Deep Networks with Internal Selective Attention through Feedback Connections

Marijn Stollenga `marijn@idsia.ch`[*],
Jonathan Masci `jonathan@idsia.ch`[*],
Faustino Gomez `tino@idsia.ch`, and
Juergen Schmidhuber `juergen@idsia.ch`

[*]Shared first author.

June 24, 2019

Abstract

Traditional convolutional neural networks (CNN) are stationary and feedforward. They neither change their parameters during evaluation nor use feedback from higher to lower layers. Real brains, however, do. So does our Deep Attention Selective Network (dasNet) architecture. DasNets feedback structure can dynamically alter its convolutional filter sensitivities during classification. It harnesses the power of sequential processing to improve classification performance, by allowing the network to iteratively focus its internal attention on some of its convolutional filters. Feedback is trained through direct policy search in a huge million-dimensional parameter space, through scalable natural evolution strategies (SNES). On the CIFAR-10 and CIFAR-100 datasets, dasNet outperforms the previous state-of-the-art model.

1 Introduction

Deep convolutional neural networks (CNNs) [1] with max-pooling layers [2] trained by backprop [3–5] on GPUs [6] have become the state-of-the-art in object recognition [7–10], segmentation/detection [11, 12], and scene parsing [13–15] (for an extensive review see [16]). These architectures consist of many stacked feedforward layers, mimicking the bottom-up path of the human visual cortex, where each layer learns progressively more abstract representations of the input data. Low-level stages tend to learn biologically plausible feature detectors, such as Gabor filters [17]. Detectors in higher layers learn to respond to concrete visual objects or their parts, e.g., [18–21]. Once trained, the CNN never changes its weights or filters during evaluation.

Evolution has discovered efficient feedforward pathways for recognizing certain objects in the blink of an eye. However, an expert ornithologist, asked to classify a bird belonging to one of two very similar species, may have to think for more than a few milliseconds before answering [22, 23], implying that several feedforward evaluations are performed, where each evaluation tries to elicit different information from the image.

Since humans benefit greatly from this strategy, we hypothesize CNNs can too. This requires: (1) the formulation of a non-stationary CNN that can adapt its own behaviour post-training, and (2) a process that decides *how* to adapt the CNNs behaviour.

This paper introduces Deep Attention Selective Networks (dasNet) which model selective attention in deep CNNs by allowing each layer to influence all other layers on successive passes over an image through special connections (both bottom-up and top-down), that modulate the activity of the convolutional filters. The weights of these special connections implement a control policy that is learned through reinforcement learning *after* the CNN has been trained in the usual way via supervised learning. Given an input image, the attentional policy can enhance or suppress features over multiple passes to improve the classification of difficult cases not captured by the initially supervised training. Our aim is to let the system check the usefulness of internal CNN filters automatically, omitting manual inspection [24].

In our current implementation, the attentional policy is evolved using Separable Natural Evolution Strategies (SNES; [25]), instead of a conventional, single agent reinforcement learning method (e.g. value iteration, temporal difference, policy gradients, etc.) due to the large number of parameters (over 1 million) required to control CNNs of the size typically used in image classification. Experiments on CIFAR-10 and CIFAR100 [26] show that on difficult classification instances, the network corrects itself by emphasizing and de-emphasizing certain filters, outperforming a previous state-of-the-art CNN.

2   Maxout Networks

In this work we use the Maxout networks [10], combined with dropout [27], as the underlying model for dasNet. Maxout networks represent the state-of-the-art for object recognition in various tasks and have only been outperformed (by a small margin) by averaging committees of several convolutional neural networks. A similar approach, which does not reduce dimensionality in favor of sparsity in the representation has also been recently presented [28]. Maxout CNNs consist of a stack of alternating convolutional and maxout layers, with a final classification layer on top:

Convolutional Layer.   The input to this layer can be an image or the output of a previous layer, consisting of c input maps of width m and height n: $x \in \mathbb{R}^{c \times m \times n}$. The output consists of a set of c' output maps: $y \in \mathbb{R}^{c' \times m' \times n'}$. The convolutional layer is parameterized by $c \cdot c'$ filters of size $k \times k$. We denote the filters by $F_{i,j}^{\ell} \in \mathbb{R}^{k \times k}$, where i and j are indexes of the input and output maps and ℓ denotes the layer.

$$y_j^{\ell} = \sum_{i=0}^{i=c} \phi(x_i * F_{i,j}^{\ell}) \qquad (1)$$

where i and j index the input and output map respectively, $*$ is the convolutional operator, ϕ is an element-wise nonlinear function, and ℓ is used to index the layer. The size of the output is determined by the kernel size and the stride used for the convolution (see [10]).

