Autonomous Driving in Simulation using Domain-Independent Perception

Audun Wigum Arbo*, Even Dalen*, and Frank Lindseth

Norwegian University of Science and Technology, Trondheim, Norway
audun.wa@gmail.com, even.dalen@live.no, frankl@ntnu.no

Abstract. There have been great advancements within the fields of computer vision and autonomous driving in recent years. Autonomous vehicle systems have typically consisted of mostly handmade algorithms and rules, where neural networks have been assisting in perception tasks. Increased usage of neural network have shown promising results, and many state-of-the-art systems now utilize these for increasingly complex tasks.
This paper builds upon recent work in building end-to-end networks for autonomous driving. Conditional imitation learning is combined with a perception network that generate high level abstractions from RGB images. We examine the driving performance implications of learning to drive from raw RGB images, semantic segmentation, and depth estimates. Secondly, we propose a driving network which uses raw RGB images to predict semantic segmentation and depth maps of the scene, which furthermore is used to predict output a steering angle and a target speed for the vehicle. The models are evaluated in CARLA, an open-source simulator for autonomous driving research, in various environments: an urban town, a rural road with surrounding farms and fields, and in greatly varied weather conditions.
Our experiments show that the driving network trained on higher-level abstractions generalize better than a model trained directly on RGB images in simulation, even when the perception model is trained on real-world data. We also show that the perception model trained on several tasks using multi-task learning, leads to better-performing driving policies than learning only semantic segmentation.

Keywords: End-to-end Autonomous Driving · AV Domain Transfer · Multi-task Learning · Conditional Imitation Learning · Semantic Segmentation · Depth Estimation

1 Introduction

Autonomous vehicles have been a popular research domain for many years, and there has recently been large investments from both technology and car com-

* These authors contributed equally to this work.
 Colour and Visual Computing Symposium 2020, Gjøvik, Norway, September 16-17, 2020.

panies to be the first to solve the problem. The most prominent approach in recent years has been the modular approach, where the driving is divided into several sub-tasks such as perception, localization, and planning. The modular approach often results in a very complex solution, where each module has to be fine tuned individually. The scalability of this approach can therefore become an issue when expanding the approach to more complex situations.

Another rising approach is the end-to-end approach, where the entire driving policy is generated within a single system. The system takes sensor-input and converts it directly to driving commands, similar to how humans drive vehicles. End-to-end systems for autonomous vehicles require large amounts of data, and the ability to train on many different scenarios. Therefore, simulated environments have been explored for training in different scenarios and creating large datasets. These environments, however, differ significantly from the real world, and the learned driving policy does not transfer adequately between environments.

This paper attempts to improve the ability for driving policies to be transferred between domains by abstracting away both the perception task, and the raw throttle and brake control of the vehicle, focusing mainly on the perception task. The Mapillary Vistas dataset [21] is used for learning perception in a real-world driving environment, and the autonomous vehicle simulator CARLA [9] is used to learn both driving and perception. The ultimate goal of this paper is to reduce the amount of real-world data required to train an autonomous vehicle, by utilizing simulated environments for training.

The paper is structured as follows: Section 2 investigates related work, while additionally providing a brief history of the field itself. Section 3 presents our method, including the data, neural network architectures, and evaluation metrics. Section 4 describes our experiments, their results, and discussion related to these. Section 5 discusses the overall implications of the experimental results, and compare our results with conclusions from related work. Section 6 draws a final conclusion of the work conducted, and addresses the paper's merits, weaknesses, and potential future work.

2 Related Work

[27] arrange autonomous vehicle control algorithms into two categories: modular approaches and end-to-end approaches. Modular approaches divides the responsibility of driving into several sub-tasks, such as perception, localization, planning, and control. Conversely, end-to-end approaches can be defined as a function $f(x) = a$ where x is any input needed to make decisions — typically sensor data and environmental information — and a are the output controls that are sent to the vehicle's actuators.

