Rotational Invariance and Hilbert Curves

Peter Chang
90r Report

December 28, 2017

Problem Summary

Within the context of image classification, rotational invariance is a sought after property in classification algorithms. This property means that an algorithm will be able to classify images even when said images are rotated. A solution such as adding additional training data is only reasonable up to a certain point so further development is needed on the algorithmic side.

Leading Solution

The current leading method towards achieving rotational invariance was developed by Worrall, Garbin, Turmukhambetov, and Brostow and is outlined in their paper "Harmonic Networks: Deep Translation and Rotation Equivariance". They did this by replacing the typical CNN filter layer with circular harmonics. This method has achieved the highest accuracy on the rotated MNIST dataset of any method and also performs well on similar rotational recognition problems. Within the rotational MNIST problem, they brought the error rate to below 2 percent.

Common Issue

The convolutional neural network is the standard algorithm used for image recognition problems. On a non-rotated dataset it works very well but once rotations are introduced, the accuracy drops to 95 percent.

Typically for a 1-dimensional CNN, the pixels of a 2 dimensional image are read in row by row into a 1 dimensional array before the classification algorithm is run on them. However, this process poorly maintains the information from the original image. For example, when a 28 by 28 pixel image is read into an array row by row, the pixel at [0,27] and the pixel at [1,0] will be directly next to each other while the pixel at [0,0] and the pixel at [1,0] will be far away from each other. With this property it would be assumed that the pixels [0,27] and [1,0] have a strong correlation and that the pixels [0,0] and [1,0] would have a weak correlation when in fact the opposite is true.

So this method uses the mapping:

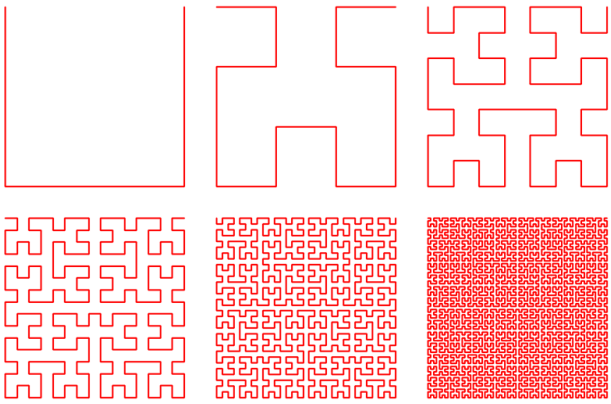
$$d = 28y + x \tag{1}$$

Where d is distance of a pixel along the flattened array and (x,y) are the original coordinates of the pixel. The problem with this mapping is that pixels which are close on the (x,y) plane do not necessarily have similar values of d (particularly when y is different).

Idea: Hilbert Curves

A Hilbert curve is a fractal space filling curve which in 2-dimensions fills a 2^n by 2^n square space. (see Figure 1). It works by recursively dividing the space into 4 quadrants then looping through each of these 4 quadrants. The idea to be tested is whether flattening an image along this Hilbert curve would help a CNN gain high accuracy.

Figure 1: 6 iterations of a Hilbert curve



Potential Advantages to Flattening Along Hilbert Curve

Using this could have an advantage since the Hilbert curve makes loops such that points which are near each other on the y-dimension will also be closer to each other in distance along the Hilbert curve. There is a deterministic formula to map from the (x,y) position to the distance d along the curve, as well as the inverse mapping. The method for this, coded in C, can be seen in Figure 2.

Hopefully this different flattening method will preserve some of the 2-dimensional information and allow for a more accurate classification.

Figure 2: Source-Wikipedia

```
//convert (x,y) to d
int xy2d (int n, int x, int y) {
    int rx, ry, s, d=0;
    for (s=n/2; s>0; s/=2) {
        rx = (x & s) > 0;
        ry = (y & s) > 0;
        d += s * s * ((3 * rx) ^ ry);
        rot(s, &x, &y, rx, ry);
    }
    return d;
}

//convert d to (x,y)
void d2xy(int n, int d, int *x, int *y) {
    int rx, ry, s, t=d;
    *x = *y = 0;
    for (s=1; s<n; s*=2) {
        rx = 1 & (t/2);
        ry = 1 & (t ^ rx);
        rot(s, x, y, rx, ry);
        *x += s * rx;
        *y += s * ry;
        t /= 4;
    }
}
```

Method

Data

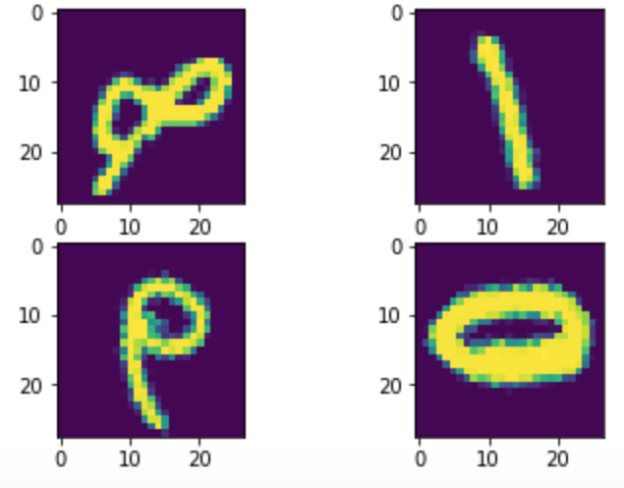
The dataset for this was the rotational MNIST dataset. This is the same dataset used by the team that produced the current leading method. The benchmark accuracies on the rotational dataset are 98.3 percent accuracy for the circular harmonics method and 95 percent for the standard CNN.