Pooling Layer.   A pooling layer is used to reduced the dimensionality of the output from a convolutional layer. The usual approach is to take the maximum value among non- or partially-overlapping patches in every map, therefore reducing dimensionality

along the height and width [2]. Instead, a Maxout pooling layer reduces every b consecutive maps to one map, by keeping only the maximum value for every pixel-position, where b is called the block size. Thus the map reduces c input maps to $c' = c/b$ output maps.

$$y^\ell_{j,x,y} = \max_{i=0}^{b} y^{\ell-1}_{j \cdot b+i,x,y} \qquad (2)$$

where $y^\ell \in \mathbb{R}^{c' \times m' \times n'}$, and ℓ again is used to index the layer. The output of the pooling layer can either be used as input to another pair of convolutional- and pooling layers, or form input to a final classification layer.

Classification Layer. Finally, a classification step is performed. First the output of the last pooling layer is flattened into one large vector $\vec{x}$, to form the input to the following equations:

$$\bar{y}^\ell_j = \max_{i=0..b} F^\ell_{j \cdot b+i} \vec{x} \qquad (3)$$

$$\mathbf{v} = \sigma(F^{\ell+1} \bar{y}^\ell) \qquad (4)$$

where $F^\ell \in \mathbb{R}^{N \times |\vec{x}|}$ (N is chosen), and $\sigma(\cdot)$ is the softmax activation function which produces the class probabilities $\mathbf{v}$. The input is projected by F and then reduced using a maxout, similar to the pooling layer (3).

3   Reinforcement Learning

Reinforcement learning (RL) is a general framework for learning to make sequential decisions order to maximize an external reward signal [29, 30]. The learning agent can be anything that has the ability to *act* and *perceive* in a given environment.

At time t, the agent receives an observation $o_t \in O$ of the current state of the environment $s_t \in S$, and selects an action, $a_t \in A$, chosen by a policy $\pi : O \to A$, where S, O and A the spaces of all possible states, observations, and action, respectively.[1] The agent then enters state s_{t+1} and receives a reward $r_t \in \mathbb{R}$. The objective is to find the policy, π, that maximizes the expected future discounted reward, $E[\sum_t \gamma^t r_t]$, where $\gamma \in [0, 1]$ discounts the future, modeling the "farsightedness" of the agent.

In dasNet, both the observation and action spaces are real valued $O = \mathbb{R}^{dim(O)}$, $A = \mathbb{R}^{dim(A)}$. Therefore, policy π_θ must be represented by a function approximator, e.g. a neural network, parameterized by θ. Because the policies used to control the attention of the dasNet have state and actions spaces of close to a thousand dimensions, the policy parameter vector, θ, will contain close to a million weights, which is impractical for standard RL methods. Therefore, we instead evolve the policy using a variant for Natural Evolution Strategies (NES; [31, 32]), called Separable NES (SNES; [25]). The NES family of black-box optimization algorithms use parameterized probability distributions over the search space, instead of an explicit population (i.e., a conventional ES [33–35]). Typically, the distribution is a multivariate Gaussian parameterized by mean μ and covariance matrix Σ. Each epoch a generation is sampled from the distribution, which is then updated the direction of the natural gradient of the expected fitness of the distribution. SNES differs from standard NES in that

---

[1]In this work $\pi : O \to A$ is a deterministic policy; given an observation it will always output the same action. However, π could be extended to stochastic policies.

