MDPI

Article

Why Dilated Convolutional Neural Networks: A Proof of Their Optimality

Jonatan Contreras, Martine Ceberio, and Vladik Kreinovich

University of Texas at El Paso, El Paso TX 79968, USA; jmcontreras2@utep.edu, mceberio@utep.edu, vladik@utep.edu

* Correspondence: vladik@utep.edu (V.K.)

Abstract: One of the most effective image processing techniques is the use of convolutional neural networks, where we assign, to each spatial point (pixel) from the original grid, a combination of data corresponding to several neighboring spatial points. To speed up computations (while retaining the same accuracy), researchers have developed a dilated version of this technique, in which only data from *some* neighboring points is processed. It turns out that the most efficient case is when we select neighboring points from a sub-grid. In this paper, we explain this empirical efficiency by proving that for all reasonable optimality criteria, the optimal subset of the original grid is either a sub-grid, or a sub-grid-like set.

Keywords: convolutional neural networks; dilated neural networks; optimality

1. Introduction

At present, one of the most efficient image processing techniques is a *convolutional neural network*; see, e.g., [1].

An image is usually represented by data corresponding to different spatial points (pixels). These spatial points usually form a rectangular grid, i.e., they have the form

$$(x, y) = (x_0 + n_x \cdot \Delta, y_0 + n_y \cdot \Delta),$$

where x_0, y_0, and Δ are fixed, and n_x and n_y are integers. The spatial points constituting the image form a bounded part of the potentially infinite grid formed by all such points $(x_0 + n_x \cdot \Delta, y_0 + n_y \cdot \Delta)$.

The data corresponding to each spatial point can be different:

- For simple black-and-while images, the data corresponding to each point is simply the intensity at this point.
- For the usual color images, for each point, the corresponding data consists of three numbers corresponding to intensities at three basic colors.
- For multi-spectral images – and for other situations where, for each spatial point, we have the results of several measurements – the data corresponding to each point consists of more than three numbers.

In the convolutional neural network, on each step of the corresponding data processing, for each spatial point (x, y), to compute the new value – or values – corresponding to this point, we combine data corresponding to several spatial points (pixels) (x', y') which are close to the given point (x, y). For example, for black-and-white images, usually, either the maximum or a linear combination of the corresponding intensities is computed – and assigned to the central point of this group of neighboring pixels.

One of the features of neural networks – as well as of many other machine learning tools – is that they often take a long time to train. This is not surprising: we humans – whose brain processes are, in effect, simulated in neural networks – are not that fast to learn either. From this viewpoint, it is desirable to decrease the computational time

Citation: Contreras, J.; Ceberio, M.; Kreinovich, V. Why Dilated Convolutional Neural Networks: A Proof of Their Optimality. *Entropy* **2021**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

needed for each training cycle while preserving the learning accuracy – or, alternatively, get a better accuracy within the same computation time.

With respect to convolutional neural networks, a natural speed-up idea is to take into account that the more values we process, the longer this processing takes. Thus, to speed up computations, a reasonable idea is not to process the data corresponding to *all* the neighboring points from the grid, but only the data corresponding to *some* of these points. This idea indeed enabled researchers to achieve a better accuracy within the same computation time. The resulting networks are known as *dilated* convolutional neural networks, since skipping some points is kind of equivalent to extending (dilating) the distance between the remaining points; see, e.g., [3,5,6].

Specifically, the success was attained when, to compute the new value(s) at a point (x, y), we take the data corresponding only to neighboring points $(x + n_x \cdot \Delta, y + n_y \cdot \Delta)$ in which both n_x and n_y are divisible by some integer $k > 1$. In some cases, the best results are attained when $k = 2$, in other cases, when $k = 3$, etc. Such points form a *sub-grid* of the original rectangular grid.

In principle, we could select different points. For example, to compute the new value(s) at a point (x, y), we could take the data corresponding to points

$$(x + n_x \cdot \Delta, y + b_y \cdot \Delta)$$

at which the sum $n_x^2 + n_y^2$ is divisible by 3. However, empirical evidence shows that the sub-grid selection works the best [3,5,6].

