Fine-Grained Image Classification with Convolutional Neural Networks

Ike Sebastian Kunze

Zusammenfassung

The subject of this paper is fine-grained image classification with convolutional neural networks (CNNs), i.e. distinguishing between very similar classes of images. In this context, the general concept of CNNs will be explained first, before a network by Krizhevsky et al. [1], the AlexNet, is presented, as a lot of other approaches build on the architecture of this network. After that, two part-based approaches, the Part-Based R-CNN by Zhang et al. [2] and Ensemble of Localized Learned Features (ELLF) by Krause et al. [3], will be discussed and compared to the AlexNet, as they use other additional techniques to further enhance the performance of the CNN. Finally, it will be analyzed to what extent convolutional neural networks are currently used in medical image processing.

Keywords: CNN, R-CNN, ELLF, Medical Image Processing

1 Introduction

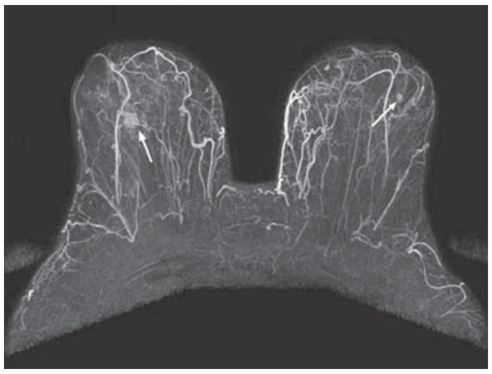
Fine-grained image classification is a difficult task, although this might not be obvious. In general, human beings can easily differentiate between certain categories (coarse-grained classification), as well as between corresponding subcategories (fine-grained classification), as long as one is familiar with the respective topic. For example, it should not be a problem, even for laypersons, to first identify an arbitrary finger as a finger (coarse-grained) and then to distinguish between the thumb and the index finger (fine-grained), as the index finger is longer and leaner than the thumb. However, differentiation in applied medical context is not as easy as this, e.g. when it comes to the interpretation of computed tomography (CT) or magnetic resonance imaging (MRI) images. One example for this is the detection of breast cancer in MRI images as shown in Figure 5.1. As can be seen there, healthy and cancerous regions within breasts can have subtle differences, thus making it hard to distinguish between them. Modern imaging techniques already facilitate this process by preprocessing the images in a beneficial way, e.g. increasing contrast within the image or hiding unimportant body layers. Nonetheless, mistakes may happen, as some radiologists have better expertise or more experience than others, thus making fewer mistakes, and as humans tend to make mistakes now and then in general. However, it would be desirable to reliably detect breast cancer, i.e. without making any mistakes, as such mistakes can be dangerous for human life.

This is where Computer Aided Detection (CADe) methods, i.e. computer-based image processing techniques, can help to decrease the number of errors. However, computers have not been able to achieve results anywhere near human performance for a long time.

Yet, latest progress has made it possible that computers now can classify images quite successfully, i.e. with relatively low rates of error [1]. This recent success is partly explained by technical progress, which accelerates the computations of the processing techniques and improves the quality of imaging techniques, thus yielding images of higher quality that are more suitable to further processing by computers. Apart from these improvements of the technical foundations, image processing techniques themselves have been extended, too, so that the newly created technical possibilities can be utilized efficiently.

One line of theoretical ideas has built upon convolutional neural networks (CNNs), which were first proposed by LeCun et al. [6] in 1989 and shall be the topic of this paper. First, the basic concepts of convolutional neural networks will be reviewed, because a well-founded knowledge of them is inevitable to understand further approaches. After that, the AlexNet, a CNN by Krizhevsky et al. [1], will be presented, as the architecture of this network is used as template for a lot of recent state-of-the-art approaches. Building up on that, two part-based approaches will be shown, as they augment the basic strategy with additional techniques to further improve the performance of the network. Finally, the applicability of convolutional neural networks to a medical context and the extent of current usage will be analyzed.

(a) The MRI image on the left shows a normal breast, whereas the image on the right shows a breast with potential cancerous regions. Source: [4]



(b) The MRI image shows two breasts, each having one potential cancerous region (arrows). Source: [5]

Abb. 5.1: MRI images of breast cancer

2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a special type of feed-forward artificial neural networks, which are used to process and classify images. Every feed-forward network consists of processing units called neurons which are organized in one input layer, one output layer and an arbitrary number of layers in between (as shown in Figure 5.2).

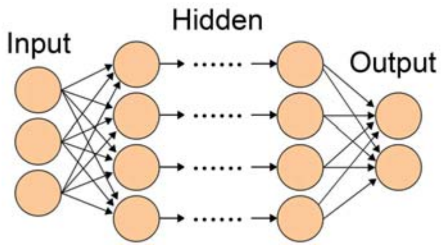


Abb. 5.2: This feed-forward neural network consists of one input, one output and several hidden layers.

Building up on these processing units, the networks compute a function on a presented input image to determine a corresponding classification. The function is adjusted during the training of the network in such a way that the computed classification becomes more and more equal to the correct classification, i.e. after some time, the training images should be classified correctly. After the training is completed, the function is fixed and, in the ideal case, suitable to correctly classify images that are not from the training dataset aswell.

As mentioned before, convolutional neural networks are one special type of feed-forward networks, which is why their concept will be described in more detail in the following. First, the processing units are characterized in Section 2.1, before different types of layers and their purposes are presented in Section 2.2. Last, the training of the network will be further elaborated in Section 2.3, i.e. it will be shown how networks can be adjusted properly.

2.1 Neurons

Neurons are the network's processing units typically having both incoming and outgoing connections to other neurons. They work on an input obtained from the incoming connections and complemented by an additional value, called bias. After applying a so-called activation function on the input, which is illustrated in Figure 5.3, the so generated output is passed along all outgoing connections.

In practice, non-linear functions are used as activation functions, which permits modeling of quite complex scenarios. Typical examples for such non-linearities are the saturating functions $f(x) = \frac{1}{1+e^{-x}}$

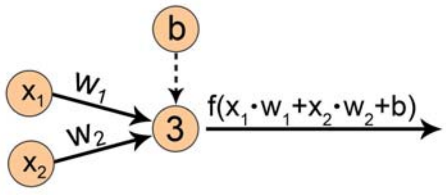


Abb. 5.3: Neuron 3 receives the outputs x_1 and x_2 of neurons 1 and 2 as input, weighted by w_1 and w_2, respectively. It then applies its function f on the weighted data, which is also complemented by the bias value and puts out the result along its outgoing connection.

(sigmoid function) and $f(x) = tanh(x)$ or the non-saturating function $f(x) = max(0, x)$, a very popular variant. Following Nair et al. [7], neurons with such an activation function are called Rectified Linear Units (ReLUs).

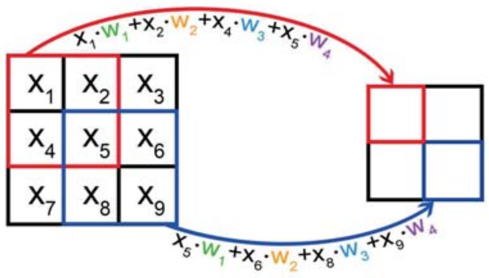
The interconnection between neurons of different layers is dependent on the layer they are in, as there are different types of layers with different properties. For example, in some layers the connections between neurons are weighted, like illustrated in Figure 5.3, whereas other layers do not make use of weights. Thus, the different types of layers shall be further explained in the now following Section 2.2.

2.2 Layers

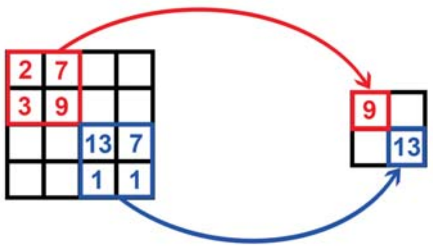
Layers are collections of neurons. There are different types of layers, which fulfill different tasks within a network. As already mentioned, the type of layer has influence on the neurons belonging to it, e.g. on the connections between the neurons.

One general constraint, which is valid for all neurons within a convolutional neural network, is that neurons of a layer $l - 1$ can only have outgoing connections to a layer l, which means that no layer can be skipped and that no connections to the same or a previous layer are allowed. This is reasoned by the fact that CNNs are feed-forward networks, i.e. data can only flow in one direction. Other constraints are highly dependent on the different types of layer, which are described hereafter.

