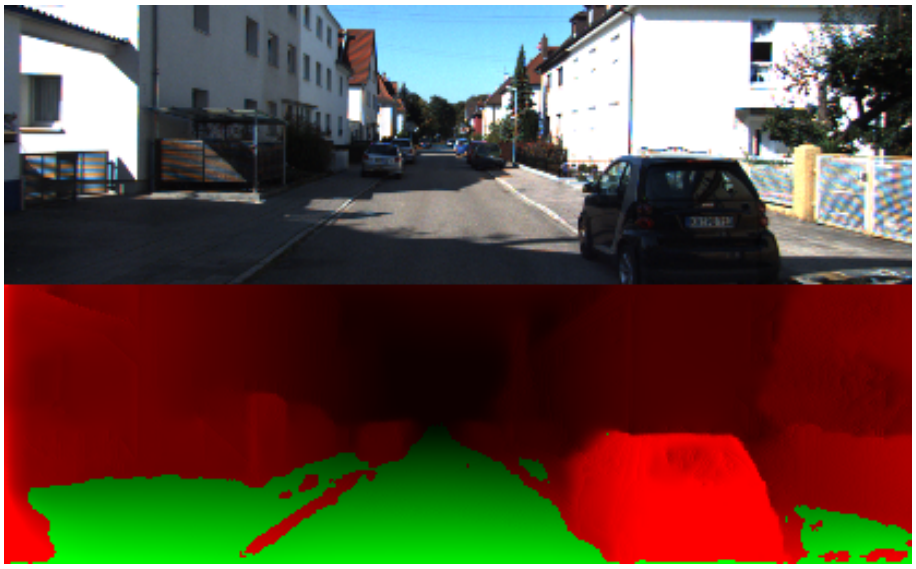
Joint Self-Supervised Depth and Obstacle Estimation from Monocular Images



by Raphael Pisoni

September 2019

Abstract

Until recently monocular depth estimation has been dominated by supervised models but newer approaches have shown that self-supervised monocular depth estimation can achieve similar and in some cases even better results while at the same time using only cheap monocular or stereo imagery. Obstacle detection and segmentation on the other hand is currently done almost exclusively by supervised models using manually generated ground truth data because unsupervised approaches do not yet provide the needed quality. Using previous best practices and a set of new methods with our proposed model we significantly improve the state of the art in self-supervised monocular depth estimation and show that these results can be used to teach the same model to learn obstacle segmentation. We show the effectiveness of a modified structural similarity loss, introduce a system that leverages inclination and roughness of the predicted depth to learn obstacle segmentation from monocular images and demonstrate the quality of our results on the KITTI benchmark as well as through visually compelling obstacle maps generated by our model.

Contents

List of Figures

List of Tables

1 Introduction

1.1 A short History of Artificial Neural Networks

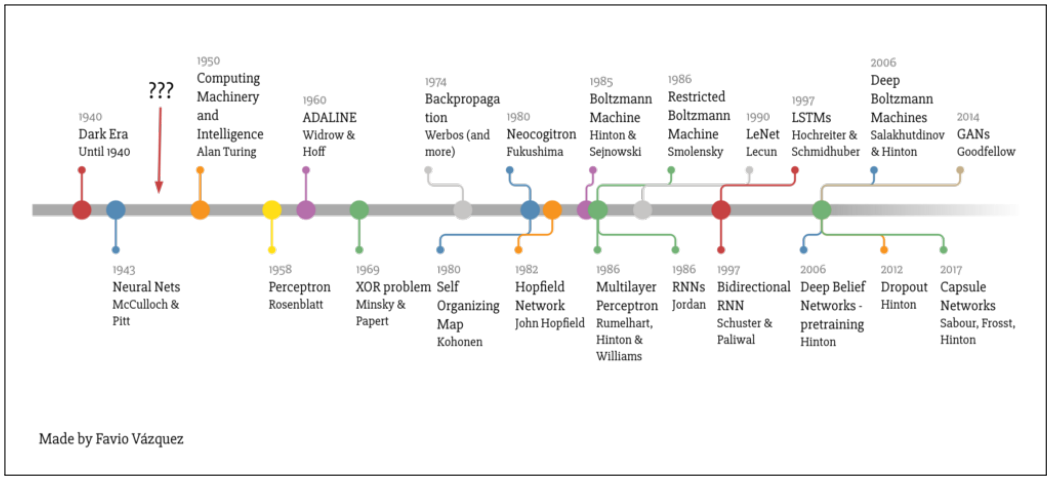


Figure 1: Time line of Neural Network Development Image source: Vázquez (2018)

Humans have long sought to reproduce their own intelligence. Starting in greek mythology with Thalos who was a mythical automaton with artificial intelligence but most prominently in the 19th and 20th century (e.g. Ada Lovelace, 1842) when inventors began to envision computers and thinking machines. In the early days of computers they were soon programmed to perform tasks that are hard for humans but easy for them, like multiplying two numbers.

Soon however it became clear that in contrast many things that feel natural and intuitive to humans like recognizing faces, were very difficult for them. To achieve more human-like behaviour scientist investigated the way the human brain works in order to build or simulate comparable machines (Figure 1). In 1943 Warren McCulloch and Walter Pitts tried to simulate intelligent behaviour through simple connected circuits which led Donald Hebb to his theory of connections getting strengthened through learning and continued use. Later McCulloch and Pitts tried to explain complex processes in the brain through a model of a neuron that takes a number of inputs, computes a weighted sum of them and outputs $zero$ if the result is below a threshold and one otherwise (Figure 2). This was the basis for the 'perceptron' introduced by Frank Rosenblatt in 1957, that introduced an additive bias to the neurons and arranged them in interconnected layers. They were used in the Mark I Perceptron machine that was designed for image recognition (Jiaconda, 2019).

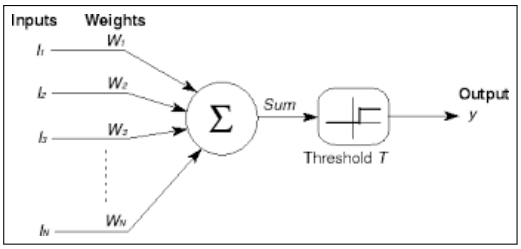


Figure 2: McCulloch-Pitts Neuron. Image source: Kawaguchi (2019)

The problem of these neurons however was that they were only able to learn linearly separable problems and e.g. failed at learning the simple but essential XOR-function (Figure 3). Furthermore the algorithms used at the time were unable to learn networks with more than a single layer that have no connection to the outside of the network (i.e. hidden layer) and were prohibitively slow to simulate on the hardware used at the time.

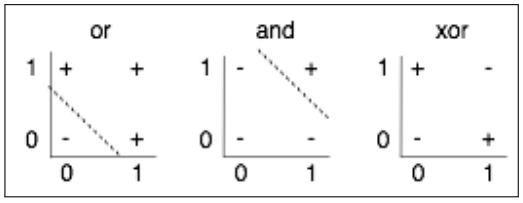


Figure 3: While some functions like AND and OR are linearly separable, for example XOR is not. Image source: Poole and Mackworth (2017)

Regardless these linear neurons were used for their first real world applications in systems designed to reduce noise in phone lines. At the idea of artificial neurons a huge hype began to rise and the true capabilities of these simple neurons were often tremendously over-interpreted or overstated and problems of the systems were ignored and overlooked. Finally the hype ended with a book named "Perceptrons" by Marvin Minsky and Seymour Papert in 1969 that clearly showed that the previous approaches could not be translated to multi-layered networks and that a large or infinite number of iterations would be needed to make perceptrons learn effectively. The publication of this and other problems erroneously led to the conclusion in the scientific community that all neural networks were unable to learn complex functions and were therefore not worth investigating any further. As a consequence almost all research and funding in the area of neural networks dried up which led to an era that today is commonly referred to as the 'AI Winter'.

Only in 1982 with the publication of a new kind of network that later came to be known as 'Hopfield Net' published by John Hopfield neural nets started to recover from their crisis (Strachnyi, 2019). They are energy based models that assign a lower scalar energy to observed variable configurations than to unobserved ones. They have the capability to learn through a simple yet powerful algorithm and are guaranteed to settle to a minimum. While they were of course not ideal they gave rise to some new interest in the concept of neural networks and allowed the field to slowly recover. Geoffrey E. Hinton and Terrence J. Sejnowski refined this new kind of network and co-invented the Boltzmann machine and its practical implementation, the Restricted Boltzmann machine that allowed for efficient training of multi-layered networks.