The end-to-end approach was first demonstrated in the ALVINN project, described by [22]. ALVINN was able to follow simple tracks, but had no means to handle more complex environments. Since then, large advancements have been done within neural networks, resulting in new research within end-to-end

vehicle control. [3] approaches the problem using modern techniques, and showcase a driving policy capable of driving on both highways and residential roads; in varied weather conditions. More recent approaches [6,20,15,14,26] are based on Conditional Imitation Learning (CIL), introduced by [6] in 2017, where the driving policy is given instructions — high-level commands (HLCs) — on which actions to take (e.g. turn left in next intersection). [6] shows that an architecture can be re-used for both simulated and physical environments, but they make no attempt to use the same model weights across the two domains. Codevilla et. al outputs a steering angle, and either throttle or brake, which are sent to the vehicle's control systems. [15] outputs the target speed of the vehicle, leaving the raw throttle and brake adjustments to a lower-level system. [20] proposes to abstract the commands even further; into several waypoints in space. Their model outputs two waypoints, 5 and 20 meters away from the vehicle, which a PID controller uses to control the vehicle's steering and velocity.

Transfer from simulation to real world. A lot of studies have been done on transferring learning from simulation to the real world. [18] used images from the driving game Grand Theft Auto, to train their object detection model, and achieved state of the art performance on the KITTI [10] and Cityscapes [8] datasets. [4] successfully used simulation to train a model for robotic grasping of physical unseen objects. Among the techniques used was applying randomization in the form of random textures, lighting, and camera position, to enable their model to generalize from the simulated source domain to their physical target domain.

Transferring driving policies between domains also require an abstraction of the perception data. [20] uses a perception model to generate segmentation maps which are forwarded to the driving model, in order to generate similar perception environments for both simulation and real-world. [26] combines ground-truth segmentation and depth data from CARLA to increase driving performance. [15] uses an encoder-decoder network with three decoder-heads — segmentation, depth estimation and original RGB reproduction — to maximize the model's scene understanding. Hawke et al. also removes the decoding-process when training their driving policy, making their driving policy model take only the compressed encoding of scene understanding as input. [19] finds that the performance of such multi-task prediction models depend highly on the relative weighting between each task's loss. Tuning these weights manually is an error prone and time-consuming process, and they therefore suggest a solution for tuning weights based on the homoscedastic uncertainty of each task. They show that the multi-task approach outperformes separate models trained individually. The uncertainty based weighing was later used by Hawke et al. and produced good results for generating optimal encoding of a driving scene. Depth images has also been proven as a useful approach in other simulation-to-real world knowledge transfers, such as robotic grasping [25,12].

3 Data and Methods

Our approach consists of two separately trained models: a perception model and a driving model. The reason for this separation is to decouple the task of scene understanding from the task of driving. This opens up the possibility of improving the tasks independently, and we can train the models separately and with different datasets. An important goal was then to make the output of the perception model domain-independent, which in turn makes the driving policy model domain-independent. Domain independence is in this context defined as the ability to generalize between multiple domains (e.g. simulated and real), as well as completely unseen domains.

The perception model is trained on datasets containing RGB images paired with semantic segmentation and depth information. These datasets can contain images not directly related to driving, as they are used to train a general scene understanding.

The driving model is trained on datasets recorded from an expert driver. The dataset contains RGB images and driving data such as steering angle, current speed and target speed. The datasets for both models can either be collected from the real world, generated from the CARLA simulator, or a combination of both real-world and simulated data.

3.1 Perception Model

The perception model takes raw RGB images as input, and tries to predict one or more outputs related to scene understanding; always semantic segmentation, and in some experiments an additional depth map. The model has an encoder-decoder structure, compressing the input into a layer with few neurons (encoder) before expanding towards one or more prediction outputs (decoders). To train the model, data from driving situations in different environments and geographical areas are used. Some experiment also use data generated from CARLA as a means to improve the model's performance in simulated environments.

Data. The *Mapillary Vistas* dataset [21] (henceforth Mapillary) was used for RGB and ground-truth semantic segmentation data. The dataset consists of 25 000 high-resolution images from different driving situations, with a large variety of weather and geographical locations. To simplify the environment for the perception network, the number of classes for segmentation was reduced from the original 66 object classes, to five classes: unlabeled, road, lane markings, humans and vehicles. To train the model's depth decoder, ground truth depth maps were generated using the Monodepth2 network from [11], as Mapillary lacks this information. Figure 1 shows a sample from this dataset.

Fig. 1: Sample of the data used when training the perception model. The left image is the original RGB. The center image is segmentation ground truth from Mapillary. The right depth map was generated from RGB images with the Monodepth2 network.