To the best of our knowledge, there is *no theoretical explanation* for this empirical result – that selecting neighboring points from a sub-grid leads to better results that selecting neighboring points from another subset of the original grid. The main *objective* of this paper is *to provide* such *an explanation*.

Comment. In this paper, we will only provide this explanation on the general abstract level. For this result to become practically useful, it is necessary to analyze which dilations and which neighborhoods work better for different practical problems and for different actual optimality criteria.

2. Analysis of the Problem

A geometric description of the convolution. Since convolution is mostly used to process 2D images, it makes sense to analyze it from the geometric viewpoint.

From the geometric viewpoint, pixels form a rectangular grid. To simplify the coordinate description of this grid, let us select one of the points on this grid as the starting point $(0, 0)$. As this starting point, we can select the center of the image, we can select one of the vertices of the images, or we can select any other meaningful point.

For the same purpose of simplifying the description of the grid, as the unit of length, let us select the distance between the two neighboring points along each axis. In the resulting coordinate system, all coordinates of all the grid points are integers: all the points on the grid have the form (x, y), where x and y are integers.

Of course, in practice, the grid is finite in both x- and y-directions, so it forms a (large) subset of the set $\mathbb{Z} \times \mathbb{Z}$ of all possible pairs of integers; see Fig. 1.

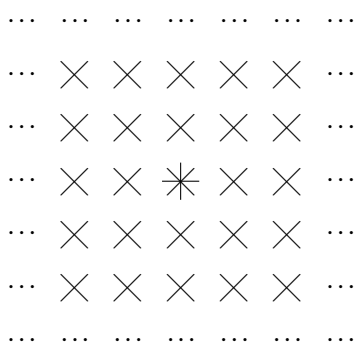

Figure 1: The original grid used for convolution

When we perform convolution, then, to each point (x, y), we assign a combination of data corresponding to the point (x, y) itself, and data corresponding to the grid points which are close to the point (x, y).

Comments.

- In view of the following discussion, it is important to mention that for all the points (x, y), we use the same grid, the only difference is that for different points (x, y), we select different neighborhoods in this same grid.
- We can select different points on the grid as starting points for the coordinate system. Instead of the originally selected point, we can select, as a starting point, a different point whose coordinates in the original coordinate system are (x_0, y_0). In this case, each grid point that had coordinates (x, y) in the original coordinate system now has coordinates $(x - x_0, y - y_0)$.

Geometric description of dilated convolution. In dilated convolution, to compute the new value(s) corresponding to the point $(0, 0)$, we consider only data from the points which are close to $(0, 0)$ and which belong to a *sub-grid* of the original grid – e.g., to the set of all the points (x, y) for which both x and y are integers divisible by 2; see Fig. 2.

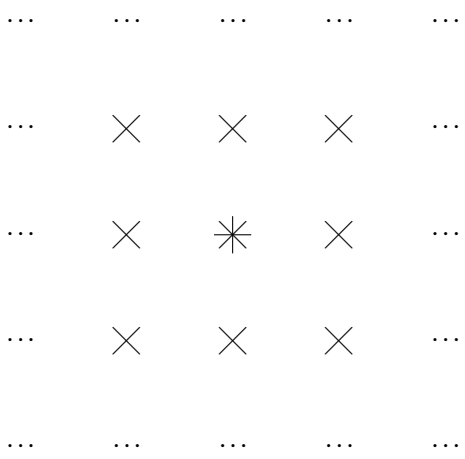

Figure 2. Grid corresponding to dilated convolution; case $k = 2$

For points $(2, 2)$, $(2, 4)$ – and, in general, for each point (x, y) in which both x and y are even – we consider data corresponding to the neighboring-to-(x, y) points from the same subset

$$\{(2n_x, 2n_y) : n_x, n_y \in \mathbb{Z}\}$$

of the original grid.

