Universität Bamberg
Angewandte Informatik

Seminar KI: gestern, heute, morgen

Understanding Convolutional Neuronal Network in Computer Vision - GoogLeNet's architecture

von

Gabriel Martin Nikol

25. Februar 2016

betreut von
Prof. Dr. Ute Schmid, Prof. Dr. Diedrich Wolter

1 Motivation

In the field of computer vision - object recognition and categorization is a very interesting task. The potential use in a wide variation of fields like: self driving cars, image categorization, intrusion detection system for security, user availability in human computer interaction (Fetter et al. (2011)) and countless more shows the need for research in this topic. The visual recognition of objects is an easy task for humans but a difficult task in computer science [1]. As a human being we learned from the first day how our environment is build and recognize it. We learn how to categorize our senses. To take one of our senses the visual system, we receive photons on the retina at the optical system and then the optic nerve forwards these information to our perceptual system. The perceptual system categorizes the information and categorizes the information contained in the image received by the optical system. The system is trained from the day we are born so a object recognition task seems to be easy for us. On the other side we can say a digital camera is an optical system for a computer. The computer with alls its hardware and software builds the perceptual System. Issues on the computer side are the large amount of data when using high scale and large images datasets, and in a software to extract information to recognize objects in the image. With faster CPU, cheaper RAM and using graphic cards the hardware challenge is still an issue but developing fitting software is still a even more challenging task. This is the reason why we take a look at the state of the art image object recognition approach GoogLeNet. In this work we focus on the understanding of the architectural components of GoogLeNet and the overall understanding of deep neuronal networks. Training of neuronal networks in detail is out of the scope of this paper. Additionally we donate a comprehensive notation for building deep neuronal networks. We start in section 2 with the introduction of the ImageNet Large Scale Visual Recognition Challenge 2014[2] to understand how GoogLeNet fits into the world of computer vision. Then we revisit the basic of neuronal network in section 3 needed and understand convolutional neuronal network. After the basics of neuronal networks we investigate the components of convolutional neuronal networks in section 4 by take a look at a typical example LeNet-5[3]. In section 5 we finally inspect GoogLeNet's component the inception module and its architecture.

2 ImageNet Large Scale Visual Recognition Challenge 2014

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC2014) is annual competition and challenge since 2010. It is also the de facto standard benchmark in object category classification and detection tasks. Among the participants are institutions like universities, companies and other organizations. The institutions taking part in state-of-the-art computer vision challenge comparing each contestant's approaches and humans as an contestant. The competition is to label given objects classes in three different

[1]`http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.0040027`
[2]`http://www.image-net.org/challenges/LSVRC/2014/`
[3]`http://yann.lecun.com/exdb/lenet/`

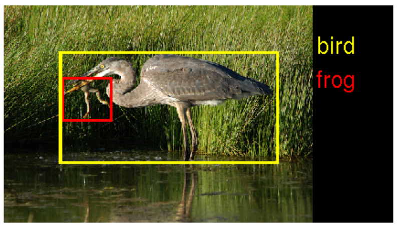

Figure 1: Example Image from ILSVRC2014 with bounding boxes for each object
Source: `http://image-net.org/challenges/LSVRC/2014/index#data`

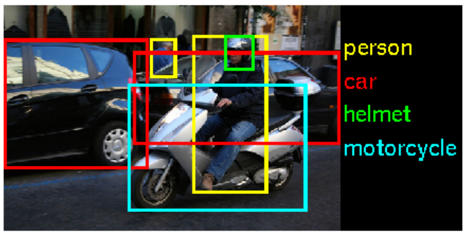

Figure 2: Another example Image from ILSVRC2014 with bounding boxes for each objects
Source: `http://image-net.org/challenges/LSVRC/2014/index#data`

tasks to images (Russakovsky et al. (2015)). We will now have a look at the dataset and the challenge tasks.

2.1 ILSVRC Dataset

Each image in the datasets are photographs like shown in Figure 1 and Figure 2 but without the bounding boxes. The ILSVRC releases for all participants two public dataset (a detection and classification and a localization dataset) each containing a set of manually annotated *training* images and a set of *test* images. The *training* images are to train the approaches and the test images used in the competition. The *test* images contain no annotation when given to the participant but are manually annotated for the evaluation of the competition. The detection dataset has so-called image-level annotations labeling the present or absence of an object class in an image. The detection dataset contains 200 different object classes, 456567 training and 40152 testing images. The classification and localization dataset contains so called object-level annotation giving a bounding box and a class label around an object instance in an image. The dataset contains 1000 different categories, 1.2 million of training and 150 000 test images[4] (Russakovsky et al. (2015).

