CITY
UNIVERSITY OF LONDON
EST 1894

Evaluation of self-driving cars using CNNs in the rain

Individual Project Proposal

Daniel Sikar - MSc Data Science PT2
Supervisor: Artur Garcez

April 13, 2021

1 Introduction

Land vehicles that can drive autonomously, without human intervention, have been increasingly researched in the last 4 decades, harnessing the development in computer hardware, where decreasing size and increasing computational power allow such vehicles to carry mobile computing systems capable of performing the signal and algorithmic processing required to successfully self-drive along a path. Parallel to this development, improvements in one area of research in artificial intelligence (AI) known as machine learning has generated models known as convolutional neural networks (CNNs), successfully applied to the field of computer vision, where current state-of-the-art technology has surpassed human accuracy. Computer vision is a core component of self-driving, which relies heavily on images.

From the Stanford Cart, a camera-equipped roving robot, with a remote time-sharing computer as its ”brain”, and ”taking about five hours to navigate a 20 meter course” (Turk et al. 1988) to the current Tesla fleet with billions of accumulated self-driven miles (Karpathy 2020), self-driving cars are an increasing presence on public roads.

Automakers such as Tesla, Nissan, Audi, General Motors, BMW, Ford, Honda, Toyota, Mercedes and Volkswagen, and technology companies such as Apple, Google, NVidia and Intel, are currently researching self-driving vehicles (Ni et al. 2020). This involves a number of tasks related to decision-making and resulting motion control: path planning, scene classification, obstacle detection, lane recognition, pedestrian detection, traffic signage detection (including traffic lights). Reliability of self-driving cars under changing weather conditions is one important factor, considered since the earlier days of research (Turk et al. 1988).

Intel subsidiary MobileEye, NVIDIA and Tesla have developed, and continue to develop dedicated self-driving compute platforms, EyeQ, NVIDIA DRIVE AGX and HW respectively. All three platforms have custom-designed processors, optimised for AI computing. Dedicated inputs receive camera data, and sensor data to measure object proximity, using a combination of radar, sonar and lidar (radio, sound and light waves respectively). These inputs translate into three-

dimensional position and orientation information used for path planning. Tesla has used both MobileEye and NVIDIA platforms, before developing its own (Wikipedia contributors 2020).

Elon Musk, co-founder and CEO of Tesla, with respect to self-driving cars states that "(...) right now AI and Neural Nets are used really for object recognition (...), identifying objects in still frames and tying it together in a perception path planning layer thereafter. (...) Over time I would expect that it moves really to (...) video in, car steering and pedals out" (Musk 2019).

Based on this scenario, our **research question** is how do different CNN architectures compare in the rain, the **purpose of this research** is "to find evidence to inform practice" (Oates 2006) on autonomous vehicles transitioning to a CNN and computer-vision-only solution. The **product of this research** is an evaluation and the **intended beneficiary** is primarily the author, hoping this work will create academic and professional opportunities, as well as anyone else researching in the area that may use this work as a stepping stone or starting point.

2 Critical Context

A CNN is a class of artificial neural network, or simply neural network, having a basic unit known as a neuron, a mathematical model developed by McCulloch and Pitts (1943), based on biological models of the human brain. Neurons are defined as real numbers, in the form of weighted inputs and a bias, and an activation function, which, given a sum of input values multiplied by weights, plus the bias, generates an output value. The combination of weights and bias represents an encoding, the biological equivalent being a memory.

Based on the McCulloch-Pitts neuron, Donald Hebb (1949) created the first learning algorithm, that enables a neural network consisting of one single neuron to "learn", or encode, a memory, through an iteration process until, given a cost (or error) function, a set of weights and bias is found such that the same desired output is produced after a number of iterations, and the weights and bias reach stable values while the error is minimized. The Hebbian learning algorithm was able to learn simple tasks such as how to compute the OR truth table, but not more complex tasks such as the XOR truth table. This hindrance is known as a linear separability problem, where, using the inputs as coordinates, the output classes (0,1) of the XOR truth table plotted on a Cartesian plane, cannot be separated by a straight line. Solutions were eventually found, one involving the addition of another layer containing one neuron, known as a "hidden" layer, plus a connection between neurons, plus connections between inputs and the hidden layer neuron. The intuition being, a neural network with more neurons and more connections is able to learn more complex tasks. Such neural network architectures are known as multi-layer perceptrons (MLP) (Garcez 2018). In the model previously described, inputs are multiplied by weights. In the CNN model, inputs are "convolved" with weights. The concept is borrowed from the digital signal processing domain, where a vector known as a filter or kernel, is combined with a signal to generate a filtered output signal. The operation is expressed by:

