Convoluational Neural Networks for German Roadsign Detection

Roman Schulte-Sasse

December 6, 2016

1 Introduction

Extracting the contents of a digital image has been proven to be a hard problem for computers. Since for them, an image is only a matrix of values, knowing what structures a human would recognize in this image, is a non-trivial problem.
Yet it is a problem crucial to solve for a wide range of modern problems. In autonomous driving, for instance, an autonomous car has to manage to drive in a system designed for humans.
Since the environment of a street is highly dynamic and subject to often unpredictable changes, the car must not solely rely on its GPS sensors in combination with maps to know the current speed limits or similar.

Simply put, for a autonomous car to be able to react to such rapid changes in the environment (eg. road works, accidents, etc), it must acquire information from road signs. Such a problem can be interpreted as a classical machine learning problem on images, in which a computer has to determine to which class an image belongs. The german road sign recognition challenge (GRSRC) tries to solve that problem by supplying a training dataset from which a computer can learn the structure of images belonging to each of the classes.

Deep neural networks have become the state-of-the-art solution to extract meaning from computer images or videos . They have been proven to achieve maximum accuracy when trying to classify such material into different classes while being able to

cite

2 Neural Networks & Convolutional Neural Networks

Artificial neural networks have become the state-of-the-art machine learning system for applications in computer vision, natural language processing and robotics. [1,6,7] They can be seen as an ensemble of different linear classifiers that are put together to solve a non-linear problem. While each of the individual classifiers only solves a linear sub-problem, the network is able to solve highly complex classification and regression tasks.
In a more formal way, we can define a fully connected feed forward neural network as a directed and acyclic graph in which the nodes are arranged in layers. Nodes within one layer have no connections to each other but every node from layer i is connected to all nodes in layer $i + 1$ and $i - 1$.
The connections between nodes have weights associated with them. Nodes

can also be called neurons, or units. To classify a data vector, it is clamped to

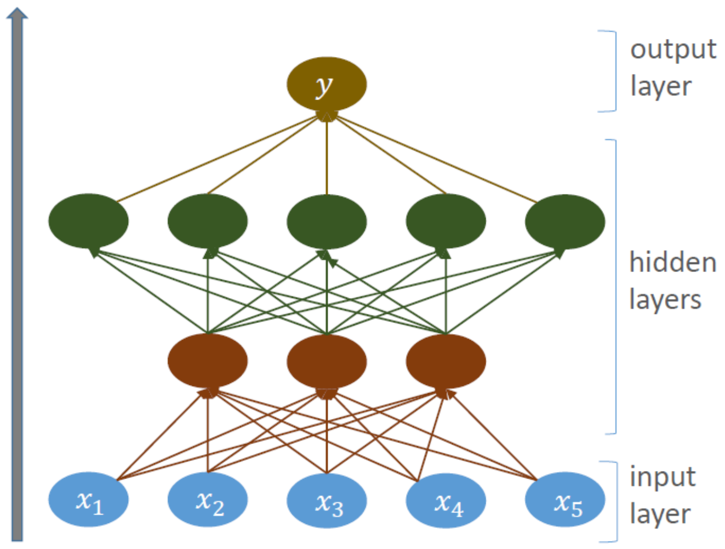

Figure 1: A schematic view of a fully connected feed forward network. The 5-dimensional input vector is first forced to be represented by only three dimensions (red layer). This reduced representation is then seen by the next layer (green layer) which tries yet again to find another representation of the input data.Finally, a one-dimensional output classifies the input vector.

the lowest layer of the network and then propagated upwards. This way, each neuron can be viewed as a linear classifier that takes a certain responsibility in the search space. In image recognition, a neuron might be responsible for detecting horizontal edges in an image, while a neuron further up the network might be responsible for deciding whether there is a cat in the image or not. Feed forward neural networks are a supervised method, which means they need labels for input vectors in order to learn from the data. When a new training vector is shown to the network, the information is first propagated up and compared to the expected output. From there, an error is calculated and its derivative with respect to the weights connecting units makes the network minimize the classification error.
Calculating the partial derivative of the error function with respect to one of the weights is an iterative process. First, the derivatives for the deepest layer are calculated and from there it becomes possible to calculate them for the second-deepest one and so one. This top-down approach is why the algorithm is called backpropagation, because it starts at the back and propagates the partial derivatives back to the front.

