Image Evaluation using Deep Learning

Michael Neuder, under the supervision of Dr. Michael Mozer

***Abstract*—Image evaluation is an important aspect of image analysis and computer vision. The goal of this project is to create a deep learning model that quantifies the quality of a lossy reconstruction in comparison to a reference image. Not only do we want to be able to measure the reconstruction similarity, but also accurately predict which of two reconstructions a human prefers as a better representation of the reference image. To do this we built a model that captures a current state of the art image evaluation metric (Multi-Scale Structural Similarity) in a large convolutional neural network. The next step which is currently in progress is to use human judgment data to further fine tune the network and try and capture more of the subtleties of human perception.**

I. Introduction

The process of image evaluation is undoubtedly important. Creating a metric that quantitatively describes the quality of an image reconstruction is especially helpful when exploring new areas of image compression. Having insight as to what humans find important in image quality can allow for intelligent choice in what information is important to preserve when making choices with the compression algorithm. It also could be useful in tasks regarding super resolution and image enhancement by providing a fast and efficient way of deciding which representation is preferred by humans.

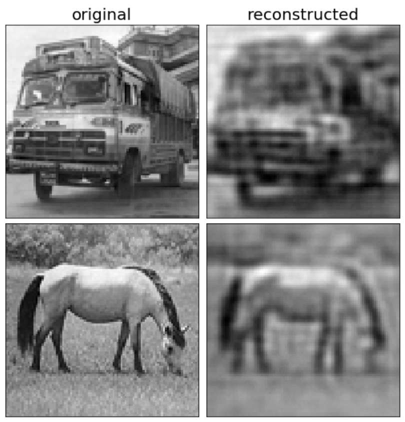

Fig. 1. Training data example. The left images are original 96x96 photos and the right images are lossy reconstructions.

Figure 1 provides a sample of the type of data we are using; given these image pairs, we want to be able to quantify how well the reconstruction represents the original. The current state of the art image evaluation metrics are typically analytic functions applied in a moving window to an image. These methods are completely deterministic and thus easy to compute and use. They typically do a good job in quantifying the quality of a reconstruction and as we will describe in the results and future work, they also perform well in a predictive setting. We want to see if using a deep learning approach, we can embed some of this value into a neural net, and then fine tune it with a subset of human preference data to outperform the analytic functions.

II. Methods

The image evaluation metrics that this work is grounded in are Structural Similarity (SSIM) [1] and a resolution based varient of this calculation called Multi-Scale Structural Similarity (MS-SSIM) [2]. These metrics are based on three main components of an image: structure, contrast, and luminance, and use them in conjunction to evaluate the quality of an image reconstruction. The scope of this paper is the implementation of these metrics in convolutional neural nets, and the step forward that is in progress is incorporating the human judgment data into the networks.

A. Structural Similarity

1) Method: The first step to take was to create a network that imitated the process of structural similarity. Structural similarity calculates statistics on local patches of an image and returns a pixel by pixel score for the similarity between two images. Starting at the top left of an image, the calculation grabs 11x11 square patches of both the original and the reconstructed image. These windows are flattened and statistics are calculated over them to return a single value. This value is then placed into the output map at the location of the center of the 11x11 patches. Figure 2 visually represents this process.

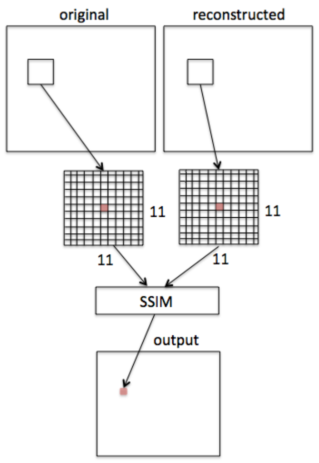

Fig. 2. The process of calculating the pixel wise SSIM between two images.

Below are the functional forms of the luminance, contrast, and structure that are calculated over the signals. Let x be the set of 121, (11 x 11 flattened), original pixels, and y be the corresponding set of reconstructed pixels.

