entropy

MDPI

Article

A New Pooling Approach Based on Zeckendorf's Theorem for Texture Transfer Information

Vincent Vigneron [1,2,*], Hichem Maaref [1] and Tahir Q. Syed [3]

[1] Computer Science Department, Univ Evry, Université Paris-Saclay, 91190 Saint-Aubin, France; hichem.maaref@univ-evry.fr
[2] School of Applied Sciences (FCA/UNICAMP), Limeira, Sao Paolo 13484-350, Brazil
[3] Computer Sciences Department, Institute of Business Administration, Karachi, Sindh 75270, Pakistan; tqsyed@iba.edu.pk
* Correspondence: vincent.vigneron@univ-evry.fr

Abstract: The pooling layer is at the heart of every convolutional neural network (CNN) contributing to the invariance of data variation. This paper proposes a pooling method based on Zeckendorf's number series. The maximum pooling layers are replaced with Z pooling layer, which capture texels from input images, convolution layers, etc. It is shown that Z pooling properties are better adapted to segmentation tasks than other pooling functions. The method was evaluated on a traditional image segmentation task and on a dense labeling task carried out with a series of deep learning architectures in which the usual maximum pooling layers were altered to use the proposed pooling mechanism. Not only does it arbitrarily increase the receptive field in a parameterless fashion but it can better tolerate rotations since the pooling layers are independent of the geometric arrangement or sizes of the image regions. Different combinations of pooling operations produce images capable of emphasizing low/high frequencies, extract ultrametric contours, etc.

Keywords: deep learning; pooling function; Zeckendorf theorem; Fibonacci; LBP; image representation; segmentation; glioblastoma

check for updates

Citation: Vigneron, V.; Maaref, H.; Syed, T.Q. A New Pooling Approach Based on Zeckendorf's Theorem for Texture Transfer Information. *Entropy* **2021**, *23*, 279. https://doi.org/10.3390/e23030279

Academic Editor: Jerry D. Gibsone

Received: 12 December 2020
Accepted: 21 February 2021
Published: 25 February 2021

1. Introduction

Deep neural networks (DNN) have revolutionized orthodox tasks of image analysis in which they have accomplished outstanding results and continually do so [1–3]. By employing modifications to the architectures and introducing various techniques (often greedy), considerable improvements have been achieved.

Convolutional Neural Networks (CNNs) architectures are increased through multiresolution (pyramidal) structures, which come from an idea that the network needs to see different levels of detail to produce good results. A CNN stacks four different processing layers: convolution, pooling, and fully connected (dense) layers [4].

The pooling layer receives multiple feature maps from convolutional layers and applies the pooling function to each of them. The pooling layer (a) reduces the number of parameters in the model (subsampling) and calculations in the network while preserving their important characteristics, (b) improves the efficiency of the network and prevents overlearning [4]. To do this, the maximum pooling function downsamples the input representation by reducing its dimensionality: the image is split into regular cells without overlapping, then the maximum value is kept within each cell. Thus, the pooling layer makes the network less sensitive to the position of features: the fact that a feature value is a little higher or lower, or even that it has a slightly different orientation *should not* lead to a drastic change in the image classification.

The weaknesses of pooling functions are well identified [5]: (a) they do not preserve all the spatial information well by reducing the spatial resolution, (b) the discrete maximum chosen by the maximum pooling in the pixel grid may be not the true maximum,

(c) average pooling assumes a single mode with a single centroid. Hence, the question is how to take into account in an optimal way the characteristics of the input image being grouped in the pooling operation [6]? Part of the response sets in Lazebnik's work, which demonstrated the importance of the spatial structure of pooling neighborhoods [7]. These local spatial variations of image pixel intensities (named textures in popular image processing) characterize an "organized area phenomenon" [8], which cannot be captured in pooling layers.

This paper proposes a new pooling operation, independent of the geometric arrangement or sizes of image regions, which can therefore better tolerate rotations. The operation is based on the Zeckendorf theorem for the decomposition of integers, which is also simple to implement. Zeckendorf theorem is mainly used in cryptography [9], e.g. to design small microcontrollers that can resist certain Fault Attack.

The rest of article is organized as follows: Section 2 presents the related works on pooling strategies. Zeckendorf additive partition is exposed in Section 3 and its implementation is explained in Section 4. Numerical experimentations and results are presented in Section 5. Finally, experimental works are discussed and future works are mentioned in Section 6.

2. Related Works

Throughout this paper small Latin letters $a, b, \ldots$ represent integers. small bold letters $\mathbf{a}, \mathbf{b}$ are put for vectors and capital letters A, B for matrices or tensor depending of the context. '$\{\ldots\}$' brackets indicate set of values. $|\cdot|$ is put for the cardinal operator.

2.1. *Pooling Strategies in Image Processing*

Convolutions in CNNs are discrete convolutions of an image V with a kernel K. Without loss of generality an input image V in a high dimensional space can be reduced into a vector $\mathbf{v}$. Let's define $N(i)$ as the set of all indices of elements in $\mathbf{v}$ which are neighbors of v_i in the neighborhood defined by the convolution kernel K

$$N(i) = \{j \in \mathbb{N} | v_j \in \text{ neighborhood of } v_i \text{ given by } K\} \tag{1}$$

As the structure of the neighborhood is fixed, we assume that $N(i, j) \in \{1, 2, \ldots, |N(i)|\}$, which is the index of j in $N(i, j)$. The discrete convolution can then be defined as

$$c(\mathbf{k}, \mathbf{v})_i = \sum_{j \in N(i)} k_{N(i,j)} v_j. \tag{2}$$

where $\mathbf{k}$ are the the weights of the convolution kernel K.

The exponential growth of the number of parameters makes convolutions with large kernel sizes computationally expensive. Therefore, most CNN architectures keep the kernel size at 3×3 or 5×5. However, how does one do a sensitive prediction for an entire image, if a single convolution "sees" only a 3×3 neighborhood? The solution is the stacking of convolutional layers. With two layers following each other, the last one can "see" a 4 × 4 neighborhood. This means a lot of convolutions must be stacked to have a receptive field as large as a reasonable input image. The increase in receptive field by convolution can be considerably higher when the image is downsampled to a lower resolution between two convolution operations. Various methods exist for resampling a given feature layer at multiple rates prior to convolution such as dilated convolution that "inflate" the kernel by inserting holes between the kernel elements [10] or astrous convolution [11].

Maximum pooling is a popular choice for this downsampling operation. The pooling operations have been little revised beyond the main current maximum, average, and stochastic pooling options despite indications that choosing multiple pooling functions can improve performance [12].

Sharma et al. analyzed and discussed qualitatively the performances of pooling strategies on different datasets [13]. Lee et al. [6] experimentally demonstrate that their pooling operations combining maximum and average pooling provide an increase in invariance properties over the conventional pooling. Lee et al. proposed to combine pooling filters that are themselves learned. In [14], Gulcehre et al. investigate a novel nonlinear unit, called L_p unit that generalizes a number of conventional pooling operators such as mean, root mean square, and maximum pooling.

Agostinelli learned activation functions to improve DNN in [15]. Boureau et al. analyze theoretically why max pooling works well in a wide variety of contexts, even if similar or different factors come into play in each case [16].

Many researchers are working on the development of advanced pooling mechanisms to effectively use these essential features of pooling [13], in particular on how to bring learning to the characteristics of the region being pooled into the pooling operation [6]?

2.2. Pooling and Statistics

In Statistics, "pooling" describes the practice of bringing together small datasets that are assumed to have the same value of a characteristic, e.g., a mean, and using the larger combined set (the "pool") to get a more precise estimate of this feature. *Poolability* can be formulated on the basis of the concept of statistical equivalence. Sheskin compiled in [17] a bibliography dealing with pooling procedures, for example to combine several independent tests of the same hypothesis.

The goal of pooling is to transform the convolutional characteristics into a new representation that preserves important information while ignoring irrelevant details. For instance, if a t-test between the two within-group slopes is not "passed", these characteristics cannot be grouped [18].

In some way, many other ensemble techniques, where a set of weak learners are combined to create a stronger learner, are very near to this notion of pooling [19].

