Biologically Extending the Gen 2 ANN Model

Leslie Knope
Pawnee Parks and Recreation Department
100 State Street
Pawnee, Indiana 47998
Leslie.Knope@gmail.com

Abstract

In this paper the generations of artificial neural networks (ANN) are surveyed. The assumptions present in Gen 1 and 2 ANNs are enumerated. In the process of reformulating the Gen 2 ANN an extension was observed that could increase the biological plausibility of the model. This new model makes use of the neurological interneuron structures that provide inhibition and input gain control in the cortical regions of the brain. The resultant interneuron neural network (INN) is applied to the MNIST data set. The first attempt at applying the INN achieves a higher accuracy than an equivalent ANN. The application of the model serves as the initial validation for the derivation of the model and associated backpropagation.

Background

Increased understanding of the neuron has lead to many advances over the past century. Medically speaking myriad diseases, toxins, and afflictions have been addressed in ways inconceivable without a detailed view of the mechanisms within the architecture of the nervous system. Computational models such as the perceptron in (Rosenblatt, 1958), the leaky integrate and fire (LIF) neuron summarized in (Abbott, 1999), and the Nobel winning Hodgkin-Huxley neural model in (Hodgkin and Huxley, 1952). have allowed for analysis and theorization within the realm of neuroscience, while in computer science they have provided unforeseen mechanistic processes that allow for neuromorphic computation.

Because our understanding of the topic is continually increasing, every aspect of the current neural models is under scrutiny and dispute. With this in mind we first attempt to describe briefly the agreed upon functions and the models that attempt to capture them. We then offer analyses which form the foundation for the proposed extension. The majority of biological understanding here is based on (Purves *et al.*, 2004).

Biological Neural Networks

In Figure1 a simple neuron is shown. The major features which will be referred to are the dendrites, cell body, and synaptic terminals.

The dendrites provide the input field for the neuron. Typically, this is the output of other neurons. Synaptic terminals projecting from other neurons connect to the dendrites of the target neuron. Just as the synaptic terminals of the target neuron terminate on other neurons. These terminals release neurotransmitters at the dendrite and provide excitation to the neuron.

With the arrival of each action potential at a synaptic terminal, neurotransmitters are released and carried into the neuron by way of ion channels and pumps. The ions change the voltage present across the membrane of the cell body. If the membrane voltage increases past the action potential threshold, an action potential will occur and propagate along the axon to the synaptic terminals. A membrane voltage that has been elevated is referred to as depolarized. In homeostasis a neuron attempts to enforce polarization, remaining ready to receive input leading to an action potential. To transform these terms into the common computer science vernacular, perform the following substitutions; dendrites are the input layer, action potentials are neuron activations, and synapse terminals are outgoing edge connections to other neurons.

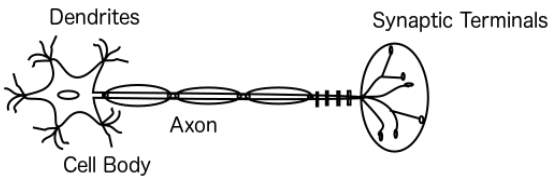

Figure 1: Neuron Diagram

From this simple functional description it should be apparent that the firing rate of the target neuron is dependent upon the rate of depolarization present at the associated dendrites. This concept is solidified in Eq 1. Where, r_{pol} is the internal rate of polarization for the target neuron. Then r_{depol} is the rate at which a specific synaptic connection is firing. The parameter α is the amount of neurotransmitter released with each pulse at the synaptic terminal. The result r_{action} is the rate of action potential propagated along the axon. Summation is required to account for connection of

more than one synapse to the target neuron. Eq 1 does not represent a full characterization of a neuron but is intended to provide intuitive understanding of biological neural activity.

$$r_{action} = -r_{pol} + \sum \alpha * r_{depol}, \quad \alpha \geq 0 \tag{1}$$

An important structure found in the biological neural network is the inhibitory interneuron network. It uses gamma aminobutyric acid or GABA as a neurotransmitter, resulting in a polarizing rather than depolarizing effect on the dendrite. These interneurons are in contrast to traditional neurons referred to as principal cells. Interneurons are seen to perform many important tasks but one is of particular importance to this study. According to (Freund and Kali, 2008) interneurons allow for selective gating of inputs to different regions of the dendrite. This means that interneurons are believed to facilitate dynamic changes in the relative contribution of inputs. This is stated in Eq 2. This equation does not represent a full picture of interneuron behaviour but rather provides the relevant understanding.

