Why Max and Average Poolings are Optimal in Convolutional Neural Networks

Ahnaf Farhan, Olga Kosheleva, and Vladik Kreinovich

University of Texas at El Paso, El Paso, Texas 79968, USA
afarhan@miners.utep.edu, olgak@utep.edu, vladik@utep.edu

1. Need for Data Processing

- The main objectives of science and engineering are:
  - to describe the world,
  - to predict the future behavior of the world's systems, and
  - to find the best way to improve this behavior.
- The current state of the world is described by numerical values of different physical quantities.
- Some of these values can be directly measured; e.g., we can measure:
  - the distance to a nearby city,
  - the temperature, humidity, and wind speed at different Earth locations.
- Other quantities are difficult (or even impossible) to measure directly.

2. Need for Data Processing (cont-d)

- Examples:
  - the distance to a nearby star,
  - the temperature on the surface of the Sun, etc.
- Since we cannot measure these quantities y directly, we have to determine them indirectly: namely,
  - we measure the values of easier-to-measure quantities $x_1, \ldots, x_n$ which are related to y, and then
  - use the measurement results $\widetilde{x}_1, \ldots, \widetilde{x}_n$ to compute an estimate $\widetilde{y}$ for the desired quantity y.
- The corresponding computations form an important case of *data processing*.
- Similar computations are needed to estimate:
  - the future values of the quantities of interest and
  - the values of necessary control.

3. Need for Machine Learning

- In some cases, we know the exact relation $y = f(x_1, \ldots, x_n)$.
- E.g., we can predict the future locations of planets.
- In other cases, we need to determine the corresponding relation from the available data.
- Namely, in several situations $k = 1, \ldots, K$:
  - we measure the values $x_1^{(k)}, \ldots, x_n^{(k)}, y^{(k)}$, and
  - then use this data to find a dependence $f(x_1, \ldots, x_n)$ for which $y^{(k)} \approx f\left(x_1^{(k)}, \ldots, x_n^{(k)}\right)$ for all k.
- Algorithms for reconstructing the dependence from empirical data are known as *machine learning*.
- At present, the most efficient machine learning algorithms are the algorithms of *deep neural networks*.

4. Need to Take Uncertainty into Account

- In the ideal situation, when all the values are known exactly, it is often easy to find the dependence; e.g.:
  - if it turns out that all the values corresponding to the dependence $y = f(x_1)$ fit a straight line,
  - we conclude that the dependence is linear.
- In reality, measurements are never absolutely accurate.
- There is always measurement uncertainty; as a result:
  - even if the actual dependence is linear,
  - we corresponding pairs $\left(\widetilde{x}_1^{(k)}, \widetilde{y}^{(k)}\right)$ do not lie on the same straight line.

5. Need for Convolutional Neural Networks

- In many practical situations, the available data comes:
  - in terms of *time series* – when we have values measured at equally spaced time moments – or
  - in terms of an *image* – when we have data corresponding to a grid of spatial locations.
- Neural networks for processing such data are known as *convolutional neural networks*.

6. Need for Pooling

- We want to decrease the distortions caused by measurement errors.
- For that, we take into account that usually, the actual values at nearby points in time or space are close to each other.
- As a result,
  - instead of using the measurement-distorted value at each point,
  - we can take into account that values at nearby points are close, and
  - combine ("pool together") these values into a single more accurate estimate.

7. Which Pooling Techniques Work Better: Empirical Results

- In principle, we can have many different pooling algorithms.
- It turns out that empirically, in general, the most efficient pooling algorithm is *max-pooling*:

$$a = \max(a_1, \ldots, a_m).$$

- The next efficient is *average pooling*, when we take the arithmetic average $a = \dfrac{a_1 + \ldots + a_m}{m}$.
- In this paper, we provide a theoretical explanation for this empirical observation.
- Namely, we prove that max and average poolings are indeed optimal.

8. What Is Pooling: Towards a Precise Definition

- We start with m values $a_1, \ldots, a_m$, and we want to generate a single value a that represents all these values.
- In the case of arithmetic average, we select a for which $a_1 + \ldots + a_m = a + \ldots + a$ (m times).
- In general, pooling means that:
  - we select some combination operation $*$ and
  - we then select the value a for which $a_1 * \ldots * a_m = a * \ldots * a$ (m times).
- For example:
  - if, as a combination operation, we select $\max(a, b)$,
  - then the corresponding condition $\max(a_1, \ldots, a_n) = \max(a, \ldots, a) = a$ describes the max-pooling.
- From this viewpoint, selecting pooling means selecting an appropriate combination operation.

