Lateral Inhibition Nets with Nonlinear Connections

Liu Ying
School of Computer and Control
University of Chinese Academy of Sciences
Beijing, China 100000
Email: yingliu@ucas.ac.cn

Xiang Chao
School of Computer and Control
University of Chinese Academy of Sciences
Beijing, China 100000
Email: xiangchao215@mails.ucas.ac.cn

***Abstract*—Convolutional Neural Networks(CNNs) have achieved great success in many computer vision tasks, especially in image recognition. However, as neural networks grow deeper and deeper, to some extend, we've found them becoming difficult to train, and requiring samples in large scale dramatically, even with the help of Dropout and Dropconnect methods, which do improve the accuracy a bit but burdens the training process as a sacrifice. To overcome this, we proposed a novel method to generate dynamic graphs of computation for varied inputs. As our computation graphs are determined by input samples and proved to be pretty sparse, we call them Data-driven Sparse Connections(DSCs). We've applied our DSCs to a few popular image recognition tasks, and it is shown deep CNNs with DSCs win over the state-of-the-art on many tasks, such as MNIST, FICAR-10, and FICAR-100, to name a few.**

I. Introduction

Deep convolutional neural networks (CNNs) is firstly realized by Fukusima [1] with max-pooling layers [2] trained by backprop [3] on GPUs [4] have become the state-of-the-art in object recognition [5]–[8], segmentation/detection [9], [10], and scene parsing [11], [12] (for an extensive review see [13]). 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 [14]. Detectors in higher layers learn to respond to concrete visual objects or their parts, e.g., [15]. Once trained, the CNN never changes its weights or filters during evaluation.

II. Nonlinear Connections

A. Conventional Connections: Linear Computation Model

In connection models known so far, the most popular and probably the most useful one [19] is :

$$y = \sigma\left(\sum_{i=1}^{N} w_i x_i + b\right)$$

While, the activation function $\sigma(\cdot)$ can be the sigmoid function $sigmoid(x) = \frac{1}{1+e^{-x}}$, or $tanh(x)$, or ReLU function $ReLU(x) = \max(0, x)$ [20]. In addition, a different form of activation function, $maxout(x) = \max_{i=1}^{N}(w_i x_i)$, is being used in CNNs recently [7]. However, these activation functions are all neuron-based, and simply suppose that each connection acts as simple as a linear function, $o(x_i) = w_i x_i$, in which, w_i is a float number without even a bound. While in biological neural system, synapses, mimicked by artificial connections, are not really behaving linearly.

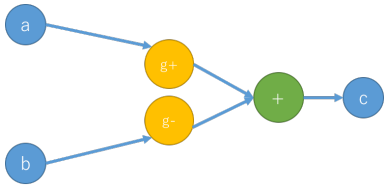

Fig. 1. A Net with Nonlinear Connections

B. Nonlinear Computation Model for Connections

Inspired by biological neuron synapses [16]–[18], we propose a new computation model for connections between neurons. In synapses, signals are transformed from electrical form to chemical one, and transferred by some proteins to a connected neuron, then again, transformed back to electrical form to inhibit or activate the connected neuron [21], [22]. In this process, signals can be amplified or reduced, but not in a linear way. Futhermore, signal strengths have lower and upper bounds which differs from the denotation of connections in CNNs.

Suppose, there're 3 neurons, $\mathcal{N}_a$,$\mathcal{N}_b$, $\mathcal{N}_c$, and 2 connections, $\mathcal{C}_{a\to c}$, $\mathcal{C}_{b\to c}$, the graph is shown in figure (1).