2.2 Challenge Tasks

There are two challenge tasks. The first one is the image classification task using the detection dataset. The algorithm shall produce a bounding box and confidence score in a given image for each object class The confidence score tells how sure the algorithm is to detect the object class in the image at the given bounding box. The second one is the object detection task using the classification and localization dataset. The algorithm shall produce 5 class labels in decreasing order of the confidence score including a bounding box for each label and the class label is one of the 1000 different categories[5](Russakovsky

[4]`http://image-net.org/challenges/LSVRC/2014/index#data`
[5]`ttp://image-net.org/challenges/LSVRC/2014/index#data`

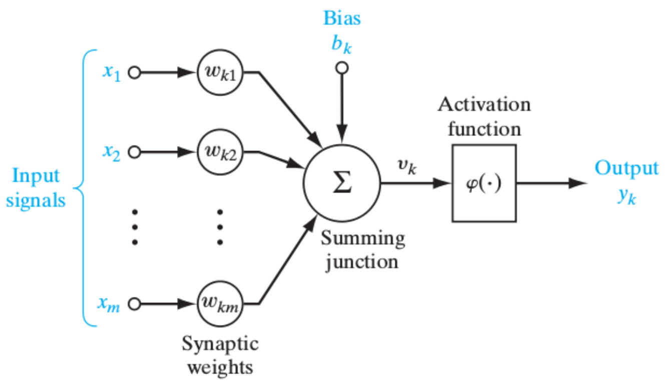

Figure 3: Nonlinear model of a neuron labeled k
Source: Haykin (2009) page 11

et al. (2015)). The GoogLeNet shown in this paper focuses on the classification part of the object detection task.

2.3 Different approaches leading to CNNs

In the first two years ILSVRC (2010 and 2011) different supporting vector machines (SVM) with scale-invariant feature transform (SIFT) features reached in the challenge in the image classification task a single-object localization error (CLS) of 25.8% and in the object detection task a localization error (LOC) of 42.5% (Russakovsky et al. (2015)). The turning point was in 2012 when large, deep convolutional neural network entered the challenge and decrease the CLS to 16.4 % and the LOC to 34.2%. In 2014 the best CLS was 6.7 % from the GoogLeNet and reached the second place with an LOC of 26.4 %, The first place in LOC had only 25.3%. Overall the GoogLeNet was the winning image classifier in the ILSVRC2014 (Russakovsky et al. (2015)).

3 Basics of Neuronal Networks

To understand the GoogLeNet later we first have to take a look at the basics. In this section we will see the contents of neuronal networks namely neurons, activation function, layer types. In this section we only take a look at the general layer types. Layer types specialized for convolutional neuronal networks will be described later.

3.1 Neuron

The neuron is the processing unit and the fundamental part of a neuronal network.

We can see a nonlinear neuron model labeled k in Figure 3. We will now describe the different components from left to right. The input into a neuron consists of a set of connecting links (input signals) $(x_1, x_2, \dots, x_m)$ where each link has a weight (synaptic weights) $(w_k1, w_k2, \dots, w_km)$. In most cases inputs are normalized in a range of $[0, 1]$.

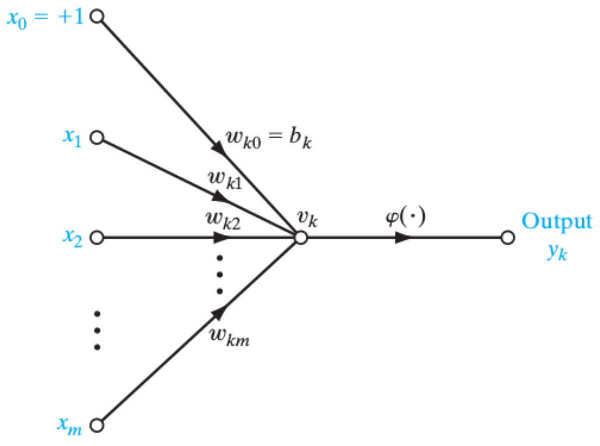

Figure 4: Singal-flow graph of a neuron labeled k
Source: Haykin (2009) page 17

Then as an adder (summing junction) sums the input signals with referring to their weights. Normally its a linear combiner like $u_k = \sum_{j=1}^{m} w_{kj} \cdot x_j$. Also an external bias is applied called b_k increasing or lowering the input signal. Last an activation function φ is applied to u_k generating the output y_k of the neuron k. The activation function can be written as $y_k = \varphi(u_k + b_k)$ to take care of the bias. For the understanding GoogLeNet the bias is not crucial so we will not include it in further descriptions.