9. Natural Properties of a Combination Operation

- The combination operation transforms:
  - two non-negative values – such as intensity of an image at a given location
  - into a single non-negative value.
- The result of applying this operation should not depend on the order in which we combine the values.
- Thus, we should have $a * b = b * a$ (commutativity) and $a * (b * c) = (a * b) * c$ (associativity).

10. What Does It Mean to Have an Optimal Pooling?

- Optimality means that on the set of all possible combination operations, we have a preference relation $\preceq$.
- $A \preceq B$ means that the operation B is better than (or of the same quality as) the operation A.
- This relation should be transitive:
  - if C is better than B and B is better than A,
  - then C should be better than A.
- An operation A is optimal if it is better than (or of the same quality as) any other operation B: $B \preceq A$.
- For some preference relations, we may have several different optimal combination operations.
- We can then use this non-uniqueness to optimize something else.

11. What Is Optimal Pooling (cont-d)

- Example:
  - if there are several different combination operations with the best average-case accuracy,
  - we can select, among them, the one for which the average computation time is the smallest possible.
- If after this, we still get several optimal operations,
  - we can use the remaining non-uniqueness
  - to optimize yet another criterion.
- We do this until we get a *final* criterion, for which there is only one optimal combination operation.

12. Scale-Invariance

- Numerical values of a physical quantity depend on the choice of a measuring unit.
- For example, if we replace meters with centimeters, the numerical quantity is multiplied by 100.
- In general:
  - if we replace the original unit with a unit which is λ times smaller,
  - then all numerical values get multiplied by λ.
- It is reasonable to require that the preference relation should not change if we change the measuring unit.
- Let us describe this requirement in precise terms.

13. Scale-Invariance (cont-d)

- If, in the original units, we had the operation $a * b$, then, in the new units, the operation will be as follows:
  - first, we transform the value a and b into the new units, so we get $a' = \lambda \cdot a$ and $b' = \lambda \cdot b$;
  - then, we combine the new numerical values, getting $(\lambda \cdot a) * (\lambda \cdot b)$;
  - finally, we re-scale the result to the original units, getting $a R_\lambda(*) b \stackrel{\text{def}}{=} \lambda^{-1} \cdot ((\lambda \cdot a) * (\lambda \cdot b))$.
- It therefore makes sense to require that if $* \preceq *'$, then for every $\lambda > 0$, we get $R_\lambda(*) \preceq R_\lambda(*')$.

14. Shift-Invariance

- The numerical values also change if we change the starting point for measurements.
- For example, when measuring intensity:
  - we can measure the actual intensity of an image,
  - or we can take into account that there is always some noise $a_0 > 0$, and
  - use the noise-only level a_0 as the new starting point.
- In this case, instead of each original value a, we get a new numerical value $a' = a - a_0$.

15. Shift-Invariance (cont-d)

- If we apply the combination operation in the new units, then in the old units, we get a slightly different result:
  - first, we transform the value a and b into the new units, so we get $a' = a - a_0$ and $b' = b - a_0$;
  - then, we combine the new numerical values, getting

$$(a - a_0) * (b - a_0);$$

  - finally, we re-scale the result to the original units, getting $a S_{a_0}(*) b \stackrel{\text{def}}{=} (a - a_0) * (b - a_0) + a_0$.
- It makes sense to require that the preference relation not change if we simply change the starting point.
- So if $* \preceq *'$, then for every a_0, we get $S_{a_0}(*) \preceq S_{a_0}(*')$.

16. Weak Version of Shift-Invariance

- Alternatively, we can have a weaker version of this "shift-invariance".
- Namely, we require that shifts in a and b imply a possibly different shift in $a * b$, i.e.,
  - if we shift both a and b by a_0,
  - then the value $a * b$ is shifted by some value $f(a_0)$ which is, in general, different from a_0.
- Now, we are ready to formulation our results.