The rotational MNIST images came in a 1-dimensional array of length 28^2. They were already flattened via the row by row method (I rebuilt some of them in Figure 3 as an example). I started by restacking these into the 2-dimensional images and adding 4 blank rows and columns to the bottom and right of the images. This would make conversion via the Hilbert curve much easier and shouldn't affect the accuracy since CNNs are translationally invariant. I then converted these into 1-dimensional arrays of length $32^2 = 1024$ which were flattened along the Hilbert curve instead of the row by row method (path shown in Figure 4 starting in top left).

Given a distance d_0 along the 1-dimensional array flattened row by row, we can get it's respective distance along the Hilbert curve via:

$$d_{Hilbert} = xy2d(5, d_0 mod 28, \lfloor \frac{d_0}{28} \rfloor) \tag{2}$$

Figure 3: Examples of rotated MNIST



Where the function xy2d is given in Figure 2 [1]. By using this conversion and iterating this over all the images in the rotational MNIST dataset, I ended up with a series of 1 dimensional arrays flattened along the Hilbert curve. These, aside from being longer in length, had the same data structure as the original rotational MNIST dataset.

Replicating Baseline Results

I started by replicating the baseline results on rotational MNIST by the standard CNN. The first convolution layer computes 32 features over 4x4 patches with a stride of one. Then the image is max pooled over 2x2 blocks. The second convolution computes 64 features for each 4x4 patch followed by another max pooling. Next there are two fully connected layers with a dropout layer in between.

I started by testing this architecture on the standard MNIST dataset. The only change I made was to convert all the images into 32 by 32 pixel by adding blank spaces to the ends of the images. This was for the sake of consistency with the rest of the tests. Training this on the 12,000 training images for 20,000 iterations of 50 images per batch gave a test accuracy of 98.5 percent.

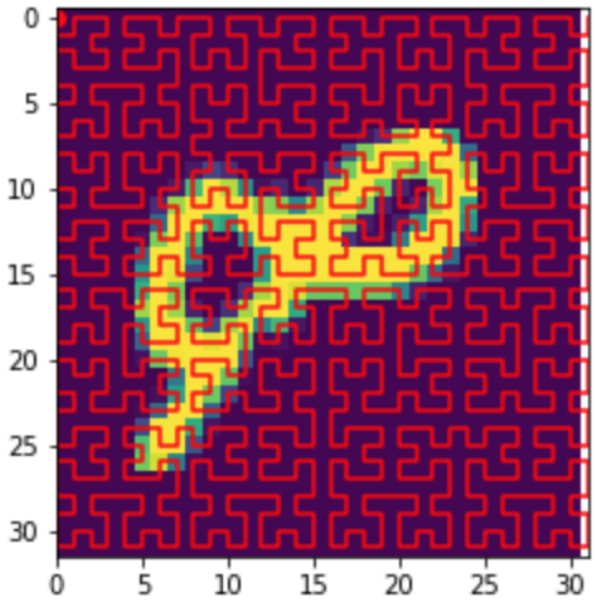
I then trained this model on the 12,000 images in the training set of the rotational MNIST package. These images had also been expanded to be 32 by 32 pixels in order to remain consistent in comparison to the Hilbert flattened test later. There were 20,000 training iterations and each iteration had a batch size of 50 images. The test result over 50,000 images yielded an accuracy of 94.6 percent. This is consistent with previous results of a CNN on rotated MNIST.

Testing Hilbert Method

In order to maintain consistent results for comparison, the general architecture for this CNN was kept the same as the CNN used for the baseline result with translations made to

[1] When implemented I used https://github.com/galtay/hilbert_curve/blob/master/hilbert.py

Figure 4: Flattening path



account for the 1-dimensional structure of the Hilbert flattened images.

The first convolution layer computes 32 features over a one dimensional patch. A 16x1 patch along a Hilbert curve is equivalent to a 4x4 patch in the 2D image but I also tested other convolution sizes. Then the image is max pooled over 4x1 blocks, which is equivalent to pooling over 2x2 blocks in 2D images. The second convolution computes 64 features for each one dimensional patch (of same size as the first convolution). This is followed by another max pooling over 4x1 blocks. Finally, there were two fully connected layers with a dropout layer in between. These layers were put in the same order as was tested in the baseline.

