stream convolutional neural networks

A CNN that receives more than one visual input modality can be described as having a multi-stream architecture. Here, a stream refers to a part of a network that one input modality goes through before it gets fused with streams of other input modalities. Four different fusion methods can be summarized: early, middle, late, and multi-level fusion[8]. The proposed model uses the early fusion method, which directly concatenates the input images along the channel dimension.

1.3.8. Self-supervised learning

Self-supervised learning (SSL) is a method to train neural networks with automatically generated labels [9]. This means, unlike supervised learning, SSL does not depend on humans to annotate the data manually, but that labels are extracted from e.g. a simulator as it is done in this work.

1.4. Related work

In the following, existing datasets are described and discussed, which were considered to be used for this thesis (see 1.4.1). Then related literature from the fields of reinforcement learning (see 1.4.2) and action classification (see 1.4.3) is presented. Last, literature about successful representation learning model architectures is presented (see 1.4.4).

1.4.1. Datasets

Multiple datasets and simulators were considered in preparation for this work. For the method described, a dataset has to meet the following three requirements.

Firstly, it is crucial to have an RGB video dataset that also includes the corresponding depth, segmentation, and optical flow data. Although predicting these modalities from RGB with e.g. pre-trained neural networks for depth is possible[10], this method was not chosen for the thesis. It has been shown that training a neural network on action recognition from optical flow is highly dependent on the quality of the images[11]. It can be expected that such predictions from RGB always include noise and errors, as they are just estimates. Since the downstream task of predicting future video frames is already complex enough by itself, these input inaccuracies would propagate through the network and may further hinder it from learning the actual task. Manually labeled datasets are also not accurate enough. As a first step, it is therefore preferred to isolate the model from such inaccurate data and only use ground-truth labels for all visual modalities. Thus, real-world datasets are omitted. Therefore, synthetic datasets seem to be the only suitable option. These also have the additional advantage that theoretically, they are not limited by the number of samples because more data can be produced at a low cost.

Secondly, to evaluate whether the learned representations contain any underlying physical concepts of the dataset e.g. the camera's speed, it is crucial that the dataset also includes ego-motion data. Even though this thesis does not cover such an evaluation, this requirement was kept for future research.

Thirdly, in order to be able to predict video frames at all, the video clips must be predictable. This may seem obvious, but datasets like SceneNet RGB-D [12] do not meet this requirement. They introduced a synthetic video dataset for random 3D camera trajectories through arbitrarily generated static indoor environments, including ground truth data for depth, semantic segmentation, instance segmentation, optical flow, as well as per-frame camera positions. This almost suits the needs for a dataset. However, random camera trajectories make it impossible to predict the next frames.

Synthetic video datasets can be obtained through most 3D graphical software. Not only does this include 3D simulators, which were created for the purpose of generating datasets but also video games. [13] e.g. used Microsoft DirectX® rendering API to extract ground truth labels for instance segmentation, semantic labeling, depth estimation, optical flow, intrinsic image decomposition and instance tracking from the video games GTA V, The Witcher 3, FarCry Primal among others.

On one hand, these game-based solutions can produce a very rich dataset. The worlds they play in are usually very large and allow for a diverse set of scenarios with highly realistic graphics. On the other hand, these games are usually not open source and not developed with data readout in mind, so it is quite tedious to gather datasets from them. The data is also not reproducible in case some data was forgotten to be collected in the first place.

No dataset was found that fulfilled all the above-mentioned requirements. Therefore, generating a new dataset was necessary. For that, a dataset generator with CARLA[14], which is an open-source urban driving simulator, was designed. Details about the dataset are described in 2.2.2.

1.4.2. Reinforcement learning

In deep RL an agent must learn a policy from which it performs its next action. A policy, in other words, is a representation that must be learned by compressing the input and extracting the task-relevant information from it. In several works, vision-based deep RL agents were used to train the policies directly from raw images[15, 16]. [17] names two drawbacks of this approach: first, training these agents is very inefficient and requires massive amounts of data, and second, the learned policies are not robust for even modest visual changes in the environment i.e. a change in lighting may result in significantly worse results. To mitigate these problems, [17] first used a set of 24 pre-trained neural network feature encoders from [18] to get different visual modalities from the RGB input video e.g. depth estimation, object classification, and surface normals. Training the policy on sensorimotor agents for different indoor navigation tasks with these 24 feature encodings outperformed the StoA RGB-only agents significantly, which were trained from scratch in terms of performance, generalization, and learning speed[17].

[19] made the same observation. They further showed that using ground-truth visual modalities improved the performance against visual modalities generated from pre-trained U-Nets, even further. They also evaluated the relevance that each of the tested modalities albedo, optical flow, depth, and semantic segmentation have on different tasks and found that semantic segmentation and depth estimation provide the highest boost across all tasks[19].

1.4.3. Action classification

Action classification is the task of predicting an action that is shown in a video. It has become a wide-used practice to either predict the action directly from optical flow or to augment the RGB video with optical flow. It was shown that optical-flow-only networks can outperform RGB-only networks in action classification[11]. Intuitively, this makes sense because an action can usually be determined by just the movement, and optical flow represents these motion cues. Optical flow is also invariant to appearance i.e. the color of clothing of a person kayaking does not matter for the classification of the action "kayaking" [20] argues, and thus using RGB exclusively as input can be a hurdle for the network's ability to generalize. But one can also imagine that combining RGB and optical flow can further improve both RGB-only and optical-flow-only approaches because a network could then detect similarities between the appearance across the different action samples from the RGB input. In the kayaking example from above, the network may recognize water in the RGB video and associate that with all water sports, which is one first valid step toward the kayaking class. For example [21] argues that 3D ConvNets should be sufficient enough to learn these motion cues directly from RGB inputs, they have made the observation that using two-stream networks (RGB and optical flow) outperformed the single-stream ones (RGB or optical flow).

1.4.4. Architecture

The task of the developed model is to encode an input video, of one or more visual modalities, into a fixed-length representation vector and later use the representation to generate a prediction video of another visual modality. Therefore, the input video needs to be compressed and decompressed in both spatial and temporal domains.

3D CNNs can be used to do this. These networks use 3D convolutional kernels to encode and decode the input in the spatial and temporal domain simultaneously. However, [22] concludes 3D CNNs tend to overfit on small datasets like UCF[23], ActivityNet[24], and HMDB51[25]. Only datasets like Kinetics[21] were large enough to prevent the models from overfitting. The proposed dataset has 10 hours of video data, and thus does not come close to Kinetics-400's 444 hours[26].