It should be stated also that there are multiple ways of dissecting the space of interneuron biologies. One useful way is to look at whether they provide feedforward or feedback modification. This paper exclusively uses feedforward interneurons. This is done for simplicity and isolation of variables.

$$r_{action} = -r_{pol} + \sum \alpha * \beta_{Inter} * r_{depol}, \quad 0 \leq \beta \leq 1 \tag{2}$$

Gen 1, 2, and 3 ANNs

Many models have been invented throughout the study of neuroscience. These range from the simplistic perceptron, containing only a step activation function, to the Hodgkin-Huxley and LIF models that are commonly used in spiking network simulations. These network models have come to be classified into three groups, generation 1, 2, and 3. The descriptions provided are for the elementary units of each ANN Generation. A network from any of the following generations can be constructed by connecting the output of one set or layer of artificial neurons to the input of another set.

Assumptions associated with the specific training algorithms for each generation of neural model are not included in the detail. Also, there are of course more complete models that implement more of the surrounding biology, like that presented in (Postnov *et al.*, 2007). However, these have not yet gained much traction and are excluded from discussion.

Gen 1 ANNs Gen 1 ANNs are considered those limited to classification. They are incapable of performing regression because the output assumes only the values one or zero. The perceptron is such a model. This model characterizes neurons as switches as shown in Eq 3. The weight w_j represents the relative amount of neurotransmitters released or, α in Eq 1, for the j^{th} synapse connection on the neuron. While, I_j is the firing rate present at the j^{th} synapse. b is the bias associated with this neuron. The bias can be understood to be synonymous with the rate of polarization, r_{pol}.

$$net = \begin{cases} 0 & b + \sum w_j * I_j \geq 0 \\ 1 & b + \sum w_j * I_j < 0 \end{cases} \tag{3}$$

As with any model, there exists a set of assumptions made that justify validity in a certain context for each generation of ANN. The assumptions for the common Gen 1 model are shown in Table 1.

Table 1: Gen 1 ANN Model Assumptions

Number	Assumption
1	Firing Frequency Encoding
2	Steady State
3	Unity Static Firing Rate
4	Learned Inhibition
5	Unconditional Weighting

The first assumption is based on Eq 1. If it is true that neurons fire with a rate defined by the rate of input activations and the associated weights of the activations, then a transformation is possible such that simulation occurs completely in the frequency domain. This transformation occurs without consideration of transition from one firing rate to another giving rise to assumption 2. The network is only evaluated once the underlying neurons have reached steady state.

Assumption 3 deals with the way Gen 1 models implement activation. A step function is commonly used to show that any value greater than 0 input to the neuron causes activation. The model was intended to capture the all or nothing activation and activation threshold present in biological neurons. However, this would seem incompatible with the frequency domain transformation. The implication for the model can be seen in Eq 4. The firing rate is now restricted to a logical 1 or 0 regardless of the magnitude of input. Thus, the model assumes that neurons only fire with a unity static firing rate.

$$r_{action} = \begin{cases} 0 & -r_{pol} + \sum \alpha * r_{depol} \geq 0 \\ 1 & -r_{pol} + \sum \alpha * r_{depol} < 0 \end{cases} \tag{4}$$

Assumption 4 is the result of the training operation. During training w_j is updated to become consistent with the desired linear separation. This is true to the point that w_j may become negative. This is not seen in the biological neurons as in general they are only capable of excitation. Inhibition is performed predominately by interneurons.

The perceptron does not implement the relative gating seen in the behavior of the interneuron network. Therefore, assumption 5 recognizes that the weight w_j remains static once the desired value is found and no other method is present to allow for dynamic changes.

