Analysis of Feature Maps Selection in Supervised Learning Using Convolutional Neural Networks

Joseph Lin Chu and Adam Krzyżak

Department of Computer Science and Software Engineering, Concordia University, Montreal, Quebec, Canada
jo_chu@encs.concordia.ca, krzyzak@cs.concordia.ca

Abstract. Artificial neural networks have been widely used for machine learning tasks such as object recognition. Recent developments have made use of biologically inspired architectures, such as the Convolutional Neural Network. The nature of the Convolutional Neural Network is that each convolutional layer of the network contains a certain number of feature maps or kernels. The number of these used has historically been determined on an ad-hoc basis. We propose a theoretical method for determining the optimal number of feature maps using the dimensions of the feature map or convolutional kernel. We find that the empirical data suggests that our theoretical method works for extremely small receptive fields, but doesn't generalize as clearly to all receptive field sizes. Furthermore, we note that architectures that are pyramidal rather than equally balanced tend to make better use of computational resources.

1 Introduction

The earliest of the hierarchical Artificial Neural Networks (ANNs) based on the visual cortex's architecture was the Neocognitron, first proposed by Fukushima & Miyake [7]. This network was based on the work of neuroscientists Hubel & Wiesel [8], who showed the existence of Simple and Complex Cells in the visual cortex. Subsequently, LeCun et al [10] developed the Convolutional Neural Network (CNN), which made use of multiple Convolutional and Subsampling layers, while also using stochastic gradient descent and backpropagation to create a feed-forward network that performed exceptionally well on image recognition tasks. The Convolutional Layer of the CNN is equivalent to the Simple Cell Layer of the Neocognitron, while the Subsampling Layer of the CNN is equivalent to the Complex Cell Layer of the Neocognitron. Essentially they delocalize features from the visual receptive field, allowing such features to be identified with a degree of shift invariance. This unique structure allows the CNN to have two important advantages over a fully-connected ANN. First, is the use of the local receptive field, and second is weight-sharing. Both of these advantages have the effect of decreasing the number of weight parameters in the network, thereby making computation of these networks easier.

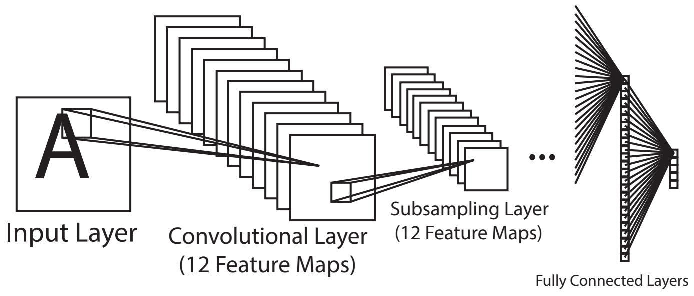

Fig. 1. The basic architecture of the CNN.

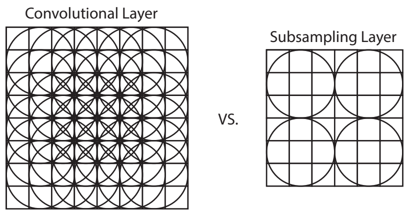

Fig. 2. A comparison between the Convolutional layer and the Subsampling layer. Circles represent the receptive fields of the cells of the layer subsequent to the one represented by the square lattice. On the left, an 8 x 8 input layer feeds into a 6 x 6 convolutional layer using receptive fields of size 3 x 3 with an offset of 1 cell. On the right, a 6 x 6 input layer feeds into a 2 x 2 subsampling layer using receptive fields of size 3 x 3 with an offset of 3 cells.

2 Theoretical Analysis

CNNs have particularly many hyper-parameters due to the structure of the network. Determining the optimal hyper-parameters can appear to be a bit of an art. In particular, the number of feature maps for a given convolutional layer tends to be chosen based on empirical performance rather than on theoretical justifications [14]. Numbers in the first convolutional layer range from very small (3-6) [12], [10], to very large (96-1600) [9], [3], [4], [5].

