How to Best Apply Deep Neural Networks in Geosciences: Towards Optimal "Averaging" in Dropout Training

Afshin GHOLAMY, Justin PARRA, Vladik KREINOVICH,
Olac FUENTES, & Elizabeth ANTHONY

Abstract The main objectives of geosciences is to find the current state of the Earth – i.e., solve the corresponding *inverse problems* – and to use this knowledge for predicting the future events, such as earthquakes and volcanic eruptions. In both inverse and prediction problems, often, machine learning techniques are very efficient, and at present, the most efficient machine learning technique is deep neural training. To speed up this training, the current deep learning algorithms use *dropout* techniques: they train several sub-networks on different portions of data, and then "average" the results. A natural idea is to use arithmetic mean for this "averaging", but empirically, geometric mean works much better. In this paper, we provide a theoretical explanation for the empirical efficiency of selecting geometric mean as the "averaging" in dropout training.

Key words: geosciences, deep learning, dropout training, averaging, geometric mean, optimization

1 Introduction

Main objectives of science. The main objectives of science are:

- to determine the state of the world, and
- based on this knowledge, to predict the future state of the world.

Afshin Gholamy and Elizabeth Anthony
Department of Geological Sciences, University of Texas at El Paso, El Paso, Texas 79968, USA, e-mail: afshingholamy@gmail.com, eanthony@utep.edu

Justin Parra, Vladik Kreinovich, Olac Fuentes
Department of Computer Science, University of Texas at El Paso, El Paso, Texas 79968, USA, e-mail: jrparra@miners.utep.edu, vladik@utep.edu, ofuentes@utep.edu

For example, in geosciences:

- we want to determine the density at different depths and at different locations based on the observed data – seismic, gravitational, etc. (this is known as the *inverse problem*) and
- based on this knowledge, we would like to be able to predict catastrophic events such as earthquake and volcanic eruptions (this is known as the *prediction problem*).

Machine learning is often needed. In some situations, we know the equations describing the physical phenomena, and we can use these equations to make necessarily determinations and predictions. This is how, e.g., weather is predicted.

In many other situations, however, we either do not know the exact equations – or these equations are too difficult to solve. In such situations, instead of using specific equations, we can use general *machine learning* tools. In both problems:

- we start with a tuple of measurement results x, and
- we would like to estimate the tuple y of the desired quantities – e.g., the density values or the values describing the future volcanic activity.

To make this prediction, we need to have a database of *patterns*, i.e., pairs $\left(x^{(k)}, y^{(k)}\right)$ corresponding to past situations in which we know both x and y.

For example, if we are interested in predicting volcanic activity at least a week in advance, we need to use patterns in which $y^{(k)}$ is the observed volcanic activity and $x^{(k)}$ are measurement results performed at least a week before the corresponding activity.

Which machine learning techniques should we use: need for deep learning. Learning is what living creatures have to do in order to survive. To learn, living creatures use signals processed a network of special cells – neurons. It is reasonable to assume that as a result of billions of years of improving winner-takes-all evolution, nature has come up with an optimal – or near-optimal – way of learning. And indeed, artificial neural networks – that are based on simulating networks of biological neurons – are, at present, the most efficient machine learning technique; see, e.g., [10].

Specifically, the most efficient technique involves *deep learning*, where we have a large number of layers with reasonably few neurons in each layer; the advantages of such an arrangement are presented in [2, 10].

Deep neural networks has indeed been efficient in geosciences, both in inverse problem (see, e.g., [9]) and in prediction problem (see, e.g., [8, 12, 13, 14]).

Need to speed up the learning process. To get a good description of the corresponding phenomenon, it is desirable to have a large number of patterns. As a result, training on all these patterns takes time. It is thus desirable to speed up computations.

When a person has a task that takes too long to do it by him/herself, a natural idea of speeding it up is to ask for help and to have several people performing this task in parallel. Similarly, a natural way to speed up computations is to perform them in parallel, on several processors.

Need to speed up the learning process naturally leads to dropout training. For traditional neural networks, when we had a large number of neurons in each layer, parallelization was reasonably natural: we just divide the neurons into several groups, and have each processor simulate neurons from the corresponding group.

However, for deep neural networks, there is a relatively small number of neurons in each layer, so we cannot apply the above natural parallelization. A natural alternative idea is:

- to use different parts of the data for training (in parallel) on different sub-networks of the network, and
- then to "average" the results.