3.2 Neuronal Network as Directed Graphs

In general neuronal networks are directed graphs. The inputs $(x_1, x_2, \ldots, x_m)$ are the source and the signal-flow goes from the connection input through the neurons function to the output. The output $y_k$is used in another neuron as an input. In Figure 4 the flow of the data is visualized by the arrow. When a neuron is seen as a node in a graph, we can connect neurons and build a directed graph. This is shown in Figure 6.

3.3 Rectified linear activation function ReLU

We now take a look at the activation function we need in this paper. It is only the so called rectified linear activation. There are also other activation function like threshold, sigmoid functions, softplus and many more, but these are not used in GoogLeNet and therefore not interesting.

The rectified linear activation function is a simple activation function dropping negative inputs to zero. It is also often called rectified linear unit and can be written as $ReLU(u_k) = max(0, u_k)$ (Glorot et al. (2011)). Figure 5 shows the plot of ReLu function (red) over $[-3, 0, 3]$. Negatives values are zero and positives values keep their value.

The advantages of the function is that we easily obtain a sparse representation and less connection links between layer when initializing a network with standard normal distribution because 50% of the connection links are lower than zero and the neuron

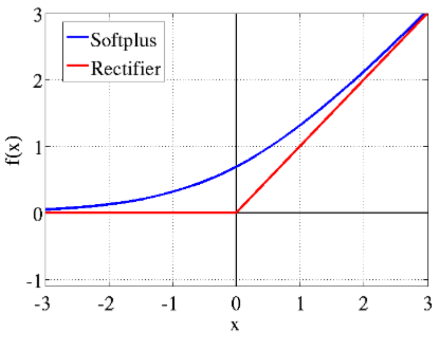

Figure 5: Plot of rectified linear activation function (red)
Source: Glorot et al. (2011)

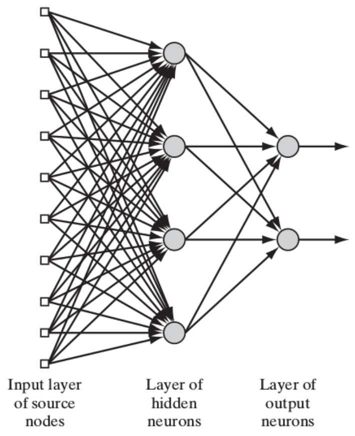

Figure 6: Fully connected 10-4-2 feedforward network with one hidden layer and one output layer.
Source: Haykin (2009) page 22

will not have any impact in the network (preventing overfitting) (Szegedy et al. (2014), Glorot et al. (2011)).

3.4 Multilayer feed forward Networks

There are several network architectures, the one we use is the convolutional neuronal network which is basically a multilayer feed forward network. Multilayer feed forward networks consist of an input layer which is in our case an image, at least one hidden layer which is not seen (hidden) by a potential user who only puts images into the network's input layer and obtain the output. And after some hidden layers there is a output layer which shall represent the result of the network (Haykin (2009)). For an example in Figure 6 we can see a 10-4-2 network witht a input layer of the size $1 \times 1 \times 10$, a hidden layer of the size $1 \times 1 \times 4$ and a output layer of the size $1 \times 1 \times 2$. All layers are fully connected.

We can build also a $m - h_1 - h_2 - \ldots - h_l - q$ network with a input layer m, $(h1, h2, \ldots, h_l)$ hidden layers and the output layer q (Haykin (2009)).

3.5 Layers

There are a large amount of different layer types. Now we will see how layers are connected and how the different layers are put together to so called feature maps. We only take a look at the basic layers namely fully connected, softmax layer, local response normalization here.

3.5.1 Layer sizes and Connections

Layers can be seen as a 2-dimensional matrix of neurons of the size $X \times Y$. Layers have "height" and "width" defined by X and Y. More layers of the same type are often put together we call this layer map and these layer maps can be seen as a 3 dimensional matrix $Z \times X \times Y$ where Z indicates the count of layer put together. We use the name layer also for a layer map fom now on. This notation is inspired by Krizhevsky et al. (2012).

3.5.2 Fully Connected Layer

So without any transformation we can fully connect a layer l_1 of the size $X \times Y$ to the previous layer l_2 of the size $X \times Y$. When we have a layer map l_1 with the size of $Z \times X \times Y$ then we can fully connect a layer map l_2 with the same size by connecting layer $Z_i \times X \times Y$ in l_1 with $Z_j \times X \times Y$ in l_2 were $i = j$. We can connect all layers $(i = j = 0, i = j = 1, \ldots, Z)$ with each other between l_1 and l_2 too. We can observe a fully connected layers in Figure 6 between the input and the hidden layer, and the hidden and output layer. In following networks let $(FC_1, FC_2, \ldots, FC_l)$ be the fully connected layer from the first layer FC_1 feed by the network to the last convolution layer FC_l.