One wonders then, if there is some sort of theoretical rationale that can be used to determine the optimal number of feature maps, given other hyper-parameters. In particular, one would expect that the dimensions of the receptive field ought to have some influence on this optimum.

A receptive field of width r consists of r^2 elements or nodes. If we have feature maps m then, the maximum number of possible feature maps before duplication, given an 8-bit grey scale image, is 256^{r^2}. Since the difference between a grey level of say, 100 and 101, is roughly negligible, we simplify and reduce the number of bins in the histogram so to speak from 256 to 2. Looking at binary features as a way of simplifying the problem is not unheard of [2]. So, given a binary image the number of possible binary feature maps before duplication is

$$\Omega = 2^{r^2}, \tag{1}$$

which is still a rapidly increasing number.

Let's look at some very simple receptive fields. Take one of size 1x1. How many feature maps would it take before additional maps become completely redundant? Ω would suggest two. For receptive field sizes 2x2, 3x3, and 4x4, we get 16, 512, and 65536, respectively. These values represent an upper bound, beyond which additional feature maps should not improve performance significantly. But clearly, not even all of these feature maps would be all that useful. If we look again at a 2x2 receptive field, regarding those 16 non-redundant feature maps, shown in Figure 3, are all of them necessary? Assuming a higher layer that combines features, many of these are actually redundant in practice.

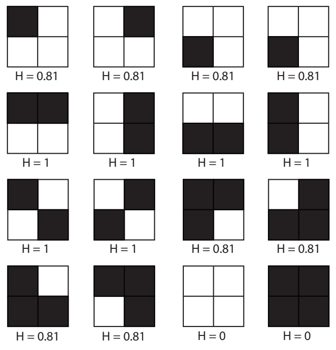

Fig. 3. The 16 possible binary feature maps of a 2x2 receptive field, with their respective entropy values.

So how do we determine which ones are useful? Borrowing from Information Theory, we can look at how much information each map encodes. Consider Shannon entropy or the amount of information given

$$H(X) = -\sum_{i=1}^{n} p(x_i) log_2 p(x_i). \tag{2}$$

We can calculate the Shannon entropy of each feature map, again, shown in Figure 3. What's interesting here is that we can group the Shannon entropy values together. In the 2x2 case, there are six patterns equal to an entropy of 1, while there are eight patterns with an entropy of around 0.81, and two patterns with an entropy of 0. Thus we have three bins of entropy values so to speak.

Thus, we hypothesize a very simple theoretical method, one that admittedly simplifies a very complex problem. Shannon entropy has the obvious disadvantage that it does not tell us about the spatial relationship between neighbouring pixels. And again, we assume binary feature maps. Nevertheless, we propose this as an initial attempt to approximate the underlying reality.

The number of different possible entropy values for the total binary feature map set of a particular receptive field size is determined by considering the number of elements in a receptive field r^2. The number of unique ways you can fill the binary histogram of the possible feature maps then is r^2+1. But roughly half the patterns are inverses of each other. So the actual number of unique entropy values is $(r^2+1)/2$ if r is odd. Or $(r^2)/2+1$ if r is even.

Given the number of different entropy values

$$h(r) = \begin{cases} \frac{r^2+1}{2} & \text{if } r \text{ is odd} \\ \frac{r^2}{2}+1 & \text{if } r \text{ is even} \end{cases} \tag{3}$$

the number of useful feature maps is

$$u = h + s, \tag{4}$$

where s is a term that describes the additional useful feature maps above this minimum of h. We know from the 1x1 receptive field feature map set that s is at least 1, because when $r = 1$, the receptive field is a single pixel filter, and optimally functions as a binary filter. In such a case, $u = \Omega = 2$. In the minimum case that $s = 1$, then in the case of 1x1, $u = 2$. In the case of 2x2, $u = 4$. In the case of 3x3, $u = 6$. This is a lower bound on u that we shall use until we can determine what s actually is.