I trained this model on the 12,000 images from the rotational MNIST which had been flattened into 1-dimensional arrays of length 1024 via the Hilbert curve. I did 20,000 training iterations with a batch size of 50 and computed a testing accuracy every 400 iterations, keeping the highest test accuracy result.

Results

Varying Convolution Size

I started by varying the size of the patches that the convolution patches evaluate over. I tested a 16x1 patch, which would correspond to a 4x4 patch in 2-dimensions, a 64x1 patch, which would correspond to an 8x8 patch in 2-dimensions, and a 25x1 patch because some CNN models use 5x5 patches for MNIST.

Of these trials, the 16x1 convolution performed the best. This seems to make sense since it 16x1 would exactly fill an iteration of the Hilbert fractal and is not too large like the 64x1 case.

Test Accuracy on Rotated MNIST	
Patch Dimensions	Accuracy
16x1	.907
25x1	.906
64x1	.893

Varying Stride Length

I then increased the stride length since in this 1-dimensional case, a stride of 1 would be too small. I ran trials with stride lengths of 2, 4, 8, and 16. These were chosen to match the geometry of the Hilbert curve. The convolution layers evaluated 16x1 patches for all of these.

Of these trials, the stride length of 16 performed the best and performed marginally better than the trial with stride length 4.

Test Accuracy on Rotated MNIST	
Stride Length	Accuracy
1	.907
2	.907
4	.91
8	.905
16	.912

Aggregate

When running the convolutional neural network on the Hilbert flattened data, the highest accuracy I achieved was 91.2 percent, which was achieved when the convolution layers evaluated 16x1 patches with a stride length of 16. This is lower than the baseline model which we were comparing to which had an accuracy of 94.6 percent.

Test Accuracy on Rotated MNIST			
Algorithm	Convolution	Stride	Accuracy
Hilbert CNN	64x1	1	.893
	16x1	8	.905
	25x1	1	.906
	16x1	1	.907
	16x1	2	.907
	16x1	4	.91
	16x1	16	.912
Baseline 2D CNN	4x4	1	.946
H-Net (leading method)			.983

I had expected the Hilbert CNN model to perform at least as well as the baseline model, even if it did not reach that of the leading method. However in my trial this did not seem to be the case.

I did not seek to significantly alter the architecture of the CNN in the Hilbert method because I wanted a more clear comparison in its performance to the baseline model.

Further Tests

Throughout testing, I ran some other trials with the results here:

Hilbert CNN on Standard MNIST

I tested using the Hilbert flattening method on the non-rotated MNIST dataset to see how it performed. Using the 16x1 convolutions, a stride of 1, and 4x1 max pooling, this model achieved an accuracy of 97.4 percent. This is one percentage point lower than that of the regular CNN on the non-rotated MNIST.

Standard CNN with 5x5 Patches

Some CNN models for the non-rotated MNIST dataset use convolutions over 5x5 patches. I tested this to see if there would be any noticeable difference between my baseline model but received an almost identical testing accuracy of 98.4 percent.

1D CNN

I ran the same CNN architecture used in my final Hilbert CNN but on the original data which was a 1-dimensional array flattened row by row. I did not stack these into 2D images before running the CNN on them.

Running this on the non-rotated MNIST dataset gave an accuracy of 97.7 percent, which is roughly the same as the Hilbert CNN's performance on the non-rotated MNIST.

Running this on the rotated-MNIST dataset gave an accuracy of 91.6 percent, which is again similar to the performance of the Hilbert CNN.

These results would seem to imply that there is little performance difference in two 1-dimensional CNNs when the flattening is done row by row versus when it is done via the Hilbert curve.

Possible Explanations

Since I seemed to get very similar results when flattening row by row and when flattening by the Hilbert Curve, it seems like the difference in accuracy between the Hilbert CNN and the baseline CNN could be more due to the differences in the 2D and 1D CNNs than from the way the image is reshaped.

Row by row flattened images have the problem that pixels directly above or below each other are separated when put in the 1-dimensional array. The idea for the Hilbert curve was that it should reduce this problem, however there are still going to be points on the Hilbert curve which are close in 2-dimensions and very far apart when stretched into 1-dimension.

For example, on 32 by 32 pixel image, the points (15,0) and (16,0) are close in 2-dimensions but are separated by a distance of several hundred on the Hilbert curve.

This type of separation happens to any two close points which lie on opposite sides of the middle of the image, both in the x and y directions. This could possibly pose a problem when training the MNIST data since the pixels in the center of the number would get very separated along the Hilbert curve. This happens again at every smaller component of the Hilbert curve, where within each quadrant points close to each other at the center of the quadrant may be separated by a large distance. This effect does diminish significantly at the smaller levels though.

It's possible that this separation effect is large enough for flattening along the Hilbert curve to have similar problems that flattening row by row does. This could explain why the accuracy of the Hilbert CNN is almost identical to that of the CNN run on row by row flattened images.