17. Definitions

- *By a* combination operation, *we mean a commutative, associative operation $a * b$ that:*
  - *transforms two non-negative real numbers a and b*
  - *into a non-negative real number $a * b$.*
- *By an* optimality criterion, *we need a transitive reflexive relation $\preceq$ on the set of all combination operations.*
- *We say that a combination operation $*_{\rm opt}$ is* optimal *w.r.t. $\preceq$ if $* \preceq *_{\rm opt}$ for all combination operations $*$.*
- *We say that $\preceq$ is* final *if there exists exactly one $\preceq$-optimal combination operation.*
- *We say that an optimality criterion is* scale-invariant *if for all $\lambda > 0$, $* \preceq *'$ implies $R_\lambda(*) \preceq R_\lambda(*')$, where:*

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

18. Definitions and First Result

- *We say that an optimality criterion is* shift-invariant *if for all a_0, $* \preceq *'$ implies $S_{a_0}(*) \preceq S_{a_0}(*')$, where:*

$$aS_{a_0}(*)b \stackrel{\text{def}}{=} ((a - a_0) * (b - a_0)) + a_0.$$

- *We say that $\preceq$ is* weakly shift-invariant *if for every a_0, there exists $f(a_0)$ s.t. $* \preceq *'$ implies $W_{a_0}(*) \preceq W_{a_0}(*')$,*

$$\textit{where} \quad aW_{a_0}(*)b \stackrel{\text{def}}{=} ((a - a_0) * (b - a_0)) + f(a_0).$$

- **Proposition 1.** *For every final, scale- and shift-invariant $\preceq$, the optimal combination operation $*$ is*

$$a * b = \min(a, b) \text{ or } a * b = \max(a, b).$$

- *This result explains why max-pooling is empirically the best combination operation.*
- Note that this result does not contradict uniqueness as we requested.

19. Results (cont-d)

- Indeed, there are several different final scale- and shift-invariant optimality criteria.
- For each of these criteria, there is only one optimal combination operation.
- For some of these optimality optimality criteria, the optimal combination operation is $\min(a, b)$.
- For other criteria, the optimal combination operation is $\max(a, b)$.
- **Proposition 2.** *For every final, scale-invariant and weakly shift-invariant $\preceq$, the optimal $*$ is:*

$$a * b = 0, \quad a * b = \min(a, b), \quad a * b = \max(a, b), \quad \textit{or}$$

$$a * b = a + b.$$

- *This result explains why max-pooling and average-pooling are empirically the best combination operations.*

20. Acknowledgments

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

21. General Part of the Two Proofs

- Let us first prove that the optimal operation $*_{\rm opt}$ is itself scale-invariant: $R_\lambda(*_{\rm opt}) = *_{\rm opt}$ for all $\lambda > 0$.
- The fact that $*_{\rm opt}$ is optimal means that $* \preceq *_{\rm opt}$ for all $*$.
- In particular, $R_{\lambda^{-1}}(*) \preceq *_{\rm opt}$ for all $*$.
- Due to scale-invariance of the optimality criterion, this implies that $* \preceq R_\lambda(*_{\rm opt})$ for all $*$.
- Thus, the operation $R_\lambda(*_{\rm opt})$ is also optimal.
- But since the optimality criterion is final, there is only one optimal operation, so $R_\lambda(*_{\rm opt}) = *_{\rm opt}$.
- Scale-invariance is proven.
- Shift-invariance is proven similarly.
- For Proposition 2, we can similarly prove that the optimal $*$ is weakly shift-invariant: $W_{a_0}(*_{\rm opt}) = *_{\rm opt}$.

22. Proof of Proposition 1

- Let $a * b$ be the optimal combination operation.
- We have shown that this operation is scale-invariant and shift-invariant.
- Let us prove that it has one of the above two forms.
- For every pair (a, b), we can have three different cases: $a = b$, $a < b$, and $a > b$.
- Let us consider them one by one.
- Let us first consider the case when $a = b$.
- Let us denote $v \stackrel{\text{def}}{=} 1 * 1$.
- From scale-invariance with $\lambda = 2$, from $1 * 1 = v$, we get $2 * 2 = 2v$.
- From shift-invariance with $s = 1$, from $1 * 1 = v$, we get $2 * 2 = v + 1$.