Algorithm 1 TRAIN DASNET ($\mathbf{M}$, μ, Σ, p, n)

1: **while** True **do**
2:    $images \Leftarrow \text{NEXTBATCH}(n)$
3:    **for** $i = 0 \rightarrow p$ **do**
4:       $\theta_i \sim \mathbb{N}(\mu, \Sigma)$
5:       **for** $j = 0 \rightarrow n$ **do**
6:          $\mathbf{a}_0 \Leftarrow \mathbb{1}$ {Initialize gates a with identity activation}
7:          **for** $t = 0 \rightarrow T$ **do**
8:             $\mathbf{v}_t = \mathbf{M}_t(\theta_i, x_i)$
9:             $\mathbf{o}_t \Leftarrow h(\mathbf{M}_t)$
10:            $\mathbf{a}_{t+1} \Leftarrow \pi_{\theta_i}(\mathbf{o}_t)$
11:          **end for**
12:          $L_i = -\lambda_{\text{boost}} d \log(\mathbf{v}_T)$
13:       **end for**
14:       $\mathcal{F}[i] \Leftarrow f(\theta_i)$
15:       $\Theta[i] \Leftarrow \theta_i$
16:    **end for**
17:    $\text{UPDATESNES}(\mathcal{F}, \Theta)$
18: **end while**

---

instead of maintaining the full covariance matrix of the search distribution, uses only the diagonal entries. SNES is theoretically less powerful than standard NES, but is substantially more efficient.

4 Deep Attention Selective Networks (dasNet)

The idea behind dasNet is to harness the power of sequential processing to improve classification performance by allowing the network to iteratively focus the attention of its filters. First, the standard Maxout net (see Section 2) is augmented to allow the filters to be weighted differently on different passes over the same image (compare to equation 1):

$$y_j^\ell = a_j^\ell \sum_{i=0}^{i=c} \phi(x_i * F_{i,j}^\ell), \tag{5}$$

where a_j^ℓ is the weight of the j-th output map in layer ℓ, changing the strength of its activation, *before* applying the maxout pooling operator. The vector $\mathbf{a} = [a_0^0, a_1^0, \cdots, a_{c'}^0, a_0^1, \cdots, a_{c'}^1, \cdots]$ represents the action that the learned policy must select in order to sequentially focus the attention of the Maxout net on the most discriminative features in the image being processed. Changing action $\mathbf{a}$ will alter the behaviour of the CNN, resulting in different outputs, even when the image x does not change. We indicate this with the following notation:

$$\mathbf{v}_t = \mathbf{M}_t(\theta, x) \tag{6}$$

where θ is the parameter vector of the policy, π_θ, and $\mathbf{v}_t$ is the output of the network on pass t.

    Algorithm 1 describes the dasNet training algorithm. Given a Maxout net, $\mathbf{M}$, that has already been trained to classify images using training set, X, the policy, π,

is evolved using SNES to focus the attention of $\mathbf{M}$. Each pass through the `while` loop represents one generation of SNES. Each generation starts by selecting a subset of n images from X at random. Then each of the p samples drawn from the SNES search distribution (with mean μ and covariance Σ) representing the parameters, θ_i, of a candidate policy, π_{θ_i}, undergoes n trials, one for each image in the batch. During a trial, the image is presented to the Maxout net T times. In the first pass, $t = 0$, the action, $\mathbf{a}_0$, is set to $a_i = 1, \forall i$, so that the Maxout network functions as it would normally — the action has no effect. Once the image is propagated through the net, an observation vector, $\mathbf{o}_0$, is constructed by concatenating the following values extracted from $\mathbf{M}$, by $h(\cdot)$:

1. the average activation of *every* output map $Avg(y_j)$ (Equation 2), of each Maxout layer.

2. the intermediate activations $\bar{y}_j$ of the classification layer.

3. the class probability vector, $\mathbf{v}_t$.

While averaging map activations provides only partial state information, these values should still be meaningful enough to allow for the selection of good actions. The candidate policy then maps the observation to an action:

$$\pi_{\theta_i}(\mathbf{o}) = dim(A)\sigma(\boldsymbol{\theta}_i\mathbf{o_t}) = \mathbf{a_t}, \tag{7}$$

where $\boldsymbol{\theta} \in \mathbb{R}^{dim(A) \times dim(O)}$ is the weight matrix of the neural network, and σ is the softmax. Note that the softmax function is scaled by the dimensionality of the action space so that elements in the action vector average to 1 (instead of regular softmax which *sums* to 1), ensuring that all network outputs are positive, thereby keeping the filter activations stable.