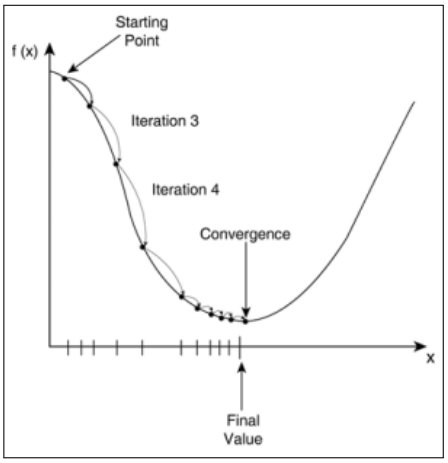


Figure 4: Backpropagation and Gradient Descent allow to find the minimum of a loss function. Image source: Champandard (2004)

Another major discovery was the Backpropagation algorithm whose utility for neural networks was already suggested in the 60's by Paul Werbos but was only rediscovered and applied practically in 1985 by Hinton, Rumelhart, and Williams. Together with their imple-

mentation they published a text that specifically addressed the problems and drawbacks laid out by Minsky and Papert in 1969 in the paper that caused the 'AI Winter' and were thus able to convince many critics that still did not believe in the usefulness of neural networks (Jiaconda, 2019). Backpropagation together with Gradient Descent are still the main workhorses of modern neural networks (Figure 4). They allow to compute the gradient of the cost-function with respect to all weights and biases in a networks, and to subsequently move them towards their optimum. While this is no guarantee to find the global optimum of the cost function, with the right tuning it can deliver impressive results in practice. Since the early 90's neural networks again capture the imagination of many and while now they finally deliver on some of their original promises the expectations again are growing much faster than the actual technology ever could.

1.2 Introduction to Modern Neural Networks

Originally artificial neural networks were meant to approximate the functioning of the human brain since that is the most intelligent system known today. However while still being inspired by neuroscience, today's research is becoming more and more independent of it. The neuron used in modern neural networks is still composed of the kind of neurons developed by Frank Rosenblatt in 1958 (Figure 5). It takes a weighted sum of all of it's inputs, and adds a bias to it. During the learning process both weights and biases are adjusted to move the resulting value towards an output that minimizes the target-function (i.e. loss-function). In its purest form a neural network is arranged in layers where the first is called the input-layer, the last is called the output-layer and all layers in between are hidden layers that don't directly interact with input or output values. This structure allows the data to be gradually transformed into an internal representation such that each layer gets a more elaborate set of features than the one before it.

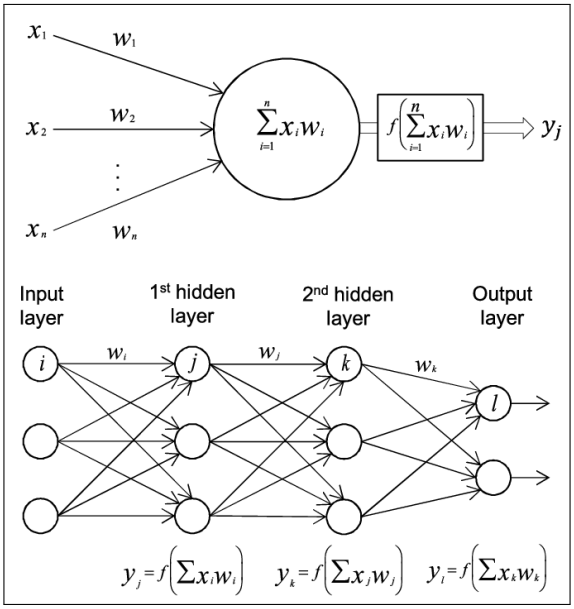


Figure 5: (top) The modern artificial neuron (bias omitted). (bottom) Simple fully connected network architecture. Image source: Vieira et al. (2017)

While historical neural networks were not able to generate outputs that were not linearly separable, in modern networks this is done through a nonlinear activation-function (Figure 6). It receives the output of the neuron as input, manipulates it and passes it on to the next layer or the output. While many modern activation functions exist and have their different advantages and drawbacks, all of them have the same basic goal of letting the network approximate nonlinear functions.

Another addition to the historical network structure has been essential to make neural nets more stable and more easily trainable. Since all weights and biases are manipulated in parallel this can lead to exploding or vanishing gradients that lead to network outputs over- or under-flowing or them to get so close to zero that they are not able to react to losses anymore since their gradient is zero. The most common proposed solution to this problem is called 'batch-normalization' and consists of an operation that normalizes all neuron outputs by a mean and scaling factor. This allows the network to reduce the co-variance shift, reduces over-fitting through the noise introduced through the normalization and finally allows to use higher learning rates by avoiding very high or very low activation outputs (Doukkali, 2018). For the same reasons it is important to normalize all network inputs. Usually they are scaled to a range between zero and one or from minus one to one (Matthew Stewart, 2019).

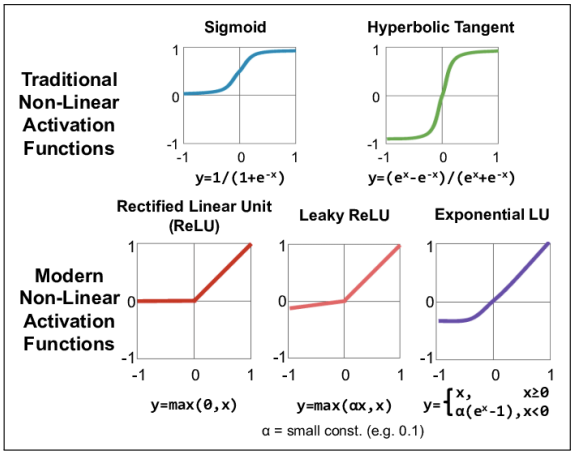


Figure 6: The most common activation functions. Image source: Sze et al. (2017)

While Backpropagation lets us find the gradients for all of our variables relatively easily the question of how to best use these gradients remains an unsolved problem. The algorithms that are used to manipulate the weights and biases are generally called optimizers and over the recent years a great number of them have been proposed. In general it is their job to move along the axes of an n-dimensional surface in order to find the global minimum or an acceptable local minimum (Figure 8). The main problems that optimizers have to deal with are local minima, saddle points, poor conditioning of the target function as well as the choice of the ideal learning-rate which is essentially the step size for the movement along the axes.

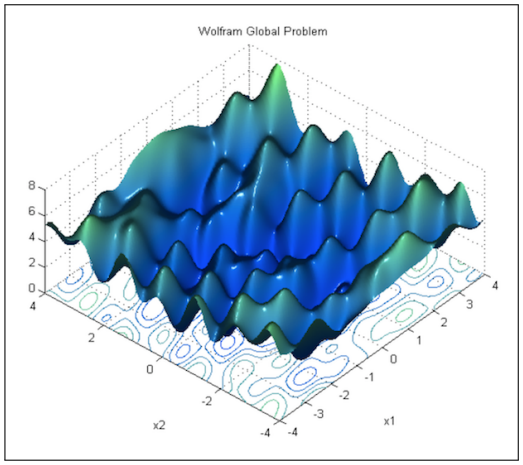


Figure 7: Example of a non-convex error surface with two parameters. In neural networks we have to efficiently minimize error-surfaces composed of millions of parameters but the algorithms stay the same. Image source: Bengio (2019)

In an infinite high-dimensional search-space local minima can be almost impossible to distinguish from a global minimum (Figure 7). What can make it easier is taking a number of measurements in the neighbourhood of our point of interest and using the mean gradient to move further. In today's neural networks this is achieved through the so called batch-size that denominates the number of random samples that is taken in a particular point. While the batch size in case of a smooth error function

could be seen as computation overhead, a big enough sample size can achieve steady progress even under very rough conditions since this avoids converging to sharp local minima (Keskar et al., 2016).

Recent studies have shown that in high dimensional space saddle points are much more common than local minima (Matthew Stewart, 2019). They can create areas of gradients very close to zero that prevent the network from making further learning progress. The fact that error surfaces have a great number of saddle points and very flat areas is a result of them being non-convex and usually conditioned poorly. This can be shown by looking at the Hessian, the second order partial derivatives of a function, that describes the local curvature of a function (Matthew Stewart, 2019). If the Hessian is poorly conditioned a small change in the input will have a large effect on the output which is unfortunate because it leads to a slow and unstable optimization in despite the presence of strong gradients because of the resulting oscilations (Matthew Stewart, 2019).

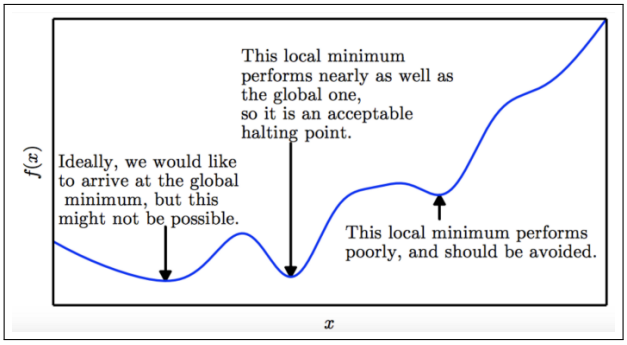


Figure 8: Minima of an error surface. Image source: Huang (2018)

In many optimizers these problems are tackled through so called momentum which is inspired by linear momentum from physics (Figure 9). When applying the current gradients an exponentially decaying fraction of the past gradients is added. This allows the optimizer to move past ill-conditioned or flat regions faster because of the 'velocity' it retains from previous steps. This also causes opposing gradients to cancel each other out such that we come closer to the fastest path of descent. Mathematically momentum can be described as follows:

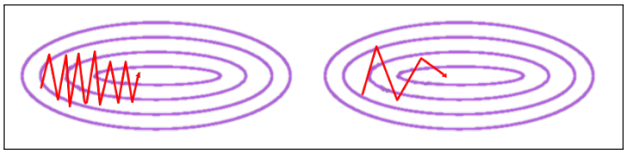


Figure 9: Momentum can reduce oscillations in a poorly conditioned system (left) and lead to a faster convergence (right). Image source: Genin (2018)

$$v_n = \eta * v_{n-1} - \alpha \nabla L \tag{1}$$

In the first part of the equation η is the so called friction coefficient that exponentially decreases the influence of past gradients (v_{n-1}) which is commonly defined as 0.9. The second part of the equation consists of the loss-function gradient multiplied by the learning rate α.