3.6 Images as input Layer

In most cases the image is a grey-scale or a RGB image. For a grey-image the input layer has a size of $1 \times X \times Y$ where the 1 at the beginning represents the abscence of colours (grey-scale) and the X and Y determines the size of the image. The grey-scale values have to be normalized between 0 and 1 to fit better in the networks. When the image is RGB like in ILSVRC2014 the size of the input layer is $3 \times X \times Y$ where the 3 represents the red, green and blue part. So we have a layers for both color spaces.(Krizhevsky et al. (2012)).

3.7 Vectors as output layer

The output layer is a layer with the size $1 \times 1 \times Y$ where Y is the number of classes we want to distinguish. A confidence score between 0 and 1 is assigned to each class. In

ILSVRC2014 in Task 1: $Y = 200$ and in Task 2 $Y = 1000$. The output is produce by a trainable classifier like the softmax-layer (Lecun et al. (1998), Szegedy et al. (2014)).

3.8 Local response Normalization Layer

The local response normalization layer has no specific size because it is the same as the previous layer. It applies a normalization function to its input. The local response activity is given by the expression $b^i_{x,y} = \frac{a^i_{x,y}}{(k+\alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} (a^i_{x,y})^2)^\beta}$ where $a^i_{x,y}$ is the activity (output) of a neuron by applying kernel i at position (x,y) and then applying the ReLU activation. The sum runs over n kernel maps and N is the total number of kernels in the layer (Krizhevsky et al. (2012)). The order of the kernel maps is arbitrary. This layer implements a form of lateral inhibition like in real neurons. The constants k, n, alpha and beta are globally set and determined by using validation sets[6] (Krizhevsky et al. (2012)). In following networks let $(LNorm_1, LNorm_2, \ldots, LNorm_l)$ be the local response normalization layer from the first layer $LNorm_1$ feed by the network to the last layer $LNorm_l$.

3.9 Softmax Layer

The softmax layer is the multi-class classifier used in GoogLeNet. Classifier are trained at the end of a network providing the output of the network. The softmax is given by $o(z)_i = \frac{e^{zj}}{\sum_{k=1}^{K} e^{zj}}$ for $j = 1, \ldots, K$ where $o(z)_i$ is a vector of the size K (or the output layer of the size $1 \times 1 \times K$). The input is a vector e of the size K. The input in a neuronal network is in most common cases a fully connected layer of the size $1 \times 1 \times K$. The softmax layer produces the confidence score between 0 and 1 for ILSVRC Tasks. In following networks let $(SoftMax_1, SoftMax_2, \ldots, SoftMax_l)$ be the softmax layer from the first layer $SoftMax_1$ feed by the network to the last layer $SoftMax_l$.

3.10 Deep Networks

In general deep means more than one stage of non-linear feature transformation. The features come from the learned weights in the hidden layer. The t-th feature can be seen in the h_n layer in the $t \times X \times Y$ feature map t. Additionally a standard architecture with $m - h_1 - h_2 - \ldots - h_l - q$ layers is used. These networks take in CV as input layer m an image and as q a trainable classifier like the softmax-layer. The hidden layers $(h_1, h_2, \ldots, h_l)$ will extract features after the network is trained properly. The lower layer (e.g. h_1, and h_2) will extract low-level features in the image, where layers in the middle $(h_{l/2})$ will extract mid-level features base on the low-level features. Hidden layers at the end will aggregate the mid-level features to high-level features and feed the classifier (softmax-layer) with these information (Arora et al. (2013a), Arora et al. (2013b)).

[6] `http://caffe.berkeleyvision.org/tutorial/layers.html#data-layers`

3.11 Issues with Fully Connected Deep Networks

There exist several issues with fully connected deep networks which is the reason CNNs are used nowadays. The major problem is the huge amount of parameters to train. For a small example a network with only three hidden layer h_1, h_2, h_3 where each hidden layer is of the size $1 \times 1 \times D$ has $3D^2$ connections. For grey-scale images of the size $D = 32 \times 32 = 32^2$ the hidden layers have $3(32^2)^2 = 3145728$ parameters to train. The result is a huge amount of time we need to train the network and additionally we need a huge amount of training images. Furthermore these networks are very prone to over-fitting and are hard to initialize. This leads to the need of layers and network architectures which can extract and aggregate features with less parameters.