So, should we pool or not? Or, putting it a little differently, when should we pool and when should we not? The answer depends on the training context. Moorthy et al. in [20] proposed to weight the image quality measures by visual importance to improve the correlations with subjective judgment. Achieving invariance to changes in position or lighting conditions, robustness to size, and compactness of representation are all common goals of pooling. We demonstrate experimentally here that these properties are achieved successively with the Z pooling operator, based on Zeckendorf number theorem.

Experimental validation is continued in Section 5 on predefined architectures and obtained by replacing the standard pooling operations with Z pooling.

2.3. Texture Coding

Most of image descriptors that encode local structures e.g., local binary patterns (LBP) (and its variants) [21,22] depend on (a) the size of the neighborhood, (b) the reading order of the neighbors, (c) the mathematical function that is used to compute the feature distance between neighboring pixels. The new pixel value $L_R(P)$ in the image is an integer in the range of 0 to 255 (for a 8-bit encoding) given by:

$$L_R(P) = \sum_{p=0}^{P-1} 2^p \cdot t(g_p - g_c), \text{ with } t(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

where P is the number of pixels in the neighborhood considering the distance R between central pixel g_c and the neighboring pixels $\{g_p | p = 0, \ldots, P-1\}$. In Equation (3), LBP computes a pixel value from a $8-bit$ string from the 3×3 neighborhood by computing the Heaviside function $t(\cdot)$ of the difference between neighboring pixels and the central pixel, $(g_i - g_c)$ (Figure 1).

image example

121	201	200
190	100	164
78	77	65

$\rightarrow$

thresholded

1	1	1
1	🟦	1
0	0	0

$\rightarrow$

LBP weights

1	2	4
128	🟦	8
64	32	16

Pattern	$(10001111)_2$
LBP	128 + 8 + 4 + 2 + 1 = 143

Figure 1. Example of 3×3 image neighborhood ($P = 8$ and $R = 1$).

LBP-like texture descriptors have evolved into almost all fields of computer vision, because of their robustness to monotonic gray-scale changes, illumination invariance, and computational simplicity. Invariance w.r.t. any monotonic transformation of the gray scale is achieved by considering in (Equation (3)) the signs of the differences $t(g_i - g_c), i = 0, \ldots, P-1$. The local texture can be represented as a joint distribution of the values of the differences at the center pixel g_c. Assuming the *independence* of g_c with respect to the differences $(g_i - g_c), i = 0, \ldots, P-1$. However, under certain circumstances such as very low or high values of g_c, the range of possible differences and so, LBP can miss the local structure as it does not consider the central pixel. To reduce the noise sensitivity, mostly in uniform regions, a three-level operator has been proposed by Tan and Trigg [23], which describes a pixel relationship with its neighbors by a ternary encoding, i.e., $-1, 0, 1$ rather than a binary code, i.e., $0, 1$. The size of this code is reduced by splitting it into two LBP (Positive and Negative) codes, which results into two 8-bit strings thus needing a 16 bit space for representation.

In the next section, an algorithm is proposed for generating Z images, which could be utilized in contour detection or image segmentation.

3. Z Representation

3.1. Zeckendorf Additive Partition

In this section, an algorithm is proposed for so called Z pooling. In [24] the Belgian mathematician Édouard Zeckendorf states that any integer N may be *uniquely* represented as the sum of distinct Fibonacci numbers so that the sum does not include any *two non-consecutive* Fibonacci numbers. The Fibonacci series $1, 1, 2, 3, 5, 8, \ldots$ is a sequence of numbers $f(n)$ such that $f(n)$ is the sum of the 2 previous values with initial conditions $f(0) = f(1) = 1$:

$$f(n) = f(n-1) + f(n-2) \quad \text{for} \quad n \geq 0. \tag{4}$$

Here we have a second-order linear constant coefficient difference equation that we want to solve. Specifically, consider the following by rewriting it in a slightly different form:

$$f(n) - f(n-1) - f(n-2) = \delta(n-1) \tag{5}$$

The solution to the Equation (4) may be found using z-transforms as follows: $F(z) - z^{-1}F(z) - z^{-2}F(z) = z^{-1}$. Solving for $F(z)$ we have $F(z) = \dfrac{z^{-1}}{1 - z^{-1} - z^{-2}}$.

Theorem 1 (Zeckendorf's additive theory). *Any positive integer N can be expressed as a sum of distinct Fibonacci numbers* $(f(1), f(2), f(3), \ldots, f(m)$ *with appropriate coefficients* $\sigma_i \in \{0, 1\}$ *such as*

$$N = \sum_{i=0}^{m} \sigma_i f(i). \tag{6}$$

such that $\sigma_i \sigma_{i+1} = 0, \; i = 1, 2, \ldots$.

Proof. For any positive integer N, there is always a positive integer m such that $f(m) \leq n \leq f(m+1)$. If $n \neq f(m)$,

$$0 < N - f(m) < f(m+1) - f(m) = f(m-1). \tag{7}$$

Since $N - f(m)$ is positive, there exists a positive integer p such that

$$f(p) \leq N - f(m) < f(p+1). \tag{8}$$

Now $f(p) \leq N - f(m) < f(m-1)$ implies $p \leq m-2$, i.e., $f(p)$ and $f(m)$ are not consecutive Fibonacci numbers. If $N - f(m) \neq f(p)$, there exists a positive integer $q \leq p - 2$ such that

$$f(q) \leq N - f(m) - f(p) < f(q+1) \tag{9}$$

and the process continues. Ultimately, we must reach the point where the partial sum equals a Fibonacci number—say $f(t)$—and thereby obtain the desired representation

$$N = f(m) + f(p) + f(q) + \ldots + f(t). \tag{10}$$

□

Zeckendorf partition is *complete* and *canonical*, i.e., every positive integer is the sum of distinct elements of Fibonacci series and, in the binary base, the sequence $\sigma_k, \sigma_{k-1}, \ldots \sigma_3, \sigma_2$ with $\sigma_i \in \{0,1\}$ in Equation (6) contains the smallest number of 1. The number of Fibonacci sequences of length $k-1$ is exactly $f(k+1)$.

An 8-bit gray scale image has the intensity values in the range $[0,255]$. The first Fibonacci numbers below 255 is the discrete set $\mathcal{F} = \{1,2,3,5,8,13,21,34,55,89,144,233\}$ of cardinality $|\mathcal{F}| = 12$. So the Fibonacci sequence can be used for 12-bit image coding and each pixel intensity of an image can be encoded as a sum of distinct consecutive or non-consecutive Fibonacci numbers. For instance the pixel value 255 can be represented by the sequences $(1,21,233)_{\text{fi}}$ $(1,8,13,233)_{\text{fi}}$ $(1,21,89,144)_{\text{fi}}$ $(1,3,5,13,233)_{\text{fi}}$ $(1,8,13,89,144)_{\text{fi}}$ $(1,21,34,55,144)_{\text{fi}}$ $(1,3,5,13,89,144)_{\text{fi}}$ $(1,8,13,34,55,144)_{\text{fi}}$ $(1,3,5,13,34,55,144)_{\text{fi}}$ but Zeckendorf decomposition $(233,21,1)_{\text{Zck}}$ is unique. bit patterns.

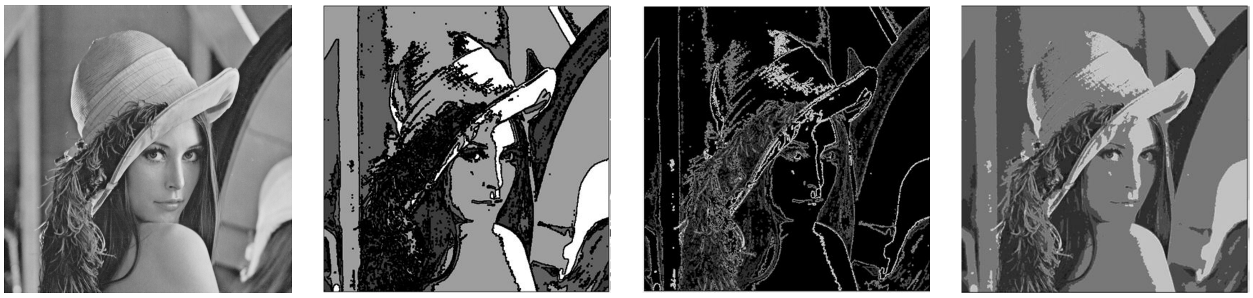
From this additive property of integers, a new image encoding is proposed (see Algorithm 1), which encodes the local dependencies of pixels by combining a pooling operation and an integration operation, both chosen from supremum (max), infimum (min), summation, intersection ($\cap$) or set difference ($\setminus$) [25]. A *texel* is a texture element or texture pixel.