However, to describe the new value(s) corresponding to the point $(0,1)$, we cannot use the same subset: indeed, to compute the new value(s) corresponding to the point $(0,1)$, we also need to take into account the data corresponding to the original spatial point $(0,1)$ itself, and this point does not belong to the above sub-grid. In the dilated convolution, to describe the new value(s) corresponding to the point $(0,1)$, we combine data corresponding to neighboring points from a *different* subset

$$\{(2n_x, 2n_y+1) : n_x, n_y \in \mathbb{Z}\}.$$

Other grid points correspond to yet different subsets: for example, the point $(1,0)$ corresponds to the subset

$$\{(2n_x+1, 2n_y) : n_x, n_y \in \mathbb{Z}\},$$

and the point $(1,1)$ corresponds to the subset

$$\{(2n_x+1, 2n_y+1) : n_x, n_y \in \mathbb{Z}\}.$$

Overall, while the original convolutional neural network can be described by a single grid (in which we select neighboring points), the $k=2$ dilated network needs all four above subsets of this grid – so that we process data corresponding to the neighboring points from the corresponding sub-grid.

Similarly, for a dilated grid with $k=3$, we need 9 sub-grids; for a dilated grid with $k=4$, we need 16 sub-grids, etc.

General case: a geometric description. As we have mentioned earlier, in principle, we do not need to a priori restrict ourselves to sub-grids. We can, instead, consider neighboring points from different subsets of the original grid $\mathbb{Z}\times\mathbb{Z}$.

Let $S_0 \subset \mathbb{Z}\times\mathbb{Z}$ be a set of points that is used to determine the new value(s) at the point $(0,0)$: to be more precise, the new value(s) corresponding to the point $(0,0)$ comes from processing data from all the points from this set S_0 which are close to $(0,0)$. Similarly, for any other point (x,y) from this set S_0, we can compute the new value(s) corresponding to this point by processing data from all the points from this set S_0 which are close to this point (x,y).

However, for a point $(x,y) \in \mathbb{Z}\times\mathbb{Z}$ which does not belong to the set S_0, we cannot use only neighboring points from this set S_0 – since we would like the new value(s) in this point to take into account the original data in this same point. Thus – similar to the dilated case – we need to have *several* such subsets S. In other words, to describe a general case, we need to have a *family* a of subsets of the grid $\mathbb{Z}\times\mathbb{Z}$.

Let us describe reasonable properties that this family must satisfy.

First property: for each family, the union of all its sets must be equal to the original grid. For each point $(x,y) \in \mathbb{Z}\times\mathbb{Z}$, we need to decide which other points to use to compute the new value(s) at this spatial point. So, we need to have a set S from the family a – so that we will use data corresponding to points from this set S which are close to the given point (x,y). As we mentioned earlier, in these calculations, we want to use the data corresponding to this same point (x,y) as well. So, we must have $(x,y) \in S$. In other words, every point from the original grid should be covered by one of the sets from the family a.

Thus, the union of all these sets should indeed coincide with the original grid.

Second property: different sets from a family must be disjoint. For each point (x,y), we want to know for which exactly neighboring points, the corresponding data should be processed to compute the new value(s) corresponding to this point (x,y). So, we cannot have a point (x,y) belong to two different sets $S, S' \in a$ – in this case, we would have two different possible sets of points to use.

Thus, different sets from the family a cannot contain the same point – in other words, these sets must be disjoint.

Third property: each family must contain at least one set containing more than one points. In principle, we can have each set S from the family a consist of a single point. This would mean that to compute the new value(s) corresponding to the spatial point (x, y), we use only the data corresponding to this same point. This is not what convolution is about, convolution is about processing data corresponding to *several* spatial points.

Thus, since we are interested in convolution, we should avoid this trivial case and we should assume that at least one set S from the family a more than one point.

What does "optimal" mean? Out of all possible families a, we want to select an *optimal* one. What does "optimal" mean?