$$conv(s1, s2)[t1] = \sum_{t=0}^{N2-1} s1[t1 - t]s2[t] \tag{1}$$

where $s1$ is the input signal, $s2$ is the kernel, $N2$ is the length of $s2$ and $t1$ is the time when input signal $s1$ was acquired. Convolution is similar to cross-correlation (where a measure of similarity between two signals is obtained) except convolution involves "flipping" one of the inputs. This can be observed by the indexing in $s1[t1 - t]$ (Pauwels and Weyde 2020).

For the case of a two-dimensional input and kernel, the operation takes the form:

$$J(x,y) = K * I = \sum_{n,m} K(n,m)I(x-n, y-m) \tag{2}$$

Where J is the convolved signal, K is the kernel, I is the input signal, and n, m are the kernel indexes. We see by the input signal indexing $I(x-n, y-m)$ that both input signal axes are "flipped".

A typical CNN architecture will have a number of convolutional layers, following by a fully connected MLP, that is, where every neuron is connected to each other. The convolutional layers are able to compress the input, without losing discriminative information, into a lower dimensional space, where different input categories can be efficiently represented. The fully-connected MLP layers then perform classification. Convolutional layers in neural networks with the ability to "self-organize", were introduced by the neocognitron network, proposed by Fukushima 1980, particularly efficient for image pattern recognition.

Finding optimal weights and biases for a neural network in a large search space is a mathematically intractable problem that can be solved by the use of back-propagation and gradient descent algorithms, like proposed by Rumelhart, Geoffrey E. Hinton, and Williams 1986 and Lecun et al. 1998, which lead to the wide adoption of CNNs.

The concepts described have been implemented in several machine learning libraries such as Facebook's PyTorch, Google's Tensorflow and MATLAB toolkits. Designing a network from scratch can amount in some cases to writing a few lines of code.

Self-driving vehicles have generally relied on designs where various sensor inputs are combined to generate a control signal (Turk et al. 1988) such as the pioneering Autonomous Land Vehicle (ALV) (Figure 1) . The ALV computer vision system, Vision Task Sequencer (VITS), treats the task of identifying the road ahead as a general classification problem with each pixel in any given image belonging to one of two classes, road and non-road. This is achieved by feature extraction, clustering and segmentation. Feature extraction is the process of engineering features that allow classes to be distinguished. Clustering partitions the feature space into mutually exclusive regions and segmentation assigns each image pixel to one regions The goal being to define regions that can be separated by a hyperplane (with dimension of the feature space minus one) boundary. Colour is used as the feature space. Other image features such as texture, saturation and reflectance from the laser scanner were not used.

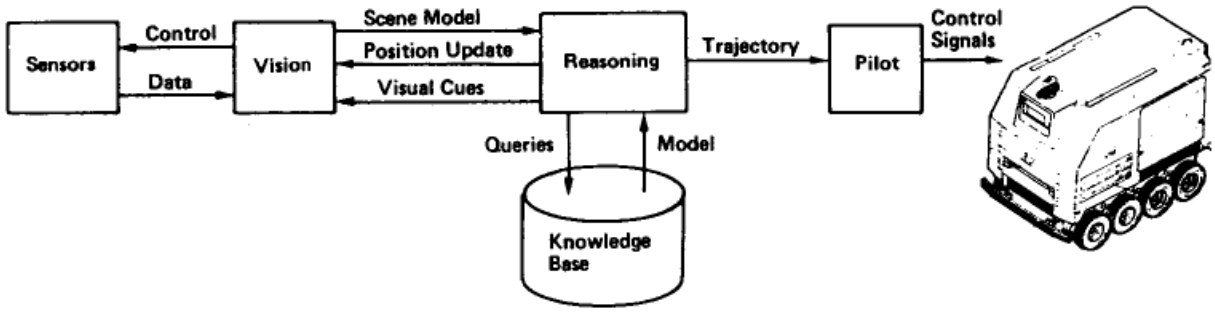

Figure 1: The ALV system configuration

Back-propagation was used to train the neural network for the next-generation ALV, the ALVINN, Autonomous Land Vehicle In a Neural Network (Pomerleau 1989), which has a 3-layer network

designed for the task of road following, taking images from a camera and a laser range finder (such as lidar) as input and producing the direction the vehicle should travel in order to follow the road as output. The neural network architecture consists of 30x32 video inputs, 8x32 laser range finder inputs 29 hidden units, one road intensity feedback unit and 45 direction output units. The road intensity feedback unit indicates whether the road is lighter or darker than the non-road in the previous image. The network is fully connected, that is, each of the input units is fully connected to the hidden layer, and each unit of the hidden layer is fully connected to the output layer. These improvements eventually lead to the RALPH system, used for a self-driving vehicle that in 1995 drove almost entirely autonomously across the USA.