Gen 2 ANNs Gen 2 ANNs are where the bulk of research in computer science has been focused. The ability to output a continuous value from zero to one provided the ability to learn non-linear regressive approximations thanks to continuously differentiable activation functions. Most neural models employed are Gen 2 models. The typical equation for

a Gen 2 ANN is presented in Eq 5. Much is the same as the perceptron. The main difference is the step function has been replaced by the sigmoid function represented by σ.

$$net = \sigma\left(b + \sum w_j * I_j\right) \tag{5}$$

The effect of the change from step activation to sigmoidal is made obvious by Table 2. The unity static firing rate assumption is no longer present. So, the model is now functionally equivalent to Eq 1 with the other assumptions detailed previously still in effect.

Table 2: Gen 2 ANN Model Assumptions

Number	Assumption
1	Firing Frequency Encoding
2	Steady State
3	Learned Inhibition
4	Unconditional Weighting

Gen 3 ANNs Gen 3 ANNs are those that attempt to relax all of the considered assumptions. They are most commonly referred to as spiking neural networks (SNN). The most common neural model of this sort is the leaky integrate and fire neuron. In this model, the neuron is considered to be an electrical circuit possessing a time constant, τ, such that the depolarization rate is some natural exponential $e^{f(\tau)}$.

Spiking neural networks are gaining traction and in (Arthur and Boahen, 2007) were implemented on chip with interneurons. Though spiking ANNs are an interesting topic, little attention is given them in this paper. They are included as part of a general explanation of the state of affairs. However, it is important to note that the main assumption that seems to remain beyond a particular mathematical model for spiking neurons is that of how the neurons learn and how they encode information.

Gen 2 Required Assumptions A good question to ask at this point is what assumptions are intrinsic to Gen 2 ANNs. Or to put it another way, where is the line between Gen 2 and Gen 3. This paper takes the opinion that the boundary lies in assumptions 1 and 2 from Table 2. The boundary between Gen 1 and 2 seems to lie along the non-linear regressive ability of Gen 2. Likewise, the boundary between Gen 2 and 3 lies in the time domain implementation of SNNs.

About the Research

In this section we establish the contribution, the motivation, and the related work relevant to this paper.

Motivation

In the process of reviewing and reformulating the Gen 2 ANN we find that there are interesting divergences. These divergences we have presented as the assumptions in Table 2. Some of these assumptions can be seen as intrinsic to the Gen 2 ANN model and some not. Specifically, assumption 4 is of interest. A common belief in neuroscience holds that a separate structure to the simple neural network, namely the interneuron, is specifically capable of allowing dynamic relative input weight adjustment. This functionality has not been captured in a Gen 2 model. We became motivated to address this for two reasons: biological plausibility and improved predictive modeling.

Improved Biological Plausibility Biologically inspired, heuristic algorithms have been very successful. Particle swarm optimization, genetic algorithms, and ANNs are all forms of biological algorithms or ecorithms to use the term coined in (Valiant, 2013). It is believed that these ecorithms exist because they have been endowed with a superior combination of motility and utility. So, by building algorithms that more closely move with Einstein's music of the spheres it is believed that their performance will benefit. Furthermore, by building better approximations of neurological processes we hope to provide qualified answers to neurological questions.

Better Predictive Modeling As stated it is believed and hoped that by more closely modeling a natural process we will gain some benefit in the performance of the algorithm. Specifically, since neuroscience has postulated on the gating ability of inputs for neural networks by way of interneurons it is hoped that they will provide better input feature selection. There is reason to believe that if this is true it should offer benefits in network generalization and multi-task learning. The justification for this hope is grounded in the related works.

Contribution

This paper contributes a Gen 2 ANN model which is extended to include interneurons. The combined model is referred to as an INN. The interneurons are capable of selectively gating the input along any edge to edge connection between neurons to allow for input gain control with a unity max gain. This paper also presents the backpropagation algorithm for the INN. Further the network is applied to the MNIST dataset and is shown to outperform the same base ANN possessing no interneuron gating. While this is true, it should be understood that the point of this paper is not to build a better performing ANN. It is the beginning of investigation towards a new more biologically plausible model.

Related Work

Most related work to this is only related in the sense that it mathematically implements something notably similar. There has been no work that was found to implement interneuron structures in a Gen 2 framework.