The way these operators are combined results in images that could be directly used in the computer vision pipeline for object segmentation or contour extraction. The result of applying various arithmetic operations after the intersection leads to different types of image variations.

Four of these variations on Lenna's image are shown on Figure 2. Each produces a characteristic inference line, which we explore below.

The first Figure 2a is produced by applying the supremum operation followed by another supremum. The edges are quite smooth and many edges are missed due to the maximum operation. This operation leaves smaller values in the intersection, resulting in fewer or no edges. Figure 2b is constructed by applying the supremum operation followed by an infimum. As expected, the max operator at the initial stage will produce the set of relatively larger values leaving small Fibonacci numbers. A minimum operator at the end slightly overcomes the maximum effect by selecting the minima for the central pixel. Figure 2c could be considered the complete opposite of the second. All the minimum values are first extracted using the infimum operator, then the supremum of the set is taken. It is totally intuitive to think of it as a double of the second image. Figure 2d is produced by applying a summation operator, which is then followed by the minimum operation.

The difference between the fourth and the second images is that the values are out of range for some pixels due to the intensity ranges saturating the summation operator.



(**a**) original Lenna (**b**) $\cap$ with max(min). (**c**) $\cap$ with min(max). (**d**) $\cap$ with max(max).

Figure 2. Z-images resulting from the application of Algorithm 1 on Lenna picture by combining $\cap$ with four arithmetic operations from top left clockwise.

Algorithm 1: Image Z coding.

```
Data: Image $I$ of size $J \times K$
Result: Z-image of size $J \times K$ of input image $I$
Input: Initialization: $Z \leftarrow \emptyset$; $j = 2, k = 2$; $N$: neighborhood size around center pixel $I_0$ of a $w \times w$ sliding
       window $W$, $w \geq 3$ an odd number
for $j = \frac{w+1}{2}$ to $J - \frac{w-1}{2}$ do
    for $k = \frac{w+1}{2}$ to $K - \frac{w-1}{2}$ do
        $I_0 = I(j,k)$; /* central pixel of $W$ */
        $\mathbf{t}(I_0) \leftarrow \mathbf{0}$ /* $N$-vector of neighboring pixel values around $I_0$ */
        $Z^0 \leftarrow$ Zeck $(I_0)$
        for $i = 1$ to $N$ do
            $T(i) \leftarrow$ Zeck$(W(j,k))$; /* $T$ is a stack filled with Fibonacci numbers */
            $stack \leftarrow Z^0 \bigwedge T(i)$ /* $\bigwedge$: pooling operator */
            if ($stack = \emptyset$) then
                $F(i) \leftarrow I_0$ /* $F$: Fibonacci number */
            else
                $F(i) \leftarrow$ Zeck$^{-1}(stack)$/* integer reconstruction from Fibonacci numbers */
        $Z(j,k) \leftarrow \bigvee(F(1), \dots, F(N))$ /* $\bigvee$: integrating operator */
return $Z$
Function nearestSmallerEqFib($n$):
    if ($n = 0$ or $n = 1$) then
        return $n$
    $f_1, f_2, f_3 = 0, 1, 1$ /* Find the greatest Fibonacci number $f$ smaller than $x$ */
    while ($f_3 \leq n$) do
        $f_1 = f_2$; $f_2 = f_3$; $f_3 = f_1 + f_2$;
        return $f_2$;
Function Zeck($x$):
    /* finds representation of $x$ as sum of non-neighbouring Fibonacci numbers. */
    while $x \geq 0$ do
        $f =$ nearestSmallerEqFib($x$)
        $x = x - f$
    return $f$
```

In Algorithm 1, e.g., for $w = 3$, the list of neighbor pixels surrounding I_0 is $W(j,k) = [I(j-1,k-1)\ I(j-1,k)\ I(j-1,k+1)\ I(j,k+1)\ I(j+1,k+1)\ I(j+1,k)\ I(j+1,k-1)\ I(j,k-1)]$.

Example 1 (Z coding). *Consider a pixel I_0 of intensity* 183*, surrounded by the eight neighbor pixels of values* $\mathbf{t} = (210, 106, 231, 233, 79, 142, 209, 188)^T$.

The Zeckendorf decomposition T of the neighbor pixels are respectively $(144, 55, 8, 3)_{Zck}$ $(89, 13, 3, 1)_{Zck}$ $(144, 55, 21, 8, 3)_{Zck}$ $(233)_{Zck}$ $(55, 21, 3)_{Zck}$ $(89, 34, 13, 5, 1)_{Zck}$ $(144, 55, 8, 2)_{Zck}$ $(144, 34, 8, 2)_{Zck}$ *and for the central pixel* $Z^0 = (144, 34, 5)_{Zck}$*. Then* $T(1) = (144, 55, 8, 3)_{Zck}$*,* $T(2) = (89, 13, 3, 1)_{Zck}$*, etc.*

Following Algorithm 1, consider first the pooling operation $\cap$ (line 10) applied to the center pixel and the first neighbor pixel. Then $Z^0 \cap T(1) = (144, 34, 5)_{Zck} \cap (144, 55, 8, 3)_{Zck} = (144)_{Zck}$*. Therefore,* $F(1) = 144$*.*

Similarly for the second pixel $Z^0 \cap T(1) = (144, 34, 5)_{Zck} \cap (89, 13, 3, 1)_{Zck} = \emptyset$*. Therefore,* $F(2) = I_0 = 183$*.*

Finally, for the last pixel $(144, 34, 8, 2)_{Zck}$*,* $Z^0 \cap T(8) = (144, 34, 5)_{Zck} \cap (144, 34, 8, 2)_{Zck} = (144, 34)_{Zck}$*. If one choose suppremum operator, then* $F(8) = \max(144, 44) = 144$*.*

After the stack F is populated with the Fibonacci values, $F = (144, 183, 144, 183, 183, 34, 144, 144)^T$*. the supremum of this set is calculated (line 15) and treated as the Z-code, which replaces the central pixel value* 183 *in this example.*

Figure 3 shows image variations on other types of pictures.

3.2. Evaluation and Result for Segmentation

Local image descriptors perform well on various computer vision tasks such as image retrieval [26], action recognition [27,28], object detection and recognition [29] etc. We discuss the Zeckendorf representation as a local image descriptor for two of these tasks.

Algorithm 1 results in ultrametric contours or segmented images based on the association of the aforementioned operations.

The union operator was not included in this work because computer vision generally derives directly from the ability of image descriptors to be discriminating, and this is achieved by intersection or set difference operators.

Table 1 reports the performances of the top 10 algorithms and the Zeckendorf segmentation on 500 test images of BSD500 [30], combining set difference and max(max) operators. Segmented images obtained after region merging were also compared with the human annotated images using the benchmark code available at Berkeley's website in Table 2 [30]. We evaluated the quality of the extracted boundaries using Precision and Recall measures. Here, the Precision P is the probability that the extracted borderline pixel is a true borderline pixel and Recall (sensitivity) R is the probability that the real borderline pixels are correctly extracted:

$$P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN}, \tag{11}$$

where TP, FP and FN are resp. the true positives, the false positives, and the false negatives. Precision is how sure one is of true positives whilst Recall is how sure one is about not missing any positives. Due to the trade-off between the two mentioned measures, we calculated an F-score from Equation (11) to compare the results obtain after regions are merged:

$$F = \frac{PR}{\alpha P + (1 - \alpha)R} \tag{12}$$

with α an adjustable perimeter, selected here as 0.5 to compare our results with results available from other algorithms. The F-measure of Z coding is 0.6652 with an average Recall of 0.833 (the highest), indicating that the edge pixels are rarely misclassified. This

F-score could be further improved by refining certain factors such as postsegmentation region-fusion procedures.