Since for each of these trainings, we drop some of the patterns and some of the neurons, this idea is known as a *dropout*; see, e.g., [10, 15, 16].

Which "averaging" works better in deep learning-related dropout training? What is the best way to "average" the values $v_1, \ldots, v_m$ obtained from different parallel trainings? The original idea was to use an arithmetic average, i.e., to use the value v for which

- adding m identical copies of the value v leads to exactly the same result as
- adding m training results $v_1, \ldots, v_m$:

$$v_1 + \ldots + v_m = v + \ldots + v.$$

In this scheme, we get

$$v = \frac{v_1 + \ldots + v_m}{m}.$$

However, it turned out that better results are attained if, instead of addition, we use different combination rules $a * b$. In this case, as the result of such "averaging", we take the value v for which

$$v_1 * \ldots * v_m = v * \ldots * v.$$

In particular, it turned out the empirically, the best results are attained if, as a combination $a * b$, we use product instead of the sum [10, 18]. In this case, the result of "averaging" is the geometric mean:

$$v = \sqrt[m]{v_1 \cdot \ldots \cdot v_m}.$$

Comment. Usually, the values are re-scaled, so that they fit into an interval, e.g., $[0,1]$. So, without losing generality, we can assume that all the averaged values v_i are non-negative.

How can we explain this empirical success? The paper [18] has some *qualitative* explanations of why geometric mean works better than arithmetic one in deep-learning related dropout training. However, it does not provide a *quantitative* explanation of why namely the "averaging" based on multiplication works best.

What we do in this paper. In this paper, we provide an explanation for the empirical success of geometric mean in deep learning-related dropout training. To be more precise, we list all "averaging" operations corresponding to optimal combination functions – optimal under all possible reasonable optimality criteria. As a result, we get a 1-D family of possible "averaging" operations – and it turns out that this list contains arithmetic and geometric means as particular cases.

Thus, we provide a quantitative explanation of the empirical success of geometric mean in deep learning-related dropout training.

Comment. Cannot we do better and explain why *only* the geometric mean is the best? Probably this is possible if we explicitly select *one* optimality criterion. However, in our general formulation, when we allow *all* possible optimality criteria, the appearance of the arithmetic average is inevitable: it corresponds, for example, to using the Least Squares optimality criterion

$$\sum_{i=1}^{m}(v_i - v)^2 \to \min,$$

a criterion that often makes sense in machine learning; see, e.g., [4, 10].

2 What Is a Combination Operation? What Is a Reasonable Optimality Criterion? Towards Precise Definitions

What is a combination operation? A combination operation (also known as an *aggregation operation* or an *aggregation function*) $a * b$ is a function that maps two non-negative numbers a and b into a non-negative number $a * b$.

There are many different combination operations; see, e.g., [3, 5, 6, 7, 11, 17]. What are the reasonable properties of the combination operations used in deep learning-related dropout training?

First reasonable property of a combination operation used in deep learning-related dropout training: commutativity. We have several

results v_i that were obtained by using the same methodology – the only difference is that we randomly selected a different set patterns and we randomly selected a different sub-network. From this viewpoint, there is no reason to believe that some of these results are more valuable than others. Thus, it makes sense to require that the result of combining two values should not depend on the order in which they are presented, i.e., that $a * b = b * a$ for all a and b.

In other words, it is reasonable to require that the combination operation be commutative.

Second reasonable property of a combination operation used in deep learning-related dropout training: associativity. If we have three values a, b, and c, then we can:

- first combine a and b and get $a * b$, and
- add, combine the result $a * b$ with c, resulting in $(a * b) * c$.

Alternatively, we can:

- first combine b nd c into a single value $b * c$, and
- then combine a with the result $b * c$ of combining b and c, thus getting

$$a * (b * c).$$

It is reasonable to require that the result of combining the three values should not depend on the order in which we combine them, i.e., that we should have

$$(a * b) * c = a * (b * c).$$

In other words, it is reasonable to require that the combination operation be associative.

Third reasonable property of a combination operation used in deep learning-related dropout training: monotonicity. It is reasonable to require that if one of the combined values increases, then the result of the combination should also increase (or at least not decrease). In other words, it is reasonable to require that $a * b$ is a (non-strictly) increasing function of each of the variables:

- if $a \le a'$, then $a * b \le a' * b$, and
- if $b \le b'$, the $a * b \le a * b'$.