As synapses have two kinds: excitatory and inhibitory ones [23], [24], we propose two categories of connections the same way. For excitatory connections, as $\mathcal{C}_{a\to c}$ shown in figure (1), their signal transferring functions are described as equation (1), in which, $\alpha \in (0, 1)$ is guaranteed. For excitatory connections,

$$g^+(x) = \min\left(\max\left(0, \frac{x-\alpha}{1-\alpha}\right), 1\right) \tag{1}$$

For inhibitory connections,

$$g^-(x) = -g^+(x) \tag{2}$$

Activation function can be any other type that constrains the output between 0 and 1, although we set activation function as the same in equation (1) as in our model.

$$\sigma(x) = g^+(x) \tag{3}$$

The α is the only parameter, and it could be varied for different neurons, with the same bounds of that in signal

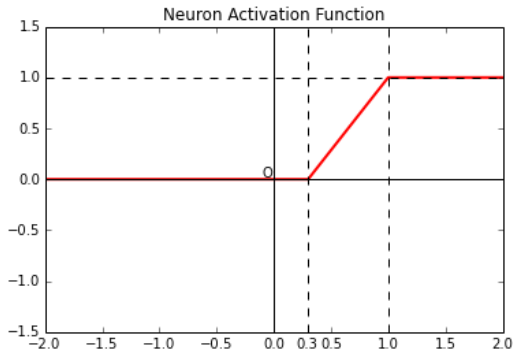

Fig. 2. Activation Function of Neuron with $\alpha = 0.3$

transferring equations. The activation function of a neuron may look like figure (2).

For a M-layer neural network, The forward-computaton equation for the k^{th} neuron on $(p+1)^{\text{th}}(p = 1, 2, \cdots, M)$ layer, $\mathcal{N}_{p_k}$, with the input layer denoted as the first layer, is

$$o_{p+1_k} = \sigma(\sum_{i=1}^{N_p} g_{i\to k}(o_{p_i})) \tag{4}$$

In which, $g_{i\to k}$ is the signal transferring function from i^{th} neuron in p^{th} layer to k^{th} neuron in $(p+1)^{\text{th}}$ layer, and o_{p_i} is the activation value of the i^{th} neuron on p^{th} layer.

For network in figure (1), we let o_a, o_b and o_c denote the activation strengths of neuron $\mathcal{N}_a$, $\mathcal{N}_b$ and $\mathcal{N}_c$ respectively, and $\alpha_{a\to c}$, $\alpha_{b\to c}$ as the parameters of connection $\mathcal{C}_{a\to c}$, $\mathcal{C}_{b\to c}$ similarly. Then we will have:

$$o_c = \sigma(g^+(o_a)|_{\alpha=\alpha_{a\to c}} + g^-(o_b)|_{\alpha=\alpha_{b\to c}}) \tag{5}$$

C. Error Backpropagation For Nonlinear Connections

While, since the equations for connections and activation function of neurons in our model are not really differentiable, the famous backprop algorithm cannot be applied in the training section. Thus, we propose a training algorithm that deals with non-differentiable situation like this.

Firstly, let's introduce two model parameters: ξ and η, they together determine the learning rate of our model. In addition, $\xi \in (0, 1)$ and $\eta \in (0, 1)$. Let $x = \sum_{i=1}^{N_p} g_{i\to k}(o_{p_i})$, $y = \sigma(x)$, and the given feedback error for current neuron $\mathcal{N}_{p_k}$ is Δy. Similarly, the feedback error for x is Δx. In case parameter α goes to 1 and fails our equations, we propose a upper-bound $\alpha_{\max}$, which is very close to 1 but less than 1. Thus, the parameter updatng rules are,

$$\Delta x = \eta\left((1-\alpha)(y+\Delta y) + \alpha - x\right) \tag{6}$$

$$\hat{\alpha} = \frac{x - y - \Delta y}{1 - y - \Delta y} \tag{7}$$

$$\Delta\alpha = \xi \cdot \begin{cases} \left(\max\left(0, \hat{\alpha}\right) - \alpha\right), & \Delta y > 0 \\ \left(\min\left(\alpha_{\max}, \hat{\alpha}\right) - \alpha\right), & \Delta y < 0 \end{cases} \tag{8}$$