To understand this, think of a receptive field that takes up the entire image. So for a 100x100 image, the receptive field is 100x100. In such a case, each feature map is in essence a template, and the network performs what is in essence template matching. Thus the number of useful feature maps is the number of useful templates. With enough templates, you can approximate the data set, any more would be unnecessary. To determine how many such templates would be useful, consider the number of different Shannon entropies in the feature set. While it is not guaranteed that two templates with the same entropy would be identical, two templates with different entropies are certainly different. Also consider the difference between that 100x100 receptive field, and a 99x99 receptive field. The differences between the two in terms of number of useful feature maps intuitively seems negligible. This suggests that s is either a constant, or at most a linearly increasing term. Also, as h increases, the distance between various entropies decreases, to the point where many of the values start to become nearly the same. Thus, one can expect that for very high values of r, u will be too high.

Any less than u and the theory predicts a drop in performance. Above this number the theory is agnostic about one of three possible directions. Either the additional feature maps won't affect the predictive power of the model, so we could see a plateau, or the Curse of Dimensionality will cause the predictive power of the model to begin to drop, or as seen in other papers such as [4], the predictive power of the model will continue to increase, albeit at a slower rate.

Thus far we have taken care of the first convolutional layer. For the second convolutional layer and beyond, the question arises of whether or not to stick to this formula for u, or whether it makes more sense to increase the number of feature maps in some proportion to the number in the previous convolutional layer. Upper convolutional layers are not simply taking the pixel intensities, but instead, combining the feature maps of the lower layer. In which case, it makes sense to change the formula for u for upper layers to:

$$u_l = v u_{l-1}, \tag{5}$$

where v is some multiplier value and l is the layer. Candidates for this value range from u_{l-1} itself, to some constant such as 2 or 4.

There is no substitute for empirical evidence, so we test the theory by running experiments to exhaustively search for the hypothetical, optimal number of feature maps.

3 Methodology

To speed up and simplify the experiments, we devised, using the Caltech-101 dataset [6], a specialized dataset, which we shall refer to as the Caltech-20. The Caltech-20 consists of 20 object categories with 50 images per category total, divided into a training set of 40 images per category, and a test set of 10 images per category. The 20 categories were selected by finding the 20 image categories with the most square average dimensions that also had at least 50 example images. The images were also resized to 100 x 100 pixels, with margins created by irregular image dimensions zero-padded (essentially blacked out). To simplify the task so as to have one channel rather than three, the images were also converted to greyscale. The training set totalled 800 images while the test set consisted of 200 images. Some example images are shown in figure 4.

Fig. 4. Images from the Caltech-20 data set.

CNNs tend to require a fairly significant amount of time to train. One way to improve temporal performance is to implement these ANNs such that they are able to use the Graphical Processing Unit (GPU) of certain video cards rather than merely the CPU of a given machine [15], [16], [13], [9]. NVIDIA video cards in particular have a parallel computing platform called Compute Unified Device Architecture (CUDA) that can take full advantage of the many cores on a typical video card to greatly accelerate parallel computing tasks. ANNs are quite parallel in nature, and thus quite amenable to this. Thus, for our implementation of the CNN, we turned to the Python-based Theano library (http://deeplearning.net/software/theano/) [1]. We were able to find appropriate Deep Learning Python scripts for the CNN. Our tests suggest that the speed of the CNN using the GPU improved by a factor of eight, as compared to just using the CPU.

CNNs require special consideration when implementing their architecture. A method was devised to calculate a usable set of architecture parameters. The relationship between layers can be described as follows. To calculate the reasonable dimensions of a square layer from either its previous layer (or next layer) in the hierarchy requires at least some of the following variables to be assigned. Let x be the width of the previous (or current) square layer. Let y be the width of the current (or next) square layer. Let r be the width of the square receptive field of nodes in the previous (or current) layer to each current (or next) layer node, and f be the offset distance between the receptive fields of adjacent nodes in the current (or next) layer. The relationship between these variables is best described by the equation below.

$$y = \frac{x - (r - f)}{f}, \tag{6}$$

where, $x \geq y$, $x \geq r \geq f$, and $f > 0$.