23. Proof of Proposition 1 (cont-d)

- Thus, $2v = v + 1$, hence $v = 1$, and $1 * 1 = 1$.
- For $a > 0$, by applying scale-invariance with $\lambda = a$ to the formula $1 * 1 = 1$, we get $a * a = a$.
- For $a = 0$, if we denote $c \stackrel{\text{def}}{=} 0 * 0$, then, by applying shift-invariance with $s = 1$ to $0 * 0 = c$, we get

$$1 * 1 = c + 1.$$

- Since we already know that $1 * 1 = 1$, this means that $c + 1 = 1$ and thus, that $c = 0$, i.e., that $0 * 0 = 0$.
- So, for all $a \geq 0$, we have $a * a = a$.
- In this case, $\min(a, a) = \max(a, a) = a$, so we have $a * a = \min(a, a)$ and $a * a = \max(a, a)$.
- Let us now consider the case when $a < b$. In this case, $b - a > 0$.

24. Proof of Proposition 1 (cont-d)

- Let us denote $t \stackrel{\text{def}}{=} 0 * 1$.
- By applying scale-invariance with $\lambda = b - a > 0$ to the formula $0 * 1 = t$, we get $0 * (b - a) = (b - a) \cdot t$.
- Now, by applying shift-invariance with $s = a$ to this formula, we get $a * b = (b - a) \cdot t + a$.
- To find possible values of t, let us take into account that the combination operation should be associative.
- This means, in particular, that for all possible triples a, b, and c for which we have $a < b < c$, we must have

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

- Since $b < c$, by the above formula, we have $b * c = (c - b) * t + b$.
- Since $t \geq 0$, we have $b * c \geq b$ and thus, $a < b * c$.

25. Proof of Proposition 1 (cont-d)

- So, to compute $a * (b * c)$, we can also use the above formula, and get $a * (b * c) = (b * c - a) \cdot t + a =$

$$((c - b) \cdot t + b) \cdot t + a = c \cdot t^2 + b \cdot (t - t^2) + a.$$

- Let us restrict ourselves to the case when $a * b < c$.
- In this case, the general formula implies that

$$(a*b)*c = (c-a*b)\cdot t+a*b = (c-((b-a)\cdot t+a))\cdot t+(b-a)\cdot t+a.$$

- So $(a * b) * c = c \cdot t + b \cdot (t - t^2) + a \cdot (1 - t)^2$.
- Due to associativity, the two formulas must coincide for all a, b, and c for which $a < b < c$ and $c > a * b$.
- These two linear expressions must be equal for all sufficiently large values of c.
- Thus, the coefficients at c must be equal, i.e., we must have $t = t^2$.

26. Proof of Proposition 1 (cont-d)

- From $t = t^2$, we conclude that $t - t^2 = t \cdot (1 - t) = 0$, so either $t = 0$ or $1 - t = 0$ (in which case $t = 1$).
- If $t = 0$, then the above formula has the form $a * b = a$, i.e., since $a < b$, the form $a * b = \min(a, b)$.
- If $t = 1$, then the above formula has the form

$$a * b = (b - a) + a = b.$$

- Since $a < b$, we get $a * b = \max(a, b)$.
- If $a > b$, then, by commutativity, we have $a * b = b * a$, where now $b < a$.
- So, either we have $a * b = \min(a, b)$ for all a and b, or we have $a * b = \max(a, b)$ for all a and b.
- The proposition is proven.

27. Proof of Proposition 2

- Let $a * b$ be the optimal combination operation.
- We have proven that this operation is scale-invariant and weakly shift-invariant.
- This means that $a * b = c$ implies $(a + s) * (b + s) = c + f(s)$.
- Let us prove that the optimal operation $*$ has one of the above four forms.
- Let us first prove that $0 * 0 = 0$.
- Indeed, let s denote $0 * 0$.
- Due to scale-invariance, $0 * 0 = s$ implies that $(2 \cdot 0) * (2 \cdot 0) = 2s$, i.e., that $0 * 0 = 2s$.
- So, we have $s = 2s$, hence $s = 0$ and $0 * 0 = 0$.
- Similarly, if we denote $v \stackrel{\text{def}}{=} 1 * 1$, then, due to scale-invariance with $\lambda = a$, $1 * 1 = v$ implies that $a * a = v \cdot a$.