The NVIDIA CNN (Figure 2) maps raw pixels from a single front-facing camera directly to steering commands. (Bojarski et al. 2016). A CNN image-in, steering-out had been previously implemented in a sub-scale radio control (RC) vehicle by the Defense Advanced Research Project Agency (DARPA) Autonomous Vehicle (DAVE) project. The end to end approach avoids the need for manual feature-extraction, clustering, segmentation and rules-based programming used to combined inputs and generate an output.

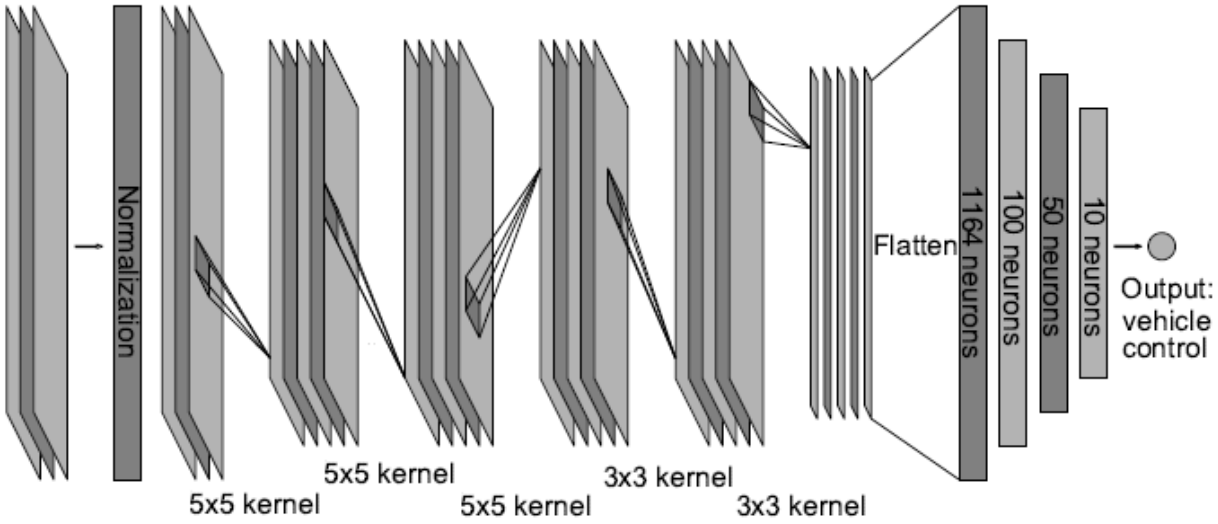

Figure 2: NVIDIA End-to-End CNN architecture

The NVIDIA CNN has approximately 27 million connections and 250 thousand parameters. The network consists of 9 layers, including a normalization layer, 5 convolutional layers and 3 fully connected layers leading to an output control value which is the inverse of the turning radius $\frac{1}{r}$ to avoid a singularity when driving straight (infinite radius). To remove a bias towards driving straight, a higher proportion of frames representing curves were added to the training data. The data was also augmented by random rotations to teach the network how to recover from unexpected shifts. The network runs on dedicated graphical processing units (GPU) hardware able to apply far more data and computational power to the task compared to previously used central processing units (CPU). This is the baseline network we are aiming to implement as a starting point to our evaluation.

3 Approaches: Methods & Tools for Analysis & Evaluation

In this section we describe the components shown in our work breakdown structure (Figure 3)

3.1 Literature Survey

We have conducted the initial literature survey using Google searches that usually led to the required publication. When we could not access an article, we used the library website and were generally able to access articles via institutional login.

3.2 Data & Tools

We identified 4 image datasets, 3 environments to train and test our models, and one image labelling crowdsourcing tool, to find relevant "rainy" sections in our data.