Final reasonable property of a combination operation used in deep learning-related dropout training: continuity. In practice, all the values are estimated only approximately. It is therefore reasonable to require that a small difference between the ideal value v_i and the corresponding approximate computational result should not drastically affect the result of the combination.

In precise terms, this means that the operation $a * b$ should be continuous.

Towards the resulting definition of a combination function. So, we define a combination operation as a commutative, associative, monotonic continuous function $a * b$ of two real non-negative variables.

What is optimality criterion: a general discussion. Out of all possible combination operations $*$, we should select the one which is, in some reasonable sense, optimal for deep learning-related dropout training. How can we describe the corresponding optimality criterion?

In many practical problems, when we talk about optimization, we have an objective function whose value we want to maximize or minimize. However, this is not the most general case of optimization.

For example, if we select an algorithm a for solving a certain problem, and we are interested in achieving the fastest possible average computation time $A(a)$, we may end up with several different algorithms a, a', ..., that have the exact same average computation time $A(a) = A(a') = \ldots$ In this case, it makes sense to use this non-uniqueness to optimize something else: e.g., the worst-case computation time $W(a)$, or the robustness $R(a)$ relative to uncertainty of the inputs. Then, the actual optimality criterion that we use to select the optimal algorithm can no longer be reduced to a single numerical objective function, this criterion is more complex. Namely, in the resulting criterion, a is better than or of the same quality as a' (we will denote it by $a \geq a'$) if:

- either $A(a) < A(a')$,
- or $A(a) = A(a')$ and $W(a) \leq W(a')$.

If there are several algorithms which are optimal with respect to this new optimality criterion, then we can use the remaining non-uniqueness to optimize something else, and thus, get an even more complex optimality criterion.

This can continue until we finally get a criterion for which there is exactly one optimal alternative.

From this viewpoint, to define an optimality criterion, we should not restrict ourselves to numerical objective functions, we should have the most general definition.

No matter how complex the criterion, what we need is to be able to compare two different alternatives:

- either a is better than b ($a > b$),
- or b is better than a ($b > a$),
- or these two alternatives are of the same quality ($a \equiv b$).

Of course, this selection must be consistent: if a is better than b and b is better than c, then we should be able to conclude that a is better than c. In other words, the preference relation should be transitive.

From this viewpoint, it is reasonable to define an optimality criterion as a *pre-ordering relation*, i.e., a relation $a \geq b$ which is transitive and *reflexive* (i.e., $a \geq a$ for all a).

Which optimality criteria are reasonable?

First reasonable property of an optimality criterion for comparing different deep learning-related combination operations: the optimality criterion should be final. As we have mentioned earlier, if the optimality criterion selects several different alternatives as equally good, this means that this optimality criterion is not final: we still need to come up with an additional criterion for selecting one of these "optimal" alternatives. Selecting this additional criterion means that we modify the original optimality criterion $\geq$.

At the end, we should end up with a final criterion, for which there is only one optimal alternative.

Comment. It goes without saying that there should be at least one optimal alternative – otherwise, if no alternative is optimal, what should we choose?

Second reasonable property of an optimality criterion for comparing different deep learning-related combination operations: the optimality criterion should be scale-invariant. As we have mentioned earlier, the values v_i are usually obtained from re-scaling. Usually, we re-scale to the interval $[0,1]$ by dividing all the values by the largest possible value of the corresponding quantity.

The resulting re-scaling is not unique: e.g., if we add one more quantity which is somewhat larger than what we have seen so far, then the maximum increases, and we need to re-scale the original values some more, i.e., replace the original values v_i with res-scaled values $\lambda \cdot v_i$.

In some cases, the values v_i are not values of the physical quantity but probabilities. In this case, the value are already in the interval $[0,1]$. However, re-scaling is possible in this case as well. Namely, most probabilities that we deal with are *conditional* probabilities, and if we slightly change the condition, this leads to a re-scaling of the corresponding probabilities. Indeed, in general, $P(A\,|\,B) = \dfrac{P(A\,\&\,B)}{P(B)}$. So, if $B \subset B'$, then for each event $A \subseteq B$, we have $P(A\,|\,B) = \dfrac{P(A)}{P(B)}$ and $P(A\,|\,B') = \dfrac{P(A)}{P(B')}$. Thus, if we replace the original condition B with the new condition B', then all conditional probabilities are re-scaled: $P(A\,|\,B') = \lambda \cdot P(A\,|\,B)$, where $\lambda \stackrel{\text{def}}{=} \dfrac{P(B)}{P(B')}$.