This learning rate is another important parameter for a successful model. It defines the step size to take in the direction of the computed gradient during the training process. While a large learning rate can accelerate training it also increases oscillations and might cause the model to step over the minima that are worth investigating. A learning rate that is to small however slows the training down unnecessarily and might let the model get suck in local minima (Figure 10). Therefor it is beneficial

to tune the learning rate in order to allow fast progress in the beginning of the training and accurate optimization once the model is close to the global minimum. Today the most popular way to achieve this is the ADAM optimizer that automates the tuning of the learning rate by estimating an optimal learning rate for each parameter at the same time applying an exponentially decaying average of the past gradients similar to momentum (Matthew Stewart, 2019).

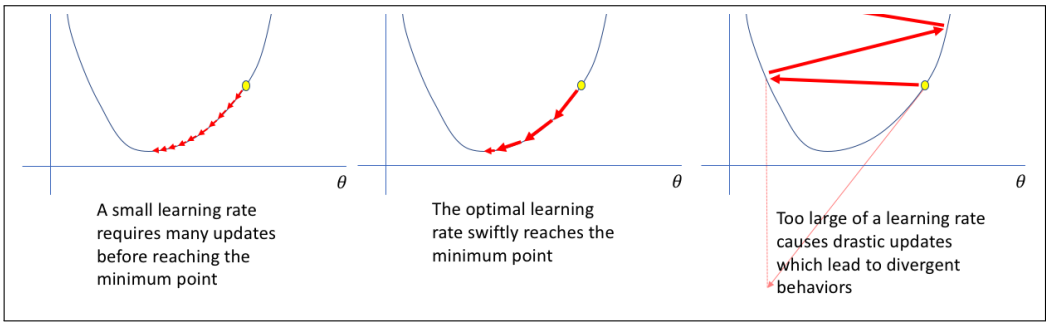


Figure 10: While a good learning rate can deliver results quickly, the wrong one can make training almost impossible. Image source: Jordan (2018)

An important task that has to be tackled before the training of neural networks even starts and that has been extensively optimized over the recent years consists of the parameter initialization at the start of training. This can have a huge effect on the stability and outcome of training and greatly speed up gradient descent. A number of heuristics for successful model initialization exist (see Xavier- or He Normal Initialization), but in general they consist of choosing small random weights with zero mean and a standard deviation that depending on the specific initialization can vary with model depth. Biases are often initialized to zero or in case of the ReLU activation being used with a small positive value (e.g. 0.1) in order to avoid saturation of the activation function.

Once the training is ready to start a way to judge the output of the network has to be found. While in supervised training we usually have the desired ground-truth and can find a measure of perceived difference from this desired value. In unsupervised training however no ground truth can be used, so other measures that can be derived from the input, auxiliary data or geometrical considerations must be used to judge the quality of the network. While it is often considered to be good practice to keep the loss positive in theory it is allowed to assume any real value.

1.3 Modern Layer- and Network Architectures

Of the seemingly infinite amount of layer and network architectures of course only a few have been used in our application. The particularly important ones will be explained here. Until now only fully connected networks have been mentioned however for images with a sensible size connecting every pixel to every neuron in a hidden layer would require a huge amount of memory and compute power. Therefor the convolutional neural network has been introduced that

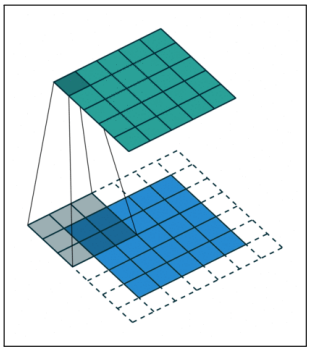


Figure 11: A convolutional layer shifts a kernel of neurons over an image and processes it patch-wise. Image source: Theano Development Team (2016)

instead of connecting all pixels to all neurons shifts the same kernel of neurons over the image for every pixel and arranges the results (Figure 11). Arranging a number of these convolutional layers allows the network to efficiently decompose an image into a more concise internal representation while reducing the number of features. This internal representation that then be used to classify the image through a fully connected layer or to recreate a new image from it through deconvolution. The resulting network stack first contracts and then re-expands its number of feature vectors and is commonly referred to as an encoder-decoder architecture (Figure 12). This architecture helps to extract the important information while redundancies or unnecessary details are reduced.

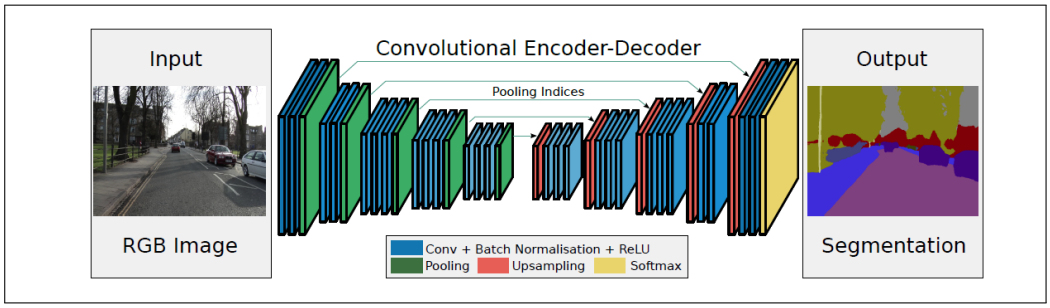


Figure 12: Encoder-decoder networks can be used to extract semantic information from images. Image source: Badrinarayanan et al. (2015)

As networks get deeper in order to extract more complex information from the input information it gets increasingly difficult for the top layers to learn because their gradients get diluted through the whole network stack. This can create the paradoxical situation that deeper networks while theoretically having more learning capacity, actually achieve worse results than their shallow counterparts. To tackle this effect residual connections have been introduced (Figure 13). This can create the paradoxical situation that deeper networks while theoretically having more learning capacity, actually achieve worse results than their shallow counterparts. To tackle this effect residual connections have been introduced. They consist of connections that skip whole blocks of layers and get then recombined with the network stack through an addition operation. Through this they create so called gradient-highways that allow networks with thousands of layers to still make reasonable progress. While this kind of depth is usually neither efficient nor necessary these residual connections have also helped more shallow networks to make faster and more reliable progress.

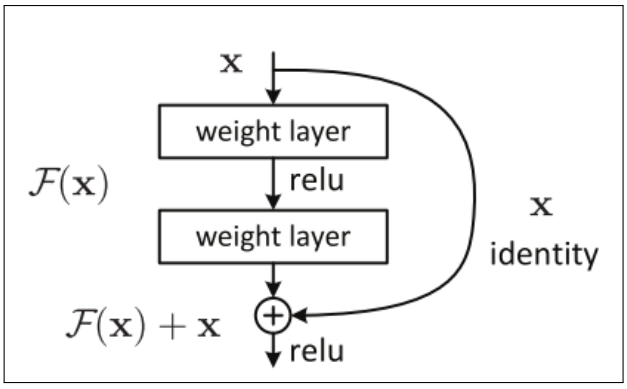


Figure 13: Example of a residual connection. Image source: He et al. (2015)

This concept of connections that skip whole blocks of layers to create a shortcut for gradients has inspired the research community to drive this idea to it's extreme in encoder-decoder networks. Previously they often had the problem of not being able to recreate the original level of detail in their output image since much of the fine-grained information gets lost during the process of down-sampling. To counter this the so called U-Net architecture has been designed (Figure 14). It has skip-connections that bridge large parts of the network stack from the encoder to the decoder and allow the network to receive fine grained information about the original image. Numerous variants of this architecture have been published since its original implementation and while today it is mainly used for semantic segmentation as will be shown later it is also highly capable of producing excellent results in depth estimation.

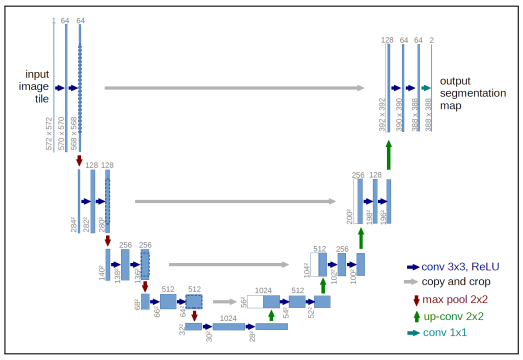


Figure 14: The original U-Net Architecture. Image source: Ronneberger et al. (2015b)

1.4 Current and Future Applications for Neural Networks

There is an endless number of tasks that neural networks are used for today, however they can be divided into three general topics. Classification, regression and data processing. It is important to note however that the boundaries between these areas are fluid since all of them rely on neural networks as general function approximators.