With an LSTM-based Seq2Seq model [27] proposed an architecture to learn fixed-length representations from videos. To learn these representations in a self-supervised way, they used input reconstruction i.e. an autoencoder and future frame prediction, as their pretext tasks. They were able to use their pre-trained encoder networks to improve results in action classification on different unrelated datasets and tasks. This shows that their models were able to extract useful representations out of the videos.

The developed model architecture is based on the "composite" archtiecture of [27]. Detailed implementation details can be found in 2.2.3.

1.5. Outline of this thesis

The methodical approach this thesis uses for trying to answer the research question is explained in 2.1. For this approach a dataset and a model framework had to be developed, which are described in 2.2. Then, multiple experiments had to be conducted, which are defined in 2.3. The results are summarized in 2.4 and analyzed in 2.5. The thesis ends with a conclusion of the results in 3, including a discussion and a set of problems that could not be covered by this thesis.

2. Main contribution

In this chapter, an answer to the research question "Does a visual representation learning system profit from additional visual input modalities?" is tried to be found. First, the approach to answering this question is formulated in which the used dataset, model, and evaluation methods are briefly described (see 2.1). The implementation is described in 2.2, where details about the develpment tools (see 2.2.1), the proposed dataset (see 2.2.2), and the proposed model (see 2.2.3) are described. The experiments are described in section 2.3, which consists of an introduction to the used loss functions and metrics (see 2.3.1), a hyper-parameter search (see 2.3.2), and the experiment setup (see 2.3.3). The results of the experiments are presented in section 2.4. An answer to the research question is formulated in section 2.5.

2.1. Methodical approach

The aim of this thesis is, to evaluate if and to what extent a representation learning network can profit from providing additional visual modalities as input. For this purpose, a model with a sequence-to-sequence architecture was designed. Specifically, it takes T video frames as input, compresses them into a fixed-length representation vector, and decodes this vector back to a video of another modality with $T + p$ frames, where p is the number of future frames the model should predict. A visualization of an example experiment where $T = 6$ and $p = 4$ is shown in Figure 2.1.

As a baseline experiment, RGB was used as the input. For all experiments, instance edges was used as the target. To evaluate if the model profits from a second input modality, this modality was concatenated with RGB along the channel dimension. These results are then compared to the results of the baseline experiment. Last, the findings are quantified and evaluated using different metrics.

The used dataset was generated within *CARLA*, an open-source urban driving simulator [14]. It contains ground-truth labels for RGB, depth, optical flow, semantic segmentation, instance segmentation, and instance edges. With this, the network could be trained in a self-supervised way.

2.2. Implementation

With this work, a parameterizable dataset generator that uses *CARLA*[14] as a simulation environment was implemented. Furthermore, a sequence-to-sequence model framework was implemented, which predicts an output sequence based on a learned representation whereby the visual input and output modalities can differ.

In the following, the design decisions for the chosen dataset and the model's architecture

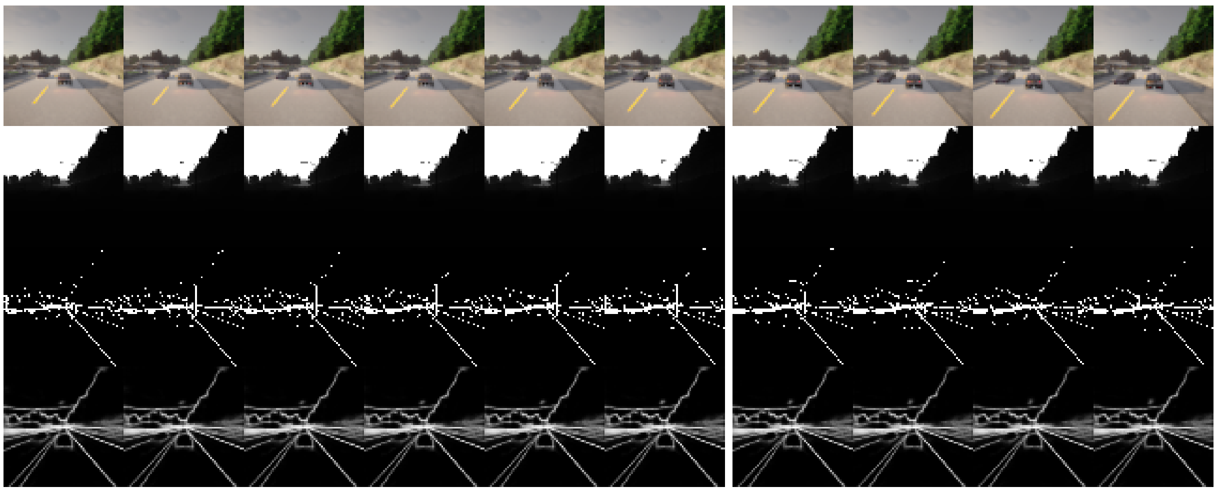
Figure 2.1.: An example visualization of an experiment from the hyper-parameter search. The model was trained with MSE and received 6 input frames of RGB and depth. It had to predict the input timesteps, as well as the next 4 timesteps in the domain of instance edges. Visual modalities top to bottom: RGB (input), depth (input), instance edges (target), output.

are described. Concrete parameters are replaced by variables because these varied across the different experiments (see 2.3). In-depth details about the implementation, however, should be derived from the code (see B).

2.2.1. Tools used for development

Both the dataset generator and the proposed model were written using *Python 3*[1]. The model was written with the machine learning library *TensorFlow*[28] in combination with *Keras*[29]. *Docker*[2] containers were used for managing dependencies and running the code independently on different machines. All experiments were executed on the conjoined high-performance cluster from *TU Berlin*[3] and *Science of Intelligence*[4], as well as on some machine from the *Computer Vision and Remote Sensing department* of *TU Berlin*[5]. For managing the experiments *Yet Another simple, stupid ML toolkit*[6] was used.

2.2.2. Dataset

A dataset was generated with *CARLA*[14].

Generating the dataset

The dataset was produced by generating 224 5-minute-long sequences of a car driving around

[1] `https://www.python.org/`
[2] `https://www.docker.com/`
[3] `https://www.tu.berlin/`
[4] `https://www.scienceofintelligence.de/`
[5] `https://www.cv.tu-berlin.de/menue/computer_vision_remote_sensing/`
[6] `https://git.tu-berlin.de/cvrs/mltk`

in one of seven maps. *CARLA* provides eight official urban maps: "Town01" - "Town07" and "Town10"[7]. However, the latter did not run stable and was excluded. The car has four forward-facing cameras attached at its front: RGB, depth, instance segmentation, optical flow. Additionally, semantic segmentation and instance edges were computed from the instance segmentation camera output. All cameras recorded at 25 frames per second, which a commonly chosen framerate for video datasets like Kinetics[26] or UCF101[23]. The cameras' resolution is 128×128 pixels and they have a 90 degree field of view. The recorded videos were saved as sequences of PNGs with values between 0 and 255. Furthermore, the car's current speed, its effecting speed limit, and traffic light state, as well as the transformation of all cars and pedestrians, were saved at the same framerate. Each sequence was captured in one of the seven maps in 15 different weather and daylight conditions[8]. After analyzing the 224 samples, one sample from the training set had to be excluded due to rendering problems. This resulted in 209 training, 7 validation, and 7 testing samples. Besides that, the number of sequences per map is distributed equally.