In addition to using real-world perception data, we generated synthesized data in CARLA. A Python script spawns a large variety of vehicles and pedestrians, and captures RGB, semantic segmentation, and depth data from the vehicles as they navigate the simulated world. The field of view (FOV) and camera yaw angle were randomly distributed to generalize between different camera setups. The simulated weather was additionally changed periodically; varying cloudiness, amount of rain, time of day, and other modifiable weather parameters in CARLA. The final size of this synthesized dataset is about 20 000 images.

Architecture. Several encoders and decoders were explored when deciding the model's architecture. Encoders tested were: MobileNet [17], ResNet-50 [16] and a vanilla CNN, while decoders tested were: U-Net[23] and SegNet[2]. To generate a network that could predict both depth and segmentation estimations, we modified the existing MobileNet-U-Net architecture to include a second U-Net decoder. The decoder was modified to predict only one value per pixel, use the sigmoid activation function, and train with a regression loss function for depth estimation, adapted from [1]. Figure 2 illustrates the new MobileNet-U-Net with two decoders.

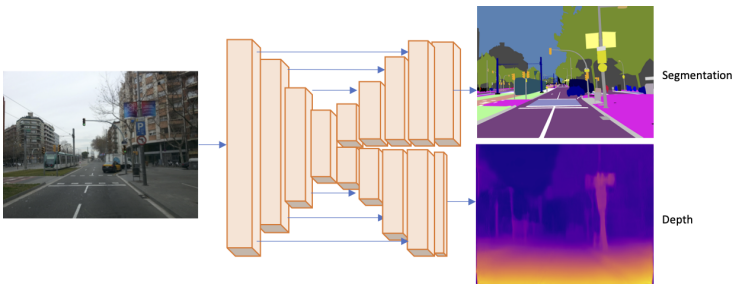

Fig. 2: A simplified illustration of the perception model. The different architectures all used a variant of the encoder-decoder architecture. The figure represents the MobileNet-U-Net model with a second depth estimation decoder, where each layer in the encoder is connected to the corresponding layer in the decoder.

Evaluation and Metrics. The segmentation prediction was evaluated using Intersection over Union (IoU), calculated with the following equation: $\frac{gt \cap p}{gt \cup p}$, where gt is the ground truth segmentation and p is the predicted segmentation. Mean IoU was used as the main indicator for performance, calculated by taking the mean of the class-wise IoU. Frequency weighted IoU was also calculated, measured as the mean IoU weighted by the number of pixels for each class.

The accuracy within threshold, as described in [5], was chosen as the metric for depth estimation. Given the predicted depth value d_p and the ground truth depth value d_{gt}, the accuracy δ within threshold th is defined as $max(\frac{d_{gt}}{d_p}, \frac{d_p}{d_{gt}}) = \delta > th$. Each pixel gets labeled as true or false based on whether the pixel is within the specified threshold or not. The accuracy of an image is then calculated by taking the average of all the pixels in the image. th is a threshold that we varied between the values 1.25, 1.25^2, and 1.25^3, as in [5].

3.2 Driving Model

The driving model runs raw RGB images through the perception model, and uses its output segmentation and depth predictions as input. These images are coupled with driving data recorded from an expert driver. The driving model processes these inputs through its own layers, before outputting a steering angle and target speed.

Data. The driving data was generated in CARLA version 0.9.9. This was done by making an autopilot control a car in various environments, and recording video from three forward-facing cameras, its steering angle, speed, target speed, and HLC (*left*, *right*, *straight*, or *follow lane*). The autopilot has access to the full state of the world, which includes a HD map, its own location and velocity, and the states of other vehicles and pedestrians. It uses this information to generate waypoints, which are finally fed into a PID-based controller to apply throttle, brake, and steering angle. The collected training data was unevenly distributed in regards to HLCs and steering angles, and we therefore down-sampled over-represented values for an improved data distribution.

Various datasets were gathered for training the driving policy, all of which were collected in *Town01*. These have different amount of complexity; steering noise magnitude the autopilot has to account for, different weather conditions and different light conditions. 30 641 samples were collected in total, where the weather varied according to CARLA's 15 default weather presets. The training data was effectively multiplied by three, as we made two copies of each data point, where we used the recorded image from each side camera instead of the main camera. To adjust for a slightly modified camera perspective, we added an offset of 0.05 and -0.05 in steering angle respectively for the left and right camera variants. This technique was first introduced by [3], and has later proved successfully in other papers [6,14].