For convolutional layers this generalizes because f = 1, to:

$$y = x - r + 1. \tag{7}$$

For subsampling layers, this generalizes because r = f, to:

$$y = \frac{x}{f}. \tag{8}$$

From this we can determine the dimensions of each layer. To describe a CNN, we adopt a similar convention to [3]. An example architecture for the CNN on the NORB dataset [11] can be written out as:

$96 \times 96 \rightarrow 8C5 \rightarrow S4 \rightarrow 24C6 \rightarrow S3 \rightarrow 24C6 \rightarrow 100N \rightarrow 5N$,

where the number before C is the number of feature maps in a convolutional layer, the number after C is the receptive field width in a convolutional layer, the number after S is the receptive field width of a subsampling layer, and the number before N is the number of nodes in a fully connected layer.

For us to effectively test a single convolutional layer, we use a series of architectures, where v is a variable number of feature maps:

$100 \times 100 \rightarrow vC1 \rightarrow S2 \rightarrow 500N \rightarrow 20N$

$100 \times 100 \rightarrow vC2 \rightarrow S3 \rightarrow 500N \rightarrow 20N$

$100 \times 100 \rightarrow vC3 \rightarrow S2 \rightarrow 500N \rightarrow 20N$

$100 \times 100 \rightarrow vC5 \rightarrow S3 \rightarrow 500N \rightarrow 20N$

$100 \times 100 \rightarrow vC99 \rightarrow S1 \rightarrow 500N \rightarrow 20N$

The reason why we sometimes use 3x3 subsampling receptive fields is that the size of the convolved feature maps are divisible by 3 but not 2. Otherwise we choose to use 2x2 subsampling receptive fields where possible. We find that, with the exception of the unique 99x99 receptive field, not using subsampling produces too many features and parameters and causes the network to have difficulty learning. The method of subsampling we use is max-pooling, which involves taking the maximum value seen in the receptive field of the subsampling layer.

For testing multiple convolutional layers, we use the following architecture, where v_i is a variable number of feature maps for each layer:

$100 \times 100 \rightarrow v_1C2 \rightarrow S3 \rightarrow v_2C2 \rightarrow S2 \rightarrow v_3C2 \rightarrow S3 \rightarrow v_4C2 \rightarrow S2 \rightarrow v_5C2 \rightarrow S1 \rightarrow 500N \rightarrow 20N$

All our networks use the same basic classifier, which is a multi-layer Perceptron with 500 hidden nodes and 20 output nodes. Various other parameters for the CNN were also experimented with to determine the optimal parameters to use in our experiments. We eventually settled on 100 epochs of training. The CNN learning rate and learning rate decrement parameters were determined by trial and error. The learning rate was initially set to 0.1, and gradually decremented to approximately 0.001.

4 Analysis and Results

The following figures are intentionally fitted with a trend line that attempts to test the hypothesis that a cubic function approximates the data. It should not be construed to suggest that this is in fact the underlying function.

Figure 5 shows the results of experimenting with different numbers of feature maps on the accuracy of the CNN trained on the Caltech-20, and using a 1x1 receptive field.

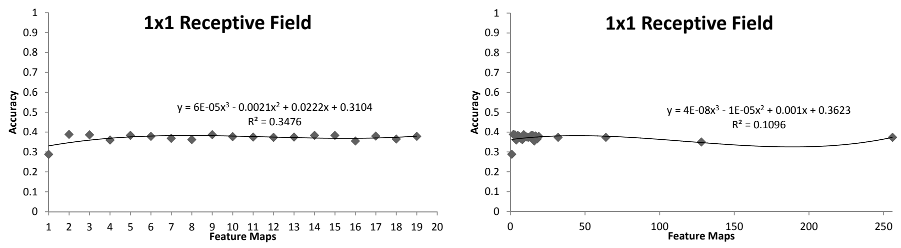

Fig. 5. Graphs of the accuracy given a variable number of feature maps for a 1x1 receptive field.

As can be seen, the accuracy quickly increases between 1 and 2 feature maps, and then levels off for more than 2 feature maps. This is consistent with the theory, albeit, the plateau beyond seems to be neither increasing nor decreasing, which suggests some kind of saturation point around 2.