Technical details

The dataset generator can be parameterized with a CSV file that contains a seed, the split name (train, test, validation), map name, framerate, duration, number of cars, number of pedestrians, weather and daytime, a speed limit factor, the cameras' position, orientation, field of view, and resolution. It took 120 hours (with a GTX 1050 and an i7-7700 CPU) to gather the 224 5-minute samples. The dataset generator was designed to be fully deterministic to be able to reproduce the dataset and to add additional sensor data on demand. However, *CARLA* 0.9.13 is still a beta version in which some minor details e.g. tree animations are still not reproducible.

Preprocessing the data

For training and inference, the images were normalized to a range of $[0, 1]$. The 5-minute long sequences had to be split into subsequences of 80 frames – or 3.2 seconds – in order to efficiently load and shuffle the samples in the RAM. Due to traffic jams, pedestrians standing on the road, and red traffic lights, subsequences with an average speed of less than 0.1 $^{km}/_{h}$ were discarded (which made out 45% of all subsequences) because in these subsequences the differences between the frames are too small, which could make the model overfit to these situations. This resulted in 11582 (10819 train + 385 test + 378 validation) subsamples – or about 10 hours of video data.

2.2.3. Model

The model is a cascade of multiple networks, similar to the composite model of [27].

Model overview

The model is an autoencoder while simultaneously predicting frames for the next p timesteps. It is visualized in Figure 2.2. A spatial encoder first compresses each frame into a spatial encod-

[7] `https://carla.readthedocs.io/en/0.9.13/core_map/`
[8] `https://carla.readthedocs.io/en/stable/carla_settings/#weather-presets`

ing vector. The compression in time of these spatial encodings is done by a temporal encoder, which outputs a representation vector. One temporal decoder reconstructs the spatial encodings from the representation vector. Another temporal decoder predicts spatial encodings for the succeeding timesteps. A spatial decoder concatenates the outputs of the two temporal decoders and predicts the output frames.

Detailed model description

Let $(x_1, \dots, x_T)$ be a sequence of frames, extracted from a video. The model's task is to compress these frames into a representation vector w of size R and output the frames $(y_1, \dots, y_{T+p})$ where p is the number of future frames the model should predict. The model's input frames x_i can be an early fusion (see 1.3.7) of multiple visual modalities, and the target frames y_j can also be of a different modality.

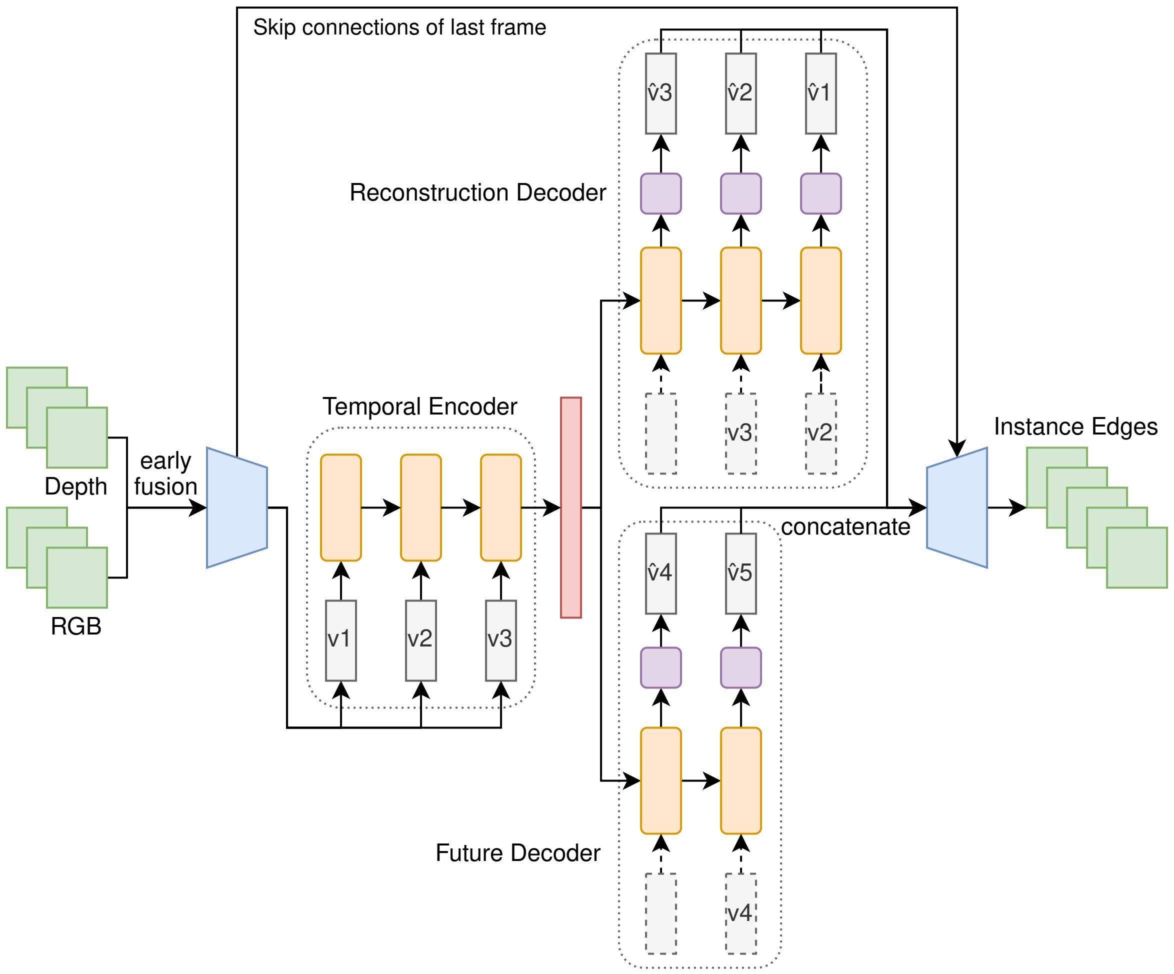

Figure 2.2.: An example of the model's architecutre, when it takes 3 frames of depth and RGB as an input and outputs 5 frames of instance edges. Green: sequence of image frames; Blue: spatial encoder/decoder; Gray: spaital encoding vector (v_j: ground-truth; $\hat{v}_j$: predictions); Orange: LSTM; Purple: fully connected layer; Red: hidden state of the temporal encoder's last LSTM step (here: representation vector); The network is unfolded in time inside the dotted boxes.