Classification is the task of assigning data to one or more predetermined classes. It is useful across many domains including pattern recognition, sequence classification or sequential decision making (Neural network, 2019). As can be seen in Figure 15 most data can not be perfectly fit with the limited information available. So instead of directly predicting a class usually a probability for each class is estimated through a softmax-activation in the final layer. The probabilities are then learned through the categorical cross entropy loss function. If a sufficient amount of data is available and overfitting is avoided (Figure 15) a good model can learn a function that is close to the original distribution that the training data was sampled from. This practice however shows that in this field classification is just a special case of regression of a probability distribution. Some applications of classification include object recognition, handwritten character recognition, tumor segmentation from X-ray or

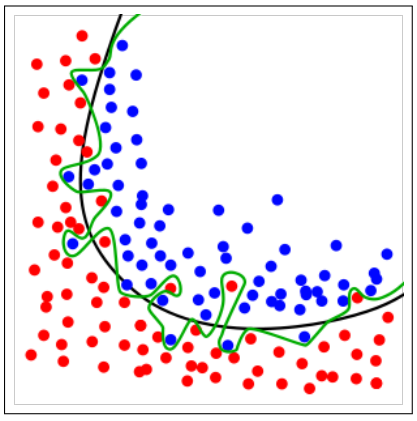


Figure 15: Often data can not be classified perfectly without overfitting. Image source: Overfitting (2019)

MRI images or recognition of recorded audio signals or speech. While most of these tasks are currently done in a supervised fashion the current trend strongly points towards unsupervised or self-supervised training methods and models. As an example in this paper we will explore self-supervised obstacle segmentation with stereo images as the only source of supervision whereby we classify each pixel of an image as either drivable surface or part of an obstacle.

Regression analysis statistically is the process of estimating the relationship between variables (Regression analysis, 2019). This is a natural application for neural networks and can be used for anything from house price estimation to weather forecasting or stock price prediction (Figure 16). An application with particular importance is probability prediction throughout a variety of topics. In particular natural language processing and understanding which relies on the estimation of conditional probabilities between words has made huge progress since the introduction of neural networks. For future applications doubtlessly graph neural networks will play a major role. They are currently being heavily researched and are concerned with the estimation of good value heuristics for traversing graphs. In particular for huge complex graphs where the generation of effective heuristics becomes increasingly difficult and standard graph searching algorithms perform poorly these nets could lead to big performance gains.

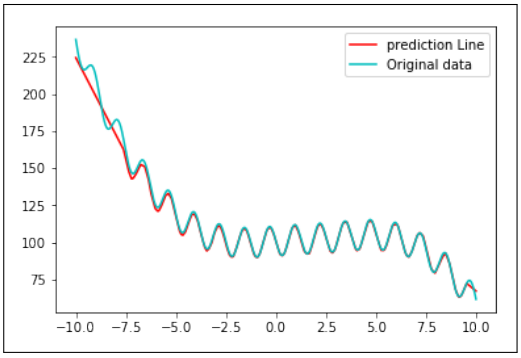


Figure 16: Complex function approximated by a neural network. Image source: fast.ai (2018)

Another research area that is evolving very quickly right now are models for data processing. As mentioned previously with the right training neural networks can separate important information from redundant or noisy data. This property makes them interesting for many tasks including data filtering, dimensionality reduction, clustering, signal separation and data compression. An example for this is multi-speaker speech recognition that allows for voice commands in crowded or noisy rooms or visualization of high-dimensional data using the popular t-Distributed Stochastic Neighbor Embedding (i.e. t-SNE) method (Figure 17).

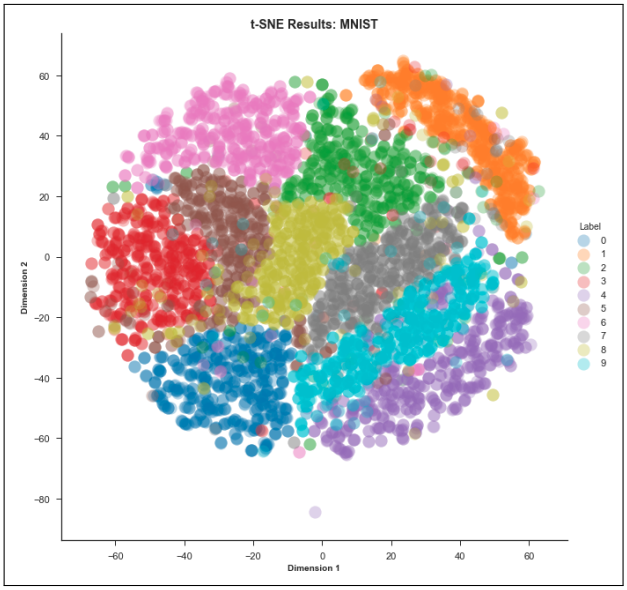


Figure 17: t-SNE allows to reduce the dimensionality of data or to display high-dimensional data in two dimensions. Image source: Violante (2018)

One impressive project that points to future uses of neural networks is for example through-wall human pose estimation by interpreting the perturbations to the WiFi signals by Zhao et al. (2018) (Figure 18). This work shows how learning systems can make information and data accessible that is hard for humans to interpret in its natural form and points to future uses in sensor-fusion, computer vision or medical imaging. For example applying similar technologies to microscopy, X-ray, or MRI technologies might lead to cheaper implementations or better image quality. A further exciting step in the evolution of neural networks for reinforcement learning is a model called AlphaStar that was built by Google and was able to defeat some of the best pro-players at Starcraft II. It was able to achieve its superhuman skill-level through a mix of supervised learning and self-play and demonstrates that neural network can learn to solve difficult problems in complex environments. A final example for research that points to future possibilities and applications is another model by Google called BERT (Devlin et al., 2018). It demonstrates how unsupervised learning and transfer learning can create a big shift in natural language processing and data processing in general. It leveraged large amounts of unlabeled data to achieve a major improvement in a number of natural language tasks while always relying on the same pretrained model. All of these examples show that neural networks are indispensable for the future of not only computer science but also many other areas of research and human life.

Figure 18: RF-Pose is able to do human pose estimation through walls from perturbations in WiFi-signals. Image source: Zhao et al. (2018)

1.5 Limitations of Neural Networks

Neural networks have long been known to be universal function approximators. This fact however has sparked great misunderstandings in the past and present that some people in the general population or even the media fall for, and that are sometimes used to sell false promises or flawed products. Neural networks are not magic and can not perform impossible tasks. While this seems like common sense to most people many false claims are either based on the lack of detailed knowledge of the technology fundamentals or are based on a misunderstanding of the 'Universal Approximation Theorem'. This theorem was first proven in 1989 for sigmoid activation functions by George Cybenko and states that a neural network with a single hidden layer and a finite number of neurons can approximate any function in R^n to an arbitrary degree (Universal approximation theorem, 2019). While this theorem by itself makes neural networks seem incredibly powerful, it does not make any claims about the learnability of this function. So while in theory a single layer neural network with a finite number of neurons can approximate any real-valued continuous function to an arbitrary degree it is not clear how we can generate said network. So while today's neural networks can be very powerful without a doubt, through the limitations of our current learning algorithms they are still far from the potential power this theorem seems to promise.

Two further limitations of neural networks that are frequently cited are the lack of interpretability and their need for a strong signal. Neural nets are often described as black box models, meaning that while their predictions might be good, the process of how those predictions are made is opaque and can therefor not be easily explained or understood. This attribute is fine for some areas where it is not necessary to have an explanation for the results of an algorithm, but some other areas where e.g. safety and security is of central importance prefer to avoid black box approaches. Thus describing neural networks as a black box is often done to disparage them for those areas. What has to be noted however is that especially simpler for neural networks there are methods to partially open the black box. The most common ones are decomposition, which approximates the weights in a neural network using a set of IF statements, and distilling the neural network into a soft decision tree. While these methods still have drawbacks and are difficult to use for complex nets they represent a first step in the young field of neural network interpretability research.

Regarding the need of neural networks for a strong signal Ockham's razor can be a very helpful heuristic. For this context essentially it states that if two models perform a regression task equally well, the one with fewer assumptions i.e. parameters should be preferred. This becomes not only relevant during model selection where complex models like neural networks should not be used for simple regression task that can easily approximated through other means (Figure 19) but also during the training of neural networks. While longer training often leads to better results on the training set, it is common practice to continually test the models performance on a test-set that it is not trained on. This allows to train the model while it keeps generalizing better to the data and to stop in time to avoid severe overfitting. This trade off is also known as the bias/variance dilemma. Bias is the inability to model the underlying features between features. Variance is the excessive modeling of random noise in the data. During training of a complex model the bias decreases while the variance constantly increases. Finding the right balance between the two is one of the main challenges during the training of neural networks.