If instead of the original values a and b, we consider re-scaled values $a' = \lambda \cdot a$ and $b' = \lambda \cdot b$, then, instead of the combined value $a * b$, we get a new combined value $(\lambda \cdot a) * (\lambda \cdot b)$. We can re-scale it back into the old units, and get a new operation

$$a *_\lambda b = \lambda^{-1} \cdot ((\lambda \cdot a) * (\lambda \cdot b)).$$

This re-scaling should not affect the relative quality of different combination operations:

- if a combination operation $*$ was better than a combination operation $*'$, i.e., if we had $* > *'$,
- then after re-scaling, we should get the same preference: $*_\lambda > *'_\lambda$.

In this sense, the optimality criterion for comparing different deep learning-related combination operations should be *scale-invariant*.

This, we arrive at the following definitions.

3 Definitions and the Main Result

Definition 1. *By a* combination operation*, we mean a commutative, associative, continuous operation $a*b$ that transforms two non-negative real numbers a and b into a non-negative real number $a*b$ and which is (non-strictly) monotonic in each of the variables, i.e.:*

- *if $a \le a'$, then $a*b \le a'*b$, and*
- *if $b \le b'$, then $a*b \le a*b'$.*

Definition 2. *By a* reasonable optimality criterion*, we mean a pre-ordering (i.e., transitive and reflexive) relation $\ge$ on the set of all combination operations which is:*

- final*, in the sense that for this criterion, there exist only one optimal combination operation $*_{\rm opt}$ for which $\forall * \, (*_{\rm opt} \ge *)$; and*
- scale-invariant*: for every $\lambda > 0$, if $* \ge *'$, then $*_\lambda \ge *'_\lambda$, where*

$$a *_\lambda b \stackrel{\rm def}{=} \lambda^{-1} \cdot ((\lambda \cdot a) * (\lambda * b)).$$

Proposition. *For every reasonable optimality criterion, the optimal combination operation has one of the following forms: $a*b = 0$, $a*b = \min(a,b)$, $a*b = \max(a,b)$, and $a*b = (a^\alpha + b^\alpha)^{1/\alpha}$ for some α.*

Discussion. What are the "averaging" operations corresponding to these optimal combination operations?

For $a*b = 0$, the property $v_1 * \ldots * v_m = v * \ldots * v$ is satisfied for any possible v, so this combination operation does not lead to any "averaging" at all.

For $a*b = \min(a,b)$, the condition $v_1 * \ldots * v_m = v * \ldots * v$ leads to

$$v = \min(v_1, \ldots, v_m).$$

For $a*b = \max(a,b)$, the condition $v_1 * \ldots * v_m = v * \ldots * v$ leads to

$$v = \max(v_1, \ldots, v_m).$$

This "averaging" operation is actually sometimes used in deep learning – although not in dropout training [10].

Finally, for the combination operation $a * b = (a^\alpha + b^\alpha)^{1/\alpha}$, the condition $v_1 * \ldots * v_m = v * \ldots * v$ leads to

$$v = \left(\frac{v_1^\alpha + \ldots + v_m^\alpha}{m}\right)^{1/\alpha}.$$

For $\alpha = 1$, we get arithmetic average, and for $\alpha \to 0$, we get the geometric mean – the combination operation which turned out to be empirically the best for deep learning-related dropout training.