Let $\Delta x_{p+1_{i\to k}}$ be the correction for connection from neuron $\mathcal{N}_{p_i}$ to $\mathcal{N}_{p+1_k}$, according to the definition of Δx in equation (6),

$$\Delta x_{p+1_k} = \eta((1-\alpha_{p+1_k})(o_{p+1_k} + \Delta o_{p+1_k}) + \alpha_{p+1_k} - x_{p+1_k}) \tag{9}$$

and,

$$\Delta x_{p+1_k} = \sum_{i=1}^{N_p} \Delta x_{p+1_{i\to k}} \tag{10}$$

We assume that the correction strength for each connection is only related to the activation value of the neuron which emits the connection. Thus we will get,

$$\Delta x_{p+1_{i\to k}} = \frac{o_{p_i}}{\sum_{j=1}^{N_p} o_{p_j}} \cdot \Delta x_{p+1_k} \tag{11}$$

Let $\alpha_{p_{i\to k}}$ be the parameter of the connection from the i^{th} neuron in p^{th} layer to the k^{th} neuron in $(p+1)^{\text{th}}$ layer. Using equation (8), we can update the parameters of connections. For excitatory connections,

$$\hat{\alpha}_{p_{i\to k}} = \frac{o_{p_i} - g^+_{p_{i\to k}}(o_{p_i}) - \Delta x_{p+1_{i\to k}}}{1 - g^+_{p_{i\to k}}(o_{p_i}) - \Delta x_{p+1_{i\to k}}} \tag{12}$$

$$\Delta\alpha_{p_{i\to k}} = \xi \cdot \begin{cases} \max\left(0, \hat{\alpha}_{p_{i\to k}}\right) - \alpha_{p_{i\to k}}, \\ \quad \Delta x_{p+1_{i\to k}} > 0 \\ \min\left(\alpha_{\max}, \hat{\alpha}_{p_{i\to k}}\right) - \alpha_{p_{i\to k}}, \\ \quad \Delta x_{p+1_{i\to k}} < 0 \end{cases} \tag{13}$$

For inhibitory connections, we only need to replace $\Delta x_{p+1_{i\to k}}$ with $-\Delta x_{p+1_{i\to k}}$, and replace $g^+_{p_{i\to k}}(o_{p_i})$ with $-g^-_{p_{i\to k}}(o_{p_i})$ in equation (12) and (13),

$$\hat{\alpha}_{p_{i\to k}} = \frac{o_{p_i} + g^-_{p_{i\to k}}(o_{p_i}) + \Delta x_{p+1_{i\to k}}}{1 + g^-_{p_{i\to k}}(o_{p_i}) + \Delta x_{p+1_{i\to k}}} \tag{14}$$

$$\Delta\alpha_{p_{i\to k}} = \xi \cdot \begin{cases} \min\left(\alpha_{\max}, \hat{\alpha}_{p_{i\to k}}\right) - \alpha_{p_{i\to k}}, \\ \quad \Delta x_{p+1_{i\to k}} > 0 \\ \max\left(0, \hat{\alpha}_{p_{i\to k}}\right) - \alpha_{p_{i\to k}}, \\ \quad \Delta x_{p+1_{i\to k}} < 0 \end{cases} \tag{15}$$

Biologically, A synapse can switch from excitatory to inhibitory [25]. In our model, we propose a training strategy which allows connections switch between excitatory and inhibitory during training process.

We propose a valve $\tau \in (0, \alpha_{\max})$, which is very close to 0 but greater than 0, if $g_{p_{i\to k}} = g^+$, $\Delta x_{p+1_{i\to k}} < 0$ and $\alpha_{p_{i\to k}} + \Delta\alpha_{p_{i\to k}} - \alpha_{\max} < \tau$, then for connection $\mathcal{C}_{p_{i\to k}}$, let $g_{p_{i\to k}} = g^-$, this update will switch connections from excitatory to inhibitory. Similarly, if $g_{p_{i\to k}} = g^-$, $\Delta x_{p+1_{i\to k}} > 0$ and $\alpha_{p_{i\to k}} + \Delta\alpha_{p_{i\to k}} - \alpha_{\max} < \tau$, then for connection $\mathcal{C}_{p_{i\to k}}$, let $g_{p_{i\to k}} = g^+$, this update will switch connections from inhibitory to excitatory.

