BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING

FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

SEMANTIC SEGMENTATION OF IMAGES USING CONVOLUTIONAL NEURAL NETWORKS

SÉMANTICKÁ SEGMENTACE OBRAZU POMOCÍ KONVOLUČNÍCH NEURONOVÝCH SÍTÍ

MASTER'S THESIS

DIPLOMOVÁ PRÁCE

AUTHOR — **Bc. FILIP ŠPILA**

AUTOR PRÁCE

SUPERVISOR — **doc. Ing. JIŘÍ KREJSA, Ph.D.**

VEDOUCÍ PRÁCE

BRNO 2020

Abstrakt
Tato bakalářská práce se zabývá návrhem, výrobou a realizací řízení nestabilního robota, balancujícího na sférické základně, známého také pod názvem ballbot. Předpokládá se kompletní návrh konstrukce, výběr pohonných jednotek, návrh, implementace a testování inteligentního řídícího algoritmu, který udrží robota v metastabilní rovnovážné poloze. Při vývoji budou využity softwarové nástroje MATLAB/Simulink. Práce také počítá s využitím mikrokontroleru dsPIC jako platformy pro finální řízení celého systému. Zadání projektu má interdisciplinární charakter a je realizováno jako týmová práce s jasně vymezenými úkoly pro jednotlivé členy.

Summary
This bachelor's thesis deals with the complete design, manufacture, and control of an unstable robot, balancing on a spherical base, also known as ballbot. The complete design of the construction, motor unit selection, design, implementation testing of an intelligent control algorithm to keep the robot in a meta-stable equilibrium is assumed. Multiple tools such as Matlab/Simulink are used for this approach. It also includes the final implementation of the code in the PIC microcontroller. The project has an interdisciplinary character and is meant to be done as teamwork whereby each team member has a strictly defined role.

Klíčová slova
Ballbot, konstrukce, inteligentní řízení, PID, MATLAB, Simulink

Keywords
Ballbot, construction, intelligent control, PID, MATLAB, Simulink

ŠPILA, F.*Semantic segmentation of images using convolutional neural networks.* Brno: Vysoké učení technické v Brně, Faculty of Mechanical Engineering, 2020. 10 s. Vedoucí doc. Ing. Jiří Krejsa, Ph.D.

Prohlašuji, že jsem svou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, software atd.) citované v práci a uvedené v přiloženém seznamu a postup při zpracování práce je v souladu se zákonem č. 121/2000 Sb.,o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V Brně 1. května 2017

Bc. Filip Špila

Děkuji Ing. Tomáši Spáčilovi, Matěji Rajchlovi a celému týmu z Mechlabu za podnětné připomínky a rady, které mi během práce poskytli. Dále děkuji své rodině za podporu jak během studií, tak během psaní této práce.

Bc. Filip Špila

Contents

1. Introduction

Image segmentation is one of the essential parts of computer vision and autonomous systems alongside with object detection and object recognition. The goal of semantic segmentation is to automatically assign a label to each object of interest (person, animal, car etc.) in a given image while drawing the exact boundary of it and to do this in the most robust and reliable way possible. Speaking in terms of machine learning, each pixel of the input image is intended to belong to a specific class.

We can see a real-world example in Figure 1. Each pixel of the image has been assigned to a specific label and represented by a different colour. Red for people, blue for cars, green for trees etc. This is unlike mere image classification task where we classify the image scene as a whole. It is appropriate to say that semantic segmentation is different from so called instance segmentation, where one not only cares about drawing boundaries of objects of a certain class but also wants to distinguish between different instances of the given class. For instance, all people in the image (each instance of the 'person' class) would all have a different colour.

It turns out that semantic segmentation has many different applications in the fields such as driving autonomous vehicles, human-computer interaction, robotics, and photo editing/creativity tools. The most recent development shows the increasing need for reliable object recognition in self-driving cars because it is crucial for the models to understand the context of the environment they're operating in.

The presented work focuses on research and implementation of one particular segmentation method that uses convolutional neural networks (CNNs). CNNs belong to the family of machine learning algorithms and got under attention mainly due to their groundbreaking success in image classification challenges (ImageNet etc,). They subsequently found their use in segmentation tasks where researchers take the most well-performing CNN architectures and use it as the first stage of the segmentation algorithm.

Figure 1.1: Segmentation of an urban road scene

2. Problem statements

The assignment of this thesis consists of several expected achievements. Firstly, a promising segmentation method using CNNs needs to be found and implemented. It is expected that the neural network will be as straightforward as possible while still being likely to be capable of giving satisfactory results for the chosen use case (segmentation of a path in outdoor environment for a robot navigation). The images will be provided by the supervisor of the thesis and used to train and validate the network performance. In addition, the author will pick an appropriate software tool for creating Ground Truths (FOOTNOTE manually created image labels that serve as a reference for the network to validate its current accuracy of prediction and to compute the needed adjustments of its parameters to get closer to the desired output) and use it to create the final training and validating datasets. Lastly, the network should be trained with various sets of hyperparameters (FOOTNOTE: hyperparameter definition) in order to get a closer idea of the network's training behaviour and to ensure the best possible results.