As shown in Figure 2.2 the spatial encoder first encodes each frame x_i individually into a spatial encoding vector v_i of size S. This results in T spatial encoding vectors $(v_1, \dots, v_T)$. The spatial encoder also outputs four skip connections per frame. Only the skip connections s_i of the last input frame x_T will later be used to preserve visual details of the input sequence. The spatial encodings are then fed into the encoder LSTM (here: temporal encoder) that outputs a representation vector w of size R. This vector is simply the hidden state output from the last time step of the temporal encoder.

The temporal encoder-decoder follows the composite architecture [27] has presented and thus has two separate temporal decoders. One reconstruction decoder for reconstructing the temporal encoder's input $(v_1, \dots, v_T)$, and one future decoder for generating the next p spatial encodings $(v_{T+1}, \dots, v_{T+p})$. In principle, both decoders work the same. Both use the representation vector as their initial state, but the reconstruction decoder is trained to generate its output sequence in reverse order i.e. it first generates the spatial encoding for the timestep that was last inputted to the temporal encoder. Teacher forcing is used for the decoder LSTMs i.e. they receive the spatial encoding vector of the last time step as an input for the next one. During training, the v_j's are used, which were generated from the ground-truth input frames. During inference, the predictions $\hat{v}_j$'s are used. For the first-time steps, they receive a beginning-of-sequence tensor that is initialized with zeros. The decoders' outputs are $(\hat{v}_1, \dots, \hat{v}_T)$ and $(\hat{v}_{T+1}, \dots, \hat{v}_{T+p})$ respectively. The spatial decoder then decodes each spatial encoding $\hat{v}_j$ together with the skip connections s_T back into a frame $\hat{y}_j$ of the output modality. Using two stacked LSTM layers instead of one inside the temporal encoder and decoders, is also possible.

Different spatial encoder-decoder networks

We have implemented the model in such a way that the spatial encoder-decoder network can be selected with a parameter. The following models are integrated: VGG[30], DCGAN[31], ResNet[5], U-Net[32], and LinkNet[32]. VGG and ResNet were originally proposed as spatial encoders, DCGAN as a spatial decoder. Therefore, implementations of these three networks were used, which include the corresponding counterpart with long skip connections[9]. U-Net and LinkNet were originally designed as encoder-decoder networks for semantic segmentation.

2.3. Experiments

Multiple experiments with the above-mentioned dataset and sequence-to-sequence network were conducted to find an answer to the research question. First, the utilized losses and metrics are described (see 2.3.1), then a hyper-parameter search was done to find parameters that lead to good metrics scores (see 2.3.2). Last, the conducted experiments are described (see 2.3.3).

2.3.1. Loss function and metrics

Different metrics were used to quantify the results. These were calculated after each training epoch, as well as after the training on samples from the validation and testing set. Structural Similarity Index Measure (SSIM) [33] and Peak Signal Noise Ratio (PSNR) [34] are two com-

[9] These networks were implemented by Manuel Wöllhaf.

monly used metrics for image similarity[35]. For both metrics a higher score is better.

Mean squared error (MSE) was used as a loss function for most of the experiments. Since instance edges, which is a binary classification problem (edge or non-edge pixel), was used as the target modality throughout all experiments, also binary cross-entropy (BCE) was tested as a loss function. However, the two classes are highly imbalanced. The edge-pixels only make out a small portion of the entire image. Therefore, a weighted version of BCE (WBCE) was implemented, which first balances the two classes[36].

2.3.2. Hyper-parameter search

Every experiment is defined by a set of more than 30 parameters specifying the input, output, optimizer, encoding dimensions, spatial encoder and decoder networks etc. Therefore, 170 short[10] experiments were conducted, to search for parameters that lead to better metric scores (PSNR and SSIM). Subjective image similarity between the output and the target was also used to choose the hyper-parameters for the next iterations. Thus, the approach and main findings are briefly summarized.

R	seed	PSNR	SSIM
32	45616	**14.334**	**0.5552**
32	290845	14.264	**0.5521**
512	45616	14.328	0.5430
512	290845	**14.375**	0.5469

S	seed	PSNR	SSIM
32	45616	14.198	0.5024
32	290845	**14.374**	0.5079
128	45616	14.287	**0.5201**
128	290845	**14.319**	0.5095
512	45616	14.192	0.4841
512	290845	14.317	**0.5171**

Table 2.1.: *Left*: Comparison of different representation vector sizes R with two different random seeds. Evaluated after 120 epochs on the test set.
Right: Comparison of different spatial encoding vector sizes S with two different random seeds. Evaluated after 45 epochs on the test set. The top two scores are marked bold.

All hyper-parameter search experiments had RGB and depth as inputs and instance edges as output modality and, unless otherwise specified, used MSE as loss function. A batch size of 70 was just small enough to not cause out-of-memory errors. Then, different learning rates and different spatial encoder and decoder networks were tested: DCGAN, LinkNet, ResNet, U-Net, VGG. LinkNet and VGG had lower metric scores and were excluded. ResNet performed best. Only ResNets were used for the following hyper-parameter search experiments. Switching from one to two stacked LSTM layers inside the temporal encoder and decoders did not result in better metric values. Changing the size R of the representation vector w in a range of $[32, 512]$ and the size S of the spatial encoding vectors v_i in a range of $[32, 512]$ also seem to have no significant effects on the results in this early stage of training (see Table 2.1). Long skip connections improved the results significantly (see 2.2).

2.3.3. Experiment setup

The same hyper-parameters were used across all experiments. Throughout all experiments, RGB was provided as one input modality and instance edges as the target modality. The base-

[10] three hours on a Tesla v100s with 32GB

Skips	PSNR	SSIM
No	13.152	0.2576
Yes	**14.103**	**0.4724**

Table 2.2.: Comparison between using and not using long skip connections. Evaluated after 120 epochs on the test set. The top scores are marked bold.

line experiment is the one in which RGB is the only input modality. Four other experiments were conducted, in which one of the following four modalities was added to the input: optical flow, depth, semantic segmentation, instance segmentation. In addition, for all five input modality combinations, three different encoder-decoder networks were tested: ResNet, U-Net, DCGAN. Another set of experiments was conducted in which WBCE was used as a loss function instead of MSE.

The experiments were backed in regular intervals, are reproducible by setting a random seed, and can be continued.