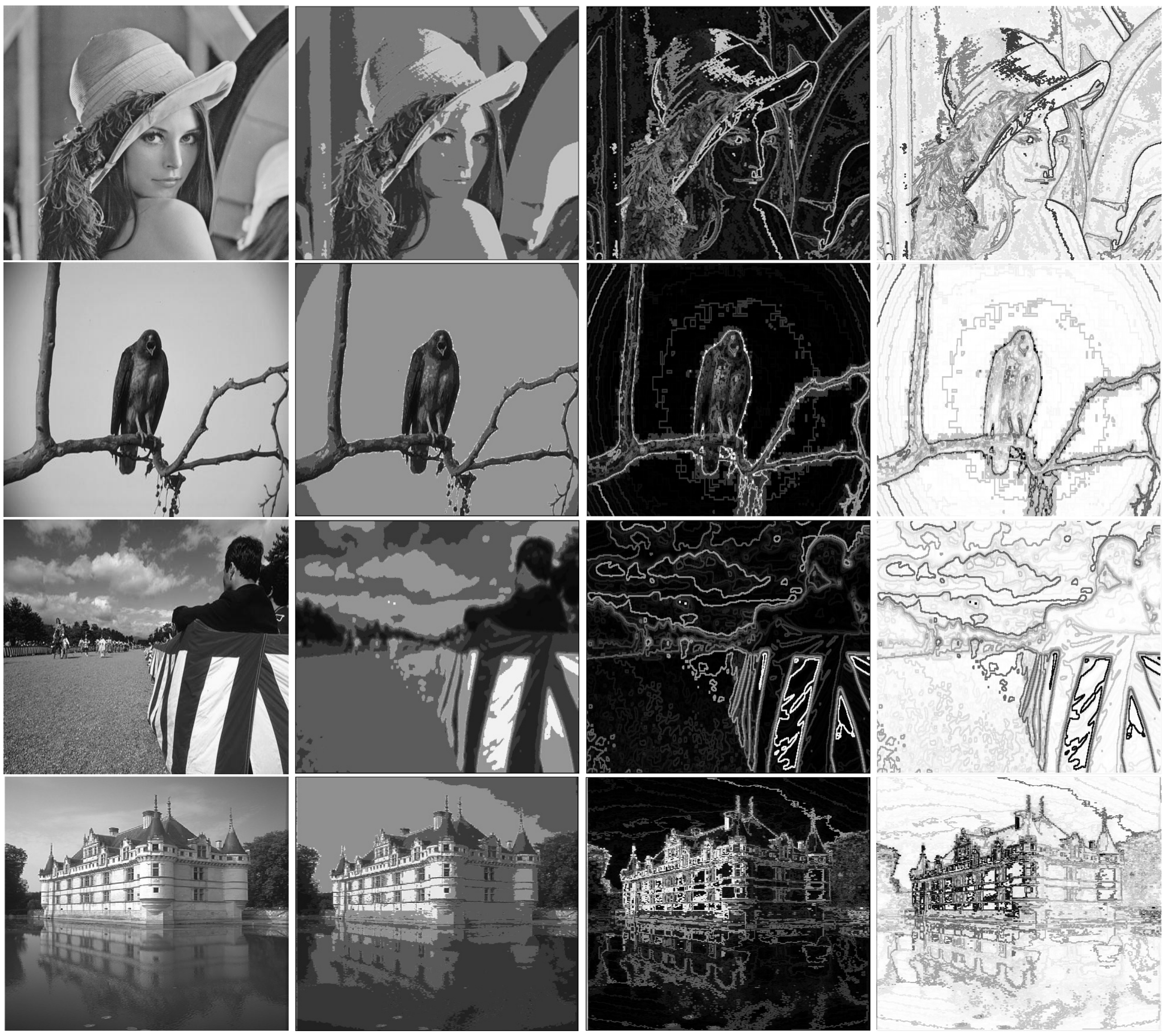

(**a**) (**b**) (**c**) (**d**)

Figure 3. Z-coded images. (**a**) original images. Segmented images obtained by combining intersection with (**b**) max(max) (**c**) min(max) (**d**) max(min).

Figure 4 illustrates the calculus of the performance metric of the segmentation process on the "horse" image from BSD500.

Table 1. Comparison of Z coding with 10 top-ranked segmentation algorithms for the BSD500 benchmark using set difference and max(max) operators. See [31] for a review.

Rank	Algorithm	Average Recall	Average Precision	Average *F*-Measure
1	gPb-UCM	0.7397	0.7241	0.7226
2	Global Probability of Boundary (GPB)	0.7261	0.6902	0.7031
3	Ren	0.7198	0.6959	0.7019
4	**Z-coding**	**0.833**	**0.5875**	**0.6652**
5	Brightness/Texture Gradients (BTG)	0.6999	0.637	0.6592
6	Boosted Edge Learning (BEL)	0.699	0.6254	0.6557
7	Brightness Gradient (BG)	0.6946	0.6011	0.6348
8	Multiscale Gradient Magnitude (MGM)	0.6562	0.5939	0.6133
9	Gradient Magnitude (GM)	0.6961	0.5677	0.6119
10	Texture Gradient (TG)	0.6231	0.6053	0.6076
11	Second order Moment Matrix (SMM)	0.6501	0.5891	0.6042

Table 2. Performance metrics when using Z coding versus human segmented images on BSD500.

Indicators	Average Recall	Average Precision	Average *F*-Measure
Proposed vs. manual benchmark	0.8326	0.2495	0.3675

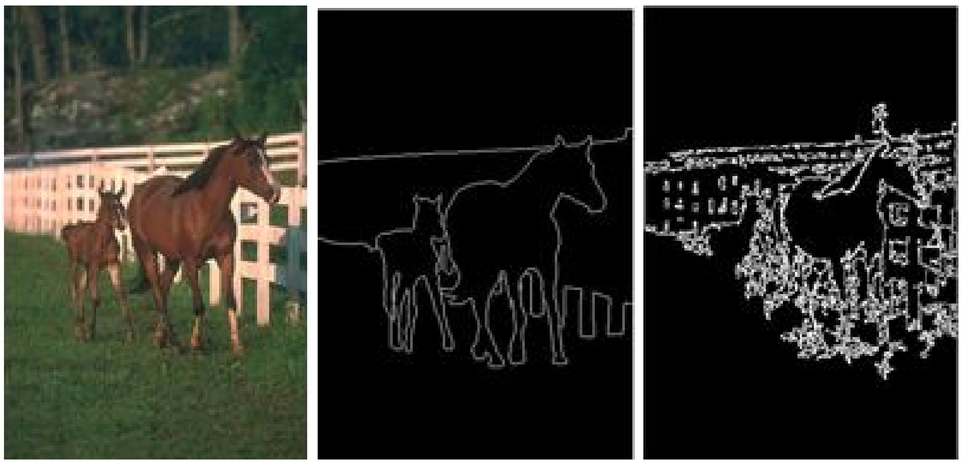

(**a**) original image. (**b**) ground truth. (**c**) z-image.

Figure 4. Z-coded color images using Zeckendorf representation. (**a**) original images (**b**) ground truth image (**c**) binary segmented images obtained with set difference and min(max) operators. Precision: 0.5833. Recall: 0.7871. *F*-Score: 0.6700.