In many cases, "optimal" is easy to describe:

- we have an objective function $f(a)$ that assigns a numerical value to each alternative a – e.g., the average approximation error of the numerical method a for solving a system of differential equations, and
- optimal means we select an alternative for which the value of this objective function is the smallest possible (or, for some objective functions, the largest possible).

However, this is not the only possible way to describe optimality.

For example, if we are minimizing the average approximation error, and there are several different numerical methods with the exact same smallest value of average approximation error, then we can use this non-uniqueness to select, e.g., the method with the shortest average computation time. In this case, we have, in effect, a more complex preference relation between alternatives than in the case when decision is made based solely on the value of the objective function. Specifically, in this case, an alternative b is better than the alternative a – we will denote it by $a < b$ – if:

- either we have $f(b) < f(a)$,
- or we have $f(a) = f(b)$ and $g(b) < g(a)$.

If this still leaves several alternatives which are equally good, then we can optimize something else and thus, have an even more complex optimality criterion.

In general, having an optimality criterion means that we are able to compare pairs of alternatives – at least some such pairs – and conclude that:

- for some of these pairs, we have $a < b$,
- for some of these pairs, we have $b < a$, and
- for some others pairs, we conclude that alternatives a and b are, from our viewpoint, of equal value; we will denote this by $a \sim b$.

Of course, these relations must satisfy some reasonable properties. For example, if b is better than a, and c is better than b, then c should be better than a; in mathematical terms, the relation $<$ must be transitive.

What we *must* have is some alternative which is better than or equivalent to all others – otherwise, the optimization problem has no solutions. It also makes sense to require that there is only one such optimal alternative – indeed, as we have mentioned, if there are several equally good optimal alternatives, this means that the original optimality criterion is not final, that we can use this non-uniqueness to optimize something else, i.e., in effect, to modify the original criterion into a final (or at least "more final") one.

Invariance. There is an additional natural requirement for possible optimality criteria, which is related to the fact that the original grid has lots of *symmetries*, i.e., transformations that transform this grid into itself.

For example, if we change the starting point of the coordinate system to a new point (x_0, y_0), then a point that originally had coordinates (x, y) now has coordinates $(x - x_0, y - y_0)$. It makes sense to require that the relative quality of two different families a and b will not change if we simply change the starting point.

Similarly, we can change the direction of the x-axis, then a point (x, y) becomes $(-x, y)$. If we change the direction of the y-axis, we get a transformation $(x, y) \to (x, -y)$. Finally, we can rename the coordinates: what was x will become y and vice versa; this corresponds to the transformation $(x, y) \to (y, x)$. Such transformations should also not affect the relative quality of different families.

Now, we are ready for the precise formulation of the problem.

3. Definitions and the Main Result

Definition.

- *By an* alternative*, we mean a family of non-empty subsets of the grid $\mathbb{Z} \times \mathbb{Z}$, a family in which:*
  - *all sets from this family are disjoint, and*
  - *at least one set from this family has more than one element.*
- *By an* optimality criterion*, we mean a pair of relations $(<, \sim)$ on the set of all possible alternatives that satisfy the following conditions:*
  - *if $a < b$ and $b < c$, then $a < c$;*
  - *if $a < b$ and $b \sim c$, then $a < c$;*
  - *if $a \sim b$ and $b < c$, then $a < c$;*
  - *if $a \sim b$ and $b \sim c$, then $a \sim c$;*
  - *we have $a \sim a$ for all a; and*
  - *if $a < b$, then we cannot have $a \sim b$.*
- *We say that an alternative a is* optimal *with respect to the optimality criterion $(<, \sim)$ if for every other alternative b, we have $b < a$ or $b \sim a$.*
- *We say that the optimality criterion is* final *if there exists exactly one alternative which is optimal with respect to this criterion.*
- *By a* transformation *T, we mean one of the following transformations: $T_{x_0,y_0}(x, y) = (x - x_0, y - y_0)$, $T_x(x, y) = (-x, y)$, $T_y(x, y) = (x, -y)$, and $T_{x,y}(x, y) = (y, x)$.*
- *For each alternative a and for each transformation T, by the* result *$T(a)$ of applying the transformation T to the family a, we mean the family $T(a) = \{T(S) : S \in a\}$, where, for any set S, $T(S) \stackrel{\text{def}}{=} \{T(x, y) : (x, y) \in S\}$.*
- *We say that the optimality criterion is* invariant *if for all transformations T, $a < b$ implies that $T(a) < T(b)$, and $a \sim b$ implies that $T(a) \sim T(b)$.*