2.1 Training Neural Networks

The backpropagation algorithm tries to answer the question, in how far each of the neurons are responsible for the error that was made by the network as a whole. That means, when we propagate the error up to the output layer, we then first calculate the output's error, using the *squared error* measure.

$$E = \frac{1}{2}\sum_{i=1}^{k}(y_i - t_i)^2$$

This error is then split up between all neurons in a top-down fashion. The key principle is here to calculate the error's partial derivatives with respect to the weights that connect that two neurons. We start by calculating the derivative for the weights connecting the output layer with the second-last layer. These can be calculated easily with the *chain rule*. From that point, it becomes possible to calculate the derivatives for the weights connecting the second-last with the third-last layer and so on. Finally, we arrive at the input layer and the backpropagation algorithm terminates for the given input. When we repeat this procedure for all data points, the network will slowly start to learn to minimize the error.
We will assume that the neural network at hand uses the *sigmoid activation function* which is given by:

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

This function is particularly handy because of its very easy derivative. We can write:

$$sig'(x) = sig(x)(1 - sig(x))$$

So the question that asks in how far one unit is responsible for the error of the whole network becomes a question of partial derivatives with respect to the connections between layers (the weights). The *chain rule* states how the derivative for functions that were applied one after the other can be written, using only the single functions. More precisely, the chain rule states:

$$(f \circ g)' = (f' \circ g) \cdot g' \tag{1}$$

When we consider the neurons as functions that are applied one after the other, we can finally calculate the partial derivatives.
During the forward step, the output and input of each neuron is stored. Let o_i^k be the output of neuron i in layer k and h_j^{k+1} be the input for neuron j in layer $(k+1)$. These values are known from the forward pass and can be used to calculate the partial derivative with respect to the weights.

$$\frac{\partial E}{\partial w_{ij}^k} = o_i^k \delta_j^{(k+1)}$$

where δ_j^k is given by:

$$\delta_j^k = \begin{cases} f'(h_j^k)(t_j - o_j^k), & \text{for output layer.} \\ f'(h_j^k)(\sum_{i \in incoming} w_{ji}\delta_i^{(k+1)}, & \text{for inner layers.} \end{cases}$$

The backpropagation algorithm has the tendency to find useful representa-

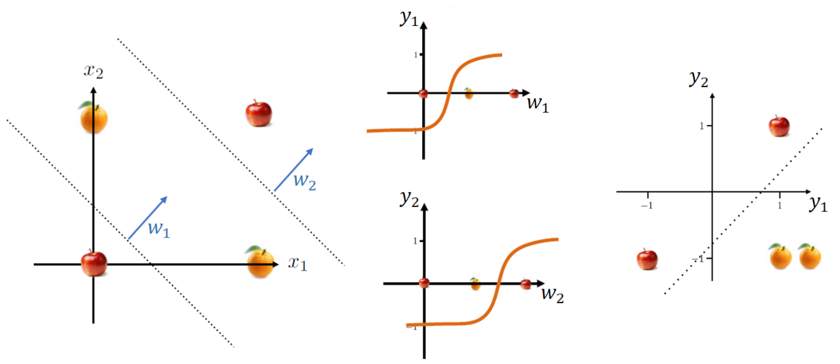

Figure 2: Example of how a neural network can solve the XOR problem. The first layer of the network only finds good data representations that serve as input for the next layer. There, the problem is suddenly linearly separable, and the problem as a whole can be solved.

tions of the data in the first layers, as is depicted in figure 2. This property yields in a high interpretability of neural networks despite their complexity.

