DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

Evaluation and Generalization of Capsule Networks in Neurorobotics

Jean A. Elsner

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

Evaluation and Generalization of Capsule Networks in Neurorobotics

Evaluation und Generalisierung von Kapsel Netzwerken in der Neurorobotik

Author:	Jean A. Elsner
Supervisor:	Prof. Dr. Alois C. Knoll
Advisor:	Alexander Kuhn
Submission Date:	TBD

I confirm that this master's thesis in robotics, cognition, intelligence is my own work and I have documented all sources and material used.

Munich, TBD Jean A. Elsner

Acknowledgments

Abstract

Contents

1 Introduction

2 State of the Art

This chapter presents an overview of state of the art approaches to object recognition, while focusing on two families of architectures, which are motivated quite differently. Object recognition techniques based on convolutional neural networks (*CNN*s) currently dominate the field, achieving state of the art performance on many datasets [2, 5]. CNNs however, are only loosely based on biological neurons. Spiking neural networks (*SNN*s) on the other hand, try to mimic the physical properties of neurons more closely and therefore constitute biologically more plausible models [6]. Generally speaking, CNNs may be regarded as a more engineering-based approach (or top-down), while SNNs are motivated by results from neuroscience and biology (bottom-up approach).

2.1 Deep Learning for Object Recognition

Recent years have seen a surge of interest in deep learning methods, especially in the field of computer vision. While the theory behind many deep learning methods has been around for many years, their recent success is mainly due to the availability of large labelled data sets and highly parallel computing powered by GPUs. One of the specific tasks, deep learning based methods excel at, is object recognition: the identification of objects in images or videos (cf. figure 2.1). The significantly better performance of deep neural networks over traditional machine learning methods can be explained by: (i) their hierarchical topology of parameterized non-linear processing units is a fundamentally better probabilistic model and prior for real world data leading to better generalization and (ii) they automatically find good features to extract based on the training data. The potential applications for a robust image classification system are myriad and range from automated driving and image-based diagnosis to robot vision and many more. As deep learning is currently the best candidate for such a system, it is well worth exploring.

2.1.1 Convolutional Neural Networks

CNN architectures are generally distinguished by their use of specific types of neuron-layers, namely, convolutional, pooling and fully connected layers. While wildly different network topologies may be found in literature, characterized by their use of skip connections, number of layers, number of paths etc., CNNs can always be reduced to these three basic layer types.

Figure 2.1: Sample images from the CIFAR-10 [4] dataset and their corresponding classes. CIFAR-10 consists of 6000 images at 32 by 32 pixels for each of the 10 classes. Datasets such as this are often used as a benchmark to evaluate the performance of novel deep learning architectures for image recognition. This is done by using a subset of the dataset to train the neural network. The remainder of the images, which the network has not seen before, are used to evaluate the accuracy.

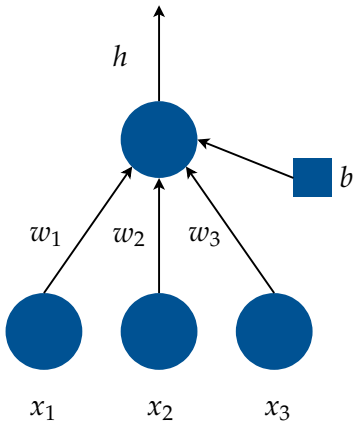

Figure 2.2: Illustration of an artificial neuron with three input connections. Artificial neurons constitute the basic non-linear computational units of neural networks. The neuron receives the activations of three lower level neurons weighted by the learnable w_i as well as a learnable bias b. After applying the nonlinearity, the neuron outputs its activation h as in equation 2.1

Fully Connected Layer

Each neuron in a fully connected layer is connected to all the activations in the previous layer. The activation of a single neuron (cf. figure 2.2) is calculated by applying a nonlinearity to the weighted sum of its inputs and a bias.

