ASSIGNMENT 1- ROTATION INVARIANCE IN CNN MODEL

A PREPRINT

Mehrad Jaloli*
Department of Electrical and Computer Engineering
University of Texas at San Antonio
San Antonio, TX 78249
mehrad.jaloli@utsa.edu

April 1, 2019

ABSTRACT

Invariance means that you can recognize an object as an object, even when its appearance varies in some way. This is generally a good thing, because it preserves the object's identity, category, (etc) across changes in the specifics of the visual input, like relative positions of the viewer/camera and the object. Figure 1 contains many views of the same statue. You (and well-trained neural networks) can recognize that the same object appears in every picture, even though the actual pixel values are quite different.

Keywords CNN, equivariance, Invariance, Rotation

1 CNN Model

Convolutional Neural Network (CNN) is a deep model working based on convolution concept. CNN is constructed of some convolution and max-pooling layers, called feature extraction part, and some fully connected layers which is the classification part. The whole idea of training in CNN model is while feeding images to the network, we have some kernels acting like filters. These kernels slide all over the images and do some filtering function based on which some features are extracted from the images. So, the output of each convolution layer would be some feature maps.

Based on the complexity of the problem and data set that we are using, the depth of the model (number of the layers) could be different. The more complex our model is, the more convolution layers we need for having a better feature extraction.

One of the main concerns in using CNN models in real life is if those models can keep their performance well even while input images have some transformations, like rotation. So, we went through this concept and tried to solve one of the translation problems, "Rotation Invariance".

Figure 1: Invariance

*https://github.com/username of your github?

2 Invariance vs Equivariance

Translation invariance means that the system produces exactly the same response, regardless of how its input is shifted. For example, a face-detector might report "FACE FOUND" for all three images in the top row. Equivariance means that the system works equally well across positions, but its response shifts with the position of the target. For example, a heat map of "face-iness" would have similar bumps at the left, center, and right when it processes the first row of images.

This is is sometimes an important distinction, but many people call both phenomena "invariance", especially since it is usually trivial to convert an equivariant response into an invariant one–just disregard all the position information)

3 Rotation Invariance

Based on section 2, we try to define rotation invariance and tied to solve it.

The procedure has been using the MNIST data set, feeding it into the model LeNet5, which is one of the well-known CNN model for image classification problem, training the model with the original data and then test the model with rotated images. Our purpose is to see whether the model can have an acceptable performance in classifying these rotated images as test data or not.

So, to go through this problem, firstly, we try to give a brief description about how each layer works in CNN model and how the layers are connected to each other.

3.1 CNN Operation

Suppose that we are using MNIST data set which are images of gray handwritten digits with the size 28*28*1 (1 is the number of color channel which for gray is equal to 1).

Once we feed the data into a convolution layer, there would be a kernel of size 5*5 (in LeNet-5 model)sliding all along the image and corresponds each block of 5 by 5 pixels of the original image to 1 pixel in the output of the first conv layer.

The output of each conv layer will be K images of size (Ns-Ks+1)*(Ns-Ks+1), while Ns*Ns is the size of original images fed to the model, Ks is the size of the kernel used for the conv layer and K is the number of kernels. Notice that each one of the images at the output of the conv layer are called **feature map**.

After the conv layer, we usually apply a pooling layer to the feature map (output of the conv layer). The purpose of using pooling layers is doing **Down Sampling** or reducing the computational cost. Here we should note that in this code we are applying kernels with **stride=2**, so it affects the size of the out put feature map.

By using a pooling layer of size 2*2 we are actually dividing the size of feature maps by 2. So, considering the MNIST dataset, after using **6** kernels of size 5*5 with stride=2 for the first conv layer we will have:

$$6*(32-5+1)*(32-5+1) = 6*28*28$$

Then by applying a pooling layer of size 2*2 we will have,

$$\text{size of the images after pooling layer} = 6*14*14$$

We can have different pooling layers. Maxpooling, takes the maximum value between the pixels that pooling block is applied while Average pooling takes the average of all the pixels the pooling block is applied.