The first type of layers are the convolutional layers. They are the most important ones in convolutional neural networks and they basically generate local descriptions of their input by using several different filters. For this, the neurons in such layers apply their standard activation function to overlapping areas of the input, which is shown in Figure 5.4 a). The influence of each part of the area, i.e. of each contained neuron, is hereby determined by a weight, which simply scales the output accordingly. Furthermore, the weights of connections between the same relative position within the overlapping areas and the neurons of the convolutional layer dealing with the respective area are shared, which is also depicted in Figure 5.4 a). For example, the output x_1 in the red area is scaled with weight w_1, which is also done for the output x_5 in the blue area. Both outputs share their relative position within their respective area, which is why their



(a) A convolutional layer of size 2 × 2 with 1 filter and filtersize of 2×2 is here applied to a 3×3 input to obtain a result of size 2 × 2 × 1.



(b) A max-pooling layer with non-overlapping 2×2 pooling is applied to a 4 × 4 input. For example, the output of the top-left neuron would be 9.

Abb. 5.4: Examples for a) convolutional and b) max-pooling layers

influence on the processing neuron of the convolutional layer is also weighted with the same shared weight. One set of such shared weights can be interpreted as a filter applied to the whole input, generating local descriptions for each area while it is slided over the input. All local descriptions combined are then called a feature map. In most cases, convolutional layers do not only have one such filter, but several different ones, each of them having a different set of shared weights yielding different feature maps. Following the interpretation from before, one could say that convolutional layers apply multiple independent filters to generate multiple different local descriptions for the input. Typically, convolutional layers represent the first layers of a network, as they only generate local information. Before this local information is put together, which is done by fully-connected layers (as they can combine local information from all over their input), the information is often condensed to reduce its dimension.

This is done by another type of layer, the pooling layers, which simply summarize areas of the preceding layer in one neuron of their own. For this, the pooling neurons apply their pooling function, which is chosen from a variety of possiblities, to areas of the preceding layer. The connections, which are used for this, do not have any weights, as the pooling layers are not adjusted during the training of the network. One way of pooling is the so-called max-pooling, where the maximum value in an area is chosen as the representative for the whole area. This is shown in Figure 5.4 b) for non-overlapping pooling regions of size 2×2. In such non-overlapping regions, one neuron can only be part of one area, which is then summarized by one neuron of the pooling layer. However, pooling regions can also be overlapping, but this limits the resulting reduction of dimension. As stated before, a lot of convolutional layers are directly followed by a pooling layer to reduce the size of the generated feature maps, before the still local information is combined by fully-connected layers.

These fully-connected layers are perhaps the most basic layer type, as they are used in a variety of different neural networks. Neurons in such layers are connected to every neuron of the previous layer by weighted connections. In contrast to convolutional layers, there is no weight sharing in fully-connected layers. Still, such layer use the same activation function and they apply it in a similar way, as their input is also composed of the sum of all weighted incoming connections and the complementing bias term. As the neurons of these layers take the input from all the neurons of the preceding layer, fully-connected layers are well suitable to combine the local information generated in convolutional layers. Thus, it is not surprising that they are typically used as the last but one layers of convolutional neural networks. They are only followed by the final classification layer which uses their results to compute the final output vector of the network.

This vector has as many entries as there are different classes in the classification task, each of them containing the probability that the input belongs to the respective class. The vector is represented by neurons of the final layer, which again have weighted connections to the previous layer. The probabilities can be computed by different kinds of probability functions, one variant being the softmax function, a generalization of the logistic function. Layers with this probability function are called softmax layers and represent the most common classification layer.

One exemplary network is shown in Figure 5.5, where all the before mentioned aspects, except for the final softmax layer, are clearly visible, above all the convolutional layers followed by pooling (sub-sampling) layers.

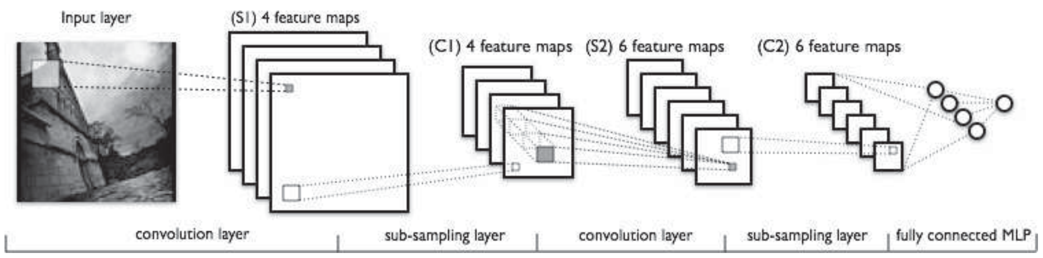


Abb. 5.5: This convolutional neural network consists of one input layer and two convolutional layers, which are followed by a pooling (sub-sampling) layer. Finally, there is a fully-connected layer and the final output layer. Source: [8].

Altogether, the network computes a chained function consisting of the functions of each layer to classify the input. As the weights of the neurons in the convolutional, fully-connected and classification layer can be adjusted, as well as the bias terms, there are a lot of parameters that can be tweaked to influence the overall output and thus to improve the obtained results, i.e. to achieve a correct classification. This is done during the training of the network, which will be looked at more closely in the now following Section 2.3.

2.3 Training Techniques

Convolutional neural networks are trained on image datasets by applying the network to the images and adjusting it to yield better results. The intention behind this is that unknown images can also be classified correctly, once the training of the network on the training dataset is completed. As stated before, the adjustment of the network is realized by tweaking the weights of connections between neurons and the values of bias terms and it is for example computed by a technique called stochastic gradient descent.

Stochastic gradient descent basically finds a new value for each weight so that the overall function moves into the direction of a minimal deviation between the network's output and the desired output, i.e. the correct classification, measured by an error function (e.g. the squared error loss function). The only requirement for the error function is derivability, as will become clear in the following.

The adjustment of the weights starts with those of the last layer. First, all partial derivatives of the error function for all weights of the last layer are computed. The so obtained gradient then indicates the direction of the largest decent of the error function, which means that it determines how the weights have to be adjusted to get closer to the minimal difference between computed and desired output. The weights are then adjusted accordingly by adding the gradient to the former values of the weights, which then moves the function in the direction of the minimal deviation. However, it might happen that the minimum is skipped by accident, if a too large step is used. For this, an additional parameter, the learning rate, is used which scales the gradient and thus determines how far the movement in the direction of the minimum shall be. Like this, the unwanted effect of skipping a minimum is treated and with suitable choices of the learning rate avoided.

After the weights of the last layer have been adjusted, the second to last layer is regarded next. This procedure is called backpropagation algorithm, which simply backpropagates the error from the final layer towards the input layer until all weights have been adjusted, i.e. the input layer is reached.

In contrast to standard gradient descent, which needs a complete run on the input dataset, stochastic gradient descent can be performed after an arbitrary number of images, which speeds up training for huge datasets. In the context of improved training techniques, pre-training and fine-tuning are two other important terms, above all when the intended training dataset is quite small. Pre-training means that the network is first trained on an additional dataset, which is (much) bigger than the (small) dataset which is supposed to be classified. Like this, the weights within the network can be trained with much more training examples. After the training on this additional dataset is completed, the final classification layer is exchanged with a new classifcation layer to fit the number of classes of the new dataset. The network is then fine-tuned on the desired dataset, which means that the network is trained as usual with the slight difference, that all weights but those of the classification layer are fixed, so that only those of the classification layer are adjusted to enable a correct classification.

Summarizing the characteristics of convolutional neural networks, one can say that they combine local descriptions generated all over the input to compute probabilities for the membership of the input in one or another class, thus classifying the input. The obtained classification is adjusted during the training of the network to improve the results, so that in the end all training images and a lot of other images belonging to the classes of the dataset can be classified correctly, atleast in the ideal case. Although this adjustment is the main strength of CNNs, it is also their main weakness, as it can happen that weights are adjusted in such a way that only the training images can be recognized correctly whereas other, potentially similar images might be classified into completely wrong classes. This is called overfitting and it is mostly the case when the used dataset is too small for the size of the used network, i.e. if there are too many parameters that need to be adjusted for too few images. Thus, it is difficult to find a suitable network architecture and suitable accompanying methods, as wrong choices, e.g. a bad or too big network architecture, can prevent good results. Still, there are a lot of successful architectures available, which will be presented in the following sections.