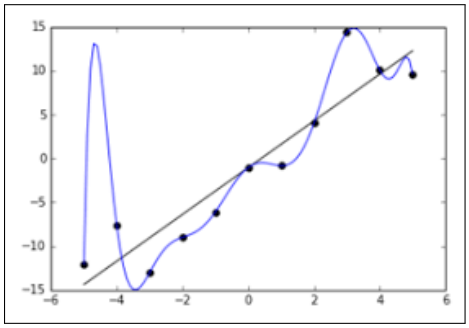


Figure 19: A model that is too complex tends to overfit more than a simple model. Image source: Overfitting (2019)

Another apparent drawback of neural networks is their susceptibility to adversarial examples. They consist of carefully tuned noise that, if added to an image, can have a huge effect on the predictive performance of a neural net. The most famous example can be seen in Figure 20 that shows the image of a panda that through the addition of seemingly meaningless noise, that is almost imperceptible to the human eye, gets misclassified as a gibbon with high probability. This however highlights a fact that, while seeming

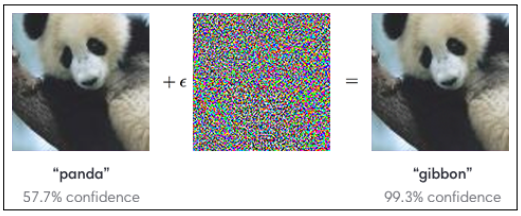


Figure 20: Apparently neural networks can be fooled by carefully tuned random noise. Image source: Goodfellow et al. (2014)

obvious, is often forgotten by many. Neural do not learn what we would like them to learn but what we train them to. This might be best explained through an example from the early days of neural network that was popularized through a number of sources but who's exact origin seems unclear (gwern, 2019). Apparently in the 60's the US military wanted to use early neural networks to distinguish pictures with tanks on them from pictures without tanks on them. They created a data-set and trained a model that seemed to work surprisingly well only to find out later that all of the pictures with tanks on them had been taken on a sunny day while all of the pictures without tanks on them had been taken on a cloudy day. So effectively all the network did was to measure the total brightness of the images, something the human eye is relatively insensitive for.

Similarly for adversarial examples a recent study confirmed what had been long suspected: Instead of bugs they are highly predictive but non-robust features in standard ML data sets that humans are insensitive to (Ilyas et al., 2019). This means that the problem of these examples in reality does not necessarily lie in the model that gets trained but in the data it gets trained on. Ilyas et al. (2019) also show ways to harden neural networks against those attacks and show that these non-robust features are so predictive that even a model trained entirely on them can perform well on standard classification (Figure 21). This shows that we should not be surprised that classifiers exploit highly predictive features that happen to be non-robust under a human-selected notion of similarity, given such features exist in real-world data sets (Ilyas et al., 2019).

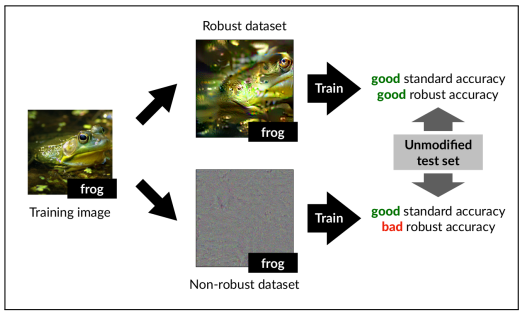


Figure 21: Not only can robust training be done but even non-robust features can be used to achieve good accuracy on the original data set. Image source: Ilyas et al. (2019)

1.6 Introduction to Depth Estimation

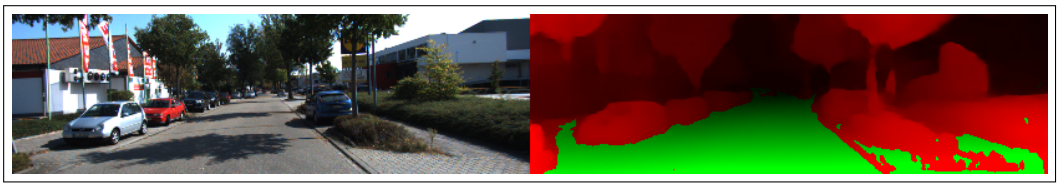


Figure 22: Original Image and Product of Estimated Disparity- and Obstacle-map.

Monocular depth estimation is a nontrivial problem in computer vision due to the fact that any image can map to an infinite number of possible depth configurations. While originally it was tackled using mostly geometric approaches, since the advent of convolutional neural networks significant progress has been achieved in this area. Until recently this task was treated mostly as a supervised problem and networks were trained on large datasets of images and their corresponding laser-scans. Godard et al. (2017) and Garg et al. (2016) introduced self-supervised monocular depth estimation

from stereo images by enforcing consistency between the left and right image. This approach has a number of advantages including the vast availability of stereo footage and the much lower cost of producing it, as well as its high quality and low sensor-noise and error-rate compared to LIDAR scans. Today multiple ways to estimate depth using self-supervised approaches from stereo (Godard et al., 2017) or motion parallax (Zhou et al., 2017) have been described and have been shown to outperform traditional geometric and sometimes even supervised models (Godard et al., 2017). In fact we will show in this work (Figure 22) that self-supervised monocular depth estimation has gotten so good that the resulting estimations can be used to derive more complex measures. We propose to use an efficient but powerful model as well as some improvements in loss computation and training to significantly improve the state of the art in monocular depth estimation.

1.7 Introduction to Obstacle Detection

In addition to depth-estimation we use a novel algorithm that uses angle and roughness of the predicted depth to train the our model to perform self-supervised obstacle detection with interesting results. In the past obstacle detection has mostly been treated as a general object detection problem from camera images, LIDAR measurements or a combination of both. While there has been limited success for unsupervised approaches most of them were supervised and therefor required a large number of human-generated labels that the network could learn (Figure 23). This is very resource-intensive and depending on the quality of the labels can be very error prone. Object detection is usually not done through segmentation but through architectures that estimate the position and size of a bounding box for an object. This is done due to the fact that segmentation is relatively inefficient for a large number of object classes. Other approaches have managed to achieve good results through street segmentation or lane detection purely from images which reduces the number of classes but might miss some kinds of obstacles without using depth information. Mancini et al. (2018) tried to combine the depth estimation and obstacle detection tasks and were able to show that this can be beneficial to both tasks, however their architecture was resource intensive and used bounding boxes which are of limited utility in real world scenarios.

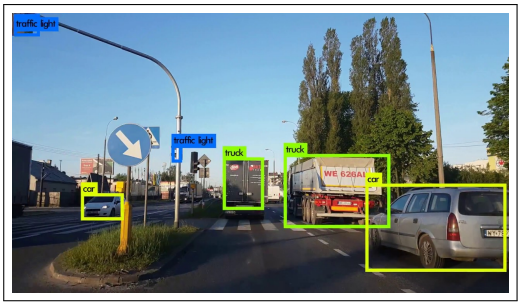


Figure 23: Obstacle detection treated as supervised object detection problem. Image source: Majek (2017)

2 Previous Work

As the notion of depth is a prerequisite for many tasks in computer vision and robotics there has been extensive research in the area of depth measurement and estimation. While currently the best direct results are achieved with modern laser scanners, their cost, weight and size is prohibitive for many areas where depth information could be of great use. In contrast cameras are passive sensors that are much smaller, cheaper and can be built to produce high quality images under most circumstances. With the rising success of learning based methods Eigen et al. (2014) have shown the feasibility of depth estimation as a supervised learning problem while others were able to refine this approach (Laina et al., 2016) to even surpass traditional methods in some cases (Yang et al., 2018a). To reduce the need for ground truth depth data, that is often hard to acquire under significantly varying real world conditions, weakly supervised approaches have tried to exploit additional information in the form of sparse supervision, unpaired depth supervision (Kundu et al., 2018), known object sizes (Wu et al., 2018) or other supervised appearance matching terms (Zbontar and LeCun, 2016), however the need for high quality depth data or other annotations still limits these approaches. Similarly synthetic data, while being a valuable alternative, is not trivial to generate in the required amounts and quality, especially including varying weather and lighting conditions.

2.1 Self-supervised Stereo Training

As an alternative to supervised training, self-supervised learning of depth as an intermediate step during the image reconstruction between two images has emerged (Garg et al., 2016). Following works used left-right consistency between two synchronized stereo pairs (Godard et al., 2017) (Figure 24) or similar techniques for monocular videos (Yang et al., 2017). While using stereo images at training time provides generally good results, they might not always be available. In contrast monocular videos are ubiquitous but many models using them encounter the problem of not handling movement well. While some approaches like ego-motion estimation and individual tracking of moving objects exist (Casser et al., 2018), to solve these problems they introduce considerable complexity and computational overhead at training time and during online learning and refinement.

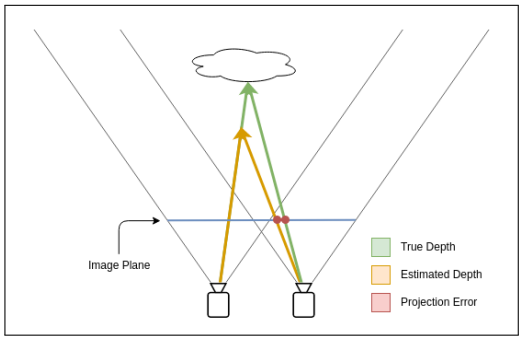


Figure 24: A misestimation in the depth of the left image can lead to discrepancies between the true and the projected right images.

2.2 Joint Depth Estimation and Obstacle Detection

Path-finding and obstacle detection are closely connected and are in many cases the main reason to use any kind of depth perception. As both play an important role in self-driving technology and robotics in general, a large amount of work has been devoted to these topics. However most authors approach them as problems for supervised learning and object segmentation from images (Badrinarayanan et al., 2016) or LIDAR (Mei et al., 2018). Another line of work has achieved remarkable results by using unsupervised features like roughness and angle of the detected surface normal from LIDAR-scans (Liu et al., 2017; Zhang et al., 2018). While during training our approach only relies on self-supervision from stereo imagery and does not use LIDAR or labeled segmentation data these are the concepts that come closest to our framework. During inference however we segment obstacles directly from monocular images using a convolutional neural network and do not use LIDAR measurements.