In (Schuster and Berrar, 2014), the authors recognize the same deficiency with static weights asserted in assumption 4 of the gen 2 model. Their aim was to improve multi-task learning in ANNs by adding a dynamic weight to the ANN. The resulting neural equation is presented in Eq 6. The key difference is that there is no selective gating and the dynamic weight is only dynamic in the sense that when a new task is to be learned, only the dynamic weight is allowed to be trained.

$$net = \sigma\left(b + \sum(w_j^{static} + w_j^{dynamic}) * I_j\right) \quad (6)$$

In (Kirkpatrick *et al.*, 2017) a method is presented that allows for a stochastic analysis to decide what the reduction in plasticity for a given neuron should be based on relative importance to the network output. This work is based on the synaptic plasticity of neurons to learn to fire more easily with the neurons that are relevant in their input field and to ignore those that are not. This provides a mechanism for learning a task B when a task A has already been learned without forgetting task A.

The last paper to be mentioned is (Wan *et al.*, 2013). Here the authors extend the concept of dropout to the dropconnect algorithm. They also derive a bound for the generalization benefit of each. This work is included as dropconnect is a stochastic process that dynamically modifies the weights of the network. However, it does so in a random fashion and in no way contributes to input feature selection or implementation of interneurons.

All discovered related work lies within the fields of network sparsification and multi-task learning. For this reason we expect to find some benefit in these domains.

The INN Model

In this section the Interneuron ANN is presented. The architecture is an extension of the existing Gen 2 ANN model.

Derivation

Before presenting the INN model it is important to clearly state the desired functionality and requirements. The biological reasoning has already been presented in the background information. Based on the understanding of information presented there, this algorithmic description is presented.

The new model should possess the power to perform relative input gating. That is to say, given a set, I, of inputs to neuron n it is required that any $I' \in I$ be modifiable such that the set $I'\Delta I$ is left unchanged. This lends itself to the addition of some weight β to the ANN equation. This can be seen in Eq 7.

A further requirement is that β must be variable depending on the inputs presented to the ANN. Any number of methods could be conceived of to accomplish this but the most biologically supported is to allow a separate ANN to provide β as an output. Then there is an ANN of principal cell neurons and an ANN of interneurons. For distinction, any variables pertaining to the principal cell ANN will remain as shown in previous equations while the interneuron network variables will be accompanied by a prime. This is shown in the Eq 8. The principal cell and interneuron ANNs together form the INN.

$$net = \sigma\left(b + \sum w_j * \beta_j * I_j\right) \quad (7)$$

$$\beta_j = \sigma\left(b' + \sum w'_j * I'_j\right) \quad (8)$$

Extending this view to a network level, Figure2 shows a view of an INN with the principal cell ANN horizontal and the Interneuron ANN vertical. Notice that only the input layer has the interneuron weights. This is not the general case but is drawn this way because it more closely matches the experimental setup. The general case allows for interneurons to be connected as desired.

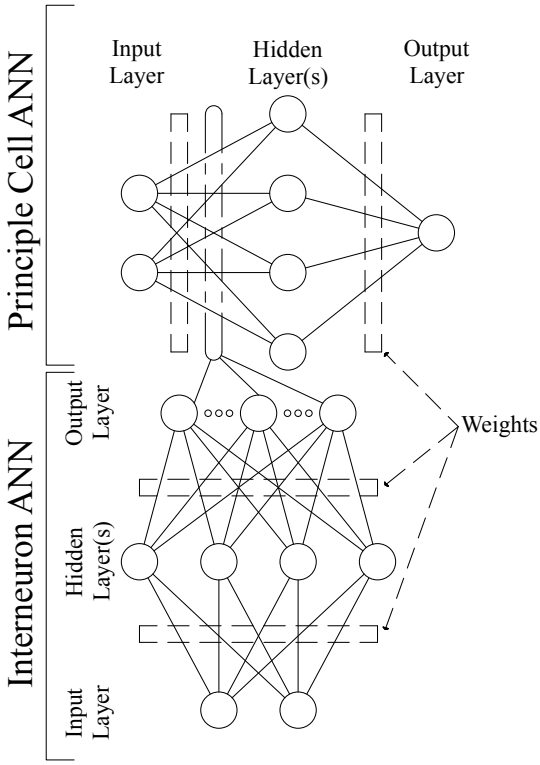

Figure 2: INN Diagram

Backpropagation

One benefit of choosing to allow an ANN to learn to output the β values is backpropagation. We can extend backpropagation so that it trains the interneuron ANN as well as the principal cell ANN. The relevant changes are presented here. Only those elements differing from standard backpropagation will be discussed. The derivatives that are not affected are listed in the following equations. Note that the addition of the term k designates the neuron from the previous layer whose output is projecting into the neuron in question. Then weight w_{kj} is the weight associated with the projection from neuron w in the previous layer to neuron j in the current.