28. Proof of Proposition 2 (cont-d)

- On the other hand, due to weak shift-invariance with $a_0 = a$, $0 * 0 = 0$ implies that $a * a = f(a)$.
- Thus, we conclude that $f(a) = v \cdot a$.
- Let us now consider the case when $a < b$ and, thus, $b - a > 0$.
- Let us denote $t \stackrel{\text{def}}{=} 0 * 1$.
- From scale-invariance with $\lambda = b - a$, from $0 * 1 = t \geq 0$, we get $0 * (b - a) = t \cdot (b - a)$.
- From weak shift-invariance with $a_0 = a$, we get $a * b = t \cdot (b - a) + v \cdot a$, i.e., $a * b = t \cdot b + (v - t) \cdot a$.
- The combination operation should be associative: $a * (b * c) = (a * b) * c$.
- When $b < c$, we have $b * c = t \cdot c + (v - t) \cdot b$.

Need for Data Processing
Need for Pooling
Which Pooling . . .
Natural Properties of a . . .
Scale-Invariance
Shift-Invariance
Definitions
Definitions and First . . .
Results (cont-d)
Home Page
Title Page
Go Back
Full Screen
Close
Quit

29. Proof of Proposition 2 (cont-d)

- We know that $t \geq 0$. This means that we have either $t > 0$ and $t = 0$.
- Let us first consider the case when $t > 0$.
- In this case, for sufficiently large c, we have $b * c > a$.
- So, by applying the above formula to a and $b * c$, we conclude that

$$a*(b*c) = t\cdot(b*c)+(v-t)\cdot a = t^2\cdot c+t\cdot(v-t)\cdot b+(v-t)\cdot a.$$

- For sufficient large c, we also have $a * b < c$.
- In this case, the general formula implies that

$$(a*b)*c = (t\cdot b+(v-t)\cdot a)*c = t\cdot c+t\cdot(v-t)\cdot b+(v-t)^2\cdot a.$$

- Due to associativity, these formulas must coincide for all a, b, and c for which

$$a < b < c, \quad c > a * b, \text{ and } b * c > a.$$

30. Proof of Proposition 2 (cont-d)

- These two linear expressions must be equal for all sufficiently large values of c.
- So, the coefficients at c must be equal, i.e., we must have $t = t^2$.
- From $t = t^2$, we conclude that $t - t^2 = t \cdot (1 - t) = 0$.
- Since we assumed that $t > 0$, we must have $t - 1 = 0$, i.e., $t = 1$.
- The coefficients at a must also coincide, so we must have $v - t = (v - t)^2$, hence either $v - t = 0$ or $v - t = 1$.
- In the first case, the above formula becomes $a * b = b$, i.e., $a * b = \max(a, b)$ for all $a \le b$.
- Since the operation $*$ is commutative, this equality is also true for $b \le a$ and is, thus, true for all a and b.

31. Proof of Proposition 2 (cont-d)

- In the second case, the above formula becomes $a * b = a + b$ for all $a \le b$.
- Due to commutativity, this formula holds for all a, b.
- Let us now consider the case when $t = 0$.
- In this case, the above formula takes the form $a * b = (v - t) \cdot a$.
- Here, $a * b \ge 0$, thus $v - t \ge 0$.
- If $v - t = 0$, this implies that $a * b = 0$ for all $a \le b$ and thus, due to commutativity, for all a and b.
- Let us now consider the remaining case when $v - t > 0$.
- In this case, if $a < b < c$, then for sufficiently large c, we have $a * b < c$, hence

$$(a*b)*c = (v-t)\cdot(a*b) = (v-t)\cdot((v-t)\cdot a) = (v-t)^2\cdot a.$$

32. Proof of Proposition 2 (cont-d)

- On the other hand, here $b * c = (v - t) \cdot b$.
- So, for sufficiently large b, we have $(v - t) \cdot b > a$, thus
$$a * (b * c) = (v - t) \cdot a.$$
- Due to associativity, we have $(v - t)^2 \cdot a = (v - t) \cdot a$, hence $(v - t)^2 = v - t$.
- Since $v - t > 0$, we have $v - t = 1$.
- In this case, the above formula takes the form $a * b = a = \min(a, b)$ for all $a \le b$.
- Thus, due to commutativity, we have $a * b = \min(a, b)$ for all a and b.
- We have thus shown that the combination operation indeed has one of the four forms.
- Proposition 2 is therefore proven.