$$h = g\left(\sum_i w_i x_i + b\right) \tag{2.1}$$

With the nonlinear function g, the learnable weights w_i, the input activations x_i and the learnable bias b. In the case of a fully connected layer, the activations can be computed using matrix multiplication. In tensor notation this may be written as:

$$\mathbf{h_l} = g_l\left(\mathbf{W_l^T h_{l-1}} + \mathbf{b_l}\right). \tag{2.2}$$

With N_l denoting the number of neurons in layer l, $\mathbf{W_l}$ is an $N_{l-1} \times N_l$ dimensional weight matrix, b_l an N_l dimensional vector and g_l the N_l dimensional vectorized activation function of layer l.

$$g_l(\mathbf{x}) = \left(g_l(x_1), ..., g_l(x_{N_l})\right)^T \tag{2.3}$$

The computational power of neural networks is shown by the universal approximation theorem. The theorem states, that networks with a single hidden layer (cf. figure 2.3) containing a finite number of neurons can approximate arbitrary continuous functions on compact subsets of $\mathbb{R}^n$ [1, 3].

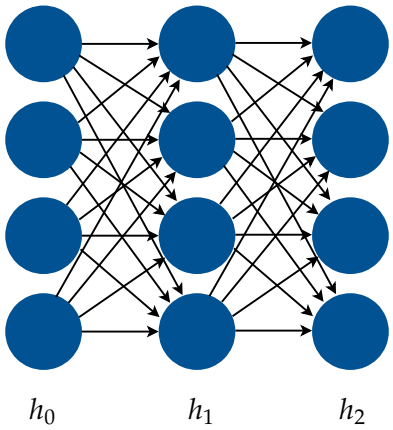

Figure 2.3: Illustration of a fully connected neural network as a directed graph with input layer h_0 and output layer h_2. Layers between the input and output layer are usually referred to as *hidden* layers. Note that the biases are not explicitly shown.

Convolutional Layer

In a convolutional layer, the activities of the input layer are convolved with a number of trainable kernels so as to create the same number of feature maps. In computer vision it is common to view the layer's neurons as two-dimensional grids of neurons arranged in channels (cf. figure 2.5). These grids correspond to pixels and color channels in the case of the input layer or activities (feature maps) resulting from convolution with different kernels in the case of intermediate convolutional layers. For a feature kernel $F^l_{m,n}$ of size $M_l \times M_l$ and an input layer $l-1$ with $N_{l-1} \times N_{l-1}$ neurons, the corresponding feature map activities of layer l are computed as

$$h^l_{m,n} = g_l\left(\sum_{m'}^{M_l}\sum_{n'}^{M_l} F^l_{m',n'} h^l_{m+m',n+n'} + b_l\right). \tag{2.4}$$

This is the same as a two-dimensional discrete cross correlation.

$$(f \star g)[n] \overset{\text{def}}{=} \sum_{m=-\infty}^{\infty} f^*[m]\; g[m+n] \tag{2.5}$$

Where f^* is the complex conjugate of the discrete function f. Strictly speaking, the use of the term *convolution* in neural network literature is therefore a misnomer, as either the filter or the image (or feature map) would have to be flipped before the operation. However, as the weights of the kernel are actually learned by the network, the result will be the same.

For a 2D single channel input layer of size $N_0 \times N_0$, the number of trainable parameters for a convolutional layer with a single $M \times M$ filter is $M^2 + 1$ for the kernel weights and bias respectively, compared to $N_0^2 N_1^2 + N_1^2$ for a fully connected layer of size $N_1 \times N_1$. The

sparsity in learnable parameters in convolutional layers compared to fully connected layers (cf. fig 2.4) is often referred to as *weight sharing* (an entire channel *shares* the weights of a single kernel).