3 Methods

The main objective of our setup is the self-supervised learning of depth estimation and the segmentation of possible obstacles. The sole source of supervision for both tasks during training are pairs of stereo images with known intrinsic and extrinsic parameters. LIDAR and other ground truth data is solely used for evaluation purposes. In the following we will describe in detail the problem setup, the architectural details of our model as well as the objective functions and compare them to architectures proposed by other authors.

3.1 Problem Setup

The input during training are two rectified stereo RGB images $(I_1, I_2) \in R^{H \times W \times 3}$ as well as the extrinsic camera parameters $(C_1, C_2) \in R^{4 \times 4}$ for them. The model θ is a fully convolutional neural network that takes a single image I_i as input and predicts a depth $D_i \in \mathbb{R}^{H \times W}$ and a map of obstacles $O_i \in R^{H \times W}$. Similar to Zhou et al. (2017) and Casser et al. (2018) through the depth estimated by θ and a fully differentiable warping operation ϕ we can project the images I_1 and I_2 between the camera viewpoints defined by C_1 and C_2 such that $\hat{I}_{i \to j}$ denotes the j-th image constructed by warping the i-th image into the j-th position. The training signal can either be computed through photometric losses between the projected image $\hat{I}_{i \to j}$ and the original image I_j (Figure 24) such as the reconstruction loss $L_{rec} = |\hat{I}_{i \to j} - I_j|$, or through losses like our obstacle detection loss that are derived from the estimated depth.

3.2 Model Architecture

The basic model architecture we use can be seen in Figure 25 and is based on a U-Net architecture (Ronneberger et al., 2015a) that consists of encoder-decoder structures with skip-connections, and applies best practices introduced in previous works (Godard et al., 2018; Casser et al., 2018). As an encoder ResNet18 is used (He et al., 2016) and as a decoder we use an improved variation of the one used in (Godard et al., 2018). We use skip connections augmented with additional blocks consisting of two convolutional layers that are then re-concatenated with their input-tensor and output the estimated depth at four scales through a single convolutional layer with sigmoid activation. Then we apply a rescaling operation of the following form: $D = 1/(a*\sigma+b)$, where σ is the output from the sigmoid activation and a and b are chosen as 10 and 0.01 to constrain the output range of D. In all other layers we use RELU nonlinearities, and reflective padding instead of zero padding to reduce border artifacts. The obstacle segmentation branch consists of a padding layer and two convolutional layers and branches out before the last convolutional layer of the depth estimation network. In total the model has little over 14M parameters which allows for relatively efficient training and execution.

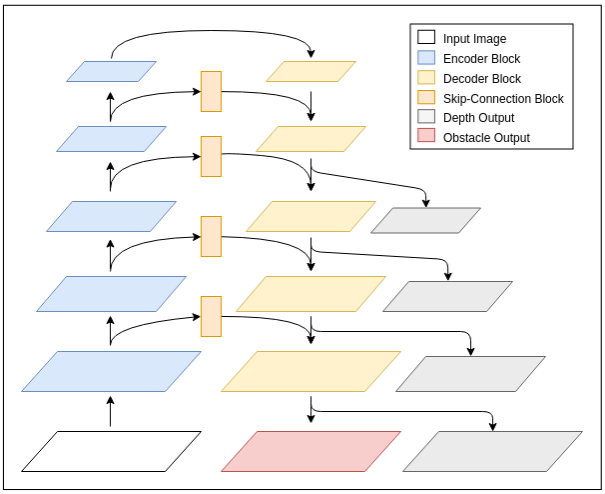


Figure 25: General Model Architecture

3.3 Objective Functions

For the depth estimation branch we use a reconstruction loss and a structural similarity loss. The reconstruction loss is computed as the sum of the absolute difference of the left and right images warped onto each other using the estimated depth:

$$L_{rec} = |\hat{I}_{1\to 2} - I_2| + |\hat{I}_{2\to 1} - I_1| \tag{2}$$

In addition we use an SSIM loss proposed by Wang et al. (2004), however instead of computing the structural similarity between the warped and the original image we compute the SSIM loss on the $atan2$ transformation of both images (Figure 26). This ensures high contrast even in under- or overexposed images and amplifies small details that are making it easier for the SSIM loss to punish even small differences. For both losses the error is only computed on areas that are not occluded

during the warping operation. As a third loss function for the depth estimation task we choose to use the extended scale invariant log RMSE introduced by Mancini et al. (2018)

$$L_{depth} = \frac{1}{n}\sum_i d_i^2 - \frac{1}{2n^2}\Big(\sum_i d_i\Big)^2 + \frac{1}{n}\sum_i \Big(\nabla_x D_i + \nabla_y D_i\Big) \cdot N_j \tag{3}$$

where $d_i = \log D_i - \log D_j$, D_i and D_j are the left and right predicted depth warped onto each other. $\nabla_x D_i$ and $\nabla_y D_i$ are the horizontal and vertical predicted depth gradients of D_i and N_j is the 3D surface normal of the projected depth D_j. While the first two terms correspond to the scale invariant log RMSE loss introduced in (Eigen et al., 2014), the third term was originally introduced to enforce the orthogonality between predicted gradients and a given ground truth normal, aiming to preserve geometric coherence by Mancini et al. (2018). We use this loss to verify the coherence between the left and right predicted depths.

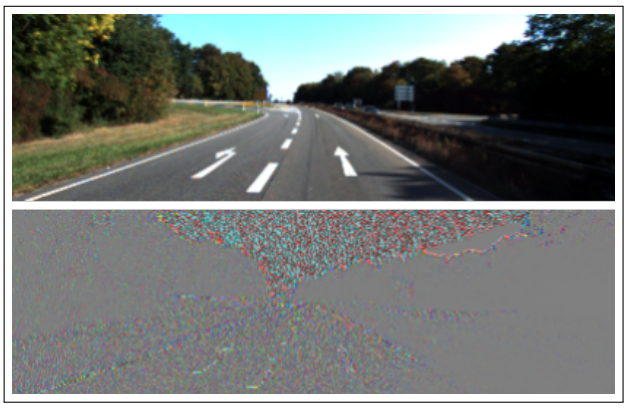


Figure 26: Image and its $atan2$ transformation.

In order to train the obstacle segmentation branch we generate a self-supervised obstacle map for every pixel p_k where $ob(p_k) = 0$ indicates drivable area and $ob(p_k) = 1$ indicates an obstacle. The values are derived from the roughness of the estimated depth $D(p_k)$ and the angle of the surface normal $\mathbf{N}(p_k)$.

If we denote the 8 connected neighboring pixels of p_k as $S(p_k)$, a pixel is marked as obstacle if one of the following conditions is true:

$$\max_{p \in S(p_k)} (|D(p_k) - S(p_k)|) > \theta_1 \ (D(p_k))^2 \tag{4}$$

$$|D(p_k) - \bar{D}(p_k)| > \theta_2 \ (D(p_k))^2 \tag{5}$$

$$\arcsin\left(\frac{|N^y(p_k)|}{\|\mathbf{N}(p_k)\|}\right) < \theta_3 \tag{6}$$

where θ_1 and θ_2 are hyperparameters set to 0.006 and 0.003, whereas θ_3 is the maximum deviation angle from the vertical axis that we set to 82°. $\bar{D}(p_k)$ is the mean of the depth of $S(p_k)$ points. $N^y(p_k)$ is the projection of the norm $\mathbf{N}(p_k)$ on the y-axis. The intermediate obstacle maps are then combined via logical inclusive-or and we remove all non-obstacle patches that are smaller than 5% of the total image area (Figure 27).

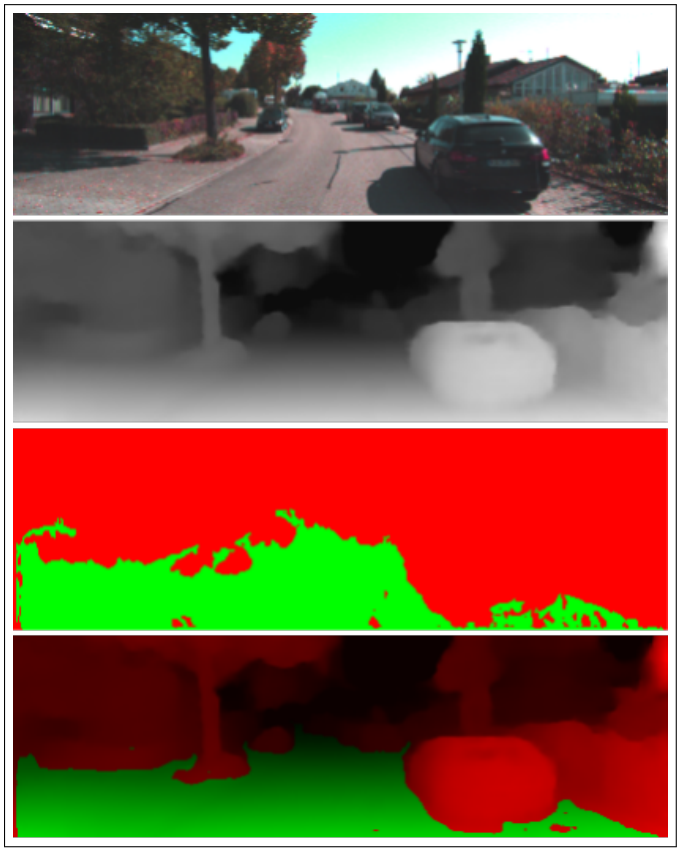


Figure 27: (from top to bottom) 1) Original Image, 2) Estimated Depth, 3) Computed Obstacle Map, 4) Product of Estimated Disparity and Estimated Obstacle Map.

