Title: Motivation for a Retinal Inspired Neural Network(RNN)

Author: Jeramiah Johnson (j.johnson.bbt@gmail.com)

Abstract:
Data processing requirements for image processing with neural networks presents a bottleneck. Traditional neural network methods attempt to process whole images resulting in a growth of 2^(2n) of data per doubling(n) of information in one dimension. When examining how image processing is handled in nature, there are some distinct features which overcome this bottleneck. This paper presents a method inspired by nature which may result in a growth as low as 4n per doubling(n) of information in one dimension. Applying only the RNNi method with MNIST Fashion images of 28x28 found a processing time reduction of 244% over conventional whole image processing, while maintaining 81.4% accuracy(vs. 88.3% for conventional whole image processing via neural networks). Due to the linear nature of data requirements growth with n using an RNNi, this method could yield significantly lower data processing times for higher resolution image processing.

Body:

RNN-I Component (Retinal Neural Network Input)

In nature, the retina of eyes have a high density of rods in the center which become less dense with an increase in radius. This motivates the following method of input preparation for the RNN-I.

1. Choose a random coordinate on an image or image center.
2. The first row of the dataset will contain a 2x2 area, or 4 pixels centered around that point.
3. The second row will contain a 4x4 area around the center point, but with a mean average(or some other average) in each 2x2 cell. Thus the entry for the second row of the dataset will also contain 4 entries.
4. The third row will continue in this manner, increasing the area to 8x8, but averaging each 4x4 cell for 4 entries.
5. Keep repeating this process, quadrupling the area size while reducing the focus for each successive row.

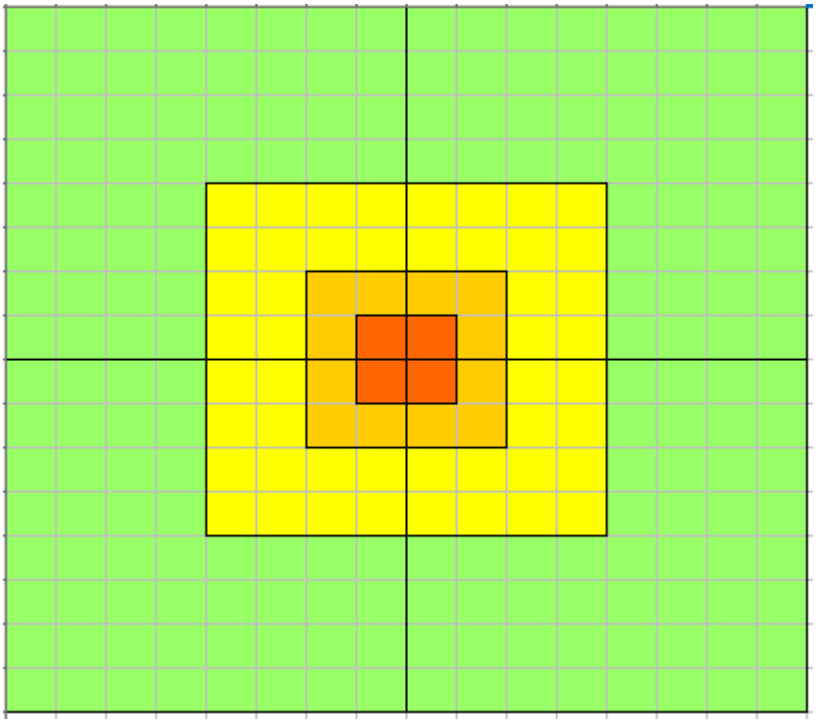


Figure 1

While a 64x64 image would normally require 4,096 pixels of data analyzed per image, the RNN-I method only needs 24 points of data per “look”.

The following chart compares the efficiency of this method based on increased dimensions of an image:

n	w,h (2^n)	area (2^(n*2))	RNN data (4n)
	0	0	0
1	2	4	4
2	4	16	8
3	8	64	12
4	16	256	16
5	32	1024	20
6	64	4096	24
7	128	16384	28
8	256	65536	32
9	512	262144	36
10	1024	1048576	40
11	2048	4194304	44
12	4096	16777216	48
13	8192	67108864	52
…	…	…	…

Training a smaller network will require less data points.