The **Ford Multi-AV Seasonal Dataset** is a multi-sensor dataset collected by a fleet of Ford autonomous vehicles at different days and times during 2017-18. They contain inertial measurement units (IMU) that provide orientation which are our required steering angles.

The **Audi dataset** provides 40,000 frames with semantic segmentation image and point cloud labels, of which more than 12,000 frames also have annotations for 3D bounding boxes. In addition, sensor data (approx. 390,000 frames) for sequences with several loops, recorded in three cities. The data need to be evaluated for steering angle labels

The **KITTI dataset** and benchmarks for computer vision research in the context of autonomous driving. The dataset has been recorded in and around the city of Karlsruhe, Germany using the mobile platform AnnieWay and has IMU labels.

The **Udacity self-driving datasets** two datasets of interest, containing labeled images with IMU and lidar values.

We intend to locate segments where rain is present with crowdsourcing **Amazon Mechanical Turk**. The expected outcome of this step is at least dozens of sequences containing rain.

We intend to use data augmentation to create synthetic data, with properties that would emulate rain characteristics, such as superimposing rain drops to images, then adding effects such as blurring, reflection and diffusion. The expected outcome of this step is at least dozens of sequences containing the augmented data plus metadata indicating the level of each applied effect.

Three environments have been identified to run CNN training and testing: the **Intel DevCloud**, the **NVidia open DRIVE Constellation** platform, and the the **Udacity Self-Driving Car Simulator**.

For initial network design we have identified **PyTorch**, **Keras** (Python Tensorflow wrapper) and **MATLAB** toolkits as potential tools.

3.3 Develop Models

Our data consists of videos, taken by cameras mounted on moving vehicles, which can be interpreted as sequences of still images taken at fixed-time intervals. Each still image is labelled with a quantity we call θ, that represents a direction from 0 to 359 degrees, in which the vehicle was travelling at the moment the image was stored. $\Delta\theta$ between two images, taken at intervals i and $i+n$, represents the amount of steering that was applied to the vehicle after n intervals. Therefore

the assumption is we are dealing with a regression problem, where, given a sequence of images as described, we want our models to approximate θ, keeping the autonomous vehicle on the road.

Our starting point is the NVIDIA CNN (Figure 2) model. Thereafter we expect to implement a number of alternative models to evaluate individually and compare to each other. The alternative models we are looking to implement are:

Our base model is the NVIDIA

AlexNet - five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To reduce overfitting in the fully-connected, dropout is used as a regularization method (Krizhevsky, Sutskever, and Geoffrey E Hinton 2012).

VGGNet - implementation showing increased depth up to 16-19 weight layers can be achieved by using an architecture with very small (3 x 3) convolution filters (Simonyan and Zisserman 2015).

Inception-v1, Inception-v2, Inception-v3 - variations of a deep convolutional neural network architecture, one implementation being GoogleLeNet, using the Network-in-Network approach in order to increase the representational power of neural networks, 1 x 1 convolutional layers are added to the network, acting as dimension reduction modules to remove computational bottlenecks, that would otherwise limit the size of the network. This allows for not just increasing the depth, but also the width of the network without a significant performance penalty (Szegedy et al. 2015).

ResNet - residual learning framework to optimize training of networks, where layers are reformulated as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions (He et al. 2015).

3.4 Evaluation

Developing the evaluation metric is part of this project. We are specifically interested in the steering aspect, and how much error (oversteering or understeering) would define an accident.

The initial step is to, once the networks have been trained and tested on our original ”dry” data, obtain a ground truth, where no accidents (autonomous vehicles driving off the road) are registered. We will then use our ”rainy” image sequences, plus augmented data edited in, and replacing original ”dry” data. The expectation is that at a given threshold of added rainy and augmented data, the autonomous network under test will output steering angles for a sufficient amount of frames that would cause the car to go off the road.

Once a few of these cases have been identified for our baseline model, the sequences will be used for all networks, and an evaluation made which should answer our research question.

3.5 Complete Report

The report writing component of this work is intended to be a continuous process, from the project start date until estimated completion date. The final reporting component being the first draft, then two rounds of reviews and addressing comments. The deliverable for this component is the final project dissertation.

4 Work Plan

Our work plan consists of a work breakdown structure (Figure 3) and a Gantt chart (Figure 4). We identified 5 principal components, each successive component has a dependency on, and cannot start before, the previous component being completed, which represents a milestone.