Architecture. The input of the driving model is a concatenation of the output from the perception model and an information vector containing the HLC (one-hot encoded), current speed, current speed limit, and the upcoming traffic light's state. The driving model is trained on simulation data with all the layers of the perception model frozen (that is, non-trainable) to preserve generalizability. Figure 3 shows an overview of the model.

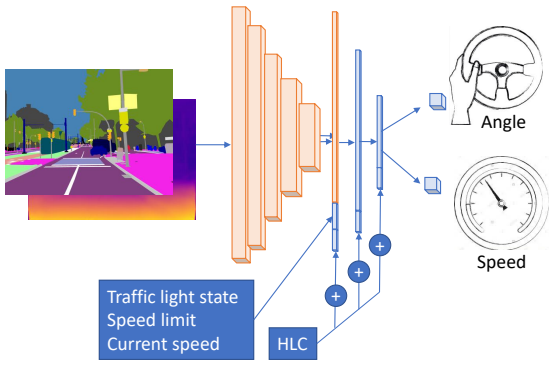

Fig. 3: A simplified illustration of the driving network. The segmentation and depth maps inputs are concatenated directly from the outputs of the perception model (shown in Figure 2).

The segmentation and depth output of the perception model are concatenated channel-wise, and resemble a RGBD (RGB + depth) image. This representation is then run through 5 convolutional blocks, each consisting of zero padding of 1, 2D-convolution with kernel 3, batch normalization, ReLu activation, and finally max-pooling with pool size 2. The filter sizes are 64, 128, 256, 256, 256, respectively. The current HLC, whether the traffic light was red or not, speed, and speed limit are concatenated with feature vectors generated from the perception data. The last layers are a combination of fully-connected layers, where we concatenate the HLC vector at each step, similar to [15]. The first output of the model is the steering prediction; one neuron outputting the optimal steering (between 0 and 1, 0 being max leftward, 1 being max rightward), later mapped to CARLA's [-1, 1] range. The second output is the optimal vehicle speed, outputted as a percentage of 100 km/h (between 0 and 1).

Evaluation and Metrics. The main metric used for measuring driving model performance was Mean Completion Rate (MCR) during real-time evaluation. This is calculated by dividing the completed distance d_c by the total route distance d_t of each run-through of a route, averaged over all run-throughs R:

$\frac{\sum_{r \epsilon R} \frac{d_c}{d_t}}{|R|}$. Traffic violations were not included as metrics, as the scope of this paper is mainly within completing routes without major incidents, and the models were therefore not trained to avoid such violations. The model's validation loss was also used as a rough metric for performance. By empirical observations we only picked models with validation loss $\lessapprox 0.03$ for further evaluation. The validation loss metric was used as an initial performance estimation because the MCR evaluation was considerably more time consuming.

4 Experiments and results

There are two main experiments conducted in this paper. The first experiment and its sub-experiments focuses on generating the best perception model to be used when training the driving network. Model architecture, dataset variants, augmentation, and multi-task learning are parameters experimented with to increase performance. The second experiment is conducted in CARLA. This experiment assess the driving policy performance given the different models derived in the first set of experiments. The generalizability of each model is tested using different unseen environments. Each perception model is then compared to a baseline model trained only on the CARLA dataset using Mean Completion Rate as the metric.

4.1 Experiment 1: Perception Model

The perception experiments use semantic segmentation and occasionally depth estimation to generalize the driving environment when training and testing the driving models. All of the perception experiments use the same dataset for evaluation, and the results can therefore be compared across experiments. The CARLA data generated and used for evaluation consists of 4 400 images with corresponding ground truth segmentation and depth maps from Town 3-4. The Mapillary evaluation dataset is a set of 2 000 images from the original Mapillary test set.

Experiment 1-1: Encoder-decoder models. This experiment attempts to find the best encoder and decoder to use for the perception network. All the encoders tested were picked because they have previously shown good results in other papers, and were implemented in a common library by [13].