Hyper-parameters used

For the main experiments, 128 was used as the size of the spatial encodings v_i, 512 as the size of the representation vector w, single LSTM layers for the temporal encoder and decoders, long skip connections between the spatial encoder and decoder and adam optimizer with an initial learning rate of 0.002 which is divided by two every 100 epochs. All frames got resized to 64×64 pixels. The model received 8 input frames $(x_1, \dots x_8)$ and outputted 8 future frames, resulting in 16 output frames $(\hat{y}_1, \dots, \hat{y}_{16})$ in total. These 16 frames were randomly chosen sequences from the 80-frames-long subsequences that were extracted from the dataset.

2.4. Results

In the following the results of the main experiments are summarized.

Comparing spatial encoder-decoder networks

To find the best encoder-decoder network, the top three performing networks from the hyper-parameter search experiments were chosen. These were ResNet, U-Net, and DCGAN. All three models were trained with all five input modality combinations RGB, RGB+depth, RGB+optical flow, RGB+instance segmentation, RGB+semantic segmentation. The target was to predict instance edges. The results for all 15 experiments are visualized in Figure 2.3.

ResNet achieved the highest PSNR scores for RGB+depth, RGB+instance segmentation, and RGB+semantic segmentation. The PSNR baseline scores of ResNet and U-Net are almost identical. DCGAN performed worst except for the optical flow experiment. As for the SSIM score, ResNet also achieved the best scores for three out of the five experiments: RGB, RGB+depth, RGB+instance segmentation. ResNet performed worst in the optical flow experiments. Since no overfitting could be observed after these experiments, ResNet was chosen for further training.

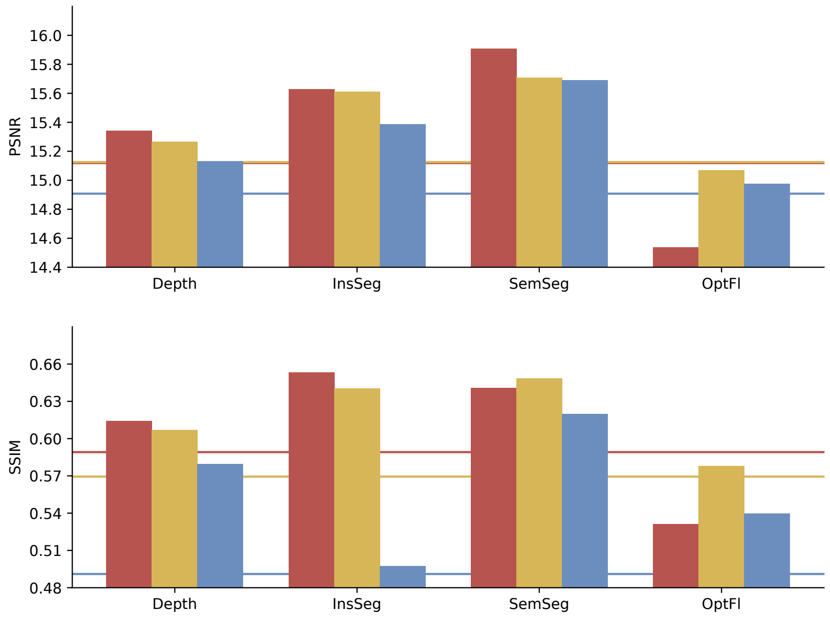

Figure 2.3.: Comparison between different spatial encoder-decoder networks and diffent modalities. Red: ResNet; Yellow: U-Net; Blue: DCGAN. The colored horizontal lines mark the corresponding results from the baseline experiment i.e. RGB as a single input modality. The x-axis shows the second input modality. Evaluated after 96 epochs on the test set. See A.3 for raw values.

Comparing modalities

All models achieved higher PSNR and SSIM scores when they received depth, instance segmentation, or semantic segmentation as a second input modality (see Figure 2.3). Optical flow only improved the results of the model that used DCGAN. The models profited more from instance segmentation and semantic segmentation than from depth.

Comparing training durations

Training the ResNet for another 98 epochs showed further improvements, except that the results for the instance segmentation model got worse (see Table 2.3). The training and validation curves for this experiment are visualized in A.2. Due to an incorrect setting of the number of validation steps[11], the validation curve fluctuates too much to say for certain that this decrease in the SSIM metric is the cause of overfitting.

Comparing BCE and WBCE as loss functions

For all above-mentioned experiments, MSE was used as the loss function. Using BCE and WBCE as a loss function showed that the WBCE experiments got worse metric scores than the BCE ones (see Table 2.4). However, the outputs from the WBCE experiments subjectively appeared more accurate. In Figure 2.4 for example, the car and the tree on the right are hardly visible in the BCE experiment, whereas with WBCE, they can be spotted easily. It appears that

[11] The validation step-size was set to 1. With a batch size of 70 this means only about a fifth of the validation data was evaluated after each training. The validation data, however, was shuffled in every epoch.

Modality	PSNR	SSIM
Baseline	0.2399	0.01890
Depth	0.1885	0.02186
InsSeg	0.1	-0.15757
SemSeg	0.2481	0.04782
OptFl	0.7364	0.09131

Table 2.3.: Difference between training the ResNet-based model 98 and 196 epochs. The values are the differences: $score_{196} - score_{98}$. Positive values are improvements. See A.4.1 and A.3.1 for raw values.

loss	PSNR	SSIM
BCE	**15.517**	**0.6407**
WBCE	11.279	0.4901

Table 2.4.: Comparison between BCE and WBCE as loss function. Evaluated after 126 epochs on the test set. The top scores are marked bold.

the used metrics SSIM and PSNR are not well suited for evaluating instance edges.

Comparing MSE and WBCE as loss functions

A comparison between MSE and WBCE is shown in Figure 2.5. The output of the experiment in which WBCE was used shows that the model learned that objects tend to come closer to the camera when the car is driving. The street light e.g. clearly gets bigger in the output towards the end of the predicted sequence. The street light in the output of the MSE-trained model just gets more blurry. In both model outputs, however, the street light still gets predicted in the last frames, even though it is not present anymore in the ground-truth. Another observation can be made when looking at the lane markings. These are clearly present in the RGB input, but not in the depth or target sequence, but the WBCE-trained model still predicts a lane marking, which comes closer. Despite that, the WBCE-trained models still got worse metric scores than the MSE-trained model (see A.1).