3 Image classification with convolutional neural networks

Recently, research has seized on convolutional neural networks using them to exceed previous results in fine-grained classification by far so that convolutional neural networks can now be considered as state of the art in a lot of areas of application, like face detection or speech recognition. A paper by Alex

Krizhevsky et al. [1], who have developed an efficient two-GPU implementation for large convolutional neural networks, has played an outstanding role in this process. The authors use their network, often called 'AlexNet', to beat previous state of the art by classifying 150,000 images of the ImageNet dataset into 1000 categories for the ImageNet Large-Scale Visual Recognition Challenge (LSVRC) [9]. Thus, the 'AlexNet' can now be regarded as state of the art, which is why it serves as principle for a lot of current approaches to image classification. Both, its architecture and how it is trained, shall be described in the following.

3.1 Network Architecture

Krizhevsky et al.'s network, the AlexNet, basically consists of 5 convolutional layers, some followed by max-pooling layers with overlapping pooling regions (layers 1,2,5), 2 fully-connected layers and a final 1000-way softmax layer. The whole network is spread across two GPUs, which is achieved by splitting up every layer into two parts of equal size and putting them on different GPUs (architecture as depicted in Figure 5.6). Like this, a higher efficiency for larger datasets can be achieved, as multiple computations can be performed simultaneously, although less efficient GPU-to-GPU communication becomes necessary due to this architecture, aswell. However, this negative aspect is limited by another trick, as the inefficient GPU-to-GPU communication is only needed in some of the layers (convolutional layer 3, the fully-connected layers and the softmax layer), whereas in the other layers the neurons only take input from neurons that reside on the same GPU (convolutional layers 2,4,5; see Figure 5.6.

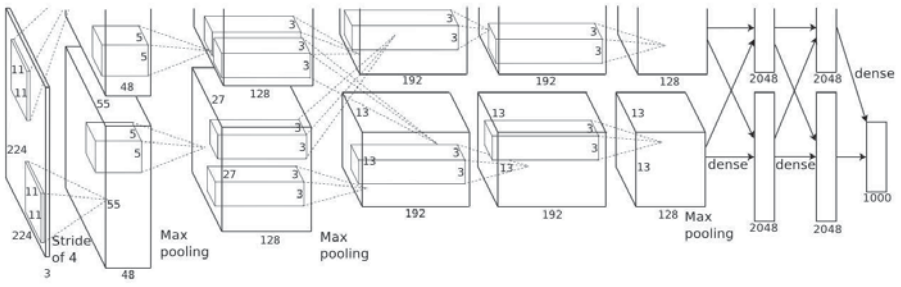


Abb. 5.6: Schematic illustration of the structure of the CNN. Source: [1]

The network's neurons are modeled as ReLUs, because the authors believe networks with such neurons train several times faster than networks with saturating neurons. This choice has contributed to the fact that ReLUs can be seen as the current standard activation function. Furthermore, the activity of neurons of the second and third convolutional layers is adjusted by Local Response Normalization (LRN), a technique that scales the activity of a neuron in relation to the activities of a certain number of other filters applied on the same spatial position. In particular, n filters inhibit the activity evoked by a filter i, which is similar to a behavior found with real neurons called 'lateral inhibition', where neurons inhibit each other to intensify contrast between them. Still, it is doubtful if this technique is really necessary, as there is no other approach which also uses it.

3.2 Training

The AlexNet is trained with standard training techniques for convolutional neural networks, i.e. with stochastic gradient descent and backpropagation. The whole training process is designed such that overfitting is reduced as much as possible. This becomes necessary as the AlexNet consists of 650,000 neurons and 60 million parameters, which need to be adjusted during the training.

One thing that is done against overfitting is an artifical augmentation of the training dataset, i.e. the network can be trained on a greater amount of images. The augmentation is performed by first altering the color of the original image using a random variable, which yields different results each time the image is used as the input. After that, the so altered image is resized to a fixed resolution and 1024 random patches are extracted. The network is finally trained on those 1024 patches and their horizontal reflections, totalling in 2048 overall training images generated from one single image in one iteration of the training process (mostly, one image is used as the input more than once, each time yielding different 2048 patches). At test time however, only 5 patches are extracted, namely the 4 corner patches and the center patch, and used as network input along with their reflections. The network's result on these 10 patches is then averaged to obtain the classification of the original image.

Another technique to avoid overfitting is called Dropout [10], which literally drops a hidden neuron out of the architecture of the network for one training image by a 50% chance, i.e. it does neither take part in the forward feeding (as its output is set to 0), nor in the backpropagation for this image. Thus, the network's architecture looks different for every training input presented which prevents neurons from relying on their neighbors too much and forces them to learn more robust features. During the training of the AlexNet, dropout is used in the first two fully-connected layers, which makes it neccessary that the out of these neurons is multiplied by 0.5 at test time to keep the results in a manageable dimension.

3.3 Results

Krizhevsky et al. performed several tests on the AlexNet, achieving record breaking results on the ILSVRC 2009 and 2010 test datasets and winning the 2012 competition. Top-1 error rates (the percentage of images, which are not classified correctly) and top-5 error rates (the percentage of images, for which the top-5 computed results of the network do not contain the correct classification) of the AlexNet and of the next best approaches are shown in Table 1 and will be presented in the following.

In the 2009 edition, top-1 and top-5 error rates of 67.4% and 40.9% are achieved, compared to 78.1% and 60.9% reported by the next best approach. Still, there is no established test dataset for this competition, which is why the much improved error rates might also be explained by the differences between the test datasets (Krizhevsky et al. simply use a random half of the dataset for training and the other half for testing)

ILSVRC-2009

Model	Top-1	Top-5
Previous best	78.1%	60.9%
AlexNet	67.4%	40.9%

ILSVRC-2010

Model	Top-1	Top-5
Previous best	45.7%	25.7%
AlexNet	37.5%	17.0%

ILSVRC-2012

Model	Top-1	Top-5
Next best	—	26.2%
AlexNet	36.7%	15.3%

Tab. 1: Comparison between the results achieved by AlexNet and previous or next best results in the ILSVRC-2009, 2010 & 2012 competitions.

In contrast to that there is a fixed dataset for the 2010 edition in which top-1 and top-5 error rates of 37.5% and 17.0% beat previous results (45.7% and 25.7%) by significant margins. Here, the AlexNet can clearly be seen as an improvement. This opinion can be substantiated by the results of the 2012 competition, where Krizhevsky et al. have actually participated and won with the AlexNet, beating the runner-up with a top-5 error rate of 15.3% compared to 26.2%.

Thus, it can be claimed that the AlexNet has set a new standard in the field of fine-grained image classification, atleast when only talking about the final classification accuracy. This is why a lot of other approaches have build on the architecture of Krizhevsky et al. One group of ideas has focused on part-based approaches, i.e. applying the network to selected parts of images. The ideas behind this will be looked at more closely in Section 4.

4 Part-based approaches to image classification

Part-based approaches try to further enhance the performance of a single convolutional neural network by using it in combination with additional region proposal algorithms, which propose certain regions of the input image with the intention of restricting the data, which is processed by the network. Ideally, this reduces the input data to the most useful one, i.e. those regions which are most suitable for discrimination. Doing so, unnecessary information, like background noise, is dropped as this is not needed for the correct classification of the image. Furthermore, part-based approaches aim at comparing the appearance of certain parts of the displayed object, rather than the appearance of certain regions within the image.

Both basic ideas can be realized in a lot of different ways, as will be shown with the help of two supplementary papers.