On the next pass, the same image is processed again, but this time using the filter weighting, $\mathbf{a}_1$. This cycle is repeated until pass T (see figure 1 for a illustration of the process), at which time the performance of the network is scored by:

$$L_i = -\lambda_{\text{boost}} d \log(\mathbf{v}_T) \tag{8}$$

$$\mathbf{v}_T = \mathbf{M}_T(\theta_i, x_i) \tag{9}$$

$$\lambda_{\text{boost}} = \begin{cases} \lambda_{\text{correct}} & \text{if } d = \|\mathbf{v}_T\|_\infty \\ \lambda_{\text{misclassified}} & \text{otherwise,} \end{cases} \tag{10}$$

where $\mathbf{v}$ is the output of $\mathbf{M}$ at the end of the pass T, d is the correct classification, and $\lambda_{correct}$ and $\lambda_{misclassified}$ are constants. L_i measures the weighted loss, where misclassified samples are weighted higher than correctly classified samples $\lambda_{misclassified} > \lambda_{correct}$. This simple form of boosting is used to focus on the 'difficult' misclassified images. Once all of the input images have been processed, the policy is assigned the fitness:

$$f(\theta_i) = \overbrace{\sum_{i=1}^{n} L_i}^{\text{cumulative score}} + \overbrace{\lambda_{L2}\|\theta_i\|_2}^{\text{regularization}} \tag{11}$$
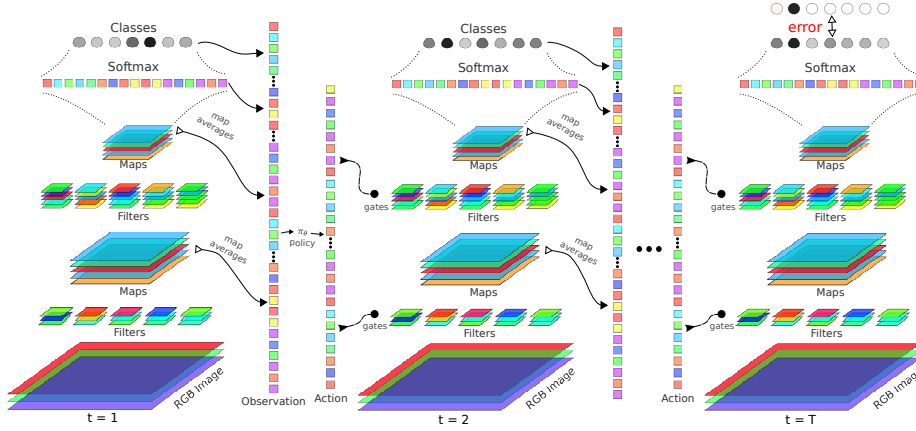
Figure 1: The dasNet Network. Each image in classified after T passes through the network. After each forward propagation through the Maxout net, the output classification vector, the output of the second to last layer, and the averages of all feature maps, are combined into an observation vector that is used by a deterministic policy to choose an action that changes the weights of all the feature maps for the next pass of the same image. After pass T, the output of the Maxout net is finally used to classify the image.

where λ_{L2} is a regularization parameter.

Once all of the candidate policies have been evaluated, SNES updates its distribution parameters (μ, Σ) according the natural gradient calculated from the sampled fitness values, $\mathcal{F}$. As SNES repeatedly updates the distribution over the course of many generations, the expected fitness of the distribution improves, until the stopping criterion is met.

5   Related Work

Human vision is still the most advanced and flexible perceptual system known. Architecturally, visual cortex areas are highly connected, including direct connections over multiple levels and top-down connections. Felleman and Van Essen [36] constructed a (now famous) hierarchy diagram of 32 different visual cortical areas in macaque visual cortex. About 40% of all pairs of areas were considered connected, and most connected areas were connected bidirectionally. The top-down connections are more numerous than bottom-up connections, and generally more diffuse [37]. They are thought to play primarily a modulatory role, while feedforward connections serve as directed information carriers [38].