3. Research and theory

First part of this section gives a thorough introduction to neural networks (NN) in general. It begins by definition of fundamental terms needed to fully understand the core principles of NNs. Due to the fact that the research in this area is still heavily ongoing, the more advanced techniques described here may soon be out of date or replaced by better-performing ones and therefore the theoretical background is limited only to the extent relevant for the particular chosen network architecture. Still, it will give a solid foundation needed to understand other similar approaches.

Second part presents some of the main approaches based on machine learning researches have recently used to tackle the semantic segmentation problem. However, not all of them use CNNs as the core algorithm. This part summarizes the main key points from the corresponding papers that contributed to this topic by presenting novel architectures and principles. It finishes by more detailed description of a method that is eventually found the most promising and thus selected for the final implementation.

3.1. Supervised learning

Artificial neural network algorithms are inspired by the architecture and the dynamics of networks of neurons in human brain. They can learn to recognize structures in a given set of training data and generalize what they have learnt to other data sets (supervised learning). In supervised learning one uses a training data set of correct input/output pairs. One feeds an input from the training data into the input terminals of the network and compares the states of the output neurons to the target values. The network trainable parameters are changed as the training continues to minimize the differences between network outputs and targets for all input patterns in the training set. In this way the network learns to associate input patterns in the training set with the correct target values. A crucial question is whether the trained network can generalize: does it find the correct targets for input patterns that were not in the training set?

3.1.1. Feedforward neural networks

The goal of a feedforward neural network is to find a non-linear, generally n-dimensional function that maps the space of the inputs x to the space of the outputs y. In other words, to learn the function [zdroj SANTIAGO]

$$f^* : \mathbb{R}^m \to \mathbb{R}^n, f^*(x; \phi)$$

where ϕ are trainable parameters of the network. The goal is to learn the value of the parameteres that result in the best function approximation, by solving the equation

$$\phi \leftarrow \ arg\,min\, L(y, f^*(x; \phi))$$

where L is a loss function chosen for the particular task. One can understand the term 'loss function' simply as a metric of 'how happy we are about the output that the network gives us for a given input' and therefore $f^*(x; \phi)$ is driven to match the ideal function $f(x; \phi)$ druing network training.

The structure of a feedforward network is usually composed of many nested functions. For instance, there might be three functions $f^{(1)}$, $f^{(2)}$ and $f^{(3)}$ connected in a chain to the form

$$f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$$

These models are reffered to as feedforward because information flows from the deepest nested function $f^{(1)}$ taking x as its direct input to other functions in the chain and finally to the output y. One can name the functions starting by $f^{(1)}$ as the first layer (input layer) of the network, $f^{(2)}$ is called the second layer, and so on. The final layer of the network is called output layer.

Recall that in supervised learning one needs a set of training data, in this case a set of matching x, y (footnote: y are often called LABELS) pairs. The training examples specify directly what the output layer must do at each point x; it must produce a value that is as close as possible to y. The behaviour of the other layers is not specified by the training data which is why we call these layers 'hidden layers'. In Figure XY , an image of a four-layer feedforward neural network with two hidden layers can be seen.

It is now very important to highlight that the neural network must be seen as something that is capable of modeling practically any function we can think of [general approximation theorem]. The power of this brings us to the definition of a classification task. In this task, the function the network approximates has discrete states, true/false. To give the most typical example, let's consider we have a large set of black and white images containing hand-written digits from 0 to 9. Now we want the network to basically associate the information encoded in the pixel values of the input image with the information of what digit that particular image shows. In this case, the vector space of inputs (flattened image array in the simplest case of size m = width*height) is projected to the space of size n = 10 categories (digits 0 to 9).

3.1.2. McCulloch-Pitts neurons

Layers in Figure XY can be further divided into distinct computational units (again, just another nested functions) called neurons. This is where the resemblance with/to biological neurons comes into play. The neurons are mathematically modelled as linear threshold units (McCulloch-Pitts neurons), they process all input signals coming to them from other neurons and compute an output. In its simplest form, the model for the artificial neuron has only two states, active or inactive. Let's stick with this simple model for a while. If the output exceeds a given threshold then the state of the neuron is said to be active, otherwise inactive. The model is illustrated in Figure 1.4. Neurons usually perform repeated computations, and one divides up time into discrete time steps $t = 0, 1, 2, 3, ...$ The state of neuron number j at time step t is denoted by

$$n_j(t) = \begin{cases} 0 & \textit{inactive}, \\ 1 & \textit{active}. \end{cases}$$

Given the signals $n_j(t+1)$, neuron number i computes

$$n_j(t+1) = \theta_H \left(\sum_j w_{ij} n_j(t) - \mu_i \right)$$

As written, this computation is performed for all i neurons in parallel, and the outputs n_i are the inputs to all neurons at the next time step, therefore the outputs have the time argument $t + 1$.