4. Z Pooling

Let h be the input volume (or image) with axis sizes n_k and g the convolution kernel with axis sizes m_k. In CNN the channel or feature axis has a special role. By convention g is the identity along the feature axis and the output of the convolution can have multiple features. The number of input features is n_N, given by h. The number of output features is $n_e tal.pha$ and a parameter of the convolution operation. This is achieved by packing multiple convolutions into one operation, one for each output feature. The set of valid indices for the input volume h is defined as $A = \{(i_1, i_2, \ldots, i_N) | i_k \in [1, \ldots, n_k], k \in [1, \ldots, N]\}$. As there are multiple output features the kernel g gets an additional axis and thus the indices of g are in the set $B = \{(j_1, j_2, \ldots, j_{N+1}) | j_b \in [1 \ldots m_k], b \in [1, \ldots, N+1], m_N = n_N, m_{N+1} = n_e tal.pha\}$. The resulting volume of the convolution operation has the same axis sizes as h except for the feature axis. The convolution with zero padding written in the terms of volumes becomes:

$$a_k = i_k - j_k + \lceil m_k/2 \rceil \quad k \in [1 \ldots N-1] \tag{13}$$

$$a_N = j_N \tag{14}$$

$$h_{a_1 a_2 \ldots a_N} = 0 \quad \forall (a_1 a_2 \ldots a_N) \notin A \tag{15}$$

$$(h * g)_{i_1 i_2 \ldots i_N} = \sum_{j_1 j_2 \cdots j_N} h_{a_1 a_2 \ldots a_N} g_{j_1 j_2 \ldots j_N} \tag{16}$$

with $*$ is the convolution operation and $\lceil \cdot \rceil$ the ceiling function. In many publications the kernel size is split into a image and feature part, i.e., the convolution operation defined by Equation (16) would be described as a convolution with a $m_1 \times m_2 \times \ldots m_{N-1}$ kernel and m_N input features/channels and m_{N+1} output features/channels. The number of input features is determined by the (known) size of the input image h and thus almost always omitted.

Maximum pooling is an important operation for contemporary neural networks defined for a input volume (or image) $h : A \to \mathbb{R}, \quad A = \{(i_1, i_2, \ldots, i_N) | i_k \in [1, \ldots, n_k], k \in$

$[1,\ldots,N]\}$ and a set $B = \{(i_1, i_2, \ldots, i_{N-1}, 0) | i_b \in [-K_b, \ldots, K^b], b \in [1, \ldots, N-1]\}$ called window where either $K_b = K^b$ or $K_b = K^b - 1$ with $K^b \in \mathbb{N}$:

$$\max \text{pool}(h, B)(x) = \max_{y \in B} h(x - y). \tag{17}$$

The x and y are indices for the volume h and B can be seen simply as a selection mask. Note that this operation looks for the maximum in a neighborhood defined by B along the image axis. Unlike the convolution the channels are not mixed in this operation. Often the maximum pooling operation is used for downsampling the volume by restricting x. This restriction is called *striding* with stride $s \in \mathbb{N}$ and A is restricted to $A' = \{(i_1, i_2, \ldots, i_N) | i_k \in [1, 1+s, 1+2s, \ldots, 1+n_s s], n_s = \lceil n_k/s \rceil - 1, k \in [1 \ldots N]\}$. The strided max pooling operation is then:

$$x' = 1 + sx, \quad \max \text{pool}(h, B, s)(x) = \max_{\mathbf{y} \in B} h(x' - y). \tag{18}$$

The strided max pooling reduces the size of the input image by only considering every s-th entry along all image axes and discarding all others. The concept of strides can be used for convolution operations as well and where fractional strides can even be used for upsampling [32].

Z pooling can easily replace maximum pooling in a CNN in Equation (17) when writing

$$\text{Z pooling}(h, B)(x) = \max_{y \in B} f(x \bigwedge \text{Z}\textsc{ck}(y))), \tag{19}$$

where $\bigwedge$ is the intersection or set difference operation. B is the mask (neighborhood) in which x is selected. Note that Z pooling is an operator without parameters as well as max pooling.

With respect to fully connected neural networks, CNN are translation invariant. The translation invariance comes from the fact that the convolution kernel W is the same for every possible position of the input. So once the network learns to recognize an object in one position on the image it automatically will recognize it at any position. However, the use of convolutions comes with a cost: the number of parameters grows with the input and output size. Different pooling operations were carried out in a categorization context to compare the behavior of the different pooling operations.

The most relevant question at this stage is: are pooling layers more efficient when they pool texels or when they pool pixels? Experiments proposed in the following section give some answers.

5. Numerical Evaluation with CNN

5.1. Implementation

The experiments using the aforementioned algorithms were implemented in Python© 3.7 using `Tensorflow` and `Keras` frameworks except for the cascaded network for which the authors provide an implementation based on Niftynet [33]. Computation were completed on a Tesla VT100 CPU @ 3.60 Ghz with 64 GB of RAM. This study focuses on the use of a magnetic resonance imaging (MRI) dataset of acquired brain tumors from the challenge of multimodal brain tumor segmentation Brain Tumor Segmentation (BraTS) challenge [34].

The BraTS publicly available dataset contains preoperative MRI scans for 285 patients. The database is divided into two categories: High-Grade Gliomas (HGG) (210 patients) and Low-Grade Gliomas (LGG) (75 patients). Four MRI modalities of each scan are presented: native (T1), postcontrast T1-weighted (T1Gd), T2-weighted (T2) and T2 Fluid Attenuated Inversion Recovery (FLAIR). The ground truth segmentation is provided (manual segmentation validated by one-to-four experienced neuroradiologists).

5.2. Miccai BraTS Dataset

Segmentation of brain tumors from multiple modalities can produce a prediction that facilitates surgical planning, postoperative analysis and radiotherapy [35].

Brain tumors require early detection and sometimes prolonged treatment. They can be benign or malignant when they have a faster growth rate, although benign tumors are slower in growth and include low-grade variants (1–4). Lower grade glioma (LGG) have a higher life expectancy and do not require immediate treatment. Both cases still require neuroimaging prior, during and after treatment. Medical imaging helps to assess tumor progression, surgical planning, and overall treatment [34]. Glioblastoma (GBM) is a very aggressive grade-4 brain tumor, the deadliest among cancers with a five-year survival rates of only 7%.

BraTS challenge requires not only the segmentation of the whole tumor but also subsequently the tumor core and enhancing tumor (Figure 5). The Dice coefficient is used to measure the quality of the segmentation.

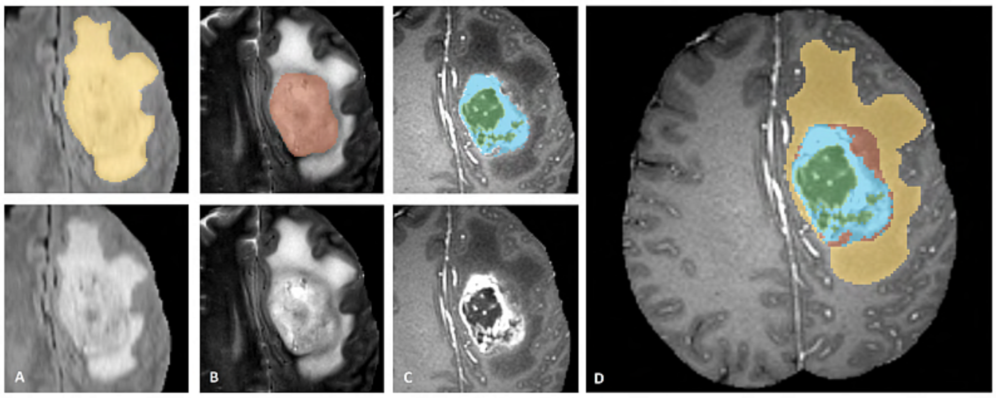


Figure 5. BraTS task description [36]. The whole tumor is visible in FLAIR (**A**), the tumor core in T2 (**B**), the enhancing tumor structures in T1c (blue), surrounding the cystic/necrotic components of the core (green) (**C**). Combined segmentation give the final labeled image (**D**): edema (yellow), nonenhancing solid core (red), necrotic/cystic core (green), enhancing core (blue).

5.3. Experiment Details

In the first experiment we consider 2D U-Net, 3D U-Net, and Cascaded Network for which the training details are presented in Table 3.