Comments.

- In our definition of an alternative, we did not require that for each family, the union of all its sets coincide with the grid $\mathbb{Z} \times \mathbb{Z}$. We omitted this requirement to make our result stronger – since, as we see from the following Proposition, this requirement actually follows from all the other requirements.
- In mathematical terms, the pair of relations $(<, \sim)$ between families of subsets forms what is called a *preorder* or *quasiorder*. This notion is more general than partial order, since, in contrast to the definition of the partial order, we do not require that if $a \le b$ and $b \le a$, then $a = b$: in principle, we can have $a \sim b$ for some $a \ne b$.

Proposition. *For every final invariant optimality criterion, the optimal alternative is equal, for some integer $k \ge 1$, to one of the following families:*

- *the family of all the sets $G_{k,x_0,y_0} \stackrel{\text{def}}{=} \{(x_0 + k \cdot n_x, y_0 + k \cdot n_y) : n_x, n_y \in \mathbb{Z}\}$ corresponding to all possible pairs of integers (x_0, y_0) for which $0 \le x_0, y_0 < k$;*
- *the family of all the sets*

$$N_{k,x_0,y_0} \stackrel{\text{def}}{=} \{(x_0 + k \cdot n_x, y_0 + k \cdot n_y) : n_x, n_y \in \mathbb{Z} \text{ and } n_x + n_y \text{ is even}\}$$

corresponding to all possible pairs of integers (x_0, y_0) for which $0 \le x_0, y_0 < k$.

Comments.

- This proposition takes care of all invariant (and final) optimality criteria. Thus, it should work for all usual criteria based on misclassification rate, time of calculation, used memory, or any other used in neural networks: indeed, if one method is better than another for images in general, it should remain to be better if we simply shift all the images or turn all the images upside down. Images can come as they are, they can come upside down, they can come shifted, etc. If for some averaging criterion, one method works better for all possible images but another method works better for all upside-down versions of these images – which is, in effect, the same class of possible images – then from the common sense viewpoint, this would mean that something is not right with this criterion.
- In the first case, we have a sub-grid – as was described in Section 1. In the second case, we have a different grid-type set; see Fig. 3.

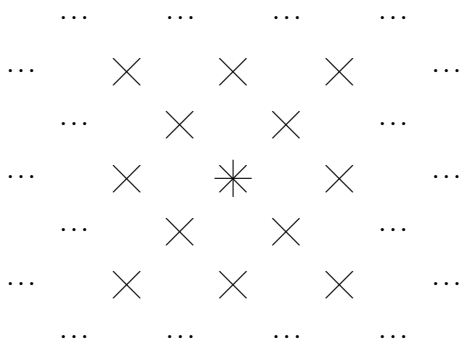

Figure 3. A grid-type set described in the Proposition

Thus, this result explains the effectiveness of sub-grids – and also provides us with a new alternative worth trying.

- The following proof is similar to several proofs presented in [4].

Proof.

1°. Since the optimality criterion is final, there exists exactly one optimal family a_{opt}. Let us first prove that this family is itself invariant, i.e., that $T(a_{\text{opt}}) = a_{\text{opt}}$ for all transformations T.