In the case of multiple input channels, the kernels actually extend through the whole depth of the input layer's volume. Equation 2.4 can be extended to accommodate multiple input channels and feature kernels. The activities of feature map k in layer l are then computed as

$$h^l_{k,m,n} = g_l\left(\sum_{k'}^{K_{l-1}}\sum_{m'}^{M_l}\sum_{n'}^{M_l} F^l_{k,k',m',n'} h^l_{k',m+m',n+n'} + b_{l,k}\right). \tag{2.6}$$

With K_l the number of channels in layer l. Most deep learning frameworks offer additional hyperparameters for convolutional layers, like stride and zero padding. Zero padding adds additional rows and columns of zero-activities around the input layer channels, effectively allowing the output feature maps to have the same size (that is height and width, depth is determined by the number of kernels) as the unpadded input layer. Stride on the other hand defines the integer value by which the kernels are moved during convolution. This can be used to keep the receptive fields from overlapping too much. Equation 2.6 is easily extended to take the stride S_l of layer l into consideration.

$$h^l_{k,m,n} = g_l\left(\sum_{k'}^{K_{l-1}}\sum_{m'}^{M_l}\sum_{n'}^{M_l} F^l_{k,k',m',n'} h^l_{k',S_l m+m',S_l n+n'} + b_{l,k}\right) \tag{2.7}$$

The spatial dimension of this feature map can be computed as a function of the input layer size $N_{l-1} \times N_{l-1}$ and the amount of zero padding P_l.

$$N_l = \frac{N_{l-1} - M_l + 2P_l}{S_l} + 1 \tag{2.8}$$

Pooling Layer

Pooling is a form of sub- or downsampling using a sliding window similar to convolution. Most often the stride is set in a way, such that the windows do not overlap. An operator is applied to each window, which selects a subset of neurons. Common operators are maximum, average or L_2-Norm. The pooling is applied to each channel and leads to a reduction in the spatial dimensions. This reduction corresponds to a loss of information or reduction in parameters and therefore reduces the amount of computation required as well as the possibility of overfitting.

2.1.2 Spiking Neural Networks

2.2 Limits of Deep Learning Approaches

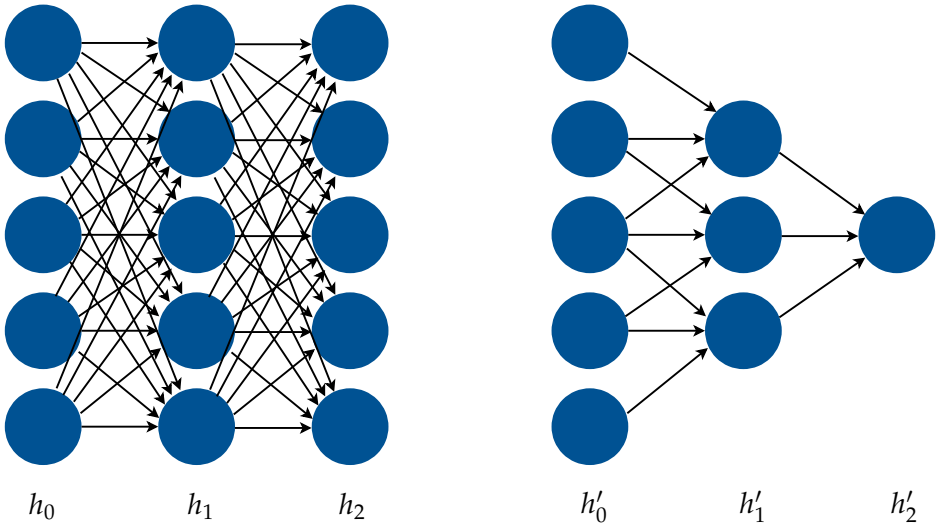

Figure 2.4: Illustration of fully connected layers compared to convolutional layers, making apparent the sparsity of the latter relative to the former. The convolutional layers h'_1 and h'_2 perform a 1D convolution with a filter of size 3. Note that due to weight sharing the number of weight parameters between h'_0 and h'_1 is actually only 3. Also note, that convolution reduces the number of neurons (or pixel resolution in the case of an image). For an input layer with N neurons and a filter kernel of size M the number of neurons in the convolutional layer computes as $N - M + 1$.