So far we've derivated the equations for updating parameters of neurons in $(p+1)^{\text{th}}$ layer and the parameters of connections from p^{th} layer to $(p+1)^{\text{th}}$ layer. With equation (9),(11) and (6), we calculate the error of the output in the p^{th} layer. Let $\Delta o_{p_{i\to k}}$ be the correction for $\mathcal{N}_{p_i}$ contributed by connection $\mathcal{C}_{p_{i\to k}}$, for excitatory connections,

$$\Delta o^{+}_{p_{i\to k}} = \eta((1-\alpha_{p_{i\to k}})(g^{+}_{p_{i\to k}}(o_{p_i}) + \Delta x_{p+1_{i\to k}}) + \alpha_{p_{i\to k}} - o_{p_{i\to k}}) \quad (16)$$

For inhibitory connections,

$$\Delta o^{-}_{p_{i\to k}} = -\Delta o^{+}_{p_{i\to k}} \quad (17)$$

Then the totol correction for the output of neuron $\mathcal{N}_{p_i}$ is,

$$\Delta o_{p_i} = \sum_{k=1}^{N_{p+1}} \Delta o_{p_{i\to k}} \quad (18)$$

III. Lateral Inhibition Nets

A. The Theory of Lateral Inhibition

Biologically, the excitation process of neurons are inhibited by nearby excited neurons, so called *Lateral Inhibition* [26]–[28]. Researches proved that the lateral inhibition play a big role in forming patterns in neural dynamics [26]. We propose an similar mechanism in artificial neural networks, and expect to achieve better presentation of feature map.

Let $\dot{\Upsilon}$ be the set of all nearby neurons, i.e., the neighbors of a neuron $\mathcal{N}_0$, and $\Upsilon = \dot{\Upsilon} \cup \{\mathcal{N}_0\}$. For general argument, if the neuron $\mathcal{N}_0$ is at position (x, y) in the i^{th} feature map of p^{th} layer in a convolutional neural network. Let $r \in \mathbf{N}^+$ be the radius of neighborhood, the center point is (x_0, y_0), thus region defined by $(x_0, y_0) \odot r$ is the neighborhood, and every neuron that is away from $\mathcal{N}_0$ by an Manhattan distance within r is counted in Υ. that is, $\Upsilon = \{\mathcal{N}_{(x,y)} \mid |x - x_0| + |y - y_0| \le r\}$, then the lateral inhibition phase is described with equation (19).

$$o'_{\mathcal{N}_0} = \frac{o_{\mathcal{N}_0}}{\max(1, \sum_{\mathcal{N}_i \in \Upsilon} o_{\mathcal{N}_i})} \quad (19)$$

In equation (19), $o_{\mathcal{N}_0}$ is the activation value of neuron $\mathcal{N}_0$, and $o'_{\mathcal{N}_0}$ is the real activation value after the lateral inhibition phase. As the forward computing flow is changed with this additional phase, the error backprop equations need to be updated, as in equation (20).

$$\Delta o_{\mathcal{N}_0} = (o'_{\mathcal{N}_0} + \Delta o'_{\mathcal{N}_0}) \cdot \max(1, \sum_{\mathcal{N}_i \in \Upsilon} o_{\mathcal{N}_i} + \Delta o'_{\mathcal{N}_0}) - o'_{\mathcal{N}_0} \cdot \max(1, \sum_{\mathcal{N}_i \in \Upsilon} o_{\mathcal{N}_i}) \quad (20)$$

B. A Visualized Instance of Lateral Inhibition

To visualize how lateral inhibition can improve the presentation of feature maps from original images, we propose a simple filter which only extracts perceived luminance distribution of an image, denoted as $\mathcal{F}(\cdot) = (0.299, 0.587, 0.114) \cdot (r, g, b)$. Such a filter has a size of receptive field as 1×1. We conduct this experiment under *Jupyter Notebook* (thanks to all supporters of this amazing staff), and the image to test (see Fig-3a) is randomly chosen from Web (no offence to the president of the U.S.).

