Deconvolution and checkboard Artifacts

Vishnu Vikas, Devireddi

Introduction:

Most of the images generated by neural networks have a strange checkerboard pattern of artifacts. They exist because when neural networks generate images, they build the images up from low resolution, high level descriptions which allows them to describe the raw image first and then scale it up to fill in the details. This operation is usually done with a deconvolution layer which allows the model to use every point in the small image to paint a square in a larger one.

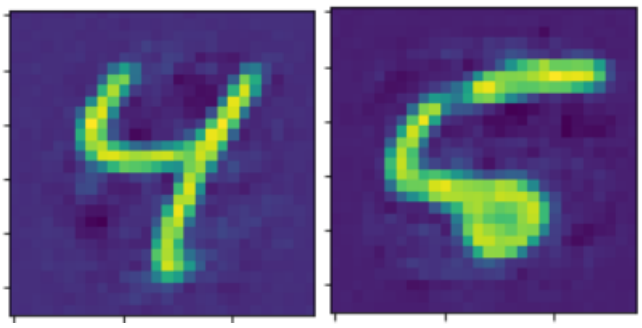
Main Problem:

Deconvolution in deep learning is concerned with mapping a set of data values to another large set of data values resulting in up-sampling the data. It can easily have an uneven overlap by putting more of the metaphorical paint in different places of the image. This usually occurs when the kernel size (filter size) is not divisible by the stride. This uneven overlap results in creating a checkerboard like pattern of varying magnitude of an image. Since the neural networks use multiple layers of deconvolution layers, iteratively building an image from a series of low resolution descriptions, the problem of this uneven overlap gets multiplied after each layer resulting in creation of artifacts on variety of scales.

Deconvolutions with stride 1 are effective at removing the artifacts of frequencies that divide their size and dampening other artifacts of frequencies less than their size. Yet, some of the artifacts still leak through because they tend to be more prominent when outputting unusual colors. Since neural networks typically have a bias they tend to output an average color from the distinguished colors resulting in requirement of more contribution from the deconvolutions to get an accurate output color.

When a neural network gets trained it learns weights for the deconvolutions where it could carefully learn to write to the unevenly overlapping positions to balance the output. While balancing is a tricky act to achieve neural networks still struggle to learn to completely avoid these patterns which causes a significant restriction on the filters.

In practice I had encountered this problem when I built the Auto-encoder-decoder network. I tried to use the transpose convolutional layers for up-sampling the image in the decoder network. This created a lot of high frequency artifacts in my image as shown below.



Possible solutions:

One approach would be to use a kernel size that can be divide by the stride to avoid this overlapping issue. Though this technique helps, the neural networks still face into the problem of creating artifacts. Another approach is to separate the up-sampling and convolutional operation where the up-sampling is done using linear approaches such as nearest-neighbor interpolation or bi-linear interpolation and then adding a convolutional layer on top of it. This approach is called as resize-convolution where it implicitly weight ties in a way that discourages high-frequency artifacts.

Artifacts in Gradients:

Transpose convolutions are done on the backward pass when gradients are computed for a convolutional layer which causes checkerboard artifacts in the gradient. This could be due to the fact that some of the neurons, many times will get the gradient of their neighbors resulting in the model caring for some particular pixels in an image than the rest.

What I have learned:

This paper has taught me about the reasons to why artifacts are created in an image and the methods I could use to reduce/dampen the artifacts generated by neural network. While I was doing my auto-encoder-decoder assignment I was unaware on how to remove the artifacts from the image generated by the transpose convolutional layer. But now I have learned that a lot of factors come into consideration while training a model to get rid of these artifacts:

1. The kernel size should be divisible by the stride.
2. Use of resize-convolutions instead of the deconvolutions (transpose convolutions).
3. Optimization parameters (computation gradients).

While every neural network differs in its own way and with respect to the dataset at hand, these factors have to be considered when training a model for the generation of an output image without these artifacts.