The weights wi j are called synaptic weights. Here the first index, i, refers to the neuron that does the computation, and j labels all neurons that connect to neuron i. The connection strengths between different pairs of neurons are in general different, reflecting different strengths of the synaptic couplings.

The argument of θ_H is often referred to as the local field

$$b_i = \sum_j w_{ij} n_j(t) - \mu_i$$

Equation (1.2) basically shows that the neuron performs a weighted linear average of the inputs nj and applies an offset (threshold) which is denoted by µi. Finally, the function θ_H is reffered to as the activation function.

3.1.3. Activation function

The general motivation for using activation functions is to bring non-linearity to the model. In the simplest case that has been discussed so far, the neurons can only have the states 0/1, which in terms of the activation function corresponds to the Heaviside function

$$\theta_H(b) = \begin{cases} 1 & \text{for } b > 0, \\ 0 & \text{for } b < 0. \end{cases}$$

In practice however, the simplest model must be generalized by allowing the neuron to respond continuously to its inputs, which is necessary for the optimization algorithms used in the training phase to perform well. To this end one replaces Eq. (1.2) by a general continuous activation function $g(b)$. An example is shown in Figure 1.6. This dictates that the states assume continuous values too, not just the discrete values 0 and 1 as given in Equation (1.1).

There are several choices for the activation function which all come with their 'pros and cons' for a particular application the network is used for. However, there's a few requirements all of these functions should meet:

- Nonlinearity. As discussed above, the non-linearity is a general ability of a neural network allowing it to model very complex functions
- Monotocity and nondecreasibility. This allows certain optimization algorithms to perform more stable as we'll see further.
- Differenciability (or at least piecewise differencialbility). This is useful not only in terms of stability of the optimization algorithms, but also for the analytical derivation of the update rule for the network parameters during optimization.

One also needs to distinguish between activation functions used in neurons in the input/hidden layers and neurons in the output layer. The reason for that comes from the definition of a classification task, where we would like to interpret the outputs of the

network as 'probabilities' of the input belonging to a certain class. For this one can usually choose between commonly used Softmax classifier (function) and Sigmoid function. In the example with hand-written digits, when we feed the network with an image showing digit 7, we want the network to spit out the 100 percent probability for the image belonging to the class 'digit 7' and 0 percent probability of it belonging to the classes 'digit 0', 'digit 1'... In such case, the output of the last layer of the network not only needs to be within 0 and 1, but in case of using Softmax the sum of all outputs must give unity as they are intepreted as (relative!) probabilities. We say 'relative' because the network's decision is only based on the features of this particular image differing in comparison with other data we fed in during training and does not reflect 'outer' probability at all.

- Sigmoid

$$g(x) = \frac{1}{1 + e^{-x}}$$

  This function used to be broadly used mainly thank to the clear interpretation of the state of the neuron - active/inactive is represented by values 0/1. Sigmoid is currently obsolete for large networks. In short, it does not have optimal properties for the learning algorithm called 'backpropagation', which is to be discussed in next chapters. Also, the fact that its mean value in non-zero doesn't have a positive impact on the learning process either. [Groman]

- Hyperbolic tangent

$$g(x) = \tanh(x)$$

  Unlike Sigmoid, the range of its output is in the interval <-1,1>. However, it still has the same limitations as the previous function and therefore is not a suitable candidate for learning via backpropagation.

- Rectified Linear unit (ReLu)

$$g(x) = \max(0, x)$$

  The authors of this concept found the inspiration in real biological neurons. The idea comes from a model for the relation between the electrical current through the cell membrane into the neuron cell, and the membrane potential. The main message is that there is a threshold below which the response of the neuron is strictly zero, as shown in Figure XY. The derivative of the ReLU function is discontinuous at $x = 0$. A common convention is to set the derivative to zero at $x = 0$.

- Parametric (leaky) ReLu

$$g(x) = \max(x, \alpha x)$$

  By modifying the previously introduced function one gets a version of ReLu intended to address the biggest drawback of ReLu, which is the fact that some neurons may

become dead (their output will be always zero) and thus not contribute to the network's output. Unfortunately there's generally no guarantee that using Leaky ReLu instead of ReLu will always mean better results.

4. Bibliography

[1] ZOUHAR, František. *Návrh konstrukce, řízení a elektroniky pro nestabilní balancující vozidlo* [online]. Brno: Vysoké učení technické v Brně. Fakulta strojního inženýrství, 2011 [cit. 2017-05-21]. Dostupné z: http://hdl.handle.net/11012/21119. Diplomová práce. Vysoké učení technické v Brně. Fakulta strojního inženýrství. Ústav mechaniky těles, mechatroniky a biomechaniky. Vedoucí práce Robert Grepl.

5. Seznam použitých zkratek a symbolů

CMU	Carnegie Mellon University

6. Seznam příloh

- Nastavení režimu External mode: *external.txt*