The three encoders showed increased performance as the complexity and size of the encoder increased. The Vanilla CNN encoder performed worst with the lowest Mean IoU score, however, it was also the fastest model during training and testing. MobileNet gave better results while keeping a lot of the speed advantage from the Vanilla CNN network. MobileNet also showed very good results, with MobileNet-U-Net displaying the best overall performance when combining scores. ResNet50 performed good as expected with a higher Mean IoU than MobileNet-U-Net, however the difference from MobileNet-U-Net was less than

Model	Mean IoU	Weighted IoU
VanillaCNN-SegNet	0.324	0.712
VanillaCNN-U-Net	0.351	0.705
MobileNet-SegNet	0.368	**0.775**
MobileNet-U-Net	0.403	0.774
ResNet50-SegNet	**0.405**	0.767
ResNet50-U-Net	0.383	0.733

Table 1: Evaluation of three different encoders (Vanilla CNN, Mobilenet and ResNet50), and two decoders (SegNet and U-Net). Each model was trained on the Mapillary dataset (18 000 samples for training and 2 000 for validation) without any augmentation. The best scores are marked in bold.

expected. MobileNet was used for futher experiments as it was significantly faster than ResNet50.

Experiment 1-2: Training data. To improve the model further some CARLA data was introduced to the Mapillary dataset. Augmentation was also introduced for further improvements and better generalization. The Mapillary+CARLA dataset consisted of 20 000 datapoints from the Mapillary dataset and 3 250 samples from *Town01* and *Town02* in CARLA. The dataset with only augmented CARLA data (CARLA+Aug) used a different dataset of 15 000 samples from Town 1-4, and 4 000 samples from Town 5 as validation. The results were evaluated on Town 3-4 as Town 1-2 was used when training Mapillary+CARLA. The augmentation included consists of among others gaussian noise, translation, rotation, hue and saturation augmentations, and was adapted from [13].

	CARLA Eval		*Mapillary Eval*	
Training dataset	Mean IoU	Weighted IoU	Mean IoU	Weighted IoU
Mapillary	0.425	0.771	0.632	0.887
Mapillary+Aug	0.436	0.809	0.574	0.873
Mapillary+CARLA	0.469	0.846	**0.633**	**0.889**
Mapillary+CARLA+Aug	0.478	0.850	0.568	0.874
CARLA+Aug	**0.572**	**0.909**	0.384	0.785

Table 2: Evaluation of different datasets on the MobileNet-U-Net model. Mapillary is the original dataset while the CARLA dataset was generated directly from CARLA. Each dataset consist of about 20 000 samples, and the Mapillary+CARLA dataset consist of about 80/20 Mapillary and CARLA data respectively. The two sets of columns show evaluation on CARLA data and Mapillary data respectively. The best scores are marked in bold.

The dataset experiment shows that including CARLA data as a component when training the perception models increases the total performance. As the model's goal is to make good predictions in both real and simulated environments, combining data from both seems to be a reasonable approach. CARLA+Aug achieves great results when evaluating on CARLA data, however the performance decreased drastically when predicting in real-world environments. Models trained on real-world data tends to generalize better to unseen simulated environments than the other way around. Incorporating some CARLA data into the real-world data in addition to augmenting the images yields the best results overall.

Experiment 1-3: Multi-task perception. Inspired by [15] we introduced a depth estimation decoder to the MobileNet-U-Net model. The model was trained with ground truth data generated by the Monodepth2 network using images from the Mapillary dataset, while depth maps for the CARLA data was included in the generated CARLA dataset.

	Segmentation		*Depth*		
Training dataset	Mean IoU	Weighted IoU	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Mapillary	0.458 (+0.03)	0.817	0.320	0.572	0.684
Mapillary+CARLA	0.520 (+0.05)	0.854	0.295	0.542	0.679
CARLA	0.717 (+0.15)	0.960	0.775	0.806	0.816

Table 3: Results after adding a depth estimation decoder to the Mobilenet-U-Net model. Each model was trained on the same dataset as in Experiment 2. Mean IoU additionally presents a difference in parantheses: the difference between these models' mean IoU and their counterparts' from Experiment 1. Depth is estimated using *accuracy within threshold*, where the set threshold is presented in the column title. A high value is best for all metrics in the table.

Including a depth estimation decoder increases the segmentation performance for each model. The mean increase in IoU on the CARLA test set is 8%, which conforms with the results reported by [24], who reported a 4.17% increase in performance when training semantic segmentation with depth estimation. An increase in overall scene understanding can also be expected as depth is introduced to the model, however this has to be verified as part of the overall driving policy experiments.