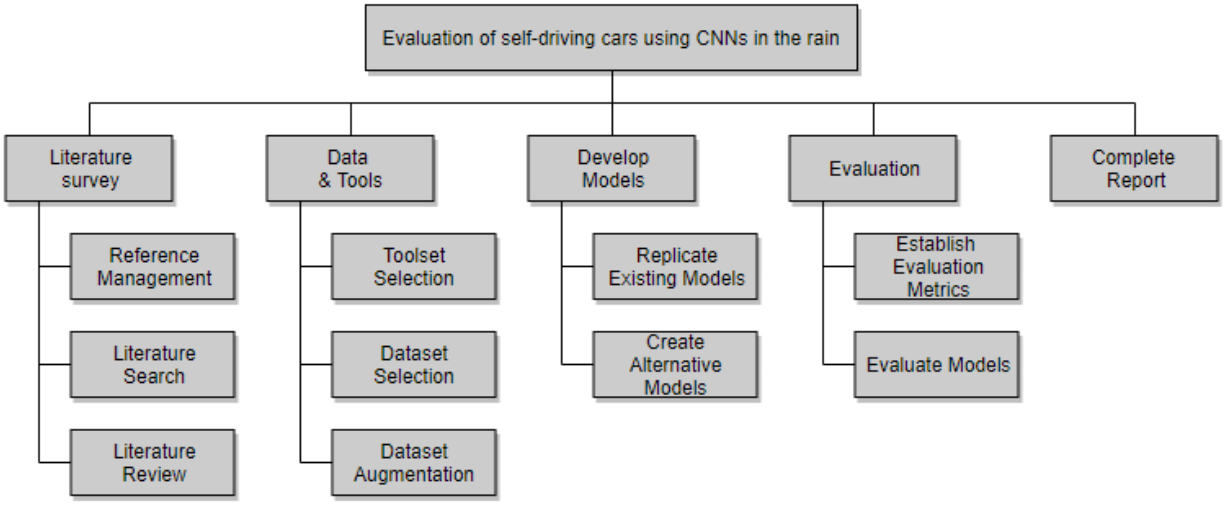

Figure 3: Work breakdown structure

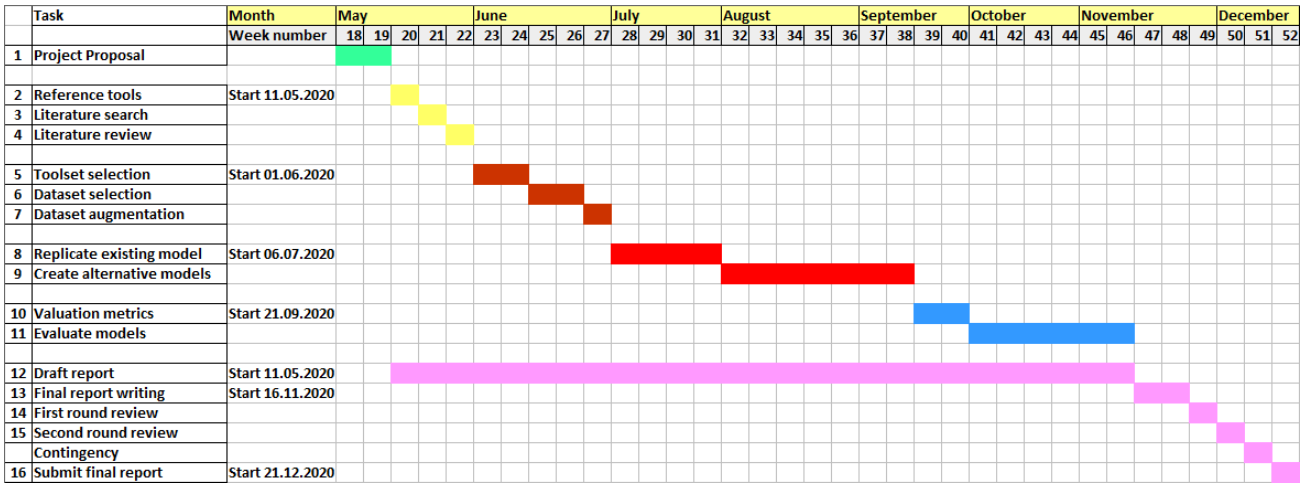

Figure 4: Gantt chart

5 Risks

To generate our risk register (Table 1), as suggested in Dawson 2009 we use:

$$I = L * C \tag{3}$$

where Likelihood L is categorized according to three-point scale Low/Medium/High, Consequence C according to five-point scale Very Low/Low/Medium/High/Very High and Risk Impact I is the product.