Analysis of response latencies to a newly-presented image lends credence to the theory that there are two stages of visual processing: a fast, pre-attentive phase, due to feedforward processing, followed by an attentional phase, due to the influence of recurrent processing [39]. After the feedforward pass, we can recognize and localize simple salient stimuli, which can "pop-out" [40], and response times do not increase regardless of the number of distractors. However, this effect has only been conclusively shown for basic features such as color or orientation; for categorical stimuli or faces, whether there is a pop-out effect remains controversial [41, 42]. Regarding the attentional phase, feedback connections are known to play important roles, such as in feature grouping [43], in differentiating a foreground from its background, (especially

6

when the foreground is not highly salient [44, 45]), and perceptual filling in [46]. Work by Bar et al. [47] supports the idea that top-down projections from prefrontal cortex play an important role in object recognition by quickly extracting low-level spatial frequency information to provide an initial guess about potential categories, forming a top-down expectation that biases recognition. Recurrent connections seem to rely heavily on competitive inhibition and other feedback to make object recognition more robust [48, 49].

In the context of computer vision, RL has been shown to be able to learn saccades in visual scenes to learn selective attention [50], learn feedback to lower levels [51, 52], and improve face recognition [53–55]. It has been shown to be effective for object recognition [56], and has also been combined with traditional computer vision primitives [57]. Iterative processing of images using recurrency has been successfully used for image reconstruction [58] and face-localization [59]. All these approaches show that recurrency in processing and an RL perspective can lead to novel algorithms that improve performance. However, this research is often applied to simplified datasets for demonstration purposes due to computation constraints, and are not aimed at improving the state-of-the-art. In contrast, we apply this perspective directly to the known state-of-the-art neural networks to show that this approach is now feasible and actually increases performance.

6   Experiments on CIFAR-10/100

The experimental evaluation of dasNet focuses on ambiguous classification cases in the CIFAR-10 and CIFAR-100 data sets where, due to a high number of common features, two classes are often mistaken for each other. These are the most interesting cases for our approach. By learning on top of an already trained model, dasNet must aim at fixing these erroneous predictions without disrupting, or forgetting, what has been learned.

The CIFAR-10 dataset [26] is composed of 32×32 color images split into 5×10^4 training and 10^4 testing samples, where each image is assigned to one of 10 classes. The CIFAR-100 is similarly composed, but contains 100 classes.

The number of steps, T, for the RL was experimentally determined and fixed at 5; enough steps to allow dasNet to adapt while being small enough to be practical. While it is be possible to iterate until some condition is met, this could be a serious limitation in real-time applications where predictable processing latency is critical. In all experiments we set $\lambda_{\text{correct}} = 0.005$, $\lambda_{\text{misclassified}} = 1$ and $\lambda_{\text{L2}} = 0.005$.

The Maxout network, $\mathbf{M}$, used in the experiments was trained with data augmentation following the suggested global contrast normalization and ZCA normalization protocol. The model consists of three convolutional maxout layers followed by a fully connected maxout and softmax outputs. Dropout of 0.5 was used in all layers except the input layer, and 0.2 for the input layer. The population size for SNES was set to 50.

Table 1 shows the performance of dasNet vs. other methods, where it achieves a relative improvement of 6% with respect to the vanilla CNN. This establishes a new state-of-the-art result for this challenging dataset.

Figure 3 shows the classification of a cat-image from the test-set. All output map activations in the final step are shown at the top. The difference in activations compared to the first step, i.e., the (de-)emphasis of each map, is shown on the bottom. On the left are the class probabilities for each time-step. At the first step, the classification is 'dog', and the cat could indeed be mistaken for a puppy. Note that in the first step,

7

Method	CIFAR-10	CIFAR-100
Dropconnect [9]	9.32%	-
Stochastic Pooling [60]	15.13%	-
Multi-column CNN [7]	11.21%	-
Maxout [10]	9.38%	38.57%
Maxout (our model)	9.61%	34.54%
dasNet	**9.22%**	**33.78%**

Table 1: Classification results on CIFAR-10 and CIFAR-100 datasets. The error on the test-set is shown for several methods. Note that the result for Dropconnect is the average of 12 models. Our method improves over the state-of-the-art reference implementation to which feedback connections are added.
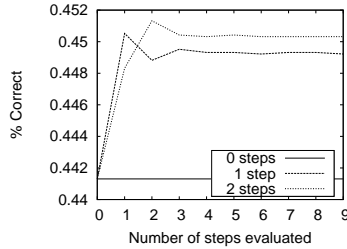

Figure 2: Two dasNets were trained on CIFAR-100 for different values of T. Then they were allowed to run for [0..9] iterations for each image. The performance peaks at the number of steps that the network is trained on, after which the performance drops, but does not explode, showing the dynamics are stable.

the network has not yet received any feedback. In the next step, the probability for 'cat' goes up dramatically, beating 'dog', and subsequently drops a bit in the following steps. The network has successfully disambiguated a cat from a dog. If we investigate the filters, we see that already in the lower layer emphasis changes significantly. Some filters focus more on surroundings whilst others de-emphasize the eyes.