Indeed, in this case, the condition $v_1 * \ldots * v_m = v * \ldots * v$ takes the form

$$(v_1^\alpha + \ldots + v_m^\alpha)^{1/\alpha} = (v^\alpha + \ldots + v^\alpha)^{1/\alpha},$$

which is equivalent to

$$v_1^\alpha + \ldots + v_m^\alpha = m \cdot v^\alpha.$$

For every real value a, we have

$$a^\alpha = (\exp(\ln(a))^\alpha = \exp(\alpha \cdot \ln(a)).$$

For small x, $\exp(x) \approx 1 + x$, so $a^\alpha \approx 1 + \alpha \cdot \ln(a)$. Thus, the above condition leads to

$$(1 + \alpha \cdot \ln(v_1)) + \ldots + (1 + \alpha \cdot \ln(v_m)) = m \cdot (1 + \alpha \cdot \ln(v)),$$

i.e., to

$$m + \alpha \cdot (\ln(v_1) + \ldots + \ln(v_m)) = m + m \cdot \alpha \cdot \ln(v),$$

and thus, to

$$\ln(v) = \frac{\ln(v_1) + \ldots + \ln(v_m)}{m} = \frac{\ln(v_1 \cdot \ldots \cdot v_m)}{m};$$

hence to $v = \sqrt[m]{v_1 \cdot \ldots \cdot v_m}$.

So, we indeed have a 1-D family that contains combination operations efficiently used in deep learning:

- the arithmetic average that naturally comes from the use of the Least Squares optimality criterion, and
- the geometric mean, empirically the best combination operation for deep learning-related dropout training.

4 Proof

1°. Let us first prove that the optimal combination operation $*_{\rm opt}$ is scale-invariant, i.e., $(*_{\rm opt})_\lambda = *_{\rm opt}$ for all λ.

Indeed, let us take any λ and consider the combination operation $(*_{\rm opt})_\lambda$. By definition, $*_{\rm opt}$ is the optimal combination operation, so $*_{\rm opt} \ge *$ for all combination operations $*$. In particular, for every combination operation $*$, we have $*_{\rm op} \ge *_{\lambda^{-1}}$. Thus, by scale-invariance, we have $(*_{\rm opt})_\lambda \ge (*_{\lambda^{-1}})_\lambda = *$. So, $(*_{\rm opt})_\lambda$ is better than or of the same quality than any other combination operation $*$. This means that the combination operation $(*_{\rm opt})_\lambda$ is optimal.

However, our optimality criterion is reasonable hence final; thus, it has only one optimal combination operation. Hence, $(*_{\rm opt})_\lambda = *$.

By definition of the re-scaling operation $*_\lambda$, this means that

$$\lambda^{-1} \cdot ((\lambda \cdot a) * (\lambda \cdot b)) = a * b,$$

i.e., equivalently, that

$$(\lambda \cdot a) * (\lambda \cdot b) = \lambda \cdot (a * b). \tag{1}$$

2°. To complete the proof of the Proposition, we can now use our result – proven in [1] – that every combination operation * that satisfied the condition (1) has one of the following forms: $a * b = 0$, $a * b = \min(a, b)$, $a * b = \max(a, b)$, and $a * b = (a^\alpha + b^\alpha)^{1/\alpha}$ for some α.

The Proposition is thus proven.

5 Conclusions

In many application areas, it is important to make accurate predictions of future events. At present, among all machine learning techniques, deep learning algorithms leads to the most accurate predictions. However, this accuracy comes at a price – deep learning algorithms require much more computation time for training than any other machine learning techniques. To speed up the training, researchers have proposed the "dropout" idea:

- we train different patterns on different sub-networks on the neural network, and then
- we "average" the results.

Which averaging operation should we use? In many similar situations, a simple arithmetic average works the best – this can be explained by the fact that in many practical cases, the errors are normally distributed, and for normal distributions, arithmetic average is indeed provably the best averaging

operation. So, researchers originally expected that arithmetic average should work the best in dropout training as well. Just in case, they also tried other statistics-motivated averaging operations. Surprisingly, it turned out that for deep learning-related dropout training, neither the arithmetic average not other statistics-motivated arithmetic operations work well. What works the best is the geometric mean, a mathematical operation that does not seem to have a direct statistical motivation.

In this paper, we provide a theoretical explanation for this surprising empirical success of geometric means. Specifically:

- We first analyze what would be reasonable properties for a combination operation used in deep learning-related dropout training and what kind of optimality criteria are appropriate for selecting the best combination operation.
- After that, we prove that for all reasonable optimality criteria, the optimal combination operation belongs to a special 1-parametric family, a family that includes both the usual arithmetic mean and the empirically efficient geometric mean.

Acknowledgments

This work was supported in part by the US National Science Foundation grant HRD-1242122 (Cyber-ShARE Center of Excellence).

The authors are thankful to Dr. Junzo Watada for his help and encouragement and to the anonymous referees for their valuable suggestions.