Finally the obstacle-loss is computed using the cross-entropy between the resulting map and the output of the obstacle-branch of the network where the weights for drivable and obstacle are 1.0 and 1.4. To keep obstacles consistent between left and right image and to remove small artifacts we also apply the SSIM loss on both estimated obstacles and their warped counterparts.

The final loss is computed as a weighted sum off all the losses, applied to both stereo images:

$$L_{total} = \alpha\ L_{rec} + \beta\ L_{ssim} + \gamma\ L_{depth} + \delta\ L_{obst} + \epsilon\ L_{obst\_ssim} \tag{7}$$

where α, β, γ, δ and ϵ are hyperparameters that we decided to set to 0.85, 0.2, 0.001, 0.01 and 0.0035. While at first it might seem more efficient to simply compute the ground truth obstacle map at inference time using the formulae listed above, we found that adding two additional layers as an obstacle branch to our network is not only faster but also leads to generally better results for obstacle detection with less noise and better left-right consistency, but also lower errors in the depth estimation.

3.4 Training

We train our model on the images from the KITTI Raw dataset and exclude all static scenes as well as the test-split used by Eigen et al. (2014). While we resize the images to a resolution of 416×128 pixels, we test the results on the ground truth depth recreated at the original image resolution. We do this by interpolating the estimated depths bi-linearly. Prior to training the encoder is pretrained on Imagenet and during training we augment images through horizontal flipping and by randomly varying the color space within the imagenet distribution. We use a batch-size of 4 and Adam (Kingma and Ba, 2015) as an optimizer with $\beta 1 = 0.9$, $\beta 2 = 0.999$. As an initial learning rate we use 0.0002 and halve it after every 100,000 batches without improvement. Even though the architecture of the model is relatively straightforward we have to apply a few tricks to allow it to train in a stable way. Our network contains few batch normalization layers in the decoder, which we assume helps with absolute scaling of the estimated depth outputs. This however makes it quite difficult to get the training started without exploding gradients. To mitigate this problem we start the training with five additional batch-normalization layers, located at the front of the decoder and between the decoder blocks. Once the model has started training successfully and the metrics have gone below a specific threshold we start removing the added normalization layers progressively starting from the end of the decoder stack. While without this system the model can start rarely if ever, with it starting the training is almost always successful. Also during the initial training we deactivate the obstacle branch and only activate it once the depth-estimation branch has started to show decent results. We train our Tensorflow implementation of the model for up to 50 epochs on a single GTX1080 which takes around 50 hours and select the checkpoint with the lowest training loss for testing.

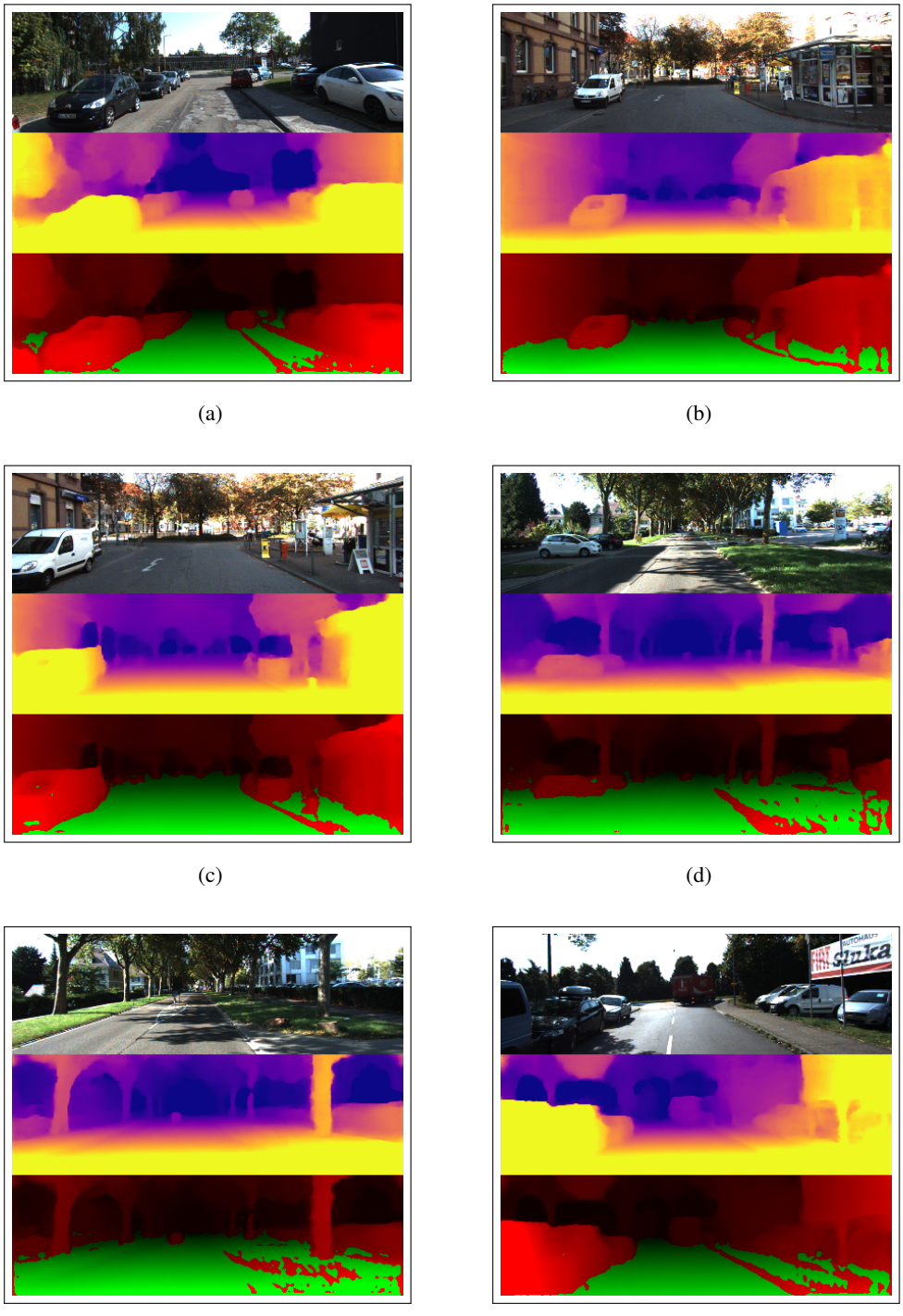
4 Experimental Evaluation

In the following section we present results from our models trained on the KITTI Raw dataset created by Geiger et al. (2013) and evaluated on the test-split proposed by Eigen et al. (2014). While we use a popular benchmark and common metrics in order to make the depth estimation results comparable, we are not aware of a benchmark to compare the results of our obstacle detection approach. The applied dataset contains around 42K rectified stereo images from 61 different scenes and the test split consists of 697 stereo image pairs. The ground truths are produced by a Velodyne laser scanner with a range of up to 80 meters and are reprojected onto the left image at the original image resolution, that slightly varies around 1392 × 512 pixels. In general ground truth data is used for evaluation only and we resize the estimations to the original resolution through bilinear interpolation. When displaying our obstacle segmentation results we combine them with the disparity, which being the inverse of the depth, helps to better display nearby details, while depth shows details equally over the whole distance, which would be less suitable for obstacle detection. As shown in Table 1 we are able to significantly improve the state of the art for self-supervised depth estimation and as shown in Figure 28, models using the obstacle-segmentation branch are able to produce convincing and reasonably detailed obstacle maps under varying circumstances. We test different variants of our model for Joint Unsupervised Depth and Obstacle Estimation (JUDO), and while all models deliver

comparably good results, we show that the addition of convolutional blocks in the skip connections as well as the addition of the obstacle-detection branch lead to a slight refinement of the model, while still running at over 60 frames per seconds during single-image inference on a single GTX1080.

Table 1: Evaluation and comparison of monocular depth estimation results on the KITTI dataset and test-set used by Eigen et al. (2014). As supervision they use ground-truth (D), motion (M), or stereo (S). All results use a maximum range of 80 meters. For pink columns lower is better while for yellow columns higher values are better.