4.1 Part-Based R-CNNs

Ning Zhang et al. [2] propose a generalization of a method by Girshick et al., called Region-CNN [11], to classify 6000 bird images of the Caltech UCSD bird dataset [12] into 200 different breeds. In Region-CNN, images are first put through a region proposal algorithm, where certain regions of the image are extracted. These regions are then fed to a CNN to generate appropriate descriptors, which are finally used to classify the image with a linear support vector machine (SVM). A (linear) SVM divides two sets of points (representing descriptors belonging to two different classes) by a (linear) function while maximizing the distance between the points located near the border on each side, as depicted in Figure 5.7. Like the network, the SVM function is trained with the descriptors of training images and fixed at test time, where a point belongs to one of the classes, if it lies on the respective side of the border. For the actual classification, the authors one one-versus-all SVM for each occurring class, i.e. SVMs which decide whether a descriptor belongs to a certain class or not.

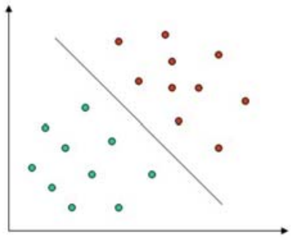


Abb. 5.7: The SVM divides the set of points in two subsets, maximizing the distance between the points located near the border. Source: [13].

4.1.1 General Structure The basic idea of Zhang et al.'s Part-Based R-CNN approach is the same like in the original Region-CNN method. Parts of the images, which are suitable to distinguish between different objects, are first proposed by a method called Selective Search [14], which uses a form of over-segmentation to propose regions, i.e. the images are first divided into a huge number of very small parts and then put back together step by step to obtain a desired amount of regions. The so proposed regions are then processed by a convolutional neural network, which uses the network architecture of the original AlexNet (see Figure 5.6), but a different implementation (Caffe [15]). After that, the computed descriptors are fed to a collection of one-versus-all SVMs. However, Zhang et al. do not only have SVMs for each class, but also for a certain number of different parts of the objects, e.g. the bird's head or body, which enable them to further perform part detection. They use this additional information to generate a representation for the object, which is based on the relation between the different parts and their respective appearance. This representation is finally used to classify the input with another SVM.

In the following, the training of the overall system will be explained first, before details of the classification process will be pointed out.

4.1.2 Training In comparison to the training of the AlexNet, the training of the Part-Based R-CNN method seems quite expensive, as there are a lot of different components that need training. However, each component can be easily trained as will be shown in the following.

The convolutional neural network is pre-trained on the ImageNet dataset and fine-tuned on the desired bird dataset afterwards, as the bird dataset is quite small compared to the ImageNet dataset (6,000 images compared to 1.2 million images). For the training of the network, Zhang et al. use the same standard techniques like Krizhevsky et al. At test time, the network architecture only differs in the final 200-way softmax layer which replaces the original 1000-way version to fit to the 200 bird breeds contained in the dataset. As the pre-trained AlexNet is publicly available, i.e. the weights of each connection, the only time in this part is spend on the fine-tuning of the network.

In contrast to that, the SVMs need a complete training. The authors do not only use one SVM for each breed of birds, but also one for each part of breed, e.g. one for the head. As the bounding boxes for the birds and a fixed number of their parts is annotated in the training data, the authors use feature descriptors of proposed regions with more than 0.7 overlap with the bounding box of a certain object or part as positive examples for the respective SVM, and regions with less than 0.3 overlap as negative examples.

The final classification SVM is trained on a pose normalized object representation, which uses the geometric relation between the detected objects and parts to filter out false detections. As this object representation is quite complex, it will be explained in the following Section 4.1.3.

4.1.3 Pose normalized object representation The pose normalized object representation is basically achieved by translating intuitive properties of correct detections into formulas.

For this, the locations of an object p_0 and its n parts $\{p_1, p_2, \cdots, p_n\}$, i.e. their bounding boxes, are denoted by $X = \{x_0, x_1, \cdots, x_n\}$, with corresponding SVM scores $\{d_0, d_1, \cdots, d_n\}$, $d_i(x)$ being the score for a feature descriptor generated at location x and put into the SVM for part i.

The first property is that the highest scored detections ought to be the right ones. This is achieved by using Equation (5.1) without $\Delta(X)$, as this simply defines an optimization problem to find that combination of proposed regions, which maximizes the product of all object and part detector scores. The solution for this is simply the combination of the highest scored regions for each object and part detector.

$$X^* = \underset{x}{\operatorname{argmax}} \left(\Delta(X) \prod_{i=0}^{n} d_i(x_i) \right) \tag{5.1}$$

However, the detectors are not perfect, which makes false detections possible. Thus, the authors use Equation (5.1) with $\Delta(X)$, as this function is designed to establish a relation between the object and part locations to filter out false detections. They experiment with several different definitions for $\Delta(X)$, each of them representing one intuitive constraint to the correct detections.

The first one, which the authors call box constraints, simply filters out those combinations, where parts are detected outside of the bounding box of the overall object, i.e. when not all parts p_1 to p_n are located inside the bounding box x_0 of object p_0. For this, they define $\Delta(X)$ as

$$\Delta_{box}(X) = \prod_{i=1}^{n} c_{x_0}(x_i). \tag{5.2}$$

Hereby, $c_{x_0}(x_i)$ is defined such that part locations x_i, which fall outside of the object's location x_0 by more than a fixed tolerance are considered to be not valid ($c_{x_0}(x_i) = 0$), whereas part locations within this tolerance limit are considered as valid ($c_{x_i}(x_i) = 1$). Thus, this definition of the scoring function $\Delta(X)$ fulfills the primary task and eliminates all detections, where certain parts are detected outside of the overall object. Like this, the simple combination of the highest scored regions is not always possible anymore.

Still, the geometric relation between the parts is not regarded by the box constraints, which allows unwanted constellations, e.g. detections of a bird's head close to the bird's feet. To avoid this, the box constraints are extended to filter out incorrect detections by enforcing constraints over the relative position between parts and their object. They define their geometric constraints as

$$\Delta_{geometric}(X) = \Delta_{box}(X) \left(\prod_{i=1}^{n} \delta_i(x_i) \right)^{\alpha}, \tag{5.3}$$

where δ_i is a function that fulfills the task of scoring the relative position of part p_i using information learned from the training data and α is an adjustable parameter. For δ, the authors propose two different definitions.

$\delta_i^{MG}(x_i)$ fits a Gaussian mixture model with 4 components to locations of part p_i learned from the training data. In other words, the location x_i of a detection for part p_i is scored by the probability distribution for the location of part p_i in the images of the training data. Thus, this definition tries to create a general model describing all possible shapes of an object category. However, this might be inaccurate, if the objects appear in different poses, which is not captured sufficiently. Just think of a data set in which a lot of birds stand upright and only few have their heads near the ground (which is also a normal pose for birds). Then the model would state that it is quite unlikely that the head of a bird is located below the body, thus scoring object and part candidates in this pose with a lesser score than candidates similar to an upright pose. Like this, detections might be discarded, although they were the correct ones, just in a pose, which is underrepresented in the dataset.

$\delta_i^{NP}(x_i)$ is an adaption of $\delta_i^{MG}(x_i)$ and tries to eliminate the described mistake. Here, the Gaussian model is not fit to the whole training data, but to 20 images, which are nearest neighbors to the top-scoring window of the object detector. This means that the location x_i of the detection for part p_i is scored by the position of part p_i in 20 images, which show the highest resemblance to the whole object detection. In contrast to the model created by $\delta_i^{MG}(x_i)$, the $\delta_i^{NP}(x_i)$ model does not describe all possible shapes of an object, but only one very specific shape, thus minding the scenario described above, as well.

The authors reportedly use both of the definitions of δ, achieving results with only slight differences. The latter one is illustrated in Figure 5.8

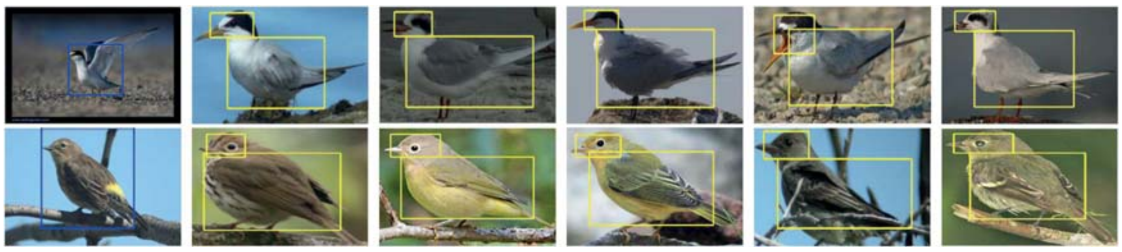


Abb. 5.8: The birds in the first column are the respective test images with bounding boxes determined by the approach of Zhang et al., the 5 other birds in each row are the top-5 nearest neighbors used in δ^{NP}. Source: [2]

As soon as the optimization problem is solved, the feature descriptors of all regions, which are used in the optimal constellation, are concatenated and used as the input for a final SVM. Thus, the final classification is performed using the pose-normalized representation created before, which makes a more stable classification possible compared to a representation that does not take different poses into account. Apart from the mere classification of an object, the authors can also use their approach to locate parts, as this is an elementary component of their system. Results for both, classification and part localization, will be shown in Section 4.1.4.