In the second layer, almost all output maps are emphasized. In the third and highest convolutional layer, the most complex changes to the network. At this level the positional correspondence is largely lost, and the filters are known to code for 'higher level' features. It is in this layer that changes are the most influential because they are closest to the final output layers. It is hard to analyze the effect of the alterations, but we can see that the differences are not simple increases or decreases of the output maps, as we then would expect the final activations and their corresponding increases to be largely similar. Instead we see complex emphasis and pattern suppression.

Dynamics To investigate the dynamics, a small 2-layer dasNet network was trained for different values of T. Then they were evaluated by allowing them to run for [0..9] steps. Figure 2 shows results of training dasNet on CIFAR-100 for $T = 1$ and $T = 2$. The performance goes up from the vanilla CNN, peaks at the $step = T$ as expected, and reduces but stays stable after that. So even though the dasNet was trained using only a small number of steps, the dynamics stay stable when these are evaluated for as many as 10 steps.

To verify whether the dasNet policy is actually making good use of its gates, their information content is estimated the following way: The gate values in the last step are taking and used directly for classification. If the gates are used properly then their activation should contain information that is relevant for classification and we would expect a

dasNet that was trained with $T = 2$ and are used as features for classification. Then using *only* the final gate-values (so without e.g. the output of the classification layer), a classification using 15-nearest neighbour and logistic regression was performed. This resulted in a performance of *40.70%* and *45.74%* correct respectively, similar to the
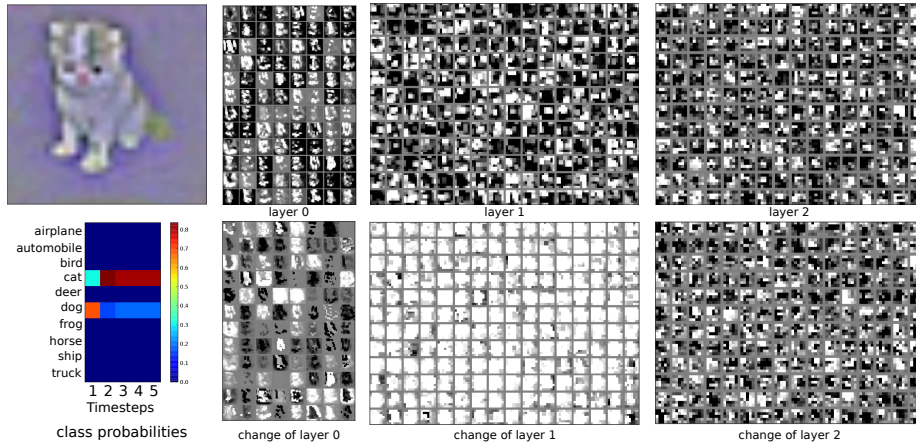
Figure 3: The classification of a cat by the dasNet is shown. All output map activations in the final step are shown on the top. Their changes relative to initial activations in the first step are shown at the bottom (white = emphasis, black = suppression). The changes are normalized to show the effects more clearly. The class probabilities over time are shown on the left. The network first classifies the image as a dog (wrong) but corrects itself by emphasizing its convolutional filters to see it is actually a cat.

performance of dasNet, confirming that they contain significant information and we can conclude that they are purposefully used.

7 Conclusion

DasNet is a deep neural network with feedback connections that are learned by through reinforcement learning to direct selective internal attention to certain features extracted from images. After a rapid first shot image classification through a standard stack of feedforward filters, the feedback can actively alter the importance of certain filters "in hindsight", correcting the initial guess via additional internal "thoughts".

DasNet successfully learned to correct image misclassifications produced by a fully trained feedforward Maxout network. Its active, selective, internal spotlight of attention enabled state-of-the-art results.

Future research will also consider more complex actions that spatially focus on (or alter) parts of observed images.