4.2 Experiment 2: Driving Model

This experiment aims to assess the overall performance of the two-part (perception and driving policy) architecture. We run real-time evaluations on variants

of our proposed architecture, including a baseline network where the complete network is trained at once. The evaluation is conducted with a custom scenario runner for CARLA, originally introduced by [14], and extended for our experiments. The real-time nature of this experiment makes it different from the previous experiments: The models' steering and speed outputs affect the camera input in subsequent simulation steps, and each prediction is therefore dependent on the ones before.

The scenario runner. The scenario runner makes each model drive through a predefined set of routes, each of which is defined by a set of waypoints. The model navigates each route using HLCs provided automatically when passing each waypoint. Each attempt at a route ends either when the vehicle completes the route, or when the vehicle enters any of the following erroneous states: stuck on an obstacle, leaving its correct lane and not returning within five seconds, or ignoring a HLC. The models are then compared on their mean route completion rate.

Environments and Routes. The models were tested in two environments, *Town02* and *Town07*. *Town02* is similar, but not identical to the one in the driving policy's training data, which is *Town01*. *Town07* is quite different, and is rural with narrow roads (some without any centre marking), fields, and barns. There are three routes in each environment, which the models will try to complete in six different weather conditions. Three of the weather conditions have already been observed in the training data, while the three remaining are unknown to the policy. The training data only contain samples from day-time weathers, but two of the unknown weathers are at midnight.

Results. Table 4 summarizes the driving performance of the different models. The model trained only on driving data and without a frozen perception model, *RGB*, was the best-performing model on *Town02*, but it struggles with *Town07*.

The model names starting with *SD* indicates that they use the segmentation and depth perception model (henceforth SD). *SD-CARLA*, which uses SD trained only on perception data from CARLA, outperforms all other models when ranked by Mean Completion Rate (MCR) over both towns. To demonstrate its performance, we made a video (`https://youtu.be/HL5LStDe7wY`) showing some of its good performing moments. *SD-Mapillary* uses SD as well, but only had perception training data from Mapillary. While not performing as good as SD-CARLA, it still has impressive results. Its perception model has not seen any CARLA data, but is still able to predict segmentation and depth good enough for the driving model to beat even the RGB model. *SD-Combined* used perception data from both Mapillary and CARLA, and performs a little bit worse than SD-Mapillary.

The model names starting with *S* indicates that they use the segmentation-only perception model (henceforth S). *S-CARLA* is the S-counterpart of SD-CARLA, and it performs very well in *Town02*. In *Town07* however, it struggles

	Seen weather			*Unseen weather*			
Model	**Clear (D)**	**Rain (D)**	**Wet (S)**	**Clear (N)**	**Rain (N)**	**Fog (S)**	**Mean**
RGB	100.00 %	28.72 %	36.05 %	100.00 %	11.22 %	100.00 %	62.67 %
SD-CARLA	100.00 %	43.30 %	55.36 %	100.00 %	23.46 %	44.96 %	61.18 %
S-CARLA	93.83 %	7.23 %	43.51 %	100.00 %	24.61 %	66.66 %	55.97 %
SD-Mapillary	88.51 %	42.16 %	67.22 %	100.00 %	11.59 %	24.91 %	55.73 %
SD-Combined	93.53 %	9.92 %	47.42 %	100.00 %	9.92 %	46.53 %	51.22 %
S-Combined	90.71 %	44.02 %	23.12 %	72.12 %	2.72 %	7.23 %	39.98 %
S-Mapillary	72.12 %	43.30 %	40.85 %	39.34 %	2.72 %	2.72 %	33.51 %

(a) Results for *Town02*.

	Seen weather			*Unseen weather*			
Model	**Clear (D)**	**Rain (D)**	**Wet (S)**	**Clear (N)**	**Rain (N)**	**Fog (S)**	**Mean**
SD-CARLA	84.95 %	61.14 %	84.95 %	60.81 %	88.88 %	17.19 %	66.32 %
SD-Mapillary	77.28 %	77.28 %	55.54 %	51.61 %	33.33 %	14.84 %	51.65 %
SD-Combined	50.52 %	61.14 %	57.39 %	38.60 %	66.67 %	33.33 %	51.27 %
S-Combined	77.83 %	55.10 %	55.10 %	17.32 %	50.65 %	33.33 %	48.22 %
RGB	44.49 %	44.49 %	44.49 %	44.49 %	49.75 %	44.49 %	45.37 %
S-CARLA	55.10 %	55.10 %	57.39 %	17.32 %	17.32 %	61.87 %	44.02 %
S-Mapillary	51.61 %	50.52 %	44.49 %	55.10 %	44.49 %	0.00 %	41.04 %

(b) Results for *Town07*.