$$E = \frac{1}{2}\sum_j (t_j - net_j)^2 \tag{9}$$

$$\frac{\delta E}{\delta net_j} = -(t_j - net_j) \tag{10}$$

$$\frac{\delta net_j}{\delta out_j} = out_j(1 - out_j) \tag{11}$$

$$\frac{\delta E}{\delta w_{kj}} = \frac{\delta E}{\delta net_j}\frac{\delta net_j}{\delta out_j}\frac{\delta out_j}{\delta w_{kj}} \tag{12}$$

$$\Delta w_{kj} = -\alpha\frac{\delta E}{\delta w_{kj}} \tag{13}$$

First, we will discuss the changes to the equations regarding principal ANN backpropagation. The equations not needing a change are presented in Eqs 9 - 13. The first change to note is the change to the derivatives of the neuron output out_{kj}. The derivative is based on Eq 7. The value out_{kj} is the input to the σ function of the j^{th} neuron in the k^{th} layer. The out_{kj} term is derived with respect to each of the three constituents. First, it is derived with respect to the weight w_{kj} to be able find the new weight value. Likewise, it is derived with respect to the input I_{kj}. This is for the purpose of continuing to backpropagate the error into the network. The new β_{kj} term is the dynamic weight provided by the interneuron ANN. Finding $\frac{\delta out_{kj}}{\delta \beta_{kj}}$ backpropagates the error into the β value. The relevant changes are shown in Eqs 14 - 16.

It is worth noting that when β is equal to one, the derivatives evaluate to the standard ANN backpropagation equations. So, standard backpropagation is a special case of INN backpropagation. This may prove to be of use in multi-task learning.

$$\frac{\delta out_j}{\delta w_{kj}} = \beta_{kj} I_{kj} \tag{14}$$

$$\frac{\delta out_j}{\delta I_{kj}} = \beta_{kj} w_{kj} \tag{15}$$

$$\frac{\delta out_j}{\delta \beta_{kj}} = I_{kj} w_{kj} \tag{16}$$

$$\frac{\delta out'_j}{\delta w'_{kj}} = I'_{kj} \tag{17}$$

$$\frac{\delta out'_j}{\delta I'_{kj}} = w'_{kj} \tag{18}$$

$$\Delta w'_{kj} = -\alpha' \frac{\delta \beta_{kj}}{\delta w'_{kj}} \tag{19}$$

The rest of the important changes are those regarding the interneuron ANN. As we did with the principal ANN, the equations that are left unchanged are presented in Eqs 17 - 19. First, the INN total network error must be propagated to the interneuron output. This is captured in $\frac{\delta E}{\delta \beta_{kj}}$ below. This value is the magnitude of error change with respect to the β dynamic weight. Then the goal from this point of view is to backpropagate this new error term into the interneuron ANN, allowing it to update the interneuron generation weights. The important derivative here is $\frac{\delta \beta_{kj}}{\delta w'_{kj}}$. The equations that were edited for the purpose of interneuron ANN backpropagation are listed in Eqs 20 - 22.

$$\frac{\delta E}{\delta \beta_{kj}} = \frac{\delta E}{\delta net_j}\frac{\delta net_j}{\delta out_j}\frac{\delta out_j}{\delta \beta_{kj}} \tag{20}$$

$$\frac{\delta \beta_{kj}}{\delta w'_{kj}} = \frac{\delta E}{\delta \beta_{kj}}\frac{\delta \beta_{kj}}{\delta out'_j}\frac{\delta out'_j}{\delta w'_{kj}} \tag{21}$$

$$\frac{\delta \beta_{kj}}{\delta out'_j} = out'_j(1 - out'_j) \tag{22}$$

Applying the Model to MNIST

MNIST is a well known problem in machine learning. It consists of 60000 training images of handwritten digits and 10000 test images. While convolutional neural networks have performed very well in image recognition due to their shift invariance, other forms of ANN have been very successful at MNIST image classification. We have applied the INN model to learning the MNIST problem.