Indeed, the fact that the family a_{opt} is optimal means that for every family a, we have $a < a_{\text{opt}}$ or $a \sim a_{\text{opt}}$. Since this is true for every family a, it is also true for every family $T^{-1}(a)$, where T^{-1} denotes inverse transformation (i.e., a transformation for which $T(T^{-1}(x, y)) = (x, y)$). Thus, for every family a, we have either $T^{-1}(a) < a_{\text{opt}}$ or $T^{-1}(a) \sim a_{\text{opt}}$. Due to invariance, we have $a = T(T^{-1}(a)) < T(a_{\text{opt}})$ or $a \sim T(a_{\text{opt}})$. By definition of optimality, this means that the alternative $T(a_{\text{opt}})$ is also optimal. However, since the optimality criterion is final, there exists exactly one optimal family, so $T(a_{\text{opt}}) = a_{\text{opt}}$.

The statement is proven.

2°. Let us now prove that the optimal family contains a set S_0 that, it its turn, contains the point $(0, 0)$ and some point $(x, y) \neq (0, 0)$.

Indeed, by definition of an alternative, every family – including the optimal family – contains at least one set S that has at least two points. Let S be any such set from the optimal family, and let (x_0, y_0) be any of its points. Then, due to Part 1 of this proof, the set $S_0 \stackrel{\text{def}}{=} T_{x_0,y_0}(S)$ also belongs to the optimal family, and this set contains the point

$$T_{x_0,y_0}(x_0, y_0) = (x_0 - x_0, y_0 - y_0) = (0, 0).$$

Since the set S had at least two different points, the set $S_0 = T_{x_0,y_0}(S)$ also contains at least two different points. Thus, the set S_0 must contain a point (x, y) which is different from $(0, 0)$.

The statement is proven.

3°. In the following text, by S_0, we will mean a set from the optimal family a_{opt} whose existence is proven in Part 2 of this proof: namely, a set that contains the point $(0, 0)$ and a point $(x, y) \neq (0, 0)$.

4°. Let us prove that if the set S_0 contains a point (x, y), then this set also contains the points $(x, -y)$, $(-x, y)$, and (y, x).

Indeed, due to Part 1 of this proof, with the set S_0 the optimal family a_{opt} also contains the set $T_y(S_0)$. This set contains the point $T_y(0, 0) = (0, 0)$. Thus, the sets S_0 and $T_y(S_0)$ have a common element $(0, 0)$. Since different sets from the optimal family must be disjoint, it follows that the sets S_0 and $T_y(S_0)$ must coincide. The set $T_y(S_0)$ contains the points $(x, -y)$ for each point $(x, y) \in S$. Since $T_y(S_0) = S_0$, this implies that for each point $(x, y) \in S_0$, we have $(x, -y) \in T_y(S_0) = S_0$.

Similarly, we can prove that $(-x, y) \in S_0$ and $(y, x) \in S_0$. The statement is proven.

5°. By combining the two conclusions of Part 4 – that $(x, -y) \in S_0$ and that therefore $T_x(x, -y) = (-x, -y) \in S_0$, we conclude that for every point $(x, y) \in S_0$, the point

$$-(x, y) \stackrel{\text{def}}{=} (-x, -y)$$

is also contained in the set S_0.

6°. Let us prove that if the set S_0 contains two points (x_1, y_1) and (x_2, y_2), then it also contains the point

$$(x_1, y_1) + (x_2, y_2) \stackrel{\text{def}}{=} (x_1 + x_2, y_1 + y_2).$$

Indeed, due to Part 1 of this proof, the set $T_{-x_2,-y_2}(S_0)$ also belongs to the optimal family. This set shares an element

$$T_{-x_2,-y_2}(0, 0) = (0 - (-x_2), 0 - (-y_2)) = (x_2, y_2)$$

with the original set S_0. Thus, the set $T_{-x_2,-y_2}(S_0)$ must coincide with the set S_0. Due to the fact that $(x_1, y_1) \in S_0$, the element

$$T_{-x_2,-y_2}(x_1, y_1) = (x_1 - (-x_2), y_1 - (-y_2)) = (x_1 + x_2, y_1 + y_2)$$

belongs to the set $T_{x_1,y_1}(S_0) = S_0$. The statement is proven.