4 Basics of Conventional Neuronal networks

To overcome the issues described in 3.11 convolutional neuronal networks seem to fit the gap (Haykin (2009)). A CNN is a special case of a multilayer feed forward network and is well suited for classification tasks and pattern recognition (Lecun et al. (1998), Szegedy et al. (2014), Krizhevsky et al. (2012)). In ILSVRC2012 ImageNet participated the challenge and showed how well suited CNNs are in the computer vision (Krizhevsky et al. (2012)). CNNs are also deep networks because the extract local features and enable a architecture to stack feature detectors on each other. In this section we take a look at the different components of a CNNS namely convolution and subsampling layer and one of the first CNN networks LeNet-5 describing the standard architecture and buildings blocks (Lecun et al. (1998)).

4.1 Convolution Layer

A convolution layer is local feature detector applying a learn able filter (called kernel) to the previous layer[7]. Formally we have a previous layer p with the size $X_p \times Y_p$ and the (follwing) layer c with the size $X_c \times Y_c$. p has to be larger than $c:$ $X_p > X_c$ and $Y_p > Y_c$. The connections between the layers are based on a patch of a given size $X_k \times Y_k dz$ with $X_k = Y_k$ which slides over p with a stride of s. A kernel k of the size $X_k \times Y_k$ (same size as the patch) is calculated with the values of the patch. This is the train able filter. A single neuron k in the layer represents a single feature. The patch are the connection inputs for a neuron (see 3.1). The kernel contains the weights 3.1 of the connections. The patch and the kernel together restricts the amount of parameter to be trained drastically compared to a fully connected neuron (and thus to layers). Afterwards the result of the kernel is summed by the summing junction like desribed in 3.1 and put into the activation function (which is in GoogLeNet the ReLu function described in 3.3) calculating the output of the neuron. Another result is that a convolution layers size depend on the filter size (patch/kernel) and the stride. A convolution layer learns$K$ local features where K is the number of patches (and the size of the convolution layer). A layer

[7] `ttp://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution`

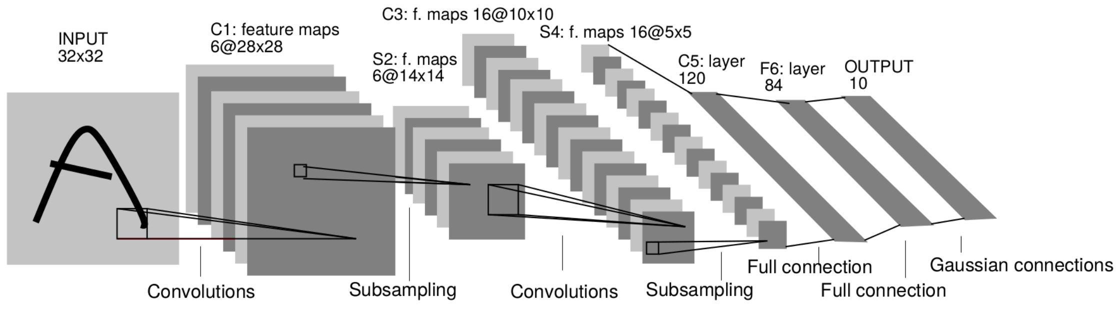

Figure 7: Architecture of LeNet-5, a CNN for digits recognition. Each plane represents a feature map.
Source: Lecun et al. (1998)

map (feature map) for convolution layer consist of several layers each has $Z \times X \times \times Y$ local features.

In following networks let $(C_1, C_2, \ldots, C_l)$ be the convolution layer from the first layer C_1 feed by the network to the last layer C_l (Lecun et al. (1998)).

4.2 Subsampling Layer

After a convolution layer normally a subsampling layer (also called pooling)is applied. The subsampling operation further reduces the number of parameters to train. Additionally the subsampling layer has no trainable weight, every weight is 1. The subsampling layer can be seen as a convolution layer which is not trained, every weight is fix with the value 1. Additionally the neurons in subsampling layer do not apply a summing junction or activation function (**?**, Krizhevsky et al. (2012),Szegedy et al. (2014)). The junction function is replaced either by a maximum function or mean-average function over the patch (Lecun et al. (1998)). The activation function of these neurons is replaced by a the identity ($identity(v_k) = v_k$). This also means similar to convolution layers the size of the subsampling layer is give by the patch size and stride. In following networks let $(Max_1, Max_2, \ldots, Max_l)$ be the subsampling or pooling layer applying the maximum function from the first layer Max_1 feed by the network to the last layer Max_l. And for following networks let $(Avg_1, Avg_2, \ldots, Avg_l)$ be the subsampling or pooling layer applying the average function from the first layer Avg_1 feed by the network to the last layer Avg_l (Lecun et al. (1998)).