However, neural networks have also some disadvantages. Because of the large amounts of parameters, the classical fully connected feed forward network requires huge amounts of training data for successful learning. Overfitting can occur easily and does so frequently. Furthermore, there is no guarantee that the solution found by the network is optimal in any way. And finally, applying these kinds of networks to images is usually not feasible due to the huge amount of parameters and the very structure of an image. When there is the same pattern of pixels somewhere in the image, we usually want it to be recognized as such, no matter where the pattern is located in the image. But imagine a neural network that is trained to see some feature (an apple, for instance) in the upper left corner. The network would be unable to recognize the very same apple located in the lower right corner of the image when this image was not part of the training set.

To overcome these problems, LeCun et al. proposed the convolutional neural network in [2], which resolves most of the problems in the field of computer vision applications.

2.2 Convolutional Neural Networks

The convolutional neural network is a variant of the conventional neural network in which the connections between two units are convolutions instead of simple connections. The convolution operation is very complex and has different applications in different scientific fields but it is worth noting that convolutions can be regarded as filters or kernels applied to images.
With this informal definition, we can define the weights connecting the input layer with the first one as filters. Their application to the image (the input layer) yields in the first hidden layer of the network. When we have a two-dimensional image of $x \times y$ input neurons, the first hidden layer typically is three-dimensional. This is because we apply many filters to the image, each one of them producing a two-dimensional matrix.
When we now look at the mathematical side of it, we only have to change the way gradients are computed from the error. Since the connection between layers is a convolution, we can also express the gradient as convolutions.

This approach solves the translation invariance issue because a filter will be applied everywhere in the image. Furthermore, the number of parameters is drastically reduced. When we use k different kernels, each of size $x_k \times y_k$, we have $k \cdot (x_k \cdot y_k)$ parameters for the convolutional layer. Typically, x_k and y_k are small and 20-30 kernels are often enough to describe the structure of an image.
For applications in image recognition or computer vision, the interpretabil-

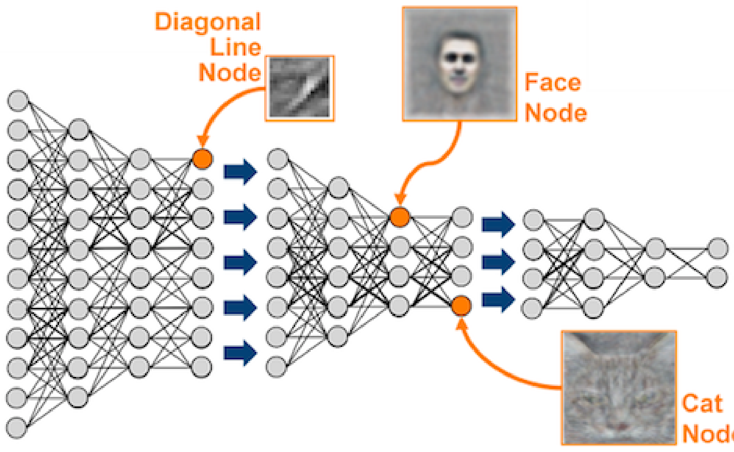

Figure 3: Schematic view of a deep convolutional neural network. The first layers capture low-level features like edge-detectors in the images. Later stages of the network capture more complex features like cats or faces. Source: [?]

ity is increased when using convolutional neural networks because the kernels have an interpretation that is known already. The kernels usually correspond to stereotypical mini-images like the cat or the face in figure 3.

3 German Road Sign Detection

To train a

References

[1] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[2] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[3] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[4] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

[5] Paul Werbos. Beyond regression: New tools for prediction and analysis in the behavioral sciences. 1974.

[6] Simon X Yang and Max Meng. An efficient neural network approach to dynamic robot motion planning. *Neural Networks*, 13(2):143–148, 2000.

[7] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.