Max pooling extracts the most important features like edges whereas, **average pooling** extracts features so smoothly. For image data, you can see the difference. Although both are used for same reason, max pooling is better for **extracting the extreme features**. Average pooling sometimes can't extract good features because it takes all into count and results an average value which may/may not be important for object detection type tasks.

It should be noted that the whole purpose of the convolution and pooling layers is feature extraction.

After 3 convolution and 2 maxpooling layers in LeNet5 (feature extractor part of the CNN) we will get to the dimension, 120*7*7. Then we will have some **Fully Connected** layers.

The purpose of fully connected layers are **Classification**. So, we use a **Flatten Layer** and transform the images to a vector (which in LeNet-5 is a vector of size 120).

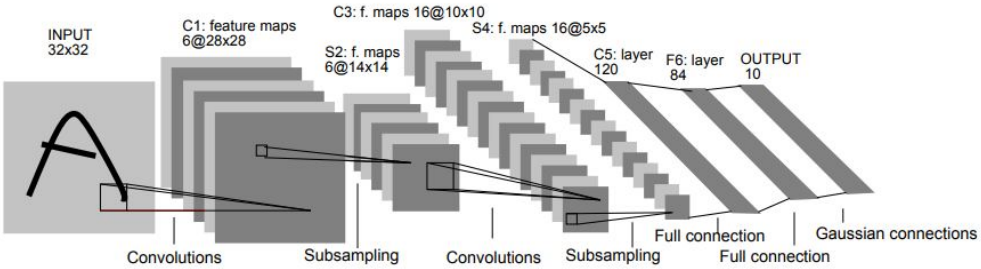

Figure 2: LeNet-5 Architecture

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1            [-1, 6, 28, 28]             156
         MaxPool2d-2            [-1, 6, 14, 14]               0
            Conv2d-3           [-1, 16, 14, 14]           2,416
         MaxPool2d-4             [-1, 16, 7, 7]               0
            Conv2d-5            [-1, 120, 7, 7]          48,120
            Linear-6                  [-1, 120]         705,720
            Linear-7                   [-1, 84]          10,164
            Linear-8                   [-1, 10]             850
================================================================
Total params: 767,426
Trainable params: 767,426
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.12
Params size (MB): 2.93
Estimated Total Size (MB): 3.05
----------------------------------------------------------------
```

Figure 3: LeNet-5 Layers

Then we use **Fully Connected** layers of size 84 and 10 respectively, and based on machine learning algorithms, specifically classification algorithms, we classify the images in 10 classes (digits from 0 to 9).

So, for the outermost layer we use a **Soft Max** layer which takes output of the last Fully Connected layer (which is a vector of size 1*10) and results 10 neurons as output corresponding to 10 classes (digits from 0 to 9)./