References

1. Autchariyapanitkul, K., O. Kosheleva, V. Kreinovich, and S. Sriboonchitta, "Quantum econometrics: how to explain its quantitative successes and how the resulting formulas are related to scale invariance, entropy, and fuzziness", In: V.-N. Huynh, M. Inuiguchi, D.-H. Tran, and T. Denoeux (eds.), *Proceedings of the International Symposium on Integrated Uncertainty in Knowledge Modelling and Decision Making IUKM'2018*, Hanoi, Vietnam, March 13–15, 2018 (2018).
2. Baral, C., O. Fuentes, and V. Kreinovich, "Why deep neural networks: a possible theoretical explanation", In: M. Ceberio and V. Kreinovich (eds.), *Constraint Programming and Decision Making: Theory and Applications*, Springer Verlag, Berlin, Heidelberg, pp. 1–6 (2018).
3. Beliakov, G., A. Pradera, and T. Calvo, *Aggregation Functions: A Guide for Practitioners*, Springer Verlag, Berlin, Heidelberg (2007).
4. Bishop, C. M., *Pattern Recognition and Machine Learning*, Springer, New York (2006).
5. Bouchon-Meunier, B. (ed.), *Aggregation and Fusion of Imperfect Information*, Physica Verlag, Berlin, Heidelberg (2013).

6. Bustince, H., J. Fernandez, R. Mesiar, and T. Calvo (eds.), *Aggregation Functions in Theory and in Practice*, Springer Verlag, Berin, Hidelberg (2013).
7. Calvo, T., G. Mayor, and R. Mesiar (eds.), *Aggregation Operators: New Trends and Applications*, Physica Verlag, Heidelberg, New York (2002).
8. Fuentes, O., J. Parra, E. Anthony, and V. Kreinovich, "Why rectified linear neurons are efficient: a possible theoretical explanations", In: O. Kosheleva, S. Shary, G. Xiang, and R. Zapatrin (eds.), *Beyond Traditional Probabilistic Data Processing Techniques: Interval, Fuzzy, etc. Methods and Their Applications*, Springer, Cham, Switzerland, to appear (2018).
9. Gholamy, A., *Backcalculation of Intelligent Compaction Data for the Mechanical Properties of Soil Geosystems*, PhD Dissertation Proposal, University of Texas at El Paso (2018).
10. Goodfellow, I., Y. Bengio, and A. Courville, *Deep Leaning*, MIT Press, Cambridge, Massachusetts (2016).
11. Grabisch, M., J.-L. Marichal, R. Mesiar, and E. Pap, *Aggregation Functions*, Cambridge University Press, Cambridge, UK (2009).
12. Parra, J., O. Fuentes, E. Anthony, and V. Kreinovich, "Prediction of volcanic eruptions: case study of rare events in chaotic systems with delay", *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics SMC'2017,* Banff, Canada, October 5–8, 2017 (2017).
13. Parra, J., O. Fuentes, E. Anthony, and V. Kreinovich, "Use of machine learning to analyze and – hopefully – predict volcano activity", *Acta Politechnica Hungarica*, 14(3), 209–221 (2017).
14. Parra, J., O. Fuentes, V. Kreinovich, E. Anthony, V. Espejel, and O. Hinojosa, "Eruption forecasting from seismic activity using neural networks", *Proceedings of International Association of Vulcanology and Chemistry of the Earth's Interior (IAVSEI) Scientific Assembly IAVCEI'2017 "Fostering Integrative Studies of Volcanism"*, Portland, Oregon, August 14–17, 2017 (2017).
15. Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhuinov, "Dropout: a simple way to prevent neural networks from overfitting", *Journal of Machine Learning Research*, 15, 1929–1958 (2014).
16. Szegedy, C., W. Liu, Y. Jia, P. Serament, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, *Going Deeper with Convolution*. arXiv:1409.4842 (2014).
17. Torra, V., R. Mesiar, and B. De Baets (eds.), *Aggregation Functions in Theory and in Practice*, Springer, Cham, Switzerland (2018).
18. Warde-Farley, D., J. J. Goodfellow, A. Courville, and Y. Bendigo, "An empirical analysis of dropout in piecewise linear networks", *Proceedings of the 2nd International Conference on Learning Representations ICLR'2014*, Banff, Canada, April 14–16, 2014 (2014).