4.3 Standard Convolutional Neuronal Network Architecture

LeNet-5 introduced a CNN for digit recognition. It provides an example of a standard architecture for CNNs. It consist of a input layer for the image, one or more convolution layer and a subsampling layer (both together) and at the end of the network are several fully connected layers with a trainable classifier at the end production the output vector. As an example we let us take a look at LeNet-5 seen in Figure 7. The CNN starts with an input layer of the size 1x32x32 because it works with grey-scale images. Then a convolution layer follows and extracts local low-level features out of the image. To reduce the size of the count of parameters a subsampling layer follows the convolution layer. One combination of convolution and subsampling layer is called a CNN building block. The building block can be repeated countlessly of times and is used to extract higher levels based on the output of underlying CNN building blocks. The network then is followed by a smaller convolution and subsampling layers extracting higher level features. The transformation from the subsampling layer S4 with the size of $16 \times 5\ times 5$ to the first fully connected layer C5 with the size $1 \times 1 \times 120$ is made by convolution, too. Then after C5 a standard fully connected layer F6 follows and feeds the output layer by a Gaussian connection (which is the classifier in LeNet-5). We skip having a deeper look into LeNet-5 because only the architecture is necessary to understand GoogLeNet.

In the notation introduced in 3.4 LeNet-5 is $m - C_1 - Max_2 - C_3 - Max_4 - C_5 - FC5 - FC6(= G) - q$ where m is INPUT with a size of 1x32x32, q is OUTPUT 1x1x10 and $FC6$ indicates the Gaussian connection (Lecun et al. (1998)).

4.4 An Enhanced Notation for Layers

For representing large scale neuronal network a better notation has to be introduced. It shall take care of the parameters of the layer and its position in the network.

We now enhance the notation, writing for the input layers $I_{z,x,y}$ were z defines the colour space (1 for grey-scale and 3 for RGB images) and x and y the size of the image. The output layer will be written as O_s as beeing unique like the input layer.

For convolution layer we write $C_{n,f,p,s}$ for the n-th layer in the network, with the feature size f, for the patch sizes p were $X_p = Y_p = p$ and the stride $s \in N$.

For subsampling layer we write $Max_{n,p,s}$ or $Avg_{n,p,s}$ for the n-th layer in the network with the patch sizes p were $X_p = Y_p = p$ and the stride $s \in N$.

For fully connected layer we compress the notation by writing $FC_{n,v}$ for the n-th layer in the network with the size $1 \times 1 \times Y_v$ and $Y_v = v \in N$.

The softmax layer will be written as $S_{n,s}$ similiar to the new fully connected layer notation.

We now can rewrite LeNet-5 very comprehensively: $I_{1,32,32} - C_{1,6,5,1} - S_{2,2} - C_{3,16,5,1} - S_{4,2} - C_{5,1,5,1} - F_{6,84} - O(= G)_{10}$

The only disadvantage in this notation is that a specific connection established between $\ldots - S_{2,2} - C_{3,16,5,1} - \ldots$ cannot be represented. This is only relevant in LeNet-5 but not in GoogLeNet due to of initialization advantage of ReLu described in 3.3 which results in no specific hand wired connections between layers made in LeNet-5 (Lecun et al. (1998), Szegedy et al. (2014)).

5 GoogLeNet

Now we take a look at GoogLeNet. It is a deep network with more then 100 layers. Thus it introduced a so-called inception module consisting of convolution and subsampling

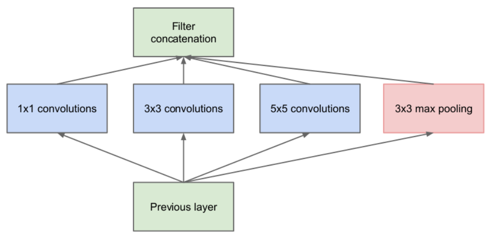

Figure 8: Inception module, naive version
Source: Szegedy et al. (2014)

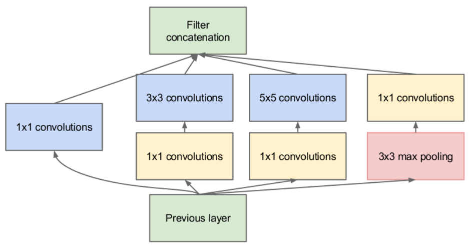

Figure 9: Inception module with dimensionality reduction
Source: Szegedy et al. (2014)

layers. In this section we start with the inception module followed by the architectural overview using our enhanced the notation describing GoogLeNet (Szegedy et al. (2014)).