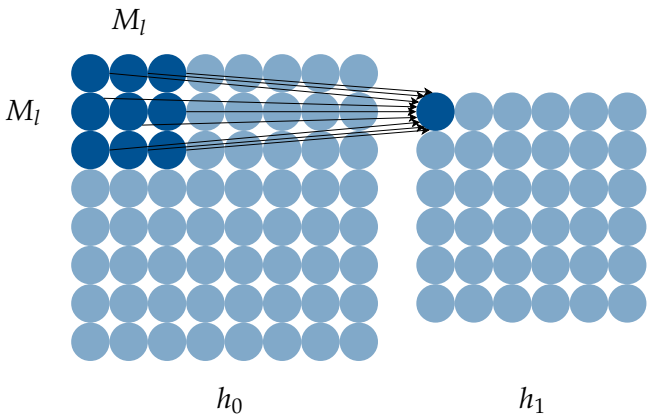

Figure 2.5: Illustration of a 2D convolution with a kernel of size $M_l \times M_l$. As the resulting feature map h_1 has a lower resolution than h_0, applying a convolutional layer is sometimes also referred to as downsampling. For an input layer with $N \times N$ neurons, the feature map's size is computed as $(N - M_l + 1) \times (N - M_l + 1)$. The area comprised of the input neurons that are used to calculate the activity of a feature map neuron (highlighted neurons in layer h_0) are known as that neuron's *receptive field*.

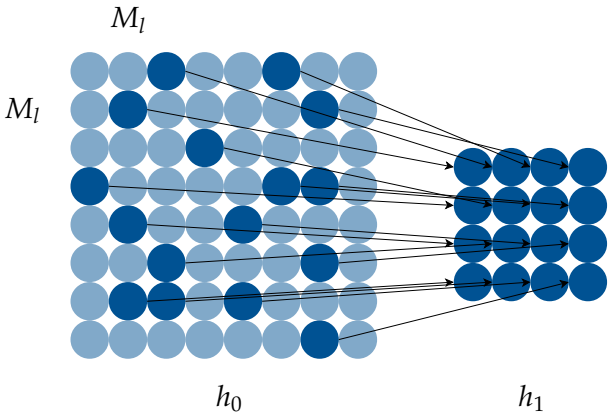

Figure 2.6: Illustration of a pooling layer.

3 Capsule Network Architectures

4 Experimental Setup

5 Results

6 Discussion

7 Conclusion

List of Figures

List of Tables

Bibliography

[1] G. Cybenko. "Approximation by superpositions of a sigmoidal function." In: *Mathematics of Control, Signals and Systems* 2.4 (Dec. 1989), pp. 303–314. DOI: https://doi.org/10.1007/BF02551274.

[2] A. Diba, V. Sharma, A. M. Pazandeh, H. Pirsiavash, and L. V. Gool. "Weakly Supervised Cascaded Convolutional Networks." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 5131–5139.

[3] K. Hornik. "Approximation capabilities of multilayer feedforward networks." In: *Neural Networks* 4.2 (1991), pp. 251–257. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(91)90009-T.

[4] A. Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. 2009.

[5] W. Ouyang, X. Zeng, X. Wang, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, H. Li, K. Wang, J. Yan, C. C. Loy, and X. Tang. "DeepID-Net: Object Detection with Deformable Part Based Convolutional Neural Networks." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.7 (July 2017), pp. 1320–1334. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2016.2587642.

[6] A. J. Schofield, I. D. Gilchrist, M. Bloj, A. Leonardis, and N. Bellotto. "Understanding images in biological and computer vision." In: *Interface Focus* 8.4 (2018). ISSN: 2042-8898. DOI: 10.1098/rsfs.2018.0027. eprint: http://rsfs.royalsocietypublishing.org/content/8/4/20180027.full.pdf.