The cost function used in LeNet-5 model is **Cross-Entropy** the outputs of which are probabilities.

Below in figure 2, you can see the architecture of LeNet-5 model.

Furthermore, the detailed architecture of Lenet-5 model based on the layers would be like figure 3.

4 Visualization of the Layers

To see if the CNN model is **Rotation Invariance** or not, we should be able to visualize the out put of each layer to see the changes after each conv layer.

In figure 4 we can visualize the kernels of the first convolution layer, which are actually the **weights** we want to obtain after training the model.

Moreover, I have also attached the output of the 2nd and 3rd convolution layer in appendix on figure 7 and 8.

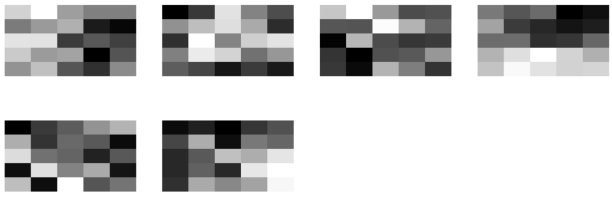

Figure 4: Kernels of the 1st Convolution Layer (6 Kernels)

To have a better understanding of what visualization means, we choose one of the images from data set and feed it to the model. Then we look at the output of each convolution layer to see how the model actually comes up with prediction of mentioned digit.

As an instance, we feed one of the images which is the handwritten of digit 4. As we go through the model, we can see the process in which model can identify the actual digit written in the picture.

Figure 5 represents the images obtained from out put of the first convolution layer which we call **Feature Maps**.

Figure 5: Out put of the Activation Function of the 1st Convolution Layer (6 Images)

As we can see, based on the architecture we have selected for our model, we have 6 feature maps at the out put of the first convolution layer so on figure 5, we see the out pt of each activation function applied to each feature map.

we can have this visualization for the other convolution layers as it is shown on figures 9 and 10 in appendix section.

5 Results

As we mentioned before, the goal of this project was investigating CNN model to see if it is Rotation Invariant or not. For this, we use visualization and try to go through the out put of each layer.

Thus, we define a function that gets images and rotate them by some specific degrees. In the code, this function is called, "**rotate_tensor**. Using this function, we rotate an image (one image of label 4) by 90, 180 and 270 degrees, as shown in figure 6.

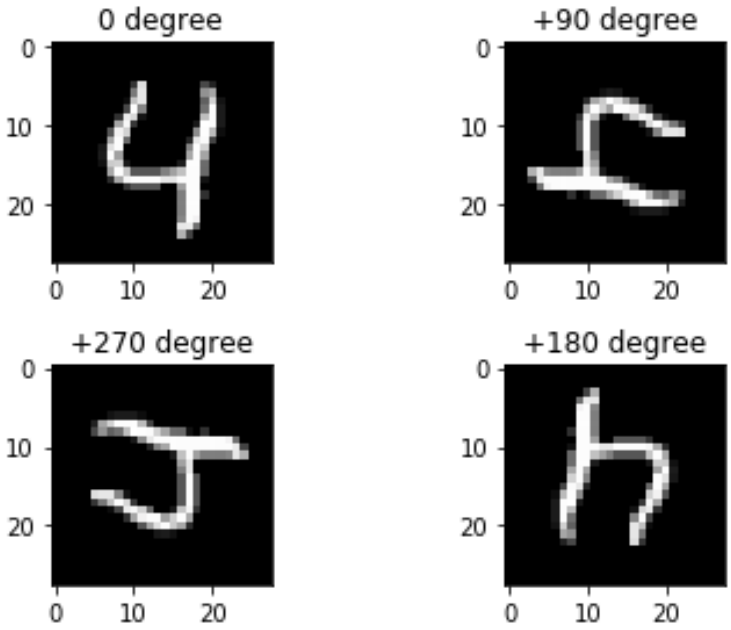