4.1.4 Results Zhang et al. use their system to classify images of the Caltech-UCSD bird dataset into the 200 different bird species contained in the dataset. They conduct several experiments in a setting where the bounding boxes of the objects and parts are unknown at test time. The classification and part localization results achieved by the different versions of the scoring function will be reported in Table 2.

Classification with bounding box unknown

Model	Accuracy
Previous best	44.94%
Δ_{null}	64.57%
Δ_{box}	65.22%
$\Delta_{geometric}$ (δ^{MG})	65.98%
$\Delta_{geometric}$ (δ^{NP})	65.96%
Δ_{box} (finetuned)	72.73%
$\Delta_{geometric}$ (δ^{MG}) (finetuned)	72.95%
$\Delta_{geometric}$ (δ^{NP}) (finetuned)	73.89%

Part localization accuracy

Model	Head	Body
Previous best	37.44%	47.08%
Δ_{null}	60.50%	64.43%
Δ_{box}	60.56%	65.31%
$\Delta_{geometric}$ (δ^{MG})	61.94%	70.16%
$\Delta_{geometric}$ (δ^{NP})	61.42%	70.68%

Tab. 2: Tables showing accuracies of the Part-Based R-CNN. Table on the left shows the overall accuracy, whereas the right table shows the accuracy of part localization, measured as percentage of correctly localized parts.

The approach of Zhang et al. beats previous state of the art by significant margins, which shows that convolutional neural networks can also be used successfully in a part-based context. Furthermore, the results are suitable to judge the effect of the different scoring functions on the overall outcome. As can be seen in the left table, the usage of a scoring function slightly improves the performance of a system without a scoring function ($\Delta_{null} = 1$, see Equation (5.1)). Moreover, the extension of Δ_{box} to $\Delta_{geometric}$ further improves the performance, although the difference might be less distinct than expected. However, the difference becomes clear, when the part localization accuracy is regarded separately. Here, the usage of $\Delta_{geometric}$ boosts the performance of a system with Δ_{box} by a more significant margin, which shows that the idea behind $\Delta_{geometric}$ actually works. The lower impact on the overall classification can possibly be explained by the fact that descriptors generated by a CNN are robust against small translations, which limits the impact of false localizations.

All in all, it can be said that the constraints enforced by the authors help to improve the results of the convolutional neural network. Moreover, they show that it is wise to rather compare the appearance of parts than the appearance of same regions within images to perform a successful classification. In the following Section 4.2, another approach is presented to show another possible strategy for realizing the basic idea of part-based approaches.

4.2 Learning Features and Parts for Fine-Grained Recognition

Krause et al. [3] use feature descriptors generated by a convolutional neural network in connection with part detectors based on a technique called HOG [16] to form an object representation: Ensemble of Localized Learned Features (ELLF). ELLF stores detected parts of an object together with their appearance, i.e. their respective feature descriptors, and is then used to classify images of a self-collected cars dataset [17] with the help of a support vector machine.

The approach introduces one novelty, as all part feature descriptors are generated from the feature descriptor of the whole image, i.e. the image is divided into parts after the application of the network. This is contrary to other part-based approaches, like Part-Based R-CNN, where first patches of the image (containing desired parts) are extracted and then used as the input for the network to obtain the respective descriptors, i.e. the division into parts happens before the application of the network.

This is only possible because of the unique architecture of Krause et al.'s convolutional neural network, which is presented in Section 4.2.3 and enables a generation of feature descriptors for specific regions of an image. Before that, the general principle of the Ensemble of Localized Learned Features representation will be described in Section 4.2.1. Thereafter, it will be described how discriminative parts of the objects are first discovered and then detected by part detectors based on HOG [16] in Section 4.2.2. Finally, the efficiency of the overall system will be pointed out with the help of results published by Krause et al.

4.2.1 Ensemble of Localized Learned Features The Ensemble of Localized Learned Features (ELLF) stores each detected part of an object together with its respective feature descriptor, e.g. if the front bumper of a car is detected, a feature descriptor for the front bumper region of the image is computed and stored as the front bumper's appearance, i.e. the object parts are directly connected to their appearances. Thus, an object is represented by the concatenation of the descriptors of each part, i.e. if an object category consists of n parts and a_i denotes the descriptor of part i, the resulting ELLF representation is $(a_1, a_2, \cdots, a_n)$.

The generation of ELLF is as follows. Assume that an object category consists of n parts and that training has already happened, i.e. parts have been discovered, the corresponding detectors are trained, etc. Given an input image, parts of the depicted object are first detected by applying the trained part detectors to the image. For each detected part, the convolutional neural network will generate a corresponding feature descriptor of the region defined by the part's bounding box. If a part is not detected, the corresponding entry in the ELLF representation (its feature descriptor) is set to 0. As soon as all entries are set, the ELLF representation will be fed to a linear support vector machine, which will then perform the final classification of the image.

In this procedure, part discovery and detection and the convolutional neural network fulfill important tasks within the overall system. Thus, the process of part discovery and part detection is presented in Section 4.2.2 and details of the unique architecture of the convolutional neural network are pointed out in Section 4.2.3.

4.2.2 Part Discovery and Detector Learning Part discovery is a key component of the approach at hand. It works unsupervised and is not customized to the classification of cars, so that it is applicable to a variety of areas and not limited to the car dataset used in this paper.

The part discovery and detector learning algorithm can be divided into 3 steps. First, sets of aligned images are searched for, i.e. images which show objects in the same position (see Figure 5.9), so that spatial positions of the images can be compared to select certain parts. After that, those parts, which differ most across the whole set of images, are selected to be the defining parts for the object class, i.e. only those parts are used, which are most suitable for differentiation. Finally, patches of the original images are used to train one part detector for each selected part, which concludes the part discovery algorithm. Details of each step will be described in the following.

As mentioned before, the algorithm starts with the search for sets of aligned images. For this, one of the images contained in the dataset is randomly selected as root image and compared to the rest of the data set in terms of HOG features. A certain number n of images that show the highest resemblance is then picked, centered to make comparison easier and stored together with the root image to form

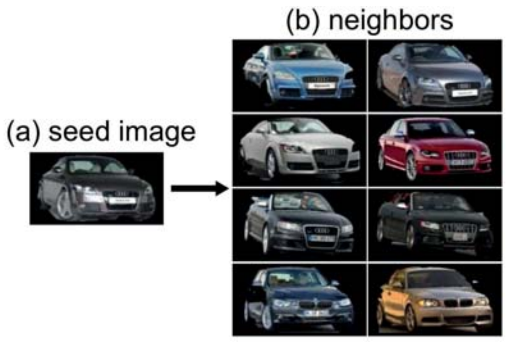


Abb. 5.9: 8 aligned images, which show nearly the same pose like the seed image. Source: [3]

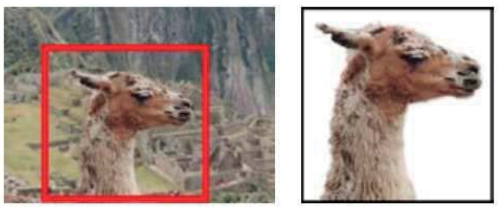


Abb. 5.10: The foreground of the region specified by the user (red frame) is extracted from the image

one set of aligned images (see Figure 5.9). As the background might have unwanted influence on the comparison, the foreground of all images is beforehand extracted using a form of image segmentation called GrabCut [18], which extracts an image's region determined by a bounding box (see Figure 5.10). As the bounding boxes for all objects are annotated in the training data, no additional manual input is needed for this step of the algorithm, which is an important aspect for this approach.