Acknowledgments

References

[1] K. Fukushima. Neural network model for a mechanism of pattern recognition unaffected by shift in position - Neocognitron. *Trans. IECE*, J62-A(10):658–665, 1979.

[2] Juyang Weng, Narendra Ahuja, and Thomas S Huang. Cresceptron: a self-organizing neural network which grows adaptively. In *International Joint Conference on Neural Networks (IJCNN)*, volume 1, pages 576–581. IEEE, 1992.

[3] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Back-propagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[4] M.A. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 2007. doi: 10.1109/CVPR.2007.383157.

[5] D. Scherer, A. Müller, and S. Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *Proc. International Conference on Artificial Neural Networks (ICANN)*, pages 92–101, 2010.

[6] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *Intl. Joint Conference on Artificial Intelligence IJCAI*, pages 1237–1242, 2011.

[7] D. C. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition CVPR 2012*, 2012. Long preprint arXiv:1202.2745v1 [cs.CV].

[8] Alex Krizhevsky, I Sutskever, and G. E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS 2012)*, page 4, 2012.

[9] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013.

[10] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *International Conference on Machine Learning (ICML)*, 2013.

[11] Dan Claudiu Ciresan, Alessandro Giusti, Luca Maria Gambardella, and Jürgen Schmidhuber. Mitosis detection in breast cancer histology images with deep neural networks. In *Proc. MICCAI*, volume 2, pages 411–418, 2013.

[12] Dan Claudiu Ciresan, Alessandro Giusti, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *NIPS*, pages 2852–2860, 2012.

[13] Clément Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1915–1929, 2013.

[14] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR'13)*. IEEE, 2013.

[15] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[16] J. Schmidhuber. Deep learning in neural networks: An overview. Technical Report IDSIA-03-14 / arXiv:1404.7828v1 [cs.NE], The Swiss AI Lab IDSIA, 2014.

[17] Dennis Gabor. Theory of communication. Part 1: The analysis of information. *Electrical Engineers-Part III: Journal of the Institution of Radio and Communication Engineering*, 93(26):429–441, 1946.

[18] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

[19] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. 2013.

[20] Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *2011 International Conference on Computer Vision*, pages 2018–2025, 2011.

[21] Quoc Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg Corrado, Jeff Dean, and Andrew Ng. Building high-level features using large scale unsupervised learning. In *ICML*, 2012.

[22] Steve Branson, Catherine Wah, Florian Schroff, Boris Babenko, Peter Welinder, Pietro Perona, and Serge Belongie. Visual recognition with humans in the loop. In *Computer Vision–ECCV 2010*, pages 438–451. Springer, 2010.

[23] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[24] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. Technical Report arXiv:1311.2901 [cs.CV], NYU, 2013.

[25] Tom Schaul, Tobias Glasmachers, and Jürgen Schmidhuber. High dimensions and heavy tails for natural evolution strategies. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 845–852. ACM, 2011.

[26] A. Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Computer Science Department, University of Toronto, 2009.

[27] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[28] Rupesh Kumar Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, and Jürgen Schmidhuber. Compete to compute. In *NIPS*, 2013.

[29] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996. doi: 10.1.1.134.2462.

[30] Richard S. Sutton and Andrew G. Barto. Reinforcement Learning I: Introduction, 1998.

[31] D. Wierstra, T. Schaul, J. Peters, and J. Schmidhuber. Natural evolution strategies. In *IEEE Congress on Evolutionary Computation*, pages 3381–3387. IEEE, 2008.

[32] T. Glasmachers, T. Schaul, S. Yi, D. Wierstra, and J. Schmidhuber. Exponential natural evolution strategies. In *12th annual conference on Genetic and Evolutionary Computation*, pages 393–400. ACM, 2010.

[33] I. Rechenberg. Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Dissertation, 1971. Published 1973 by Fromman-Holzboog.

[34] H. P. Schwefel. Numerische Optimierung von Computer-Modellen. Dissertation, 1974. Published 1977 by Birkhäuser, Basel.

[35] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[36] Daniel J Felleman and David C Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex*, 1(1):1–47, 1991.

[37] Rodney J Douglas, Christof Koch, Misha Mahowald, KA Martin, and Humbert H Suarez. Recurrent excitation in neocortical circuits. *Science*, 269(5226):981–985, 1995.