5.1 Inception Module

GoogLeNet introduces so-called inception Module. The idea of a module goes back to the idea of network in networks as a way to stack neuronal networks (in this case an inception module) on each other. The inception modules design comes from the Hebbian principle saying that neurons firing together are connected together. In images this means feature correlations tend to be local (Szegedy et al. (2014)). There exist two versions of the inception module. A naive version representing the idea and a version used in the network with dimensionality reduction (and reducing the computation blow up).

The naive version can be seen in figure 8. From the bottom to the top we can see previous layer connected to 3 convolution layer. The layers have a patch size of 1x1, 2x2 and 3x3 with a stride of 1. These three layers reperesent the handling of the Hebbian principle. The 1x1 patch size convolution layer learns the most local filter. The 3x3 patch size convolution layer learns less feateures but more spread out over the image . The 5x5 patch size convolution layer learns least features but the most spread out clusters. Additionally subsampling layers had an essential success for convolution networks in LeNet-5 and ImageNet resulting in adding a 3x3 maximum subsampling layer to the naive version (Lecun et al. (1998), Krizhevsky et al. (2012)). The three convolution layers and the subsampling layer are feed by the previous layer and are concatenated in the filter concatenation layer. The inception modules can be stacked in two different ways. First when the filter concatenation layer in the first inception module is the previous layer in the second inception module. Second when the concatenation of the first inception module is connected to the previous layer of the second module. In all inception modules layers (expect the maximum subsampling layer) the rectified linear activation function is used like mentioned earlier. The dimensionality reduction inception module is build by the idea of reducing dimension wherever the computational requirements would increase to much. To have less connection sparse representation in most places are desired. compression of signals is desired when features have to be aggregated en mass (Szegedy et al. (2014)). This leads to adding a concolution layer with the patch sie of 1×1 and a stride of 1, before the computational expense 3x3 and 5x5 patch convolution layer and behind the 3x3 patch size max subsampling layer (Szegedy et al. (2014)). We can see the finished result at Figure 9.

For the enhanced notation we define two different notations for the inception modules used by GoogLeNet. One for each of the stacking methods. For the first version of stacking let $IM_{n,f}$ where n is the number of the first inception's modules layer, with the feature size of f. For the second version let $IM_{n,f,L}$ where n is the number of the first inception module's layer, with the feature size of f. Let L be a any layer (in GoogLeNet maximum-subsampling layer). For example we can write $IM_{4,256,Max_{4,3,2}}$ where the first layer of the inception module is the maximum pooling layer beeing the 4-th layer in the network. Due to the fact that in GooGLeNet every inception module's concatenation layer is a depth concatenation we implicitly assume this in is in both notations applied. As concatenation layer implementation a so called DepthConcat operation was used which concatenates the convolution layer to a feature/layer map of the same size as the previous layer[8](Szegedy et al. (2014)).

5.2 Architectural Overview

After defining the enhanced and comprehensive notation we can write down the GoogLeNet as follows:

$Input_{3,224,244} - C_{0,64,7,2} - Max_{1,3,2} - LNorm_2 - C_{3,64,1,1} - C_{4,192,3,1} - LNorm_5 - IM_{6,256,Max_{6,3,2}} - IM_{18,480} - IM_{31,480,Max_{24,3,2}} - IM_{43,512} - IM_{55,512} - IM_{67,512} - IM_{79,832} - IM_{92,832,Max_{92,3,2}} - IM_{104,1024} - Avg_{117,7,1} - FC_{118,1000} - SoftMax_{119,1000} - Output_{1000}$

Inspectioning the layout of the network we can see the basic architecture of LeNet-5 is represented in the combination of Convoltional and maximum subsampling layer in layer 0 to 5. Afterwards 9 Inceptions modules are stacked on top of each other. The first inception module has a size of $256 \times 28 \times 28$. The number of features increase (higher level features) with higher inception module numbers. To prevent an explosion of parameter by wiring too many fully connected layers at the end GoogLeNet uses more inception modules and has only 1 fully connected layer of the size $1 \times 1 \times 1000$ at the end. Additionally the filly connected layer can be used to make the network multi-purpose by exchanging the classifier. The network is finished by a softmax layer as classifier (Szegedy et al. (2014)) .