The second experiment combines the best method in terms of the highest Dice (i.e., 2D U-Net) with the proposed enhancement methods. Hence, the results of the retrained model are presented following a curricular learning (CL) and data augmentation (DA). The third experiment considers the equally weighted majority voting performed using the 3D U-Net, Cascaded Network and the best performing model (i.e., 2D U-Net + CL) from the second experiment. When used, all the DA and CL transformation are applied on 25% of the initial training dataset.

Table 3. Experiment Details.

Method	Loss Function	Training Set Size	# of Trainable Parameters	Test Set Size	Epochs	Training Duration (Hours)
Cascaded Network	Dice Loss	100	n/a	25	30	9
2D U-Net	Dice Loss	100	31,032,451	25	300	13
3D U-Net	Dice Loss	100	14,491,619	25	300	24

Curricular learning was first proposed by Bengio [37] to deal with nonconvex optimization to avoid the local optimum issue. The intuition behind Curricular Learning is to mimic the learning of human with a gradual training process with examples sorted in an increasing level of difficulty. Following this idea, we propose to pretrain the considered models from artificially downsampled MRIs by a progressive increasing level of resolutions. This enhancement was carried out by downsampling then upsampling by successive factors equal to eight, four and two. Hence, the first model is trained with the data that is downsampled/upsampled by a factor eight. Once saved, it is retrained with the data that is downsampled/upsampled by a factor four. This process is then repeated with the data downsampled/upsampled by a factor two. Finally, the resulting model is trained with the data in its original resolution.

Data augmentation is used to improve the robustness of the model by artificially increasing the size of the training dataset. In this study, the following geometrical transformations are used with randomly chosen settings: (a) 90 degrees rotation, (b) Horizontal/vertical flip, (c) Cropping, (d) Gaussian white noise.

In order to simultaneously take benefit of all the investigated methods, this proposal consists in developing an original method which combines the predictions provided by each technique (i.e., 2D U-Net, 3D U-Net, Cascaded network). An equally-weighted majority voting is then applied to each pixel of the input MRI. For the prediction, all the methods have the same relevance (weight) to assign a score to each prediction. The final decision is set to use the prediction, which obtains the highest voting score. If several different predictions obtain an identical score, the final prediction is randomly chosen among the best proposed choices.

5.3.1. Data

BraTS dataset was split using 125 patients: 25 patients are used for test, 75% for training, 25% for validation. To improve the computation efficiency of our evaluation, each MRI of the dataset was cropped from $240 \times 240 \times 155$ to $144 \times 160 \times 60$, removing background region pixels.

5.3.2. Training Protocol

All the three supervised methods were trained using the Dice Loss Function (DLF) equal to one minus the Sørensen–Dice index:

$$\mathrm{DLF}(P,T) = 1 - \frac{2\sum_i P_i \times T_i}{\sum_i P_i + \sum_i T_i + \epsilon} \tag{20}$$

where P denotes the set of the predicted pixels (P_i being the i-th element) and T the set of the corresponding ground truth. We arbitrary defined in Equation (20) $\epsilon = 1$ to deal with the particular case when P and T only contain background values equal to zero. The 2D and 3D U-Net were trained with 300 epochs while the Cascaded Network was only trained for 30 epochs due to time constraints. The network requires separate training for each region and each of the three views, which increases training time.

2D U-NET was first proposed for biomedical image segmentation by Ronneberger et al. [38] (Figure 6). This architecture contains two paths respectively called encoder and decoder, which contain several convolutional and maximum pooling layers at the encoder level and transposed convolution (up-conv) layers at the decoder. The autoencoder is designed to find a latent representation of a dimension smaller than the input that is used for the segmentation task. Unlike the U-Net originally proposed, zero-padding is used as well rather than maximum pooling to preserve the dimension of the output at each layer, allowing more flexibility for the dimension of the input. The U-Net used in this article follows the U-Net architecture proposed by Dong et al. [39] depicted in Figure 6.

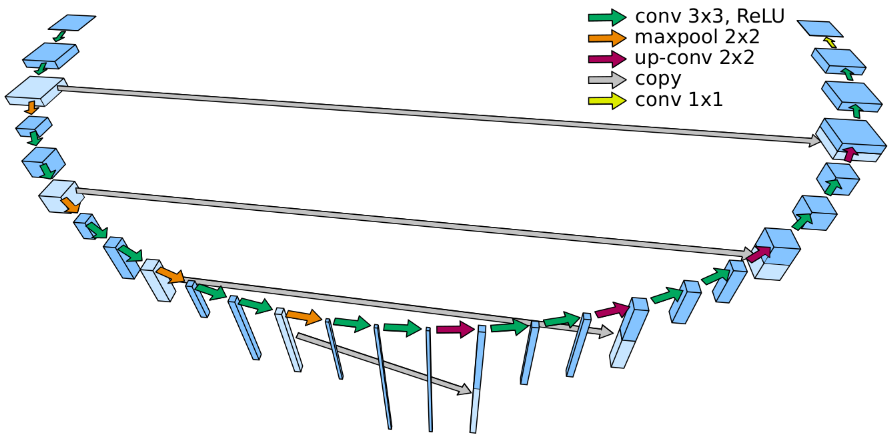
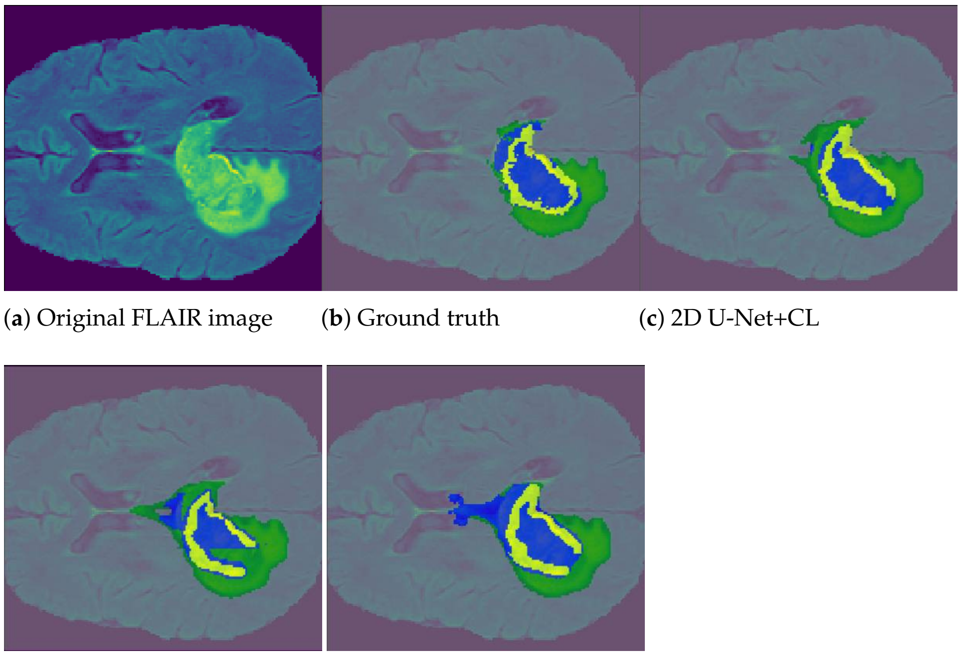


Figure 6. U-Net architecture with its encoder and decoder structure. The arrows represent the operations and the volumes are the characteristics: the height of the volume corresponds to the number of feature maps and the width and depth of the volume for the size of the feature maps. This U-Net uses 5 different resolutions.

3D U-NET extends the U-Net network for volumetric segmentation [40]. The input is taken as the voxels of the volumetric images and the resulting output is a 3D segmentation mask. All the operations are in 3D and a batch normalization of 10 has shown to improve the training convergence. Another difference is the reduction in the number of blocks in each path from five to four. The Dice loss function Equation (20) was also used for the training of this network. The encoder path contains two 3D convolutions followed by a Rectified Linear Unit (ReLU), and a $2 \times 2 \times 2$ maximum pooling with strides of two. The decoder path blocks include $2 \times 2 \times 2$ transposed convolution (up-conv) by strides of two in each dimension and two 3D convolutions followed by ReLU. The entire image is analyzed in the contracting path and subsequent extensions produce the final segmentation.