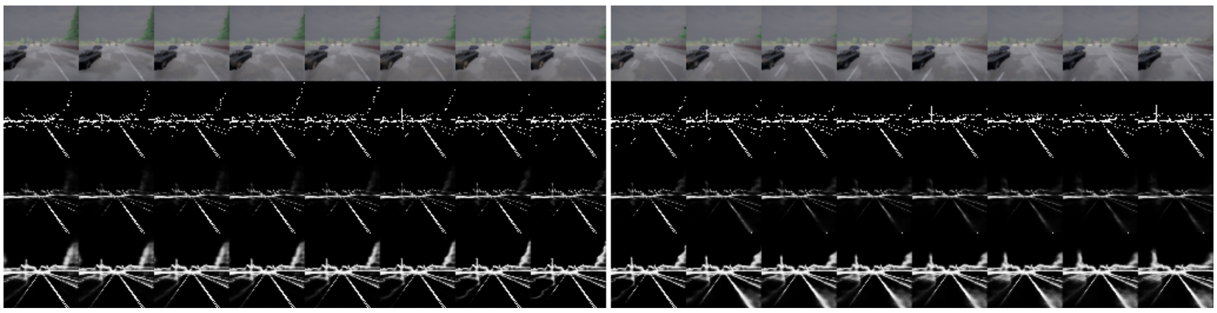

Figure 2.4.: Comparison between BCE and WBCE as loss function. Modalities top to bottom: RGB (input), instance edges (target), BCE (result), WBCE (result). The second input modality depth is not shown. Evaluated after 126 epochs on the test set.

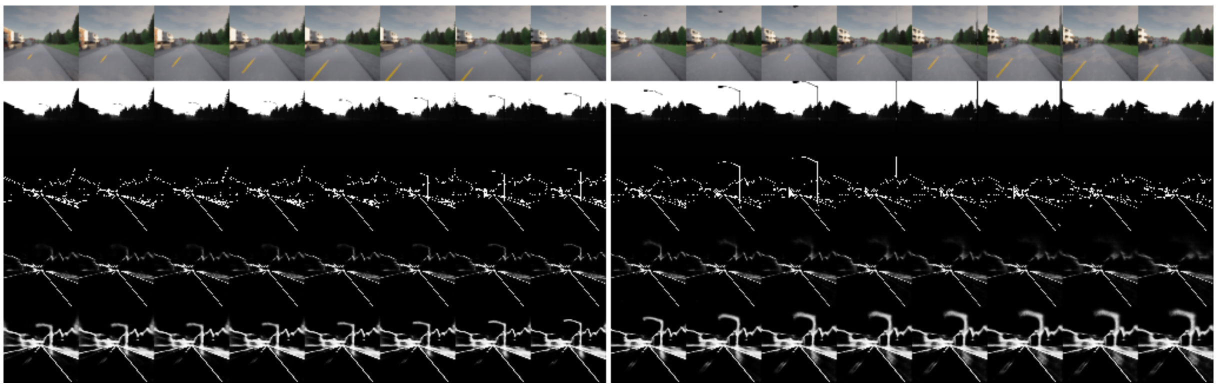

Figure 2.5.: Comparison between MSE and WBCE as loss function. Modalities top to bottom: RGB (input), depth (input), instance edges (target), MSE (result), WBCE (result). Evaluated after 196 epochs on the test set.

2.5. Interpretation

If evaluating the similarities between the models' outputs and targets only by the metrics SSIM and PSNR, it was shown that all models profit from receiving depth, instance segmentation, or semantic segmentation as an additional input to the RGB input. The latter two observations should not be surprising, as semantic segmentation and instance edges are directly derived from the instance segmentation images. However, this confirms that the models indeed took advantage of this privileged information.

When adding optical flow, only the DCGAN-based model improved in both metrics. It appears different spatial encoder-decoder networks do not utilize from visual modalities in the same way. ResNet e.g. got better scores in the baseline experiment than in the RGB+optical flow experiment.

However, these results should not be relied upon. As it was shown, when training models with WBCE instead of BCE or MSE, the SSIM and PSNR scores get worse even though the predictions subjectively get better.

As observed in the last paragraph of section 2.4, the WBCE-trained model predicted lane markings, which were not visible in the target but only in the input. This could have one of two causes: Either the model has overfitted to lane markings (such as the model from Figure 2.1) because they appear in the majority of the samples at the same pixels; Or, – since the lane markings are visible in some samples in the instance edges images –, the model learned that if a lane marking is visible in the RGB images, such a marking is usually also visible in the target images. The latter case would be preferred in a real-world situation.

3. Conclusion

For this work, a deterministic dataset generator for *CARLA* was proposed, which generates videos of a car driving around the simulation environment. Multiple different visual modalities can be extracted, as well as the motion data of all vehicles and pedestrians at every timestep.

With this data generator, a dataset was produced which includes 10 hours of video material, including RGB, depth, optical flow, instance segmentation, semantic segmentation, and instance edges. Also, the speed of the recording car, as well as the positions and orientations of all other cars and pedestrians, are included.

A highly parameterizable sequence-to-sequence model framework for video prediction is introduced, in which the spatial encoder-decoder networks can be switched out easily.

An extensive hyper-parameter search has yielded acceptable results for predicting future instance edges video frames from RGB and second visual input modalities. With a comparison between different spatial encoder-decoder networks (VGG, DCGAN, ResNet, U-Net, LinkNet), it was shown that ResNets seem to perform best in the task of predicting instance edges. It could not be answered with certainty which of the tested visual modalities profited the proposed model the most. However, it was shown that PSNR and SSIM are not well suited for evaluating image similarity between instance edges video frames. Furthermore, it was shown that models trained with WBCE as a loss function resulted in more satisfying outputs than when trained with MSE.

3.1. Discussion

In this thesis, many design decisions had to be made, as a dataset and a model framework for video prediction were implemented from scratch. Every decision must be chosen very carefully. As shown, changing only one of these decisions, as the loss function, has a high influence on the results. Working with the high-performance cluster, allowed for an extensive hyper-parameter search for trying out some of these design decisions. However, only a small portion of the possible parameter combinations could be tested.

A not reproducible problem on the cluster caused some experiments to kill a node of the cluster, and blocking it for multiple days until it got restarted. An I/O error could be the cause, however further investigation is necessary.

The models were trained with GPU support. However, the models did not utilize the GPUs to their maximum capacity. A cause for this problem could not be identified yet.

The accidental choice of setting the number of validation steps to 1 made the validation curve fluctuate too much to be representable. This should be corrected in future experiments.

3.2. Statement of problems left solved

For this work, MSE was used as the main loss function throughout the hyper-parameter search. MSE did not prove to be a good loss function for training on binary images. WBCE led to better results. Also, the metrics SSIM and PSNR were used for the evaluation of the results. Even though these metrics are commonly used for image similarity, it was shown that they do not apply well in evaluating binary image similarity. Further research is necessary to find more suitable loss functions and metrics.