From the so found sets of aligned images, certain parts are selected to become the defining parts for the object class. This is done by randomly sampling some thousand patches of various sizes as part candidates, whereby the same regions are selected on each image. The variance of the candidates' HOG features is then compared across all images, to find parts which differ most, as these are most suitable for discrimination, e.g. cars can easier be distinguished by their front bumper than by their tires. The 10 candidates with the highest variance are finally selected as object parts. Whenever a candidate is selected, all parts that overlap more than a fixed value with the selected part are removed from the list of candidates, so that the selection of redundant parts is avoided. Note that this step is also performed in an unsupervised way, as patches are randomly sampled and as the 10 parts are automatically selected.

Finally, one detector is trained for each selected object part to enable automatic part detection. For this, patches of each aligned image at the part's location are used as positive example, whereas patches from other locations, which must not overlap with the part's location, are used as negative examples. However, it cannot be guaranteed that the images are indeed well-aligned, so that the part's location might not be exactly the same in every image, which has to be considered as well.

Krause et al. propose the following learning objective

$$\min_w \left\{ \sum_j \max\{0, 1 - \max_{z_j^+} w^T h(I_j, z_j^+)\} + \sum_j \sum_{z_j^-} \max\{0, 1 + w^T h(I_j, z_j^-)\} \right\}, \tag{5.4}$$

where $h(I_j, z)$ represents the HOG features extracted from image I_j at location z. Furthermore, z_j^+ is a latent variable representing the true location of the respective part on image I_j, whereas z_j^- are randomly chosen locations on image I_j that do not overlap with the true location of the part and are thus used as negative examples.

The objective is then to find a vector w which yields maximal values when multiplied with the HOG feature vector extracted at positive locations, i.e. maximizing $w^T h(I_j, z_j^+)$, and minimal values when multiplied with vectors extracted at negative locations, i.e. minimizing $w^T h(I_j, z_j^-)$. For this, the latent variable z_j^+ is initialized with the original location z^+, which is only the true location, if the images are well-aligned. However, w is first optimized for this location and after that the best match for all possible locations z_j^+ is chosen. The objective is optimized by alternately optimizing w or z_j^+ while fixing the other variable or by including a penalty for locations z_j^+ which are too far away from the starting location z^+.

At test time, the vector w^T is multiplied by the HOG feature vector of a certain region of an image to obtain the detector score. If no part of the image surpasses a certain threshold, the part is considered to be not visible in the image, which will automatically set the associated feature descriptor in the ELLF representation to 0. The generation of those feature descriptors shall now be explained in the following Section 4.2.3.

4.2.3 Network Architecture The convolutional neural network used by Krause et al. is the second key component of their approach, as it enables the generation of feature descriptors for specific regions within an image, i.e. feature descriptors for different parts of an image can be generated without the need of applying the network to respective image patches like in Part-Based R-CNN for example.

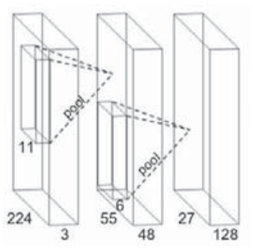


Abb. 5.11: At test time, the convolutional neural network consists of one input layer and 2 convolutional layers. Feature descriptors for specific regions are generated by performing max-pooling on the second convolutional layer. Source: [3]

This is only possible due to the unique architecture of the network at test time, as the fully connected layers are removed after training and feature descriptors are thus only generated by the remaining 2 convolutional layers (architecture at test time as shown in Figure 5.11). The feature descriptor for a specific region is then generated by performing max-pooling on the respective region of the second convolutional layer, which is not possible in a standard convolutional neural network, as spatial information is lost in fully connected layers. Convolutional layers, however, retain spatial information so that a direct relation between a value in the feature descriptor and a region within the image can be established. Like this, Krause et al. can generate feature descriptors for each part by simply pooling the overall feature descriptor at those positions, where a part is detected.

Although the network at hand is based on the AlexNet, there are some more differences to the original architecture, as the network uses 2 convolutional layers instead of 5, which is an adjustment made due to the much smaller scale dataset (17,000 images compared to 1.2 million images), and linear units in the fully connected layers (at least during training). Still, the number of fully connected layers (2), the usage of dropout and of ReLUs in the convolutional layers and the various forms of data augmentation techniques are just like in the approach of Krizhevsky et al.

What is confusing is the fact that the Krause et al. do not use pre-training for their network although their training data is quite small, which is a difference to the Part-Based R-CNN. Although they say that this aspect will not be important in the future anymore, as datasets will become much bigger, this explanation is not convincing, as pre-training could have further improved their results.

Nonetheless, their results are quite impressive, even without pretraining, as will be evaluated in the following Section 4.2.4.

4.2.4 Results Krause et al. apply their system to classify their own car dataset, which makes a comparison to other techniques harder, as no other approach has worked on the same data before. Thus, they also apply other techniques to the same dataset to compare the results and to make founded statements regarding the efficiency of their own technique.

As can be seen in Table 3, ELLF beats the results of the previous best approach by 4.4%. Moreover, it can be seen that the sole application of a convolutional neural network, e.g. the AlexNet, is also better than the previous best, which solidifies the impression obtained in the before discussed approaches that convolutional neural networks can be regarded as state-of-the-art.

The results for CNN-SPM (small and large) further show the importance of the ELLF representation. In those two adaptions of the approach at hand, the ELLF representation is adjusted in such a way that the same entries of the representations of two images refer to the same location within the image. The locations used in the representation are obtained by performing spatial pyramid matching (SPM) on the image, i.e. the image is divided into patches of same size which are then processed by a CNN to generate descriptors for each patch. In the case of CNN-SPM small, first the whole image is used, then the whole image divided into 4 patches and finally into 16; for the CNN-SPM large the image is additionally divided into 64 patches. The descriptors for each of those patches is then concatenated and used as the representation. The adjusted variants are outperformed by the original ELLF representation by 6.0% and 4.6%, respectively, and even the previous best approach is better than them.

Classifying Cars

Model	Accuracy
Previous best	69.5%
CNN-SPM (small)	67.9%
CNN-SPM (large)	69.3%
CNN	70.5%
ELLF	73.9%

Tab. 3: This table shows the performance of several models on the car classification task.

Thus, it is again shown, that the comparison of the actual appearance of parts and not same spatial positions within images is more useful for a successful classification, as was already seen in the Part-Based R-CNN approach. The results of the two previous sections will be summarized in the now following Section 4.3.

4.3 Evaluation

In this section, the most important facts that can be learned from the three different approaches will be presented in order to develop some kind of guideline concerning the application of convolutional neural networks. For this, basic components of all networks will be evaluated and pointed out in connection with other general aspects.

The main problem of all deep convolutional neural networks, i.e. networks with multiple convolutional and fully connected layers, is overfitting, which can be avoided by training the networks with huge datasets. However, such datasets are not necessarily available, which is why a lot of dataset augmenting techniques, like color perturbation or the training on image patches, are used to augment the available training data.

Another possibility is to pre-train the network on a large, additional dataset and then later fine-tune the network on the desired, smaller dataset, which is for example done by Zhang et al. It is also possible to simply use a smaller network, like Krause et al. showed in their approach. However, it one can be assumed that smaller networks without pre-training, which are not used in a part-based system, perform worse than larger networks with pre-training, as the fewer number of layers restricts the level of abstraction computed by the network.

In general, it can be said that the network's size, especially the number of convolutional layers, has to be chosen fitting to the size of the available dataset, which should always be increased as much as possible by multiple data augmentation techniques, because the larger the resulting dataset is the more convolutional layers can be used, thus deriving more and more complex descriptors of the input.

The network's neurons are preferably modeled as rectified linear units. Doing so, response normalization is not needed to prevent neurons from saturating, i.e. making training possible, as Krizhevsky et al. explain in their paper.

Several fully connected layers should be used during the network's training. After that, they can theoretically be removed from the network to obtain feature descriptors that preserve spatial information. This means that every part of the resulting feature vector can directly be associated with a specific region of the input image, which is not possible after the application of fully connected layers, as these mix up the spatial information. Thus, convolutional neural networks can be used to generate two different types of feature descriptors, namely one that can directly be related to the input image and the other being an abstract description of the image.