7°. Let us prove that if the set S_0 contains a point (x, y), then, for each integer c, this set also contains the point

$$c \cdot (x, y) = (c \cdot x, c \cdot y).$$

Indeed, if c is positive, this follows from the fact that

$$(c \cdot x, c \cdot y) = (x, y) + \ldots + (x, y) \ (c \text{ times}).$$

When c is negative, then we first use Part 5 and conclude that $(-x, -y) \in S_0$, and then conclude that the point $(|c| \cdot (-x), |c| \cdot (-y)) = (c \cdot x, c \cdot y)$ is in the set S_0.

8°. Let us prove that if the set S_0 contains points $(x_1, y_1), \ldots, (x_n, y_n)$, then for all integers $c_1, \ldots, c_n$, it also contains their linear combination

$$c_1 \cdot (x_1, y_1) + \ldots + c_n \cdot (x_n, y_n) = (c_1 \cdot x_1 + \ldots + c_n \cdot x_n, c_1 \cdot y_1 + \ldots + c_n \cdot y_n).$$

Indeed, this follows from Parts 6 and 7.

9°. The set S_0 contains some points which are different from $(0,0)$, i.e., points for which at least one of the integer coordinates is non-zero. According to Parts 4 and 5, we can change the signs of both x and y coordinates and still get points from S_0. Thus, we can always consider points with non-negative coordinates.

Let d denote the greatest common divisor of all positive values of the coordinates of points from S_0.

If a value x appears as an x-coordinate of some point $(x,y) \in S_0$, then, due to Part 4, we have $(x,-y) \in S_0$ and thus, due to Part 5,

$$(x,y) + (x,-y) = (2x,0) \in S_0.$$

Similarly, if a value y appears as a y-coordinate of some point $(x,y) \in S_0$, then we get $(0,2y) \in S_0$ and thus, due to Part 3, $(2y,0) \in S_0$.

It is a known that a common divisor d of the values $v_1, \ldots, v_n$ can be represented as a linear combination of these values:

$$d = c_1 \cdot v_1 + \ldots + c_n \cdot v_n.$$

For each value v_i, we have $(2v_i, 0) \in S_0$, thus, for

$$2d = c_1 \cdot (2v_1) + \ldots + c_n \cdot (2v_n),$$

by Part 8, we get $(2d,0) \in S_0$. Due to Part 4, we thus have $(0,2d) \in S_0$, and due to Parts 6 and 7, all points $(n_x \cdot (2d), n_y \cdot (2d))$ for integers n_x and n_y also belong to the set S_0.

If S_0 has no other points, then for sets containing $(0,0)$, we indeed conclude that these sets form a desired sub-grid, with $k = 2d$.

10°. What if these are other points in the set S_0? Since d is the greatest common divisor of all the coordinate values, each of these points has the form $(c_x \cdot d, c_y \cdot d)$ for some integers c_x and c_y. Since this point is not of the form $(n_x \cdot (2d), n_y \cdot (2d))$, this means that either c_x, or c_y is an odd number – or both.

Let us first consider the case when exactly one of the values c_x and c_y is odd. Without losing generality, let us assume that c_x is odd, so $c_x = 2n_x + 1$ and $c_y = 2n_y$ for some integers n_x and n_y. Due to Part 9, we have $(2n_x \cdot d, 2n_y \cdot d) \in S_0$, so the difference

$$((2n_x+1) \cdot d, 2n_y \cdot d) - (2n_x \cdot d, 2n_y \cdot d) = (d,0)$$

also belongs to the set S_0. Thus, similarly to Part 9, we can conclude that for every two integers c_x and c_y, we have $(c_x \cdot d, c_y \cdot d) \in S_0$. So, in this case, S_0 coincides with the sub-grid for which $k = d$.