Method	Sup	Abs Rel	Sq Rel	RMSE	RMSE log	δ_1	δ_2	δ_3
Eigen et al. 2014	D	0.214	1.605	6.563	0.292	0.673	0.884	0.957
Eigen et al. 2014	D	0.203	1.548	6.307	0.282	0.702	0.890	0.958
Zhou et al. 2017	S	0.208	1.768	6.856	0.283	0.678	0.885	0.957
Yang et al. 2017	S	0.182	1.481	6.501	0.267	0.725	0.906	0.963
Godard et al. 2017	S	0.141	1.369	5.849	0.242	0.818	0.929	0.966
Godard et al. 2018	S	0.133	1.158	5.370	0.208	0.841	0.949	0.978
Yang et al. 2018b	S	0.137	1.326	6.232	0.224	0.806	0.927	0.973
Yang et al. 2018b	M	0.131	1.254	6.117	0.220	0.826	0.931	0.973
Casser et al. 2018	M	0.109	0.825	4.750	0.187	0.874	0.958	**0.983**
JUDO$_{\text{no skip blocks}}$(ours)	S	0.0951	0.8900	4.5232	0.1825	0.9130	0.9610	0.9785
JUDO(ours)	S	0.0944	0.9005	4.5801	0.1818	0.9136	**0.9613**	0.9787
JUDO$_{\text{obstacle detection}}$(ours)	S	**0.0935**	**0.8446**	**4.4363**	**0.1817**	**0.9146**	0.9610	0.9784



(a) (b) (c) (d) (e) (f)

Figure 28: (top to bottom) 1) Original Image, 2) Estimated Depth, 3) Obstacle Disparity Product

5 Conclusion

We have presented a model for the self-supervised learning of depth-map prediction as well as obstacle segmentation from monocular images using only aligned stereo images during training. Applying current best practices as well as some novel loss variations and techniques we are able to significantly improve the current state of the art in self-supervised depth estimation and show that self-supervised obstacle segmentation can achieve promising results. This should be encouraging for future research in these areas since supervised approaches in both areas require considerable technical and financial resources and might under some conditions not be feasible at all. In the future more rigorous testing of the obstacle detection results will be tackled such that its results can be judged and explored more scientifically. Furthermore we would like to try approaches for better temporal consistency of our results and for simultaneous localization and mapping with included obstacle detection. Also the application of our system for semi-supervised learning of obstacle segmentation could be a promising topic for future exploration.

6 Discussion

While the quantitative results are strong compared to current methods, it is not absolutely clear how much the different components contribute to them. Detailed ablation studies would need to be performed to judge the individual contributions of architecture, staged training procedure or losses. Similarly the quality and utility of the obstacle detection results are currently left unvalidated and will need to be investigated further to judge them fairly. This detailed quantitative evaluation could not be performed within the limited amount of time and resources and will have to be done during future experiments. Furthermore testing on different data sets and practical implementations will provide a better understanding about how well these results can be generalized.

Bibliography

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP.

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2016). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:2481–2495.

Bengio, Y. (2019). Deep Learning:Theoretical Motivations. Retrieved from `http://videolectures.net/deeplearning2015_bengio_theoretical_motivations`. [Online; accessed 31. Aug. 2019].

Casser, V., Pirk, S., Mahjourian, R., and Angelova, A. (2018). Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. *CoRR*, abs/1811.06152.

Champandard, A. (2004). *AI game development : synthetic creatures with learning and reactive behaviors*. New Riders, Indianapolis, Ind.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805.

Doukkali, F. (2018). Batch normalization in Neural Networks. *Medium*.

Eigen, D., Puhrsch, C., and Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. In *NIPS*.

fast.ai (2018). Non-linear regression, Which network architecture? Retrieved from `https://forums.fast.ai/t/non-linear-regression-which-network-architecture/13304`. [Online; accessed 31. Aug. 2019].

Garg, R., Kumar, B. G. V., Carneiro, G., and Reid, I. D. (2016). Unsupervised cnn for single view depth estimation: Geometry to the rescue. *ArXiv*, abs/1603.04992.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *I. J. Robotics Res.*, 32:1231–1237.

Genin, V. (2018). DataScience Deep Dive. Retrieved from `http://dsdeepdive.blogspot.com/2016/03`. [Online; accessed 31. Aug. 2019].

Godard, C., Aodha, O. M., and Brostow, G. J. (2017). Unsupervised monocular depth estimation with left-right consistency. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6602–6611.

Godard, C., Aodha, O. M., and Brostow, G. J. (2018). Digging into self-supervised monocular depth estimation. *CoRR*, abs/1806.01260.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and Harnessing Adversarial Examples. *arXiv*.

gwern (2019). The Neural Net Tank Urban Legend - Gwern.net. Retrieved from `https://www.gwern.net/Tanks`. [Online; accessed 11. Aug. 2019].

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Huang, X. (2018). Retrieved from `https://cgi.csc.liv.ac.uk/~xiaowei/ai_materials/14-Gradient-Descent.pdf`. [Online; accessed 31. Aug. 2019].

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. (2019). Adversarial Examples Are Not Bugs, They Are Features. *arXiv*.

Jiaconda (2019). A Concise History of Neural Networks. *Medium*.

Jordan, J. (2018). Setting the learning rate of your neural network. *Jeremy Jordan*.

Kawaguchi, K. (2019). 2.3.3 Artificial Neuron with Continuous Characteristics. Retrieved from `http://wwwold.ece.utep.edu/research/webfuzzy/docs/kk-thesis/kk-thesis-html/node14.html#figGenNeuron`. [Online; accessed 31. Aug. 2019].

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. *ArXiv*, abs/1609.04836.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Kundu, J. N., Uppala, P. K., Pahuja, A., and Babu, R. V. (2018). Adadepth: Unsupervised content congruent adaptation for depth estimation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2656–2665.

Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., and Navab, N. (2016). Deeper depth prediction with fully convolutional residual networks. *2016 Fourth International Conference on 3D Vision (3DV)*, pages 239–248.

Liu, Z. y., Yu, S. y., Wang, X., and Zheng, N. (2017). Detecting drivable area for self-driving cars: An unsupervised approach. *CoRR*, abs/1705.00451.

Majek, K. (2017). 4K YOLO COCO Object Detection #1. Retrieved from `https://www.youtube.com/watch?v=yQwfDxBMtXg&t=177s`. [Online; accessed 31. Aug. 2019].

Mancini, M., Costante, G., Valigi, P., and Ciarfuglia, T. A. (2018). J-mod2: Joint monocular obstacle detection and depth estimation. *IEEE Robotics and Automation Letters*, 3:1490–1497.

Matthew Stewart, P. R. (2019). Neural Network Optimization. *Medium*.

Mei, J., Gao, B., Xu, D., Yao, W., Zhao, X., and Zhao, H. (2018). Semantic segmentation of 3d lidar data in dynamic scene using semi-supervised learning. *CoRR*, abs/1809.00426.

Neural network (2019). Neural network — Wikipedia, The Free Encyclopedia. Retrieved from `https://en.wikipedia.org/wiki/Neural_network`. [Online; accessed 9. Aug. 2019].

Overfitting (2019). Overfitting — Wikipedia, The Free Encyclopedia. Retrieved from `https://en.wikipedia.org/wiki/Overfitting`. [Online; accessed 31. Aug. 2019].

Poole, D. and Mackworth, A. (2017). Artificial Intelligence - foundations of computational agents – 7.3.2.1 Squashed Linear Functions. Retrieved from `https://artint.info/html/ArtInt_180.html`. [Online; accessed 31. Aug. 2019].

Regression analysis (2019). Regression analysis — Wikipedia, The Free Encyclopedia. Retrieved from `https://en.wikipedia.org/wiki/Regression_analysis`. [Online; accessed 9. Aug. 2019].

Ronneberger, O., Fischer, P., and Brox, T. (2015a). U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*.

Ronneberger, O., P.Fischer, and Brox, T. (2015b). U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer. (available on arXiv:1505.04597 [cs.CV]).

Strachnyi, K. (2019). Brief History of Neural Networks. *Medium*.

Sze, V., Chen, Y.-H., Yang, T.-J., and Emer, J. (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Universal approximation theorem (2019). Universal approximation theorem — Wikipedia, The Free Encyclopedia. Retrieved from `https://en.wikipedia.org/wiki/Universal_approximation_theorem`. [Online; accessed 9. Aug. 2019].

Vázquez, F. (2018). A "weird" introduction to Deep Learning. *Medium*.

Vieira, S., Pinaya, W., and Mechelli, A. (2017). Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications. *Neuroscience & Biobehavioral Reviews*, 74.

Violante, A. (2018). An Introduction to t-SNE with Python Example. *Medium*.

Wang, Z., Bovik, A. C., Sheikh, H. R., Simoncelli, E. P., et al. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.

Wu, Y., Ying, S., and Zheng, L. (2018). Size-to-depth: A new perspective for single image depth estimation. *CoRR*, abs/1801.04461.

Yang, N., Wang, R., Stückler, J., and Cremers, D. (2018a). Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *ECCV*.

Yang, Z., Wang, Y., Xu, W., and Nevatia, R. (2018b). Lego: Learning edge with geometry all at once by watching videos. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 225–234.

Yang, Z., Xu, W., Zhao, L., and Nevatia, R. (2017). Unsupervised learning of geometry with edge-aware depth-normal consistency. *CoRR*, abs/1711.03665.

Zbontar, J. and LeCun, Y. (2016). Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17:65:1–65:32.

Zhang, Y., Gu, S., Yang, J., Alvarez, J. M., and Kong, H. (2018). Fusion of lidar and camera by scanning in lidar imagery and image-guided diffusion for urban road detection. *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 579–584.

Zhao, M., Li, T., Abu Alsheikh, M., Tian, Y., Zhao, H., Torralba, A., and Katabi, D. (2018). Through-wall human pose estimation using radio signals. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhou, T., Brown, M. R., Snavely, N., and Lowe, D. G. (2017). Unsupervised learning of depth and ego-motion from video. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6612–6619.