Figure 6: Rotated images after applying rotate_tensor function

For investigating the Rotation Invariance property of LeNet5, we use the pre-trained model with original MNIST dataset, then we test our rotated data sets to see the accuracy of the model. Below, in table 1, you can see the results of training and testing LeNet-5 with the original MNIST data set.

	Training Loss	Test Loss	Test Accuracy
Train and Test with MNIST	0.100	0.099	97%

Table 1: Train and Test results for original MNIST data set

Furthermore, in table 2 we can see the results obtained from testing the pre-trained LeNet5 model with images rotated by 90, 180, 270 and 360 degrees.

	Test Loss	Test Accuracy
Test with 90 degree rotated images	5.9623	14%
Test with 180 degree rotated images	6.2347	33%
Test with 270 degree rotated images	5.1319	13%
Test with 360 degree rotated images	0.084	97%

Table 2: Test loss and accuracy for rotated images

Note that, we have tested with 360 degree rotated images to see if it gives the same results as the original MNIST data set or not. As we can see, based on table 1 and 2, the test accuracy for these 2 data sets are the same, proving that our function (rotate_tensor) works properly

6 Conclusion

To put this problem in to perspective, we can say that the CNN models are **not Rotation Invariant** since as it is represented in the previous section, if the model is trained with a certain data set, it can not give a good accuracy on rotated images from the same data set.

7 Appendix

Figure 7: Kernels of the 2nd Convolution Layer (6*16 Kernels)

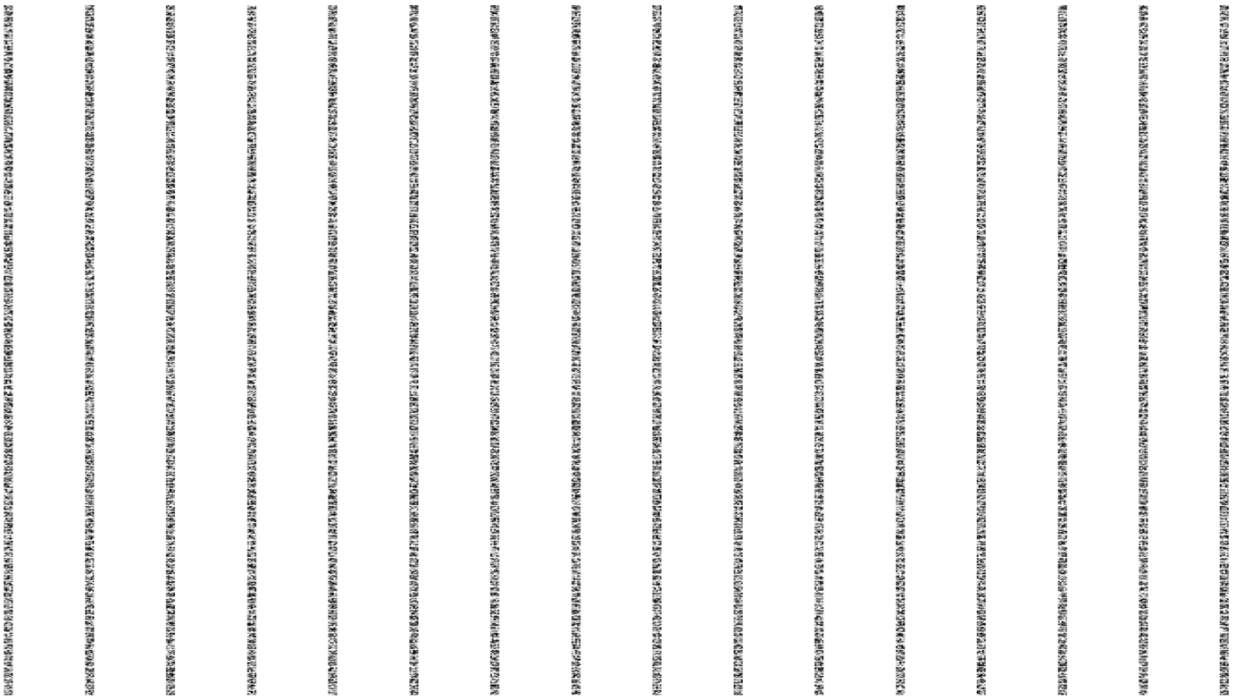

Figure 8: Kernels of the 3rd Convolution Layer (16*120 Kernels)

Figure 9: Feature Maps of the 2nd Convolution Layer (16 images)

Figure 10: Feature Maps of the 3rd Convolution Layer(120 images)

References

1. https://github.com/arundasan91/IS7033.git
2. Shield, Malcolm, and Shelley Dole. "Assessing the potential of mathematics textbooks to promote deep learning." Educational Studies in Mathematics 82.2 (2013): 183-199.
3. LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. "Gradient-based learning applied to document recognition". Proceedings of the IEEE, 86(11), pp.2278-2324.
4. Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton "ImageNet Classification with Deep Convolutional Neural Networks". In Proceedings of the NIPS 2012