Figure 6 shows the results of experimenting with different numbers of feature maps on the accuracy of the CNN trained on the Caltech-20, and using a 2x2 receptive field.

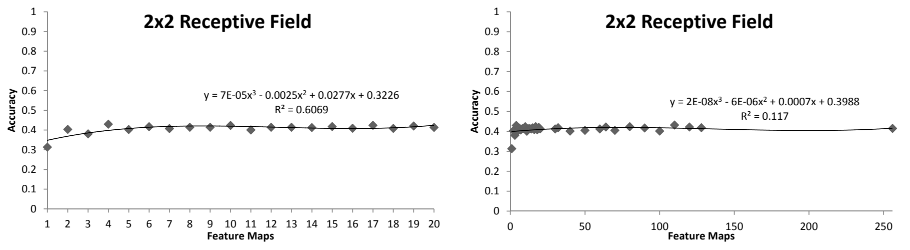

Fig. 6. Graphs of the accuracy given a variable number of feature maps for a 2x2 receptive field.

As can be seen, the accuracy quickly increases between 1 and 4 feature maps, and then levels off for more than 4 feature maps. This is consistent with the theory where $s = 1$. The plateau beyond seems to be neither increasing nor decreasing, which suggests some kind of saturation point around 4.

Figure 7 shows the results of experimenting with different numbers of feature maps on the accuracy of the CNN trained on the Caltech-20, and using a 3x3 receptive field.

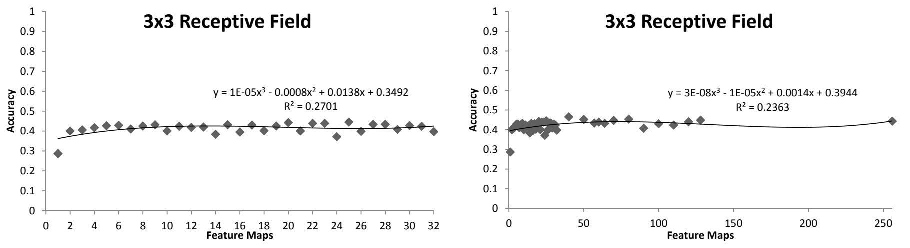

Fig. 7. Graphs of the accuracy given a variable number of feature maps for a 3x3 receptive field.

As can be seen, the accuracy increases between 1 and 6 feature maps, and then proceeds to plateau somewhat erratically. Unlike the previous receptive field sizes however, there are accuracies greater than that found at u. The plateau also appears less stable.

Figure 8 shows the results of experimenting with different numbers of feature maps on the accuracy of the CNN trained on the Caltech-20, and using a 5x5 receptive field.

As can be seen, the accuracy quickly increases between 1 and 4 feature maps, and then plateaus for a while before spreading out rather chaotically. This may

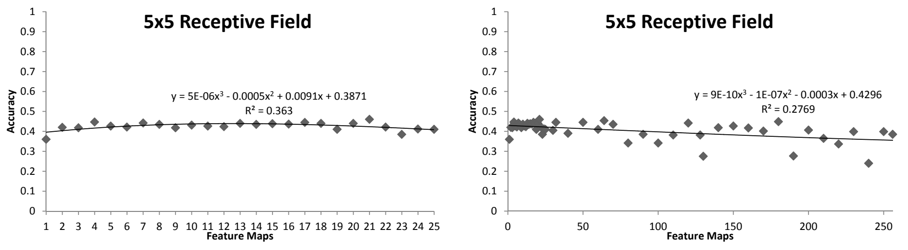

Fig. 8. Graphs of the accuracy given a variable number of feature maps for a 5x5 receptive field.

be explained by some combination of overfitting, the curse of dimensionality, or too high a learning rate causing failure to converge.

Figure 9 show the results of experimenting with different numbers of feature maps on the accuracy of the CNN trained on the Caltech-20, and using a 99x99 receptive field.