Table 4: Mean completion rate in (a) *Town02* and (b) *Town07*, in six weather conditions. Day, Sunset and Night is shortened to D, S, N respectively. The individual cells are colored on a scale where green is the best, and red is the worst. Note that no cars or pedestrians were included in the traffic during these experiments.

with night-time weather. S-Mapillary is the S-counterpart of SD-Mapillary, and it has the lowest MCR in both towns. In any run with *Fog (S)*, it fails almost immediately. *S-Combined* uses combined perception data, the same as *SD-Combined*. It is performing a bit better than S-Mapillary in *Town02*, and is the fourth best in *Town07*.

5 Discussions

5.1 Results in comparison to related work

[26] uses ground-truth semantic segmentation data generated from CARLA, not predicted as we do, and combine segmentation with both ground-truth depth maps and depth estimated by a separate network. Their results aligns with our results; using semantic segmentation data beats just using raw images, and combining both segmentation and depth performs the best. With a combination of ground truth segmentation and estimated depth, their policy is still able to beat the raw image-based policy. Our models estimate both segmentation and depth, and is still able to perform good in comparison to our baseline RGB-model.

[20] use predicted binary segmentation (road/not road) as driving input, and our work extends this with predicted depth, giving additional performance benefits. [14] achieved higher completion rates even with traffic, but focused more on the impact of larger datasets and encoding temporal information in the model, while this paper focused mainly on generalizability.

The driving model by [15] did not include the perception model's decoding layers in its architecture, which seems to be an overall more efficient approach. Because the U-Net architecture used in our paper had connections between each encoder-decoder layer, information could have been lost by not including the decoding layers. In future work, a model without connections between the encoder-decoder layers could be explored to take advantage of [15]'s approach.

5.2 Driving models

We find that models with a learned understanding of both the semantics and/or geometry of the scene are able to navigate never-before-seen environments and weather. Our real-time experiment shows that these driving models often perform better than learning from raw image inputs directly, with models utilizing both semantics and geometry performing best overall.

Variance. It is important to note that we observed a high variance when training and evaluating our models. Two models trained from the exact same setup could perform significantly different, despite having the exact same training data. We suspect that this is the same variance problem as [7] experienced. The variance was handled by training and testing the models several times to make sure

the results were representative. Still, conclusions based on the results in Experiment 2 must be drawn carefully. A more robust approach could be to train multiple models with the same parameters, and averaging their results.

6 Conclusion

Splitting end-to-end models for autonomous vehicles into separate models for perception and driving policy is shown to give good results in simulated environments. Perception models trained from public datasets such as *Mapillary Vistas* can be used to reduce the amount of driving data needed when training an end-to-end driving policy network. This approach opens up for training the driving policy in a simulated environment, while still getting good performance in real-world environments.

Future work should explore how these results transfers into the real world. Evaluating the performance of a model trained solely in simulation directly in a real-world environment will be an important next step as a means of testing the validity of these results.

References

1. Alhashim, I., Wonka, P.: High quality monocular depth estimation via transfer learning. arXiv:1812.11941 [cs] (Mar 2019), arXiv: 1812.11941
2. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. arXiv:1511.00561 [cs] (Oct 2016), arXiv: 1511.00561
3. Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., Zieba, K.: End to End Learning for Self-Driving Cars (2016)
4. Bousmalis, K., Irpan, A., Wohlhart, P., Bai, Y., Kelcey, M., Kalakrishnan, M., Downs, L., Ibarz, J., Pastor, P., Konolige, K., et al.: Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). p. 4243–4250 (May 2018). https://doi.org/10.1109/ICRA.2018.8460875
5. Cao, Y., Zhao, T., Xian, K., Shen, C., Cao, Z., Xu, S.: Monocular depth estimation with augmented ordinal depth relationships. arXiv:1806.00585 [cs] (Jul 2019), arXiv: 1806.00585
6. Codevilla, F., Müller, M., López, A., Koltun, V., Dosovitskiy, A.: End-to-end Driving via Conditional Imitation Learning. arXiv:1710.02410 [cs] (Oct 2017), arXiv: 1710.02410
7. Codevilla, F., Santana, E., López, A.M., Gaidon, A.: Exploring the Limitations of Behavior Cloning for Autonomous Driving. arXiv:1904.08980 [cs] (Apr 2019), arXiv: 1904.08980
8. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. arXiv:1604.01685 [cs] (Apr 2016), arXiv: 1604.01685
9. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Proceedings of the 1st Annual Conference on Robot Learning. pp. 1–16 (2017)

10. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. International Journal of Robotics Research (IJRR) (2013)
11. Godard, C., Mac Aodha, O., Firman, M., Brostow, G.: Digging into self-supervised monocular depth estimation. arXiv:1806.01260 [cs, stat] (Aug 2019), arXiv: 1806.01260
12. Gualtieri, M., Pas, A.t., Saenko, K., Platt, R.: High precision grasp pose detection in dense clutter. arXiv:1603.01564 [cs] (Jun 2017), arXiv: 1603.01564
13. Gupta, D.: Image segmentation keras : Implementation of segnet, fcn, unet, pspnet and other models in keras. (2020), `https://github.com/divamgupta/image-segmentation-keras`
14. Haavaldsen, H., Aasboe, M., Lindseth, F.: Autonomous Vehicle Control: End-to-end Learning in Simulated Urban Environments. arXiv:1905.06712 [cs] (May 2019), arXiv: 1905.06712
15. Hawke, J., Shen, R., Gurau, C., Sharma, S., Reda, D., Nikolov, N., Mazur, P., Micklethwaite, S., Griffiths, N., Shah, A., Kendall, A.: Urban Driving with Conditional Imitation Learning. arXiv:1912.00177 [cs] (Dec 2019), arXiv: 1912.00177
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv:1512.03385 [cs] (Dec 2015), arXiv: 1512.03385
17. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861 [cs] (Apr 2017), arXiv: 1704.04861
18. Johnson-Roberson, M., Barto, C., Mehta, R., Sridhar, S.N., Rosaen, K., Vasudevan, R.: Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? arXiv:1610.01983 [cs] (Feb 2017), arXiv: 1610.01983
19. Kendall, A., Gal, Y., Cipolla, R.: Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. arXiv:1705.07115 [cs] (Apr 2018), arXiv: 1705.07115
20. Müller, M., Dosovitskiy, A., Ghanem, B., Koltun, V.: Driving Policy Transfer via Modularity and Abstraction. arXiv:1804.09364 [cs] (Dec 2018), arXiv: 1804.09364
21. Neuhold, G., Ollmann, T., Rota Bulò, S., Kontschieder, P.: The mapillary vistas dataset for semantic understanding of street scenes. In: International Conference on Computer Vision (ICCV) (2017), `https://www.mapillary.com/dataset/vistas`
22. Pomerleau, D.A.: Advances in Neural Information Processing Systems 1, p. 305–313. Morgan Kaufmann Publishers Inc. (1989), `http://dl.acm.org/citation.cfm?id=89851.89891`
23. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. arXiv:1505.04597 [cs] (May 2015), arXiv: 1505.04597
24. Standley, T., Zamir, A.R., Chen, D., Guibas, L., Malik, J., Savarese, S.: Which tasks should be learned together in multi-task learning? arXiv:1905.07553 [cs] (May 2019)
25. Viereck, U., Pas, A.t., Saenko, K., Platt, R.: Learning a visuomotor controller for real world robotic grasping using simulated depth images. arXiv:1706.04652 [cs] (Nov 2017), arXiv: 1706.04652
26. Xiao, Y., Codevilla, F., Gurram, A., Urfalioglu, O., López, A.M.: Multimodal End-to-End Autonomous Driving. arXiv:1906.03199 [cs] (Jun 2019), arXiv: 1906.03199
27. Yurtsever, E., Lambert, J., Carballo, A., Takeda, K.: A Survey of Autonomous Driving: Common Practices and Emerging Technologies. arXiv:1906.05113 [cs, eess] (Jun 2019), arXiv: 1906.05113