Using the Luminance Filter $\mathcal{F}$, we get Fig-3b, which is generally considered as a grayscale image from the original. In conventional neural nets, such a feature map generated by this filter is connected directly to a sub-sampling or fully-connected layer, the connection volume and computation cost thereby are considerable, and often lead to overfitting [7]. To overcome this, we append lateral inhibition phase immediately to each of convolution layer before the computation process enters into next layer, expecting to make a relatively more sparse net. The result for current case is shown as Fig-3c. The darker part of image is what is expected to extract. As we see, the lateral inhibition phase performs similarly to a differential calculus of images, but with more global features (such as luminance) remained. Let's see how this works, suppose a part of single channel image is denoted as a float matrix $\mathbf{M}$,

$$\mathbf{M} = \begin{Bmatrix} 0.1 & 0.0 & 0.1 & 0.9 & 0.9 & 0.9 \\ 0.0 & \mathbf{0.4} & 0.0 & 0.9 & \mathbf{0.9} & 0.9 \\ 0.1 & 0.0 & 0.1 & 0.9 & 0.9 & 0.9 \end{Bmatrix}$$

Notice that only the emphasized part of matrix $\mathbf{M}$ is under observation. Let the lateral inhibition phase be $\Omega(r)$ in which r is the only parameter for the definition of neighborhood Υ, according to equation (19),

$$\mathbf{M}' = \mathbf{M} \otimes \Omega(r=1) = \begin{Bmatrix} 0.1 & 0.0 & 0.1 & 0.32 & 0.25 & 0.33 \\ 0.0 & \mathbf{0.4} & 0.0 & 0.25 & \mathbf{0.2} & 0.33 \\ 0.1 & 0.0 & 0.1 & 0.32 & 0.25 & 0.33 \end{Bmatrix}$$

Before lateral inhibition phase, the left half of $\mathbf{M}$ is a 'darker' place (the smaller, the darker), compared to its right one. While, after $\Omega(r=1)$ is applied, the left half remained unchanged, and the right half of $\mathbf{M}'$ is obviously suppressed, and the right observed identity becomes surprisingly far smaller than the left. While, we can see that the left half of the matrix still remained 'darker' than the right half, which means lateral inhibition can keep the global features from images, and highlights local features at the same time.

IV. Binarization of Feature Map

A. Feature Map Visualization

Although the feature map drawn by the luminance filter is clearly visualized and understood in last section, filters can be arbitrary and ambiguous in neural networks like CNNs, and it

(a) The Original (b) Luminance Filter (c) Lateral Inhibition (d) Binarization

Fig. 3. Lateral Inhibition Visualization

thereby becomes difficult or even impossible to understand what kind of features those filters really extract from the images. However, if the lateral inhibition phase is employed, we may obtain a better understanding and visualization of the feature maps, since we do not need to care what the feature map really means, but just to know what differences are presented among local regions of feature maps, as those differences are what truly determine which class the input image is related to.

On the other hand, conventional visualization of feature maps simply colorize those feature activation matrix into pactchs of image, and an average man may not get any useful information to understand how convolutional networks work, or if they are working in a way as expected. Recently a category of deconvolutional network is proposed to visualize the feature maps, and it seems working well [15]. However, the method is literally too complicated and requires much more computation, which we may not expect.

Therefore, we propose *feature map binarization* to help visualize and diagnose each feature layer of CNNs at literally no cost (the visualization is straightly one snapshot of the layer without any more procedure as you will see).