Alternatively, segments can begin with 4x4 or 8x8 or some other size, depending on the overall number of pixels being analyzed and to optimize for accuracy, for cases such as reading words/letters where more area needs to be in focus. For example, here would be a chart based on 4x4:

n	w,h (2^(n+1))	area (2^((n+1)*2))	RNN data (16n)
	0	0	0
1	4	16	16
2	8	64	32
3	16	256	48
4	32	1024	64
5	64	4096	80
6	128	16384	96
7	256	65536	112
8	512	262144	128
9	1024	1048576	144
10	2048	4194304	160
11	4096	16777216	176
12	8192	67108864	192
13	16384	268435456	208
…	…	…	…

And for 8x8:

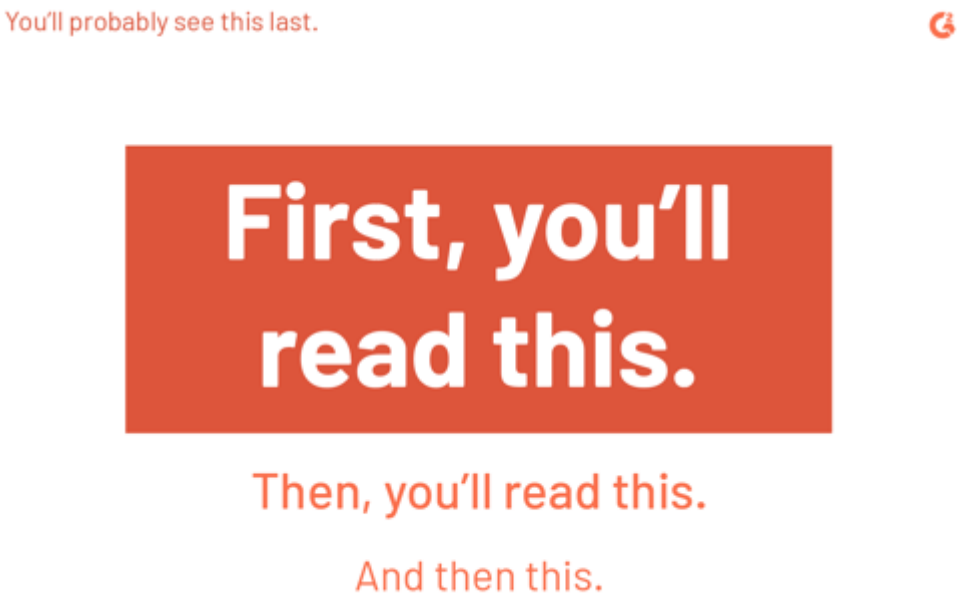
n	w,h (2^(n+2))	area (2^((n+2)*2))	RNN data (64n)
	0	0	0
1	8	64	64
2	16	256	128
3	32	1024	192
4	64	4096	256
5	128	16384	320
6	256	65536	384
7	512	262144	448
8	1024	1048576	512
9	2048	4194304	576
10	4096	16777216	640
11	8192	67108864	704
12	16384	268435456	768
13	32768	1073741824	832
…	…	…	…

So the data requirements are not very high, even for an 8x8 focal point with almost infinite range of view. Just as you wouldn't focus on a wall, the goal here is to allow the neural network to intelligently focus on the points of interest while ignoring the rest.

Outputs of the RNN-I can be the number of classifications plus 3 more outputs for transferring contextual data used by the RNN-Brn Component.

RNN-Brn Component

Oftentimes, we can identify what an image is with only one look. But in many cases, we need multiple looks in order to properly classify the image. Consider the following meme:



Our brains have developed a neural network in tandem with our visual processing which helps us to determine what is most important to focus on first.

So that is where a secondary neural network would be helpful in using this generalized RNN-I output data to determine if a) a classification can be made with the first look, and if not, b) where to look next. This will increase accuracy while requiring little in extra computation time.

The RNN-Brn can be structured with inputting the outputs of the RNN-I plus the center coordinates used by the RNN-I. The output could be three neural network outputs in tandem with one confidence filter using Deep Evidential Regression(http://www.mit.edu/~amini/pubs/pdf/deep-evidential-regression.pdf) or some other confidence analysis:

1. RNN-I confidence score; (And can loop if below a certain threshold)
2. RNN-I classification;
3. x-coordinate change for next "look" center point (used if looped)
4. y-coordinate change for next "look" center point (used if looped)

Comparing RNNi with Conventional Whole Image Processing(CWIP)
Method:
An experiment was done to compare processing times for the RNNi method with conventional image processing. Two python scripts were developed which are identical(scripts can be found at https://github.com/therealjjj77/RNNi*). The script ending in RNNi was edited to implement the above method using an initial focal point of 4x4, and then receding focus groups. Due to the images being 28x28, the final focus group was produced using a mean average of 7x7 per pixel entry into the group. Thus the image matrix was of size 16x4 for the RNNi while the matrix for the CWIP was 28x28.

The time was measured before running the neural network data processing and again after all epochs had run. Note that the time did not include RNNi data preparation (this can be improved for efficiency and should be faster when scaled due to these only involving arithmetic operations). Note that this

experiment did not utilize the above proposed RNN-Brn method, which should increase efficiency while requiring little more in the way of processing time.

Result:

The CWIP method(file ending in “-default”) completed all epochs in 36.17 seconds, while the RNNi (file ending in RNNi) completed all epochs in 14.83 seconds, yielding a 244% increase in speed. The CWIP method had an accuracy of 88.3% while the RNNi method had an accuracy of 81.4%.

*The code used was based on Neural Networks from Scratch in Python, Chapter 19: https://nnfs.io/.