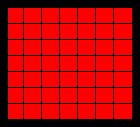
Weave Convolution:

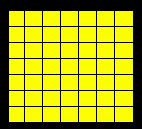
The purpose of weave convolution is to grant larger contextual information to convolution without increasing the size of the filters. Instead of increasing the size of the filters, we will use two separate passes of filters. The first set of filters will be used as local information, and the second will be used as peripheral information. Then the results of each filter will be “woven” together to create a larger image which will then be convolved on once again.

Looking at a center of an image 7 by 7 section (thus there is no need for padding, filters that go off the edge of this example are still are operating on the image) to make the example easier to understand;

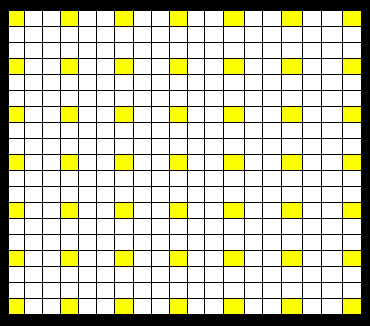
Starting center of an Image (Still works for even sized images, but pictured as odd for a nice center):



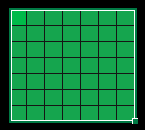
Passing a 3by3 convolution with stride 1 over the original image padded with a single layer of ones creates an image of the same size. These will be the local filters!



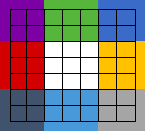
Expanding this image by 3 times (2 zero rows between each row due to the filter size of 3 by 3) results in the base image. This operation is detailed in **ZeroExplode** below.



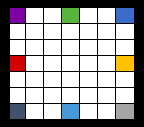
Passing a 3by3 convolution with stride 1 over the original image padded with a single layer of ones creates an image of the same size. These will be the peripheral filters!



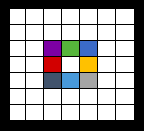
Now for the more complicated step which is described in **ArrayWeave**, we must arrange the following peripheral filters in fashion that surrounds each local filter with the neighboring peripheral filters. The next few images show this step for the center pixel. The white 3 by 3 square would be the pixels that create the local pixel and the 8 colored squares that cover the neighboring pixels exist in the green image above and must be moved into the proper place.



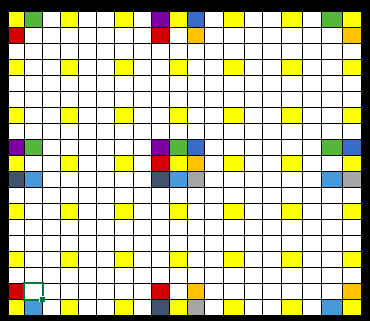
They would be located in these positions for the green image of peripheral filters.



Then this “image” of peripheral filters are “woven” (hence the name Weave Convolution) in the following way for each pixel.



This will then create an array as the same size as the **ZeroExplod** array (Giant Sparse Yellow Array). They are then combined into the same image. Here for one center square the operation was preformed, in addition it can be seen that these peripheral filters are also used in neighboring spots.



Then finally, a 3 by 3 convolution filter is passed over this larger image, padded again with a layer of ones) with a (3, 3) stride to return an image of the same size as the original.

The intuition is now that each local convolution filter will be surrounded by 8 peripheral 3 by 3 filters that can give help supply larger contextual information which can then be joined with the final filters.

Below is the Algorithm for Weave Convolution. In addition, Naïve versions of ZeroExplode and ArrayWeave are presented. I have faster versions in TensorFlow but I am still working on optimizing them.

**Weave Convolution:**

**Inputs:**

Image of Shape (*C, N, N*); *image*

Filter Size: *filter\_size* (must be odd)

Number of Filters; *n\_filters*

**Do:**

*pad* 🡨 (*filter\_size – 1) / 2*

*p\_image* 🡨**2DZeroPad(***image,* *pad***)**

*p\_image* of size (*C, N + pad, N + pad)*

*conv\_loc* 🡨 **2DConvolve** *p\_image* with *n\_filters* of size (*filter\_size, filter\_size)* with stride 1

*conv\_loc* of size (*n\_filters, N, N*)

*conv\_per* 🡨 **2DConvolve** *p\_image* with *n\_filters* of size (*filter\_size, filter\_size)* with stride 1

*conv\_loc* of size (*n\_filters, N, N*)

e\_conv\_loc 🡨 **ZeroExplode** *conv\_loc* by 2 \* *pad*

*w\_conv\_per* 🡨 **ArrayWeave** *conv\_per* by 2 \* *pad*

*conv\_total* 🡨 e\_conv\_loc + *w\_conv\_per*

*total\_image* 🡨**2DZeroPad(***conv\_total,* *pad***)**

*o\_image* 🡨 **2DConvolve** *total\_image* with *n\_filters* of size (*filter\_size, filter\_size)* with stride

*filter\_size*

*o\_image* of size (*n\_filters,* *N,* *N*)

**return** *o\_image*

**ZeroExplode:**

**Inputs:**

Image of Shape (*C, N, N*); *image*

Distance: *distance*

**Do:**

*e\_image* 🡨 zero array of size (*C, distance\**(*N-1) –* distance, *, distance\**(*N-1) –* distance)

**for** *i\_pos*  in **{**0, 1, …, *N-1*}:

**for** *j\_pos* in **{**0, 1, …, *N-1*}:

*e\_image*[: , (*distance+*1)\**i\_pos*, (*distance+*1)\**j\_pos*] 🡨 *image*[: , *i\_pos, j\_pos*]

**return** *e\_image*

**ArrayWeave:**

**Inputs:**

Image of Shape (*C, N, N*); *image*

Distance: *distance*

**Do:**

*e\_distance* 🡨 2 \* *distance* + 2

*w\_image* 🡨 zero array of size (*C, distance\**(*N-1) –* distance, *, distance\**(*N-1) –* distance)

**for** *i\_pos*  in **{**0, 1, …, *N-1*}:

**for** *j\_pos* in **{**0, 1, …, *N-1*}:

**for** *i\_change* in {-e*\_distance, 0, e\_distance*}:

**for** *j\_change* in {-e*\_distance, 0, e\_distance*}:

*new\_x* 🡨 *i\_pos*\*(1+*distance) + i\_change*

*new\_y* 🡨 *j\_pos*\*(1+*distance) + j\_change*

**if** *w\_image*[*new\_x, new\_y*] exists **and** (*i\_change, j\_change) !=* ***0***:

*w\_image*[: , *new\_x, new\_y*] 🡨 *image*[: , *i\_pos, j\_pos*]

**return** *w\_image*