For any type of layer at depth p, consisting of N_p filter(s) (actually, we may view a fully connected layer as a convolutional layer with only single filter of which the receptive field is maximumized as the same size of last layer), we get N_p feature map(s). For each feature map, $\Phi_{p_i}, i = 1, 2, 3, \cdots, N_p$, let $o'_{p_{i_j}}$ be the j^{th} element in map Φ'_{p_i}, which is a state of Φ_{p_i} following lateral inhibition phase, and $\dot{\Upsilon}_{p_{i_j}}$ be its neighborhood (the definition of neighborhood preserves the same as that in lateral inhibition phase, while the parameter r may be set alone), we apply such procedure to binarize feature map $\Phi'_{p_{i_j}}$,

$$\Xi_{p_{i_j}} = \{o'_{p_{i_k}} \mid o'_{p_{i_j}} - o'_{p_{i_k}} > \beta, o'_{p_{i_k}} \in \dot{\Upsilon}_{p_{i_j}}\} \tag{21}$$

$$o''_{p_{i_j}} = \begin{cases} 1.0, & \dfrac{\text{count}(\Xi_{p_{i_j}})}{\text{count}(\dot{\Upsilon}_{p_{i_j}})} \geq \theta \\ 0.0, & \text{otherwise} \end{cases} \tag{22}$$

In equation (21)(22), $\beta \in [0, +\infty]$ is the minimum gap of activation proposed between any element in neighborhood $\dot{\Upsilon}_{p_{i_j}}$ and the interested one $o_{p_{i_j}}$, and $\theta \in [0, 1]$ is the lowerbound of the ratio of satisfied neighbors by the filter of minimum gap of activation. Such that the binarization of feature maps is determined and only, on parameters $\langle r, \beta, \theta \rangle$.

According to equation (21) and (22), we append our binarization phase into the whole computing flow following the lateral inhibition phase. Thus we get Fig-3d as the binarization result of the feature map, for which we take $\langle r, \beta, \theta \rangle = \langle 5, 0.0, 0.85 \rangle$. From Fig-3d, we can clearly see what the luminance filter really extracts from the original image, and basically how well it behaves.

B. Input Determined Sparsity

In fact, we may present the computing flow for each filter as the transformations of its feature map,

$$\mathbf{X} \longmapsto \cdots \longmapsto \Phi_{p_i} \overset{\Omega(r)}{\longmapsto} \Phi'_{p_i} \overset{\Psi(r,\beta,\theta)}{\longmapsto} \Phi''_{p_i}$$

, with Ψ defined as the binarization phase. We take Φ'_{p_i} as the final state of feature map generated by filter $\mathcal{F}_{p_i}$, and Φ''_{p_i} is the mask of its sparsity. Obviously, Φ''_{p_i} is determined by its input $\mathbf{X}$, which may be an original image. For each different input of a net, the sparsity mask may vary, and this is different from the Dropout [32] and Dropconnect [33] regularization, which randomly reduce nodes or connections to train submodels of a net.

V. Filter Polarization

In conventional CNNs, due to huge volume of connections and sometimes insufficient training data (big data though, not big enough yet), the redundancy of filters leads to model overfitting and difficulty in training [29], [30]. Thus we propose a method called filter polarization, it generates and pretrains those filters layer by layer, dynamically determines the number of filters for each layer, and produces more polarized filters, aiming to reduce the filter redundancy of nets and speed up global training.

Filter polarization algorithm contains three phase, they are executed in distinct order, and the whole flow of three phases will iterate itself till some condition meets.

A. Filter Generation

B. Filter Pretraining

C. Filter Redundancy Check

Let $\mathcal{F}_{p_k}$ denote the k^{th} filter in layer of depth p, and o_{p_k} be its activation value. Firstly, we initiliazed the p^{th} layer with N_p filters, for which we set the connection parameters $\left\langle \alpha_{p_{k_1}}, \alpha_{p_{k_2}}, \cdots, \alpha_{p_{k_{N_p}}} \right\rangle$ randomized as strictly different values.

VI. Imperical Study

We propose a net model to achieve automatic image recognition, and it is tested on a set of popular benchmarks.

On MNIST datasets, with no convolutional layers, we've achieved accuracy and minimum training cost.

————- CHART HERE ————-

On CIFAR-10 benchmark, we use convolutional layers to reduce parameter volume and enhence model robustness. Our model performs well too.