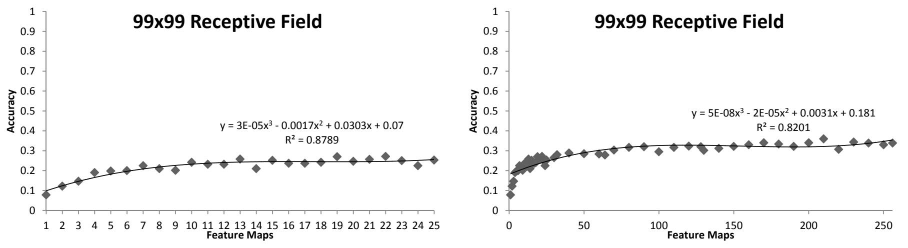

Fig. 9. Graphs of the accuracy given a variable number of feature maps for a 99x99 receptive field.

As can be seen, the accuracy steadily increases between 1 and 210 feature maps, and then begins to plateau. As expected, our u value of 4901 is much too high for the very large value of $r = 99$.

Lastly, we look at the effect of multiple convolutional layers. Figure 10 shows what happens when the first layer of a 12 layer CNN with 5 convolutional layers is held constant at 4 feature maps, and the upper layers are multiplied by the number in the layer before it. So for a multiple v of 2, the feature maps for each layer would be 4, 8, 16, 32, 64, respectively. We refer to this as the pyramidal structure.

Clearly, increasing the number of feature maps in the upper layers has a significant impact. Perhaps not coincidentally, the best performing architecture we have encountered so far was this architecture with 4, 20, 100, 500, and 2500

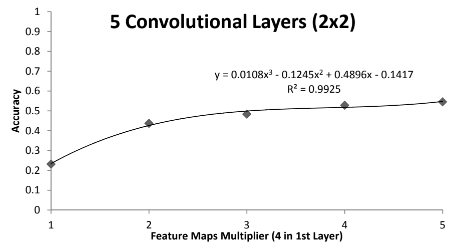

Fig. 10. Graph of the accuracy given a variable number of feature maps for a network with 5 convolutional layers of 2x2 receptive field. Here the higher layers are a multiple of the lower layers.

feature maps in each respective layer. It achieved an accuracy of 54.5% on our Caltech-20 data set. This can be contrasted with the effect of having the same number of feature maps in each layer as shown in figure 11. We refer to this as the equal structure.

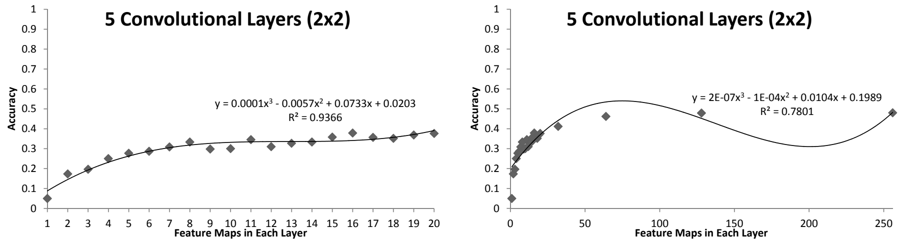

Fig. 11. Graph of the accuracy given a variable number of feature maps for a network with 5 convolutional layers of 2x2 receptive field. Here each layer has the same number of feature maps.

While the accuracy rises with the number of feature maps as well, it should be noted that for the computational cost, the pyramidal structure appears to be a better use of resources than the equal structure.

5 Discussion

It appears that the theoretical method seems to hold well for receptive fields of size 1x1, and 2x2. For larger sizes, the data is not as clear. The data from the 3x3 and 5x5 receptive field experiments suggests that there can be complicating factors involved that cause the data to spread. Such factors could include, the curse of dimensionality, and also, technical issues such as failure to converge due to too high a learning rate, or overfitting. As our experimental set up is intentionally very simple, we lack many of the normalizing methods that might otherwise improve performance. The data from the 99x99 receptive field experiment is interesting because it starts to plateau much sooner than predicted by the equation for u. However, we mentioned before that this would probably happen with the current version of u. The number of different entropies at $r = 99$ are probably very close together and an improved u equation should take this into account.

It should also be emphasized that our results could be particular to our choice of hyper-parameters such as learning rate and our choice of a very small dataset.