CASCADE NETWORK proposed by Wang et al. [33] includes a combination of three CNNs that segment each of three subregions sequentially: whole tumor, tumor core and enhancing tumor. Hence, anisotropic convolution (i.e., dependent on the direction) are used to deal with 3D MRI but it results in a higher model complexity and memory consumption. Lastly the fusion of the CNN outputs in three orthogonal views: axial, sagittal, and coronal is used to enhance the segmentation of the brain tumor. The three CNNs follow the hierarchical structure of the tumor subregions as depicted in Figure 7.

After the convolutional layer with zero padding, we get feature maps of the same size as the input. Then each feature map is passed through Z pooling with stride one and k different windows of sizes $d_1 \times d_1, d_2 \times d_2, \ldots, d_k \times d_k$ are used. The second layer is responsible for the increase of the receptive field, which is determined by the larger window size d_k. For an input of size $s \times s$ we suggest $d_k = 2s$ to ensure that the receptive field is as large as the input image. In these experiments, the multiplicity is chosen at $m = 10$ and the window size $d_i = 2^{i-1} + 1$ i.e., $d_i \in \{1, 3, 5, 9\}$. This is a good compromise between the size of the network and the expected performance. Hence $k = \lfloor \log_2(s) + 2 \rfloor$. The other window sizes determine the scales for which the information is collected. The initial convolution ensures that the features are relevant for each scale. The multiplicity m makes it possible to collect multiple features by scale. The convolution layers are followed by ELU [41] as an activation function.



(**a**) Original FLAIR image (**b**) Ground truth (**c**) 2D U-Net+CL

(**d**) 3D U-Net (**e**) Cascade network

Figure 7. Brain tumor segmentation of a MRI slice using three different methods (**c**–**e**). Given the original FLAIR image in (**a**), the different sub-regions correspond to predicted whole tumor region (green+yellow+blue), tumor core region (yellow+blue) , enhancing core (blue). Ground truth image (**b**) is obtained with manual segmentation.

5.3.3. Results and Discussion

Results presented in Table 4 (and illustrated in Figure 7) show the effectiveness of each method measured in terms of Sørensen–Dice index, Recall and Precision only on the tumor core region, the most difficult to segment. The pooling layers with configuration-2 favor segmentation unlike configuration-3, which favor ultrametric contours.

According to Table 4, the 2D U-Net obtains the highest Dice scores for the three subregions during the first experiment: tumor core = 0.65 (and for the record whole tumor = 0.65 and increased tumor = 0.46). The scores for the Cascade and 3D U-Net are not drastic. Given this result, the 2D U-Net was chosen for the improvement experiences: CL and DA. In regard to the equally weighted majority vote, the prediction of the first three methods was used to obtain the final prediction. The improvement in 2D U-Net results shows that the three improvement methods proposed improve Dice score and Precision, but the 2D U-Net formed with CL outperforms the others. Z pooling works comparatively better than maximum pooling in terms of accuracy, Recall, and Dice score. The Dice score indicates that Z pooling with max(min) misses fewer tumor cores on average than the max(max) combination. The best combination is obtained with a 2D U-Net with Curricular Learning, Zeck min(max) (Dice = 0.77, Recall = 0.8, Precision = 0.87) while comparatively 2D U-Net alone gives lower scores (Dice = 0.72, Recall = 0.79, Precision = 0.77). Note that the 2D U-Net association with DA and CL provides disappointing performances.

Table 4. Comparative results for segmentation of tumor core. Pooling layer choice: ① = (regular) maximum pooling ② = Zeck ∩ followed max(max) ③ = Zeck ∩ followed min(max).

DL Network	**Dice Score**			**Recall**			**Precision**		
Z Pooling Config	①	②	③	①	②	③	①	②	③
Cascaded Network	0.58	0.71	0.52	0.73	0.71	0.76	0.80	0.88	0.88
2D U-Net	0.65	0.62	0.72	0.77	0.77	0.79	0.81	0.86	0.77
3D U-Net	0.46	0.55	0.46	0.72	0.73	0.75	0.86	0.83	0.84
2D U-Net + CL	0.68	0.73	0.77	0.77	0.76	0.80	0.80	0.81	0.87
2D U-net+DA	0.67	0.75	0.62	0.77	0.76	0.82	0.81	0.83	0.78
2D U-Net + DA + CL	0.65	0.68	0.53	0.78	0.77	0.80	0.82	0.90	0.88
Majority Voting	0.61	0.67	0.60	0.76	0.75	0.79	0.82	0.85	0.84

CL = Curricular Learning, DA = Data Augmentation.

The intuitive explanation is that Z-pooling prepares CNN better than maximum pooling for the segmentation tasks by sharpening the edges. The weights of the CNN shall accentuate the edges during training whenever there is a significant difference between two adjacent pixels. In capturing ultrametric contours Z-pooling can be seen as a kind of pretraining of the network to accelerate the learning and enhance the segmentation result.

6. Conclusions

To conclude, the experiments presented along with the results have demonstrated a pipeline of evaluation for the supervised segmentation MRI images with Z pooling. CNNs have once again proven to excel in image processing and more specifically in learning and distinguishing characteristics that then enables segmentation. Simple or complex addition or changes based on Z pooling have proved to improve results, which reinforces the need to further advance research in this area. The goal of this research was met, which was not only to examine the presented methods but also to introduce the enhancements and enable a thorough comparison.

Earlier, we raised two questions: when should we perform pooling? Is texel pooling more efficient than pixel pooling?

It is advisable to pool when we can extract features contained in the binned subregions from the input representation (input image, hidden layer, etc.) As mentioned in the discussion, some of the enhancements in the pooling improved certain results and diminished others. However, in most of our experiments, the texelization of the pooling layer improved the image segmentation capacity of the CNN. It is because Z coding, compared to other local descriptors: (a) can be extended to any neighborhood size or geometry, (b) is invariant in shift (c) is invariant in rotation, (d) is nonlinear (e) follows a integer generating function, (f) is less sensitive to noise.

The correct scale is therefore part of the definition of texture and plays an important role. In other words, texel pooling is more efficient in general because our world is "textured" but performance decreases directly as signal-to-noise ratio gets worse.

We challenged the concept for feature extraction, which has been uncontested for three decades, the feature extraction pyramid. Our method translates the series of solutions to enhance Z pooling with different window sizes. The effective receptive field of our method can be modified freely through the pooling window sizes without affecting the parameter number, whereas traditional feature extraction pyramids have a high parameter cost associated with an increase of the receptive field.

Further investigations should target all combinations of Z pooling operators and find out a performance criterion to maximize that describes the pixel organization.

Author Contributions: V.V. and H.M.: conceptualization; V.V. and T.Q.S.: investigation; V.V. and T.Q.S.: data organization and analysis; V.V.: writing—original draft; V.V. and H.M.: writing—review and editing; V.V. and H.M.: supervision; All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Catedra franceza Project VRERI 021/2016 from Faculdade de Ciências Aplicadas, UNICAMP, Sao Paolo, Brazil.

Institutional Review Board Statement: Ethical review and approval were obtained by each individual institution which contributed with data to the BraTS dataset.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: BraTS dataset can be downloaded at URL https://www.med.upenn.edu/sbia/brats2017/data.html (accessed on 12 December 2020). Python code can be found at https://github.com/ikramabdel/tumorsegmentation (accessed on 12 December 2020).

Acknowledgments: The authors wish to express their sincere gratitude to the anonymous referee whose suggestions have led to a substantial improvement in the presentation of this article. They also want to thank Leonardo Tomazeli Duarte from FCA, Dominique Fourer from IBISC and Ikram Ibrahim for their respective contributions in the digital implementation.