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

The values C_1, C_2, C_3 are present to avoid issues of undefined functions in any case where the denominator could be zero. These are small constants and the literature [1] [2] uses the values,

$$C_1 = (L \cdot k_1)^2, \quad C_2 = (L \cdot k_2)^2, \quad C_3 = \frac{C_2}{2}.$$

Where L is the range of the pixels (0-255 in our context), and $k_1 = 0.01, \quad k_2 = 0.03$. Figure 3 shows an example of the resulting SSIM map generated from these calculations.

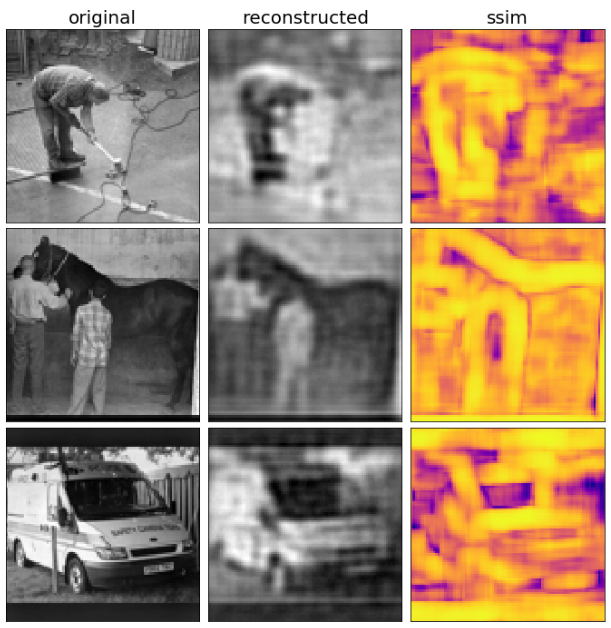

Fig. 3. A sample of the SSIM calculation. Left two columns are original image and reconstruction respectively, and right column is the calculated pixel wise SSIM map.

2) Data: The above is an example of the data that was used to train the SSIM network. We used four inputs into a convolution network; original image, reconstructed image, original image squared (pixel wise), reconstructed image squared (pixel wise). The output was the resulting pixel wise SSIM map.

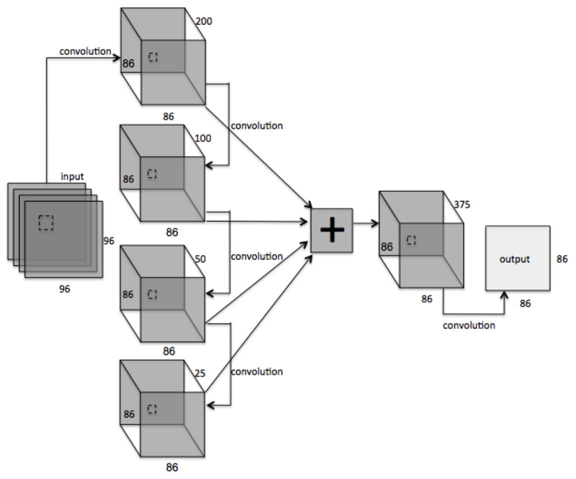

Fig. 4. The network architecture for the SSIM net. Dimensions and operations are labeled.

3) Architecture: The network architecture we used was similar to a standard convolutional neural network, but without any pooling layers. The network had four layers with 200, 100, 50, 25 filters for the respective layers. The first layer had 11x11 filters (to match the SSIM calculation), and all further layers had 1x1 filters to reduce the amount of learned parameters. In order to retain the information from each layer (primarily the first layer because it was the only one with 11x11 filters), we had the output layer convolve the sum of the output from each other layer. To do this we simply concatenated the outputs from each layer and fed that into the last layer which gave it dimension $200 + 100 + 50 + 25 = 375$. Each of the convolutional layers used no zero padding so that lead to a dimension reduction from the input layer to layer one. Because the filters were 11 x 11 the original and reconstructed image dimension was reduced to $96 - 11 + 1 = 86$. All other layers had no dimension reduction because filter size was 1x1. Figure 4 shows the network architecture. Each convolution operation used a $\tanh$ activation function except for the output layer which had linear activation.