The proposed model compresses and decompresses the video in space and time independently. As stated in 1.4.4, models like 3D CNNs exist, which compress videos in space and time at the same time. These models may lead to better frame predictions if provided with a larger dataset.

The generated dataset only consists of 10 hours of video material. In comparison to other video datasets like Kinetics-400[26], this is still comparably small. A larger dataset could possibly lead to better results. However, this is only a matter of computation time if the proposed dataset generator is used.

The raw data, coming from the dataset generator, consisted to 45% of sequences in which the recording car did not move at all. Movement of other cars or pedestrians in those sequences is very sparse. Predicting future frames of videos without movement is trivial. An algorithm would be needed to balance the dataset such that a satisfying distribution of movement is present across all sequences.

In further research, the learned representation vectors themselves could also be analyzed. Visualization technics like T-SNE could e.g. show if the model has learned a concept of the car's speed. For this, the model's predicted representation vector, as well as the current speed of the recording car, would be needed. Both already get saved when evaluating the proposed model.

4. List of Tables

5. List of Figures

6. Bibliography

[1] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press, May 2014, google-Books-ID: Hf6QAwAAQBAJ. [Page 3]

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org/ [Page 3]

[3] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into Deep Learning," *arXiv:2106.11342 [cs]*, Jul. 2021, arXiv: 2106.11342. [Online]. Available: http://arxiv.org/abs/2106.11342 [Page 3]

[4] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," *arXiv:1409.3215 [cs]*, Dec. 2014, arXiv: 1409.3215. [Online]. Available: http://arxiv.org/abs/1409.3215 [Page 4]

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs]*, Dec. 2015, arXiv: 1512.03385. [Online]. Available: http://arxiv.org/abs/1512.03385 [Pages 4 and 12]

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *arXiv:1505.04597 [cs]*, May 2015, arXiv: 1505.04597. [Online]. Available: http://arxiv.org/abs/1505.04597 [Page 4]

[7] T. Zia and U. Zahid, "Long short-term memory recurrent neural network architectures for Urdu acoustic modeling," *International Journal of Speech Technology*, vol. 22, no. 1, pp. 21–30, Mar. 2019. [Online]. Available: https://doi.org/10.1007/s10772-018-09573-7 [Page 4]

[8] Y. Li, J. Zhang, Y. Cheng, K. Huang, and T. Tan, "Semantics-guided multi-level RGB-D feature fusion for indoor semantic segmentation," in *2017 IEEE International Conference on Image Processing (ICIP)*, Sep. 2017, pp. 1262–1266. [Page 4]

[9] L. Jing and Y. Tian, "Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey," *arXiv:1902.06162 [cs]*, Feb. 2019, arXiv: 1902.06162. [Online]. Available: http://arxiv.org/abs/1902.06162 [Page 4]

[10] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep Ordinal Regression Network for Monocular Depth Estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2002–2011. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/html/Fu_Deep_Ordinal_Regression_CVPR_2018_paper.html [Page 5]

[11] G. Varol, I. Laptev, and C. Schmid, "Long-term Temporal Convolutions for Action Recognition," *arXiv:1604.04494 [cs]*, Jun. 2017, arXiv: 1604.04494. [Online]. Available: http://arxiv.org/abs/1604.04494 [Pages 5 and 6]

[12] J. McCormac, A. Handa, S. Leutenegger, and A. J. Davison, “SceneNet RGB-D: Can 5M Synthetic Images Beat Generic ImageNet Pre-training on Indoor Segmentation?” in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 2697–2706, iSSN: 2380-7504. [Page 5]

[13] P. Krahenbuhl, “Free Supervision from Video Games,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 2955–2964, iSSN: 2575-7075. [Page 5]

[14] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An Open Urban Driving Simulator,” *arXiv:1711.03938 [cs]*, Nov. 2017, arXiv: 1711.03938. [Online]. Available: http://arxiv.org/abs/1711.03938 [Pages 6, 8, and 9]

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, number: 7540 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/nature14236 [Page 6]

[16] A. Dosovitskiy and V. Koltun, “Learning to Act by Predicting the Future,” *arXiv:1611.01779 [cs]*, Feb. 2017, arXiv: 1611.01779. [Online]. Available: http://arxiv.org/abs/1611.01779 [Page 6]

[17] A. Sax, J. O. Zhang, B. Emi, A. Zamir, S. Savarese, L. Guibas, and J. Malik, “Learning to Navigate Using Mid-Level Visual Priors,” Dec. 2019. [Online]. Available: http://arxiv.org/abs/1912.11121 [Page 6]

[18] A. Zamir, A. Sax, W. Shen, L. Guibas, J. Malik, and S. Savarese, “Taskonomy: Disentangling Task Transfer Learning,” *arXiv:1804.08328 [cs]*, Apr. 2018, arXiv: 1804.08328. [Online]. Available: http://arxiv.org/abs/1804.08328 [Page 6]

[19] B. Zhou, P. Krähenbühl, and V. Koltun, “Does computer vision matter for action?” *Science Robotics*, vol. 4, no. 30, May 2019. [Online]. Available: https://arxiv.org/abs/1905.12887 [Page 6]

[20] L. Sevilla-Lara, Y. Liao, F. Guney, V. Jampani, A. Geiger, and M. J. Black, “On the Integration of Optical Flow and Action Recognition,” *arXiv:1712.08416 [cs]*, Dec. 2017, arXiv: 1712.08416. [Online]. Available: http://arxiv.org/abs/1712.08416 [Page 6]

[21] J. Carreira and A. Zisserman, “Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset,” *arXiv:1705.07750 [cs]*, Feb. 2018, arXiv: 1705.07750. [Online]. Available: http://arxiv.org/abs/1705.07750 [Pages 6 and 7]

[22] K. Hara, H. Kataoka, and Y. Satoh, “Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet?” *arXiv:1711.09577 [cs]*, Apr. 2018, arXiv: 1711.09577. [Online]. Available: http://arxiv.org/abs/1711.09577 [Page 7]

[23] K. Soomro, A. R. Zamir, and M. Shah, “UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild,” *arXiv:1212.0402 [cs]*, Dec. 2012, arXiv: 1212.0402. [Online]. Available: http://arxiv.org/abs/1212.0402 [Pages 7 and 10]

[24] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles, “ActivityNet: A large-scale video benchmark for human activity understanding,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 961–970, iSSN: 1063-6919. [Page 7]

[25] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: A large video database for human motion recognition," in *2011 International Conference on Computer Vision*, Nov. 2011, pp. 2556–2563, iSSN: 2380-7504. [Page 7]

[26] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, "The Kinetics Human Action Video Dataset," *arXiv:1705.06950 [cs]*, May 2017, arXiv: 1705.06950. [Online]. Available: http://arxiv.org/abs/1705.06950 [Pages 7, 10, and 19]