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

References

1. Shen, D.; Wu, G.; Suk, H. Deep Learning in Medical Image Analysis. *Annu. Rev. Biomed. Eng.* **2017**, *19*, 221–248. [CrossRef] [PubMed]
2. Serre, T.; Wolf, L.; Poggio, T. Object recognition with features inspired by visual cortex. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005; Volume 2, pp. 994–1000. [CrossRef]
3. Arel, I.; Rose, D.; Karnowski, T. Deep Machine Learning—A New Frontier in Artificial Intelligence Research [Research Frontier]. *IEEE Comp. Int. Mag.* **2010**, *5*, 13–18. [CrossRef]
4. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
5. Yu, D.; Wang, H.; Chen, P.; Wei, Z. Mixed Pooling for Convolutional Neural Networks. In Proceedings of the International Conference on Rough Sets and Knowledge Technology, Shanghai, China, 24–26 October 2014; pp. 364–375.
6. Lee, C.; Gallagher, P.; Tu, Z. Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree. In Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, San Diego, CA, USA, 9–12 May 2015.
7. Lazebnik, S.; Schmid, C.; Ponce, J. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, New York, NY, USA, 12–17 June 2006; Volume 2, pp. 2169–2178. [CrossRef]
8. Haralick, R. Statistical and Structural Approaches to Texture. *Proc. IEEE* **1979**, *67*, 786–804. [CrossRef]
9. Li, C.; Zhang, Y.; Xie, E.Y. When an attacker meets a cipher-image in 2018: A year in review. *J. Inf. Secur. Appl.* **2019**, *48*, 102361. [CrossRef]
10. Yu, F.; Koltun, V. Multi-Scale Context Aggregation by Dilated Convolutions. In Proceedings of the International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2–4 May 2016.
11. Yang, M.; Yu, K.; Zhang, C.; Li, Z.; Yang, K. DenseASPP for Semantic Segmentation in Street Scenes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018.
12. Scherer, D.; Müller, A.; Behnke, S. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In *Proceedings of the 20th International Conference on Artificial Neural Networks: Part III*; ICANN'10; Springer: Berlin/Heidelberg, Germany 2010; pp. 92–101.
13. Sharma, S.; Rajesh, M. Implications of Pooling Strategies in Convolutional Neural Networks: A Deep Insight. *Found. Comput. Decis. Sci.* **2019**, *44*, 303–330. [CrossRef]
14. Gulcehre, C.; Cho, K.; Pascanu, R.; Bengio, Y. Learned-Norm Pooling for Deep Feedforward and Recurrent Neural Networks. In Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases, Nancy, France, 15–19 September 2014; pp. 530–546. [CrossRef]
15. Agostinelli, F.; Hoffman, M.; Sadowski, P.; Baldi, P. Learning Activation Functions to Improve Deep Neural Networks. *arXiv* **2014**, arXiv:1412.6830.
16. Boureau, Y.L.; Ponce, J.; Lecun, Y. A Theoretical Analysis of Feature Pooling in Visual Recognition. In Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel, 21–25 June 2010; pp. 111–118.
17. Sheskin, D.J. *Handbook of Parametric and Nonparametric Statistical Procedures*, 4th ed.; Chapman & Hall/CRC: Boca Raton, FL, USA, 2007.
18. Howell, D. *Statistical Methods for Psychology*, 6th ed.; Thomson: Belmont, CA, USA, 2007.
19. Lowe, D.G. Object Recognition from Local Scale-Invariant Fea- tures. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Fort Collins, CO, USA, 23–25 June 1999; Volume 2, pp. 1150–1157.

20. Moorthy, A.; Bovik, A. Visual Importance Pooling for Image Quality Assessment. *IEEE J. Sel. Top. Signal Process.* **2009**, *3*, 193–201. [CrossRef]
21. Pietikinen, M.; Hadid, A.; Zhao, G.; Ahonen, T. *Computer Vision Using Local Binary Patterns*; Computer imaging and vision; Springer: Berlin/Heidelberg, Germany, 2011; Volume 40.
22. Ojala, T.; Pietikäinen, M.; Harwood, D. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognit.* **1996**, *29*, 51–59. [CrossRef]
23. Tan, X.; Triggs, B. Enhanced local texture feature sets for face recognition under difficult lighting conditions. In *Analysis and Modeling of Faces and Gestures*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4778, pp. 235–249.
24. Zeckendorf, E. Représentation des nombres naturels par une somme de nombres de Fibonacci ou de nombres de Lucas. *Bull. Soc. Roy. Sci. Liege* **1972**, *41*, 179–182.
25. Vigneron, V.; Syed, T.; Duarte, L.; Lang, E.; Behlim, S.; Tomé, A. Z-Images. In Proceedings of the 7th Iberian Conference on Pattern Recognition and Image Analysis, Faro, Portugal, 20–23 June 2017; pp. 177–184.
26. Yao, C.; Chen, S. Color texture retrieval, color texture segmentation, content-based retrieval, images, local edge pattern, similarity measure, texture, texture region. *Pattern Recognit.* **2003**, *36*, 913–929. [CrossRef]
27. Kellokumpu, V.; Zhao, G.; Li, S.Z.; Pietikäinen, M. Dynamic Texture Based Gait Recognition. In Proceedings of the International Conference on Biometrics, Alghero, Italy, 2–5 June 2009; pp. 1000–1009.
28. Wang, H.; Ullah, M.; Klaser, A.; Laptev, I.; Schmid, C. Evaluation of local spatio-temporal features for action recognition. In Proceedings of the British Machine Vision Conference, London, UK, 7–10 September 2009; Volume 124.
29. Chen, J.; Zhao, G.; Pietikäinen, M. An improved local descriptor and threshold learning for unsupervised dynamic texture segmentation. In Proceedings of the 2nd IEEE International Workshop on Machine Learning for Vision-based Motion Analysis (MLVMA09), Kyoto, Japan, 28 September 2009; IEEE: Kyoto, Japan, 2009; pp. 460–467.
30. Stone, Z.; Zickler, T.; Darrell, T. Autotagging Facebook: Social network context improves photo annotation. In Proceedings of the Computer Vision and Pattern Recognition Workshops, Anchorage, AK, USA, 23–28 June 2008; pp. 1–8. [CrossRef]
31. Fowlkes, C.; Martin, D.; Malik, J. Learning affinity functions for image segmentation: Combining patch-based and gradient-based approaches. In Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Madison, WI, USA, 18–20 June 2003; Volume 2, pp. 2–54. [CrossRef]
32. Shelhamer, E.; Long, J.; Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 640–651. [CrossRef] [PubMed]
33. Wang, G.; Li, W.; Ourselin, S.; Vercauteren, T. Automatic Brain Tumor Segmentation Using Cascaded Anisotropic Convolutional Neural Networks. In *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2018; Volume 10670 LNCS, pp. 178–190. [CrossRef]
34. Menze, B. The multimodal brain tumor image segmentation benchmark (BRATS). *IEEE Trans. Med Imaging* **2014**, *34*, 1993–2024. [CrossRef] [PubMed]
35. Razzak, M.; Naz, S.; Zaib, A. Deep learning for medical image processing: Overview, challenges and the future. In *Classification in BioApps*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 323–350.
36. Menze, B.H.; Jakab, A.; Bauer, S.; Kalpathy-Cramer, J.; Farahani, K.; Kirby, J.; Burren, Y.; Porz, N.; Slotboom, J.; Wiest, R.; et al. The multimodal brain tumor image segmentation benchmark (BRATS). *IEEE Trans. Med. Imaging* **2015**, *34*, 1993–2024. [CrossRef] [PubMed]
37. Bengio, Y.; Louradour, J.; Collobert, R.; Weston, J. Curriculum learning. In Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, QC, Canada, 14–18 June 2009; pp. 41–48.
38. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Munich, Germany, 5–9 October 2015; pp. 234–241.
39. Dong, H.; Yang, G.; Liu, F.; Mo, Y.; Guo, Y. Automatic brain tumor detection and segmentation using U-Net based fully convolutional networks. In Proceedings of the Annual Conference on Medical Image Understanding and Analysis, Edinburgh, UK, 11–13 July 2017; pp. 506–517.
40. Çiçek, Ö.; Abdulkadir, A.; Lienkamp, S.S.; Brox, T.; Ronneberger, O. 3D U-Net: Learning dense volumetric segmentation from sparse annotation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Athens, Greece, 17–21 October 2016; pp. 424–432.
41. Clevert, D.A.; Unterthiner, T.; Hochreiter, S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In Proceedings of the 4th International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.