Another indispensable component of convolutional neural networks are (max-)pooling layers. They are normally used to summarize information of fixed areas in between layers, but as Krause et al. have shown, they can also be converted to pool specific regions of the network's output vector. In connection with cutting off the fully connected layers, this method enables the network to generate feature descriptors for different parts of one input image. This way, Krause et al. perform unprecedented work in the area of part-based approaches, as other approaches, like Zhang et al., need to first extract the desired region from the image and then use the extraction as the network's input to generate the desired feature descriptors. However, both approaches show that it is clever to compare the appearance of parts rather than to compare certain positions within images, as both build representations connecting object parts with their feature descriptors.

Summarizing it can be said that it is not surprising that convolutional neural networks are the new state of the art in object classification, as all three approaches show the potential of these networks on basic research classification tasks. In contrast to that, the application of convolutional neural networks to medicine, a field of particular interest, shall be discovered in the following.

5 Application to Medicine

The requirements for image classification in medicine are a great deal higher than for usage in fields that do not concern life, because false detections, both false-positive and false-negative, can cause life-threatening consequences. Thus, medical image processing systems should have high precision, i.e. a high detection rate and a low number of false positive detections.

Even slight improvements in either of both aspects make such systems more applicable. Knowing that convolutional neural networks can be successfully combined with other techniques, it becomes plausible that they could be used for these improvements in connection with already existing methods. And indeed, a lot of combined methods can be found. First, early development is shortly described, before current progress, especially based on Krizhevsky et al.'s basic research, is stated.

5.1 First Attempts

First attempts in this domain were undertaken in the mid 90's, after LeCun et al. had proposed convolutional neural networks in 1989 [6].

Lo et al. [19] applied convolutional neural networks to the detection of lung nodules on chest radiographs, which can be indicators for lung cancer. As assumed above, they used the network in combination with other methods, namely to verify pre-scan results, i.e. to filter out false-positive detections (FP) and to confirm true-positive detections (TP). The results achieved by the network were later validated. The authors report that their network reduces the number of FPs by 79% and preserves 80% of TPs.

This shows that convolutional neural networks can effectively improve other techniques to come near to performances of radiologists, especially considering studies by Stitik et al. [20] from 1985, who stated that one radiologist was only able to detect 68% of lung nodules back in that time, which shows that tasks like this were even hard for human beings not too long ago.

Despite the obvious potential of convolutional neural networks in Computer Aided Detection (CADe), they were still rarely used, as a paper review by Shi et al. [21] showed in 2011. They point out that convolutional neural networks were only used in 2 of the 23 reviewed papers concerning medical object detection and recognition. Although their choice of papers might not be representative for the real application of such networks, it clearly shows that there were not many common papers proposing CNNs for medical image processing, as these would have been surely included in their work. Following the publication of Krizevshky et al. [1], this has changed however, as there have been a variety of newly developed techniques in medical image processing involving convolutional neural networks. Two of them will be exemplarily described in the next subsection.

5.2 Recent Progress

Holger Roth et al. [22] build upon state-of-the-art Computer Aided Detection methods for computed tomography (CT) images by Liu et al. [23] (mediastinal CT volumes) and Cherry et al. [24] (abdominal) to detect enlarged lymph nodes. The CADe methods can detect almost 100% of the occurrences of lymph nodes, although this high sensitivity, i.e. high number of true positive detections and low number of false negative detections, is at the expense of a relatively high number of false positive detections, namely about 30 per image. Thus, a challenge is to filter out those false detections, while simultaneously preserving correct detections.

The network used for this works with the cuda-convnet implementation by Krizhevsky et al., ReLUs and DropConnect, an advancement of Dropout which drops out single connections instead of whole neurons, with the architecture being illustrated in Figure 5.12.

The preprocessing CADe systems by Liu et al. and Cherry et al. yield marked regions in the 3-dimensional CT volumes, indicating where enlarged lymph nodes have been detected. Each marked region is then divided into one set of three 2-dimensional images, namely axial, one coronal and one sagittal image, which are all centered at the center of the region (see Figure 5.13). Doing so, the authors can use a CNN designed for RGB images by simply using each slice of the 3-D volume as the input for one of the

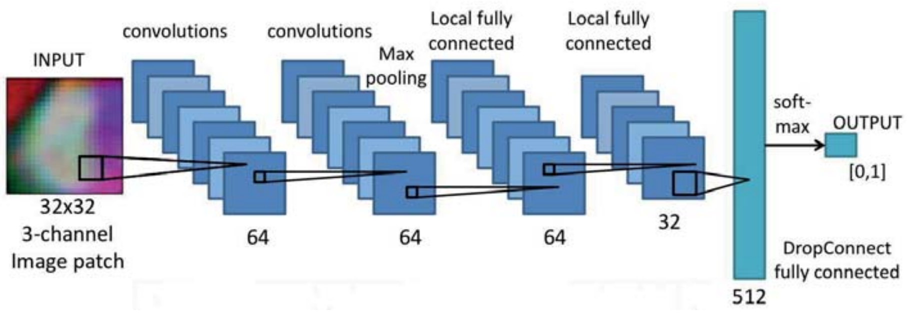


Abb. 5.12: The CNN consists of 2 convolutional layers, followed by a max-pooling layer, 2 fully-connected layers without DropConnect, 1 fully-connected layer with DropConnect and a final 2-way softmax. Source: [22]

3 input channels, i.e. the axial image for the first channel, the coronal image for the second channel and the sagittal image for the third channel.

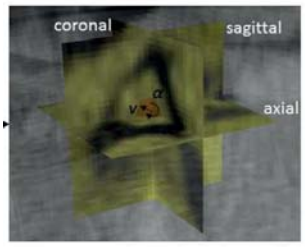
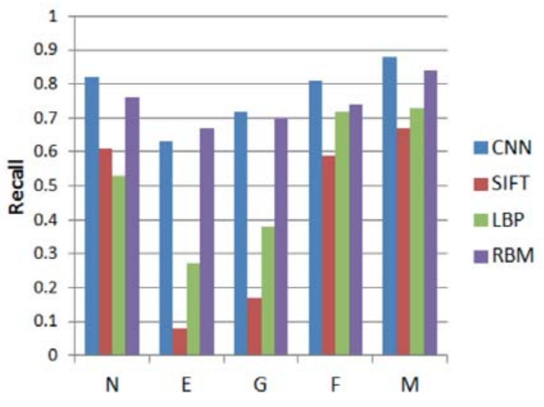


Abb. 5.13: Decomposition of the 3-dimensional CT volume into 3 2-dimensional images. Source: [22]

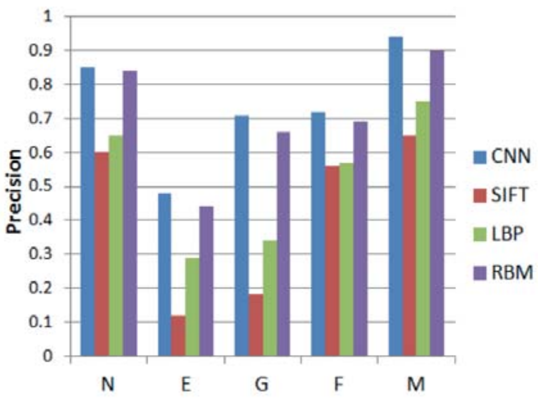
Finally, two different networks are trained, one for each category of CT volumes, i.e. one for mediastinal images and one for abdominal images. The application of convolutional neural networks improves the sensitivity of the mediastinal lymph node detection from 55% of the stand-alone CADe method (Liu et al.) to 70% and from 30% (Cherry et al.) to 83% in the abdominal regions, all achieved with a fixed rate of 3 false positive detections per image, e.g. 83% of the true positives were detected, before 3 false positives occured. This proves that the performance of existing CADe systems can be drastically improved by the application of convolutional neural networks. Furthermore, it shows one easy way to use 2-dimensional networks for 3-dimensional volumes is shown, which is important as the typical data type in medical imaging is 3-dimensional, e.g. CT or MRI volumes.