Description	**L**	**C**	**I**	**Mitigation**
Insufficient domain knowledge	2	4	8	Extend scope for literature survey and toolset selection and decrease scope for alternative models
Cannot source rain image sequences	1	5	5	Use augmented rain image sequences only
Unable to replicate NVIDIA CNN model	2	3	6	Use Udacity Unity engine as a new starting point
Sequence learning approach does not work or is too complex to implement.	2	4	8	Use single labelled image as input
Unable to create alternative models	2	4	8	Use NVIDIA CNN only, comparing ground truth ”dry” with ”rainy” data
Accidental loss of data.	2	5	10	Keep all source code, reporting and data online (github.com, overleaf.com, aws.amazon.com, camber.city.ac.uk and devcloud.intel.com)
Continued COVID-19 lockdown disruption	2	3	6	Scale down project
Final report delay	2	5	10	One additional week has been added for contingency

Table 1: Risk register

6 Ethical, Legal and Professional Issues

Having completed *Part A: Ethics Checklist* of the Research Ethics Review Form, as determined by the Computer Science Research Ethics Committee (CSREC 2020), we find this work complies with research ethics guidelines and does not require ethical approval.

References

Bojarski, Mariusz et al. (2016). "End to End Learning for Self-Driving Cars." In: *CoRR* abs/1604.07316. URL: http://dblp.uni-trier.de/db/journals/corr/corr1604.html#BojarskiTDFFGJM16.

CSREC (2020). *Research Governance Framework*. Department of Computer Science Research Ethics Committee, City, University of London.

Dawson, Christian W. (2009). *Projects in Computing and Information Systems: A Student's Guide*. 2nd. Pearson Prentice Hall. ISBN: 978-0-273-72131-4.

Fukushima, Kunihiko (1980). "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position". In: *Biological Cybernetics* 36, pp. 193–202.

Garcez, Artur (2018). *INM427 Neural Computing Lecture Notes*.

He, Kaiming et al. (2015). *Deep Residual Learning for Image Recognition*. arXiv: 1512.03385 [cs.CV].

Karpathy, Andrej (2020). *AI for Full-Self Driving, Scale ML Conference*. URL: https://www.youtube.com/watch?v=hx7BXih7zx8&t=1m30s (visited on 05/01/2020).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., pp. 1097–1105.

Lecun, Yann et al. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE*, pp. 2278–2324.

Musk, Elon (2019). *TESLA Autonomy Day Event*. URL: https://www.youtube.com/watch?v=tbgtGQIygZQ&t=2h25m52s (visited on 05/01/2020).

Ni, Jianjun et al. (2020). "A Survey on Theories and Applications for Self-Driving Cars Based on Deep Learning Methods". In: *Applied Sciences* 10.8. ISSN: 2076-3417. DOI: 10.3390/app10082749. URL: https://www.mdpi.com/2076-3417/10/8/2749.

Oates, Briony J. (2006). *Researching information systems and computing*. 1st. SAGE Publications. ISBN: 9781446235447.

Pauwels, Johan and Tillman Weyde (2020). *INM378 Digital Signal Processing and Audio Processing Lecture Notes*.

Pomerleau, Dean A. (1989). "ALVINN: An Autonomous Land Vehicle in a Neural Network". In: *Advances in Neural Information Processing Systems 1*. Ed. by D. S. Touretzky. Morgan-Kaufmann, pp. 305–313. URL: http://papers.nips.cc/paper/95-alvinn-an-autonomous-land-vehicle-in-a-neural-network.pdf.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). "Learning Representations by Back-propagating Errors". In: *Nature* 323.6088, pp. 533–536. DOI: 10.1038/323533a0. URL: http://www.nature.com/articles/323533a0.

Simonyan, Karen and Andrew Zisserman (2015). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations*.

Szegedy, Christian et al. (2015). "Going Deeper with Convolutions". In: *Computer Vision and Pattern Recognition (CVPR)*. URL: http://arxiv.org/abs/1409.4842.

Turk, M. A. et al. (1988). "VITS-a vision system for autonomous land vehicle navigation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10.3, pp. 342–361.

Wikipedia contributors (2020). *Tesla Autopilot*. [Online; accessed 02-May-2020]. URL: https://en.wikipedia.org/wiki/Tesla_Autopilot.