[27] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised Learning of Video Representations using LSTMs," *arXiv:1502.04681 [cs]*, Jan. 2016, arXiv: 1502.04681. [Online]. Available: http://arxiv.org/abs/1502.04681 [Pages 7, 10, and 12]

[28] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: https://www.tensorflow.org/ [Page 9]

[29] F. Chollet *et al.*, "Keras," 2015. [Online]. Available: https://keras.io [Page 9]

[30] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv:1409.1556 [cs]*, Apr. 2015, arXiv: 1409.1556. [Online]. Available: http://arxiv.org/abs/1409.1556 [Page 12]

[31] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," *arXiv:1511.06434 [cs]*, Jan. 2016, arXiv: 1511.06434. [Online]. Available: http://arxiv.org/abs/1511.06434 [Page 12]

[32] P. Iakubovskii, "Segmentation Models," Mar. 2022, original-date: 2018-06-05T13:27:56Z. [Online]. Available: https://github.com/qubvel/segmentation_models [Page 12]

[33] A. H. Abdulnabi, B. Shuai, Z. Zuo, L.-P. Chau, and G. Wang, "Multimodal Recurrent Neural Networks with Information Transfer Layers for Indoor Scene Labeling," *IEEE Transactions on Multimedia*, vol. 20, no. 7, pp. 1656–1671, Jul. 2018, arXiv: 1803.04687. [Online]. Available: http://arxiv.org/abs/1803.04687 [Page 12]

[34] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," *arXiv:1511.05440 [cs, stat]*, Feb. 2016, arXiv: 1511.05440. [Online]. Available: http://arxiv.org/abs/1511.05440 [Page 12]

[35] R. Rane, E. Szügyi, V. Saxena, A. Ofner, and S. Stober, "PredNet and Predictive Coding: A Critical Review," *Proceedings of the 2020 International Conference on Multimedia Retrieval*, pp. 233–241, Jun. 2020, arXiv: 1906.11902. [Online]. Available: http://arxiv.org/abs/1906.11902 [Page 13]

[36] R. Deng, C. Shen, S. Liu, H. Wang, and X. Liu, "Learning to predict crisp boundaries," *arXiv:1807.10097 [cs]*, Jul. 2018, arXiv: 1807.10097. [Online]. Available: http://arxiv.org/abs/1807.10097 [Page 13]

Appendices

A. Additional figures and data

A.1. Difference between training with WBCE and MSE

This data is referred to in 2.4.

Modality	PSNR	SSIM
Baseline	-4.305	-0.14
Depth	-4.211	-0.1452
InsSeg	-5.439	-0.1580
SemSeg	-4.39	-0.1580
OptFl	-3.995	-0.1161

Table A.1.: Difference between training the ResNet-based model with WBCE and MSE as the loss function. The values are the differences: $score_{WBCE} - score_{MSE}$ i.e. negative values indicate that the WBCE-trained model achieved a lower score than the MSE-trained model. See A.4 for raw data.

A.2. Training and validation curves for the instance segmentation model

This figure is referred to in 2.4.

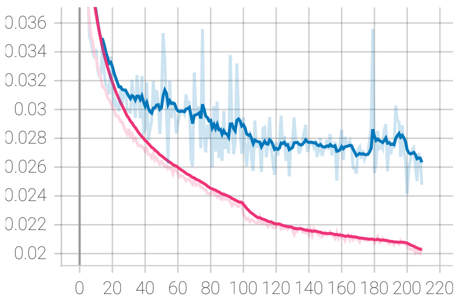

Figure A.1.: Training and validation curves for the instance segmentation model. The x-axis is the MSE loss. The y-axis are the training epochs. Pink: training curve; Blue: validation curve. Both curves were smoothened to counteract the fluctuations of the validation curve. The "pale" curves show the original training and validation losses.

A.3. Metric data for different spatial encoder-decoder networks and modalities after 98 epochs

This data is used in Figure 2.3.

A.3.1. ResNet

This data is used in Table 2.3.

ResNet	PSNR	SSIM
RGB	15.118666648864746	0.5891214609146118
Depth	15.340457916259766	0.614318311214447
InsSeg	15.628207206726074	0.6533489227294922
SemSeg	15.90740966796875	0.6405366063117981
OptFl	14.537500381469727	0.5311346650123596

A.3.2. UNet

UNet	PSNR	SSIM
RGB	15.125197410583496	0.5694199800491333
Depth	15.26447582244873	0.6068575978279114
InsSeg	15.608999252319336	0.6403436660766602
SemSeg	15.708322525024414	0.6485517024993896
OptFl	15.06726360321045	0.5779425501823425

A.3.3. DCGAN

DCGAN	PSNR	SSIM
RGB	14.90750789642334	0.49102887511253357
Depth	15.132073402404785	0.5795562267303467
InsSeg	15.385443687438965	0.4974871873855591
SemSeg	15.69117259979248	0.6198152899742126
OptFl	14.97403621673584	0.5396600365638733

A.4. Metric data for different losses and modalities after 196 epochs

This data is used in A.1.

A.4.1. MSE

This data is used in Table 2.3.

ResNet	PSNR	SSIM
RGB	15.358552932739258	0.608020007610321
Depth	15.528942108154297	0.6361757516860962
InsSeg	15.728174209594727	0.49578016996383667
SemSeg	16.155529022216797	0.6883578300476074
OptFl	15.273910522460938	0.6224478483200073

A.4.2. WBCE

ResNet	PSNR	SSIM
RGB	11.05336856842041	0.46804699301719666
Depth	11.317829132080078	0.490978479385376
InsSeg	10.288700103759766	0.2579684853553772
SemSeg	11.765873908996582	0.5303319692611694
OptFl	11.278923988342285	0.506332278251648

B. Code and dataset access

B.1. Code

Detailed explanations on how to use the code can be found in the README.md files. The code is available at TubCloud, which is password protected:

URL: [redacted]

Password: [redacted]

The code is also available on GitLab of TU Berlin. However, access permission must be requested:

URL: [redacted]

E-Mail: l.schroeder@campus.tu-berlin.de

B.2. Experiment results

To access the experiment results (486 GB including the code and checkpoints), you need to have access to [redacted].

All experiments conducted in this thesis can be found at:

[redacted]

B.3. Dataset

To access the porposed dataset (382 GB unzipped; 112 GB zipped; 44 GB TFRecords), you need to have access to [redacted].

A sample of the training set can be found at:

URL: `https://tubcloud.tu-berlin.de/s/5ygs7dbMjDJWtqX`

The entire dataset can be found at:

[redacted]

and

[redacted]