Another application to medicine is shown by Li et al. [25], who use a CNN to classify HRCT lung volumes of the interstitial lung disease (ILD) dataset [26] into 5 different classes, one of them representing healthy lungs, whereas the other classes are different types of diseases. The authors apply a quite small convolutional neural network, which is not pre-trained, to 2-D slices of the original HRCT volumes and they do not use any kind of other techniques to further improve the performance. Still, they achieve a recall, i.e. sensitivity, which is better than previous best for all but one class of disease, and a precision, i.e. the fraction of true positives on all positives (false and true positives), which is better for all classes (see Figure 5.14).

Thus, it can be concluded that convolutional neural networks also show promising results in a medical context, which implies that they could be used regularly in the near future. Moreover, there are easy ways to use the common 3-dimensional medical imaging data with standard convolutional neural networks, although this causes a loss of information, as slices of the volumes only cover a small percentage of the available data. This is why it could be beneficial to further research efficient, true 3-dimensional convolutional neural networks, which is a difficult task as they would have a huge number of parameters and the available datasets in medicine are very small, even when used with standard CNNs. Nonetheless, the research should be continued in any case to avoid a stagnation like before.



(a)



(b)

Abb. 5.14: Results for the approach by LI et al. on the ILD classification. Source: [25]

6 Conclusion

In this paper, current progress in fine-grained image classification has been shown. Although convolutional neural networks have already been proposed in 1989 by LeCun et al. [6], their application to medical problems has stagnated from the mid 90's. However, early attempts back then, like Lo et al. [19], have already shown that convolutional networks could be beneficial for medical image processing, above all as verifiers for other image processing tools. Despite that, CNNs have not been extensively used in medical context, as the results of a paper review by Shi et al. [21] from 2011 imply. This could be explained by the fact that there had not been substantial progression in the basic techniques of convolutional neural networks, which helped to overcome deficits which were crucial for the application in medicine. What can be mentioned here is e.g. the need for a huge number of training examples, which is especially difficult to obtain for rare diseases, and for a high accuracy, because both false-positive and false-negative detections can have serious consequences for a patient's health.

The paper by Krizhevsky et al. [1] in 2012 has certainly helped to bring convolutional neural networks back to the center of attention. In their work, they showed how to efficiently train deep convolutional neural networks, thus making them more useful in general. Although their approach can be regarded as new state of the art, it uses the network as a stand-alone system and works on a huge dataset, which is different to the real 'work environment' of such networks in medical context. Thus, the papers by Zhang et al. [2] and Krause et al. [3] provide important insights regarding the applicability of this new kind of networks in connection with other techniques, as they successfully use them in part-based surroundings together with techniques like HOG feature descriptors and support vector machines.

Approaches by Roth et al. [22] and Li et al. [25] transform the newly learned techniques in a medical context. They both use standard 2-dimensional convolutional neural networks to work on the common 3-dimensional volumes in medicine, taking slices of the original volumes as the input for their networks. Roth et al. use their network to verify the detections of state-of-the-art Computer Aided Detection methods with a 2-way classification network, i.e. either verifying or rejecting the CADe detections. Li et al. use a 5-way classification network to classify lung images into 5 different classes. Both achieve promising results.

Summarizing, it can be said that the importance of convolutional neural networks in medical context will certainly rise in the (near) future. On the one hand, this can be explained by the enormous positive effects that CNNs already provide today, as shown above. On the other hand, it is nearly safe to say that the performance of convolutional neural networks will further improve with progression in hardware power and the availability of larger datasets of medical images. Still, the basic principles of convolutional neural networks should also be further improved, e.g. to make true 3-dimensional convolutional neural networks feasible, as they are the only ones which really make use of all the information present in 3-dimensional volumes.

Literaturverzeichnis

[1] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems; 2012. p. 1097–1105.

[2] Zhang N, Donahue J, Girshick R, Darrell T. Part-based R-CNNs for fine-grained category detection. In: European Conference on Computer Vision; 2014. p. 834–849.

[3] Krause J, Gebur T, Deng J, Li LJ, Fei-Fei L. Learning Features and Parts for Fine-Grained Recognition. In: International Conference on Pattern Recognition; 2014. p. 26–33.

[4] Breast MRI results. Mayo Foundation for Medical Education and Research; 2014. [Online; Accessed January 9, 2015]. Available from: http://www.riversideonline.com/source/images/image_popup/mcdc7_breast_mri_results.jpg

[5] O'Leary M. MRI Breast Cancer Screening in High-Risk Women Boosts Detection Rates. Medical Digest Publications; 2011. [Online; Accessed January 9,2015]. Available from: http://medicaldigestpublications.blogspot.de/2011/09/mri-breast-cancer-screening-in-high.html

[6] LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, et al. Backpropagation applied to handwritten zip code recognition. Neural computation. 1989;1(4):541–551.

[7] Nair V, Hinton GE. Rectified linear units improve restricted boltzmann machines. In: International Conference on Machine Learning; 2010. p. 807–814.

[8] My LeNet. Theano Development Team; 2015. [Online; Accessed January 15,2015]. Available from: http://deeplearning.net/tutorial/_images/mylenet.png

[9] Berg A, Deng J, Fei-Fei L. Large scale visual recognition challenge. Technical report; 2010. [Online; Accessed January 9, 2015]. Available from: http://www.image-net.org/challenges

[10] Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580; 2012.

[11] Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv preprint arXiv:1311.2524; 2013

[12] Welinder P, Branson S, Mita T, Wah C, Schroff F, Belongie S, et al. Caltech-UCSD Birds 200. Technical report; 2010. CNS-TR-2010-001.

[13] Wikipedia. Support Vector Machine. In: Wikipedia, Die freie Enzyklopaedie; 2015. [Online; Accessed January 9, 2015]. Available from: http://de.wikipedia.org/w/index.php?title=Support_Vector_Machine&oldid=136138406

[14] Uijlings JR, van de Sande KE, Gevers T, Smeulders AW. Selective search for object recognition. International journal of computer vision. 2013;104(2):154–171

[15] Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, et al. Caffe: Convolutional architecture for fast feature embedding. In: ACM International Conference on Multimedia; 2014. p. 675–678.

[16] Dalal N, Triggs B. Histograms of oriented gradients for human detection. In: Conference on Computer Vision and Pattern Recognition; 2005. p. 886-893.

[17] Krause J, Stark M, Deng J, Fei-Fei L. 3D Object Representations for Fine-Grained Categorization. In: IEEE International Conference on Computer Vision Workshops; 2013. p. 554–561.

[18] Rother C, Kolmogorov V, Blake A. Grabcut: Interactive foreground extraction using iterated graph cuts. In: ACM Transactions on Graphics; 2004. p. 309–314.

[19] Lo SCB, Chan HP, Lin JS, Li H, Freedman MT, Mun SK. Artificial convolution neural network for medical image pattern recognition. Neural Networks. 1995;8(7):1201–1214.

[20] Stitik FP, Tockman MS, Khouri NF. Chest radiology. In: Screening for cancer; 1985. p. 163–191.

[21] Shi Z, He L. Current status and future potential of neural networks used for medical image processing. Journal of multimedia. 2011;6(3):244–251.

[22] Roth HR, Lu L, Seff A, Cherry KM, Hoffman J, Wang S, et al. A new 2.5 D representation for lymph node detection using random sets of deep convolutional neural network observations. In: Medical Image Computing and Computer-Assisted Intervention; 2014. p. 520–527.

[23] Liu J, Zhao J, Hoffman J, Yao J, Zhang W, Turkbey EB, et al. Mediastinal lymph node detection on thoracic CT scans using spatial prior from multi-atlas label fusion. In: SPIE Medical Imaging; 2014. 90350M.

[24] Cherry KM, Wang S, Turkbey EB, Summers RM. Abdominal lymphadenopathy detection using random forest. In: SPIE Medical Imaging; 2014. 90351G.

[25] Qing L, Weidong C, Xiaogang W, Yun Z, Dagan F, Mei C. Medical Image Classification with Convolutional Neural Network. In: International Conference on Control, Automation, Robotics and Vision; 2014. p. 844–848.

[26] Depeursinge A, Vargas A, Platon A, Geissbuhler A, Poletti PA, Muller H. Building a reference multimedia database for institial lung diseases. In: Computerized Medical Imaging and Graphics; 2012. p. 227–238.