Nevertheless, what we do not find, is the clear and simple monotonically increasing function seen in [4], and [5]. Rather, the data shows that after an initial rise, the function seems to plateau and it is uncertain whether it can be construed to be rising or falling or stable.

This is not the case with highly layered networks however, which do appear to show a monotonically increasing function in terms of increasing the number of feature maps. However, this could well be due to the optimal number of feature maps in the last layer being exceedingly high due to multiplier effects.

One thing that could considerably improve our work would be finding some kind of measure of spatial entropy rather than relying on Shannon entropy. The problem with Shannon entropy is of course, that it does not consider the potential information that comes from the arrangement of neighbouring pixels. We might very well improve our estimates of u by taking into consideration the spatial entropy in h, rather than relying on the s term.

Future work should likely include looking at what the optimal receptive field size is. Our experiments hint at this value as being greater than 3x3 and [4] suggests that it is less than 8x8, but performing the exhaustive search without knowing the optimal number of feature maps for each receptive field size is a computationally complex task.

As with [9], we find that more convolutional layers seems to improve performance. The optimal number of such layers is something else that should be looked at in the future.

6 Conclusions

Our experiments provided some additional data to consider for anyone interested in optimizing a CNN. Though the theoretical method is not clear beyond certain extremely small or extremely large receptive fields, it does suggest that there is some relationship between the receptive field size and the number of useful feature maps in a given convolutional layer of a CNN. It nevertheless may prove to be a useful approximation.

Our experiments also suggest that when comparing architectures with equal numbers of feature maps in each layer with architectures that have pyramidal schemes where the number of feature maps increase by some multiple, that the pyramidal methods are a more effective use of computing resources.

References

1. Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., Bengio, Y.: Theano: a CPU and GPU math expression compiler. In: Proceedings of the Python for Scientific Computing Conference (SciPy) (Jun 2010), oral Presentation
2. Boureau, Y.L., Ponce, J., LeCun, Y.: A theoretical analysis of feature pooling in visual recognition. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10). pp. 111–118 (2010)
3. Ciresan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. pp. 3642–3649. IEEE (2012)
4. Coates, A., Ng, A.Y., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: International Conference on Artificial Intelligence and Statistics. pp. 215–223 (2011)
5. Eigen, D., Rolfe, J., Fergus, R., LeCun, Y.: Understanding Deep Architectures using a Recursive Convolutional Network. ArXiv e-prints (Dec 2013)
6. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. IEEE. CVPR 2004, Workshop on Generative-Model Based Vision. (2004)
7. Fukushima, K., Miyake, S.: Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. Pattern Recognition 15(6), 455–469 (1982)
8. Hubel, D.H., Wiesel, T.N.: Receptive fields, binocular interaction and functional architecture in a cats visual cortex. Journal of Physiology (London) 160, 106–154 (1962)
9. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems 25. pp. 1106–1114 (2012)
10. LeCun, Y., Bottou, L., Bengio, Y., P., H.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
11. LeCun, Y., Huang, F., Bottou, L.: Learning methods for generic object recognition with invariance to pose and lighting. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) 2, 97–104 (2004)
12. Nguyen, G.H., Phung, S.L., Bouzerdoum, A.: Reduced training of convolutional neural networks for pedestrian detection. In: International Conference on Information Technology and Applications (2009)
13. Scherer, D., Schulz, H., Behnke, S.: Accelerating large-scale convolutional neural networks with parallel graphics multiprocessors. In: Artificial Neural Networks–ICANN 2010, pp. 82–91. Springer (2010)
14. Simard, P., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: ICDAR. vol. 3, pp. 958–962 (2003)
15. Uetz, R., Behnke, S.: Large-scale object recognition with cuda-accelerated hierarchical neural networks. In: Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on. vol. 1, pp. 536–541. IEEE (2009)
16. Uetz, R., Behnke, S.: Locally-connected hierarchical neural networks for gpu-accelerated object recognition. In: NIPS 2009 Workshop on Large-Scale Machine Learning: Parallelism and Massive Datasets (2009)