[38] J. Bullier. Hierarchies of cortical areas. In J.H. Kaas and C.E. Collins, editors, *The Primate Visual System*, pages 181–204. CRC Press, New York, 2004.

[39] Victor AF Lamme and Pieter R Roelfsema. The distinct modes of vision offered by feedforward and recurrent processing. *Trends in neurosciences*, 23(11):571–579, 2000.

[40] L. Itti. Visual salience. 2(9):3327, 2007.

[41] Carl M Francolini and Howard E Egeth. Perceptual selectivity is task dependent: The pop-out effect poops out. *Perception & Psychophysics*, 25(2):99–110, 1979.

[42] Rufin VanRullen. On second glance: Still no high-level pop-out effect for faces. *Vision research*, 46(18):3017–3027, 2006.

[43] Charles D Gilbert and Mariano Sigman. Brain states: top-down influences in sensory processing. *Neuron*, 54(5):677–696, 2007.

[44] JM Hupe, AC James, BR Payne, SG Lomber, P Girard, and J Bullier. Cortical feedback improves discrimination between figure and background by v1, v2 and v3 neurons. *Nature*, 394(6695):784–787, 1998.

[45] Jean Bullier, Jean-Michel Hupé, Andrew C James, and Pascal Girard. The role of feedback connections in shaping the responses of visual cortical neurons. *Progress in brain research*, 134:193–204, 2001.

[46] Victor AF Lamme. Blindsight: the role of feedforward and feedback corticocortical connections. *Acta psychologica*, 107(1):209–228, 2001.

[47] Moshe Bar, Karim S Kassam, Avniel Singh Ghuman, Jasmine Boshyan, Annette M Schmid, Anders M Dale, MS Hämäläinen, Ksenija Marinkovic, DL Schacter, BR Rosen, et al. Top-down facilitation of visual recognition. *Proceedings of the National Academy of Sciences of the United States of America*, 103(2):449–454, 2006.

[48] Dean Wyatte, Seth Herd, Brian Mingus, and Randall O'Reilly. The role of competitive inhibition and top-down feedback in binding during object recognition. *Frontiers in Psychology*, 3:182, 2012.

[49] Dean Wyatte, Tim Curran, and Randall O'Reilly. The limits of feedforward vision: Recurrent processing promotes robust object recognition when objects are degraded. *Journal of Cognitive Neuroscience*, 24(11):2248–2261, 2012.

[50] J. Schmidhuber and R. Huber. Learning to generate artificial fovea trajectories for target detection. *International Journal of Neural Systems*, 2(1 & 2):135–141, 1991.

[51] Randall C O'Reilly. Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. *Neural Computation*, 8(5):895–938, 1996.

[52] Kunihiko Fukushima. Restoring partly occluded patterns: A neural network model with backward paths. In Okyay Kaynak, Ethem Alpaydin, Erkki Oja, and Lei Xu, editors, *Artificial Neural Networks and Neural Information Processing ICANN/ICONIP 2003*, volume 2714 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2003.

[53] Hugo Larochelle and Geoffrey Hinton. Learning to combine foveal glimpses with a third-order boltzmann machine. *Image*, 1:x2, 2010.

[54] Benjamin Goodrich and Itamar Arel. Reinforcement learning based visual attention with application to face detection. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 19–24. IEEE, 2012.

[55] M. F. Stollenga, M. A. Wiering, and L. R. B. Schomaker. Using guided autoencoders on face recognition. In *Master's thesis*. University of Groningen, 2011.

[56] Randall C OReilly, Dean Wyatte, Seth Herd, Brian Mingus, and David J Jilk. Recurrent processing during object recognition. *Frontiers in Psychology*, 4:124, 2013.

[57] S.D. Whitehead. *Reinforcement Learning for the adaptive control of perception and action*. PhD thesis, University of Rochester, February 1992.

[58] Sven Behnke. Learning iterative image reconstruction in the neural abstraction pyramid. *International Journal of Computational Intelligence and Applications*, 1(04):427–438, 2001.

[59] Sven Behnke. Face localization and tracking in the neural abstraction pyramid. *Neural Computing & Applications*, 14(2):97–103, 2005.

[60] Matthew D. Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *CoRR*, abs/1301.3557, 2013.