The only remaining case is when not all points $(c_x \cdot d, c_y \cdot d)$ belong to the set S_0. This means that for some such point both values c_x and c_y are odd: $c_x = 2n_x + 1$ and $c_y = 2n_y + 1$ for some integers n_x and n_y. Due to Part 9, we have $(2n_x \cdot d, 2n_y \cdot d) \in S_0$, so the difference

$$((2n_x+1) \cdot d, (2n_y+1) \cdot d) - (2n_x \cdot d, 2n_y \cdot d) = (d,d)$$

also belongs to the set S_0.

Since, due to Part 9, we have $(2n_x \cdot d, 2n_y \cdot d) \in S_0$ for all n_x and n_y, we conclude, by using Part 5, that $((2n_x+1) \cdot d, (2n_y+1) \cdot d) \in S_0$. So, all pairs for which both coordinates are odd multiples of d are in S_0. Thus, we get the new case described in the Proposition.

11°. The previous results were about the sets containing the point $(0,0)$.

For all other sets S containing some other point (x_0, y_0), we get the same result about the sub-grid if we take into account that the optimal family is invariant, and thus,

with the set S, the optimal family also contains the set $T_{x_0,y_0}(S)$ that contains $(0,0)$ and is, thus, equal either to the desired sub-grid or to the new sub-grid-type set.

The proposition is proven.

4. Conclusions and Future Work

Conclusions. One of the efficient machine learning ideas is the idea of a convolutional neural network, when to each point in the orgoal image, we assign a combination of data corresponding to several neighboring points. To speed up computations, a reasonable idea is to use only a subset of the set of all of neighboring points. It turns out that out of all such subsets, the best results are obtained when we only use data corresponding to neighboring points from a sub-grid of the original grid of pixels.

In this paper, we provide a theoretical explanation for this empirical conclusion.

Future work. This paper describes a general abstract result: that for any optimality criterion that satisfies some reasonable properties, *some* sub-grid is optimal. To be practically useful, it is desirable to analyze which sub-grid is optimal for different practical situations and for specific criteria uses in machine learning, such as misclassification rate, time of calculation, used memory, etc. (and the combination of these criteria). It is also desirable to analyze what size neighborhood should we choose for different practical situations and for different criteria.

Author Contributions: All three authors contributed equally to this paper. All three authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Science Foundation grants 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science), and HRD-1834620 and HRD-2034030 (CAHSI Includes). It was also supported by the program of the development of the Scientific-Educational Mathematical Center of Volga Federal District No. 075-02-2020-1478.

Acknowledgments: The authors are greatly thankful to the anonymous referees for valuable suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Leaning*, MIT Press: Cambridge, Massachusetts, 2016.
2. Kreinovich, V.; Kosheleva, O. Optimization under uncertainty explains empirical success of deep learning heuristics", In: Pardalos, P.; Rasskazova, V.; Vrahatis, M.N. (eds.), *Black Box Optimization, Machine Learning and No-Free Lunch Theorems*, Springer: Cham, Switzerland, 2021, pp. 195–220.
3. Li, Y.; Zhang, X.; Chen, D. CSRNet: dilated convolutional neural networks for understanding the highly congested scenes, *Proceedings of the 2018 Conference on Computer Vision and Pattern Recognition CVPR'2018*, Salt Lake City, Utah, June 18–22, 2018, pp. 1091–1100.
4. Nguyen, H.T.; Kreinovich, V. *Applications of Continuous Mathematics to Computer Science*, Kluwer: Dordrecht, 1997.
5. Yu, F.; Koltun, V. Multi-scale context aggregation by dilated convolutions, *Proceedings of the 4th International Conference on Learning Representations ICLR'2016*, San Juan, Puerto Rico, May 2–4, 2016.
6. Zhang, X.; Zou, Y.; Shi, W. Dilated convolution neural network with LeakyReLU for environmental sound classification, *Proceedings of the 2017 22nd International Conference on Digital Signal Processing DSP'2017*, London, U.K., August 23–25, 2017.