5.3 Training

Until now we left training out of the scope of this paper. But for a complete view on GoogLeNet at least the basic data of training are mentioned here. The training method was asynchronous stochastic gradient descent, with a momentum of 0.9 like suggested by Sutskever et al. (2013). This also means the network is initialized suggested by Sutskever et al. (2013) meaning that they use a sparse initialization technique from Sutskever et al. (2013). A "fixed learning rate schedule decreasing the rate by 4% every 8 epochs was

[8]`http://stats.stackexchange.com/questions/184823/how-does-the-depthconcat-operation-in-going-deeper-with-c`

used"(Szegedy et al. (2014)). During training before $FC_{118,1000}$ a dropout later with 40% was inserted to prevent the network from over-fitting. The ILSVRC changed image sampling methods during the competition so Szegedy et al. (2014) says "it is hard to give a definitive guidance to the best way" to train this large network. (Szegedy et al. (2014))

6 Evaluation of GoogLeNet in ILSVRC2014

The goal GoogLeNet reached in ILSVRC2014 at the detection task a mean average precision (mAP) of 38.02% without any contextual model and without a bounding box regression (Szegedy et al. (2014)) resulting in the 1st place in this task. The classification task was performed with an CLS of 6.7% like already mentioned in section 2.3 resulting in the 1st place in this task, too. To support a comparison between the approaches of the participants Russakovsky et al. (2015) an interface was built where humans can label and annotate images. Then the performance a comparison of two humans, Annotater A1 and Annotater A2 against the network. An extra dataset of only 1500 test images out of the full set was chosen. Annotater A1 label 1352 images and Annotater A2 219 images. In Annotater A1 labed dataset GoogLeNet reached an error of 6.8% and Annotater A1 5.1% with a statistically significant result. In Annotater A2 labed dataset GoogLeNet reached an error of 5.8% and Annotater A1 12.0% with a statistically significant result. This higher error value is a result of failing to spot and consider the ground truth label as an option by the annotator A2 (Russakovsky et al. (2015)).

7 Conclusion

This paper provides and overviews over the basics of neuronal networks the components used in GoogLeNet. Furthermore it shows the basic layout and idea of convolutional neuronal network. At the end GoogLeNet was shown as a large scale deep neuronal network including the new inception module. We saw the environment GoogLeNet was developed for especially the two image recognition tasks. Then we introduce a notation even show large scale networks in a very comprehensive way. For future work it can be investigated in detail how such large scale network like GoogLeNet can be trained. But leaving GoogLeNet aside the question arise of convolutional neuronal network are not only fitting into computer vision task but even in other areas like acoustic signal classification. It is based on the winner of the ILSCRC2012's[9] ImageNet, such a CNN was successfully applied to learn underwater bioacoustics, classifying different animal species[10][11][12]. By convering the sound patterns into spectrograms an acoustics task could be transformed into an image recognition task easily applying a standard convolutional network architecture very similar to ImageNet, detecting nearly 97% of whale

[9] http://www.image-net.org/challenges/LSVRC/2012/results.html
[10] http://danielnouri.org/notes/2014/01/10/using-deep-learning-to-listen-for-whales/
[11] http://www.wired.com/2013/02/wanted-right-whale-caller-id/
[12] https://www.kaggle.com/c/whale-detection-challenge/leaderboard/public

calls. This leads to the question if we can transform other signal data like data from weather stations, radar signals, wifi signals and other acoustic signal like 40kHz sent and received pings into an image recognition task and apply well known convolutional network architectures for classification tasks.

References

Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. 2013a. Provable Bounds for Learning Some Deep Representations. *CoRR* abs/1310.6343 (2013). http://arxiv.org/abs/1310.6343

Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. 2013b. Provable Bounds for Learning Some Deep Representations. *CoRR* abs/1310.6343 (2013). http://arxiv.org/abs/1310.6343

Mirko Fetter, Julian Seifert, and Tom Gross. 2011. Predicting selective availability for instant messaging. In *Human-Computer Interaction–INTERACT 2011*. Springer Berlin Heidelberg, 503–520.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep Sparse Rectifier Neural Networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, Geoffrey J. Gordon and David B. Dunson (Eds.), Vol. 15. Journal of Machine Learning Research - Workshop and Conference Proceedings, 315–323. http://www.jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf

S.S. Haykin. 2009. *Neural Networks and Learning Machines*. Number Bd. 10 in Neural networks and learning machines. Prentice Hall. https://books.google.de/books?id=K7P36lKzI_QC

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 1097–1105. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*. 2278–2324.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115, 3 (2015), 211–252. DOI: http://dx.doi.org/10.1007/s11263-015-0816-y

Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, Sanjoy Dasgupta and David Mcallester (Eds.), Vol. 28. JMLR Workshop and Conference Proceedings, 1139–1147. `http://jmlr.org/proceedings/papers/v28/sutskever13.pdf`

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2014. Going Deeper with Convolutions. *CoRR* abs/1409.4842 (2014). `http://arxiv.org/abs/1409.4842`