The goal here was to provide proof of concept. To prove this the model learns and converges to a solution in a multi-output, large input environment. The INN has been implemented in Tensorflow. For our first revision of the algorithm, it has been implemented using a unique combination of the base available Tensorflow functions. In the future we intend to implement a generalized custom operation and make the code available via github as an extension to Tensorflow. Interneurons were only applied to the hidden layer. This limitation was placed to allow the algorithm to train more quickly.

Table 3: INN vs ANN MNIST Parameters

Data	ANN	INN Principal Cell ANN
Input Neurons	784	784
Hidden Layer 1 Neurons	256	256
Hidden Layer 2 Neurons	256	256
Output Neurons	10	10
Learning Rate	0.001	0.001
Batch Size	100	100
Training Epochs	100	100

The values presented in Table 3 show that the two networks, the ANN and the principal cell ANN are equivalent. The values for the interneuron ANN are of more interest. It was desired to have an interneuron connection on each edge to edge connection between the two hidden layers. There are 256^2 connections between the hidden layers. All other values for the interneuron ANN were the same as the principal cell ANN in the table.

The INN was trained as a whole. The principal cell ANN and interneuron ANN were optimized simultaneously with the derivatives presented herein.

Results

The INN outperformed the ANN. The generalization performance of the ANN as applied to the test set was 96.9 while the performance of the INN was 97.2. The important result here is that the model has been shown to converge and performs better than the naive case.

The reason it did not outperform the ANN more is most likely due to the problem domain. The power of the INN is its ability to contextualize the network dynamically. For this to be allowed to happen a staggered training algorithm may need to be implemented. This is part of our immediate future work as we begin to apply the now validated model on multi-task problems.

Future Work

Our future work is listed here. Primarily it deals with computational experiments and further extension of the model.

Neuroscience

The theory that has been liberally applied in this paper is a simple but powerful assertion. Interneurons possess the ability to uniquely gate the relative contribution of any input selectively. But in truth the brain is a very complex mechanism and there is little certainty regarding underlying mechanics. So, as neuroscience has not been able to make a definitive assertion about interneuron input feature selection, we intend to provide an answer using this model. We have chosen a Gen 2 framework because more is computationally known in this domain. The hope is that through the isolation of variables an answer will be approximated.

Multi-Task and Transfer Learning

The literature review was rich with similar work in multi-task learning. It is hoped that by training the principal ANN weights with unity interneuron ANN weights, the special case of INN backpropagation, the principal cell ANN will converge to task 1 and allow the interneuron ANN to learn task differentiation. A similar idea can be applied to experiment with transfer learning.

References

Larry F Abbott. Lapicques introduction of the integrate-and-fire model neuron (1907). *Brain research bulletin*, 50(5-6):303–304, 1999.

John V Arthur and Kwabena A Boahen. Synchrony in silicon: The gamma rhythm. *IEEE Transactions on Neural Networks*, 18(6):1815–1825, 2007.

Tamas Freund and Szabolcs Kali. Interneurons. *Scholarpedia*, 3(9):4720, 2008.

Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500–544, 1952.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, page 201611835, 2017.

Dmitry E Postnov, Ludmila S Ryazanova, and Olga V Sosnovtseva. Functional modeling of neural–glial interaction. *BioSystems*, 89(1-3):84–91, 2007.

D Purves, GJ Augustine, D Fitzpatrick, WC Hall, A Mcnamara Lamantia, and Williams JO. Sm (2004) neuroscience, 2004.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

Alfons Schuster and Daniel Berrar. Towards nature-inspired modularization of artificial neural networks via static and dynamic weights. In *Biomedical Informatics and Technology*, pages 219–234. Springer, 2014.

Leslie Valiant. *Probably Approximately Correct: NatureÕs Algorithms for Learning and Prospering in a Complex World*. Basic Books (AZ), 2013.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066, 2013.