———— CHART HERE ————

For small sample volume datasets like CIFAR-100 benmark, our model also outperforms most state-of-the-art ones.

———— CHART HERE ————

VII. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

References

[1] K. Fukushima, "Neural network model for a mechanism of pattern recognition unaffected by shift in position- neocognitron," *ELECTRON. & COMMUN. JAPAN*, vol. 62, no. 10, pp. 11–18, 1979.

[2] J. Weng, N. Ahuja, and T. S. Huang, "Cresceptron: a self-organizing neural network which grows adaptively," in *Neural Networks, 1992. IJCNN., International Joint Conference on*, vol. 1. IEEE, 1992, pp. 576–581.

[3] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[4] D. C. Ciresan, U. Meier, J. Masci, L. Maria Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, no. 1. Barcelona, Spain, 2011, p. 1237.

[5] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3642–3649.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[7] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio, "Maxout networks." *ICML (3)*, vol. 28, pp. 1319–1327, 2013.

[8] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[9] D. C. Cireşan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Mitosis detection in breast cancer histology images with deep neural networks," in *International Conference on Medical Image Computing and Computer-assisted Intervention*. Springer, 2013, pp. 411–418.

[10] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Advances in neural information processing systems*, 2012, pp. 2843–2851.

[11] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1915–1929, 2013.

[12] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3626–3633.

[13] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[14] D. Gabor, "Theory of communication. part 1: The analysis of information," *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering*, vol. 93, no. 26, pp. 429–441, 1946.

[15] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[16] W. S. Hall, *A Textbook of physiology*. Lea, 1905.

[17] W. Bayliss, "On reciprocal innervation in vaso-motor reflexes and the action of strychnine and of chloroform thereon," *Proceedings of the Royal Society of London. Series B, Containing Papers of a Biological Character*, vol. 80, no. 541, pp. 339–375, 1908.

[18] R. Gerard, "The interaction of neurones," *Ohio J. Sci*, vol. 41, pp. 160–172, 1941.

[19] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[20] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[21] E. G. Gray, "Axo-somatic and axo-dendritic synapses of the cerebral cortex: an electron microscope study," *Journal of anatomy*, vol. 93, no. Pt 4, p. 420, 1959.

[22] C. D. Harvey and K. Svoboda, "Locally dynamic synaptic learning rules in pyramidal neuron dendrites," *Nature*, vol. 450, no. 7173, pp. 1195–1200, 2007.

[23] K. Uchizono, "Characteristics of excitatory and inhibitory synapses in the central nervous system of the cat," *Nature*, vol. 207, no. 4997, pp. 642–643, 1965.

[24] H. R. Wilson and J. D. Cowan, "Excitatory and inhibitory interactions in localized populations of model neurons," *Biophysical journal*, vol. 12, no. 1, pp. 1–24, 1972.

[25] K. Ganguly, A. F. Schinder, S. T. Wong, and M.-m. Poo, "Gaba itself promotes the developmental switch of neuronal gabaergic responses from excitation to inhibition," *Cell*, vol. 105, no. 4, pp. 521–532, 2001.

[26] S.-i. Amari, "Dynamics of pattern formation in lateral-inhibition type neural fields," *Biological cybernetics*, vol. 27, no. 2, pp. 77–87, 1977.

[27] C. Blakemore and E. A. Tobin, "Lateral inhibition between orientation detectors in the cat's visual cortex," *Experimental Brain Research*, vol. 15, no. 4, pp. 439–440, 1972.

[28] C. Blakemore, R. H. Carpenter, and M. A. Georgeson, "Lateral inhibition between orientation detectors in the human visual system." *Nature*, 1970.

[29] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *arXiv preprint arXiv:1405.3866*, 2014.

[30] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing convolutional neural networks," *arXiv preprint arXiv:1506.04449*, 2015.

[31] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," *Artificial Neural Networks and Machine Learning–ICANN 2011*, pp. 52–59, 2011.

[32] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[33] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013, pp. 1058–1066.