4) Results: The network was trained on 500 examples and tested on 140 examples (this may seem small, but recall that there are many patches to train on per image. 500 images at $86 \cdot 86 = 7396$ patches per image is 3698000 examples). We trained for 2500 epochs with batch normalization and the order of the training data shuffled at each epoch. We used minibatches of size 4 images between weight updates. Figure 5 (following page) shows the error rate for both training and test set as a function of the epoch. The error rate is normalized by dividing by the variance of the training and testing SSIM maps, which means 100% error corresponds to no learning and 0% error corresponds to perfect accuracy. The fully trained network was averaging about 5% error on both the training and the testing data. This was a good start because it confirmed that the network could learn the structural similarity measures at a high dimension.

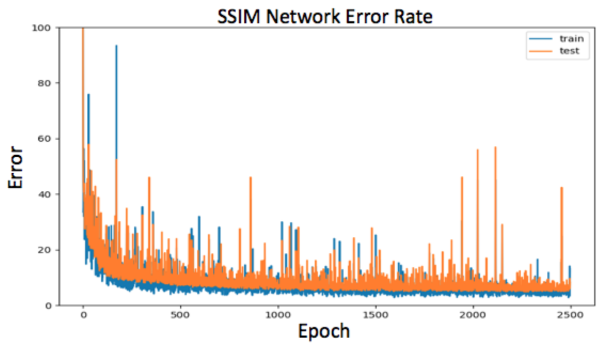

Fig. 5. Error rates for test and train data on the SSIM network. Ending error of $\approx 5\%$ of total variance in the data.

B. *Multi-Scale Structural Similarity*

It was a good sign to get the SSIM network training and performing well. But the true metric we were interested in was the MS-SSIM. The next step was to try and build this into a deep neural network.

1) Method: The calculation of MS-SSIM is quite similar to that of SSIM. The only difference is that the original and reconstructed images are downsampled and the statistics of these reduced images contribute to the calculation as well. Using the same analytic formulas for luminance, constrast, and structure, MS-SSIM is calculated as follows [2]

Step 1: Contrast and Similarity are calculated and averaged for the full dimension image. These values are then multiplied together.
Step 2: Images are down sampled (using an average pooling method) by a factor of 2.
Step 3: Repeat steps 1-2 for as many scales as desired.
Step 4: Calculate and average Luminance for the lowest resolution of the data set.
Step 5: Multiply all the levels of contrast and structure along with the average luminance of the lowest level.

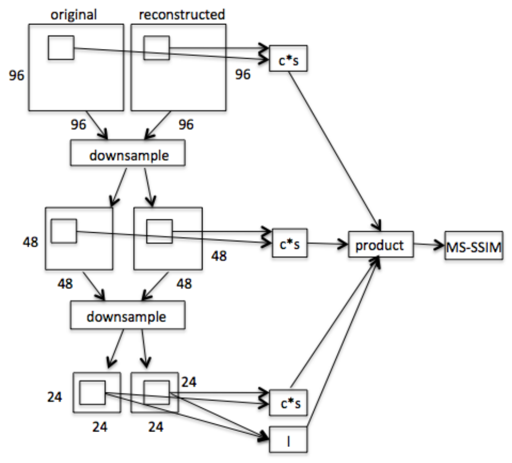

Fig. 6. Process of calculation of MS-SSIM.

Figure 6 shows how the MS-SSIM calculation was done for our images. The originals were 96 x 96 so we chose to do a two level downsample which left the lowest resolution images at 24 x 24.

2) Data: Figure 7 shows a sample training data example with both levels of downsampling and the contrast and structure calculated at each resolution image. Note that we aren't down sampling the contrast and structure metrics, but rather again running the moving window across each and calculating statistics over that moving window. Again there is no zero padding, so the resulting structure and contrast maps have dimension $96 = 11 + 1 = 86$, $48 - 11 + 1 = 38$, and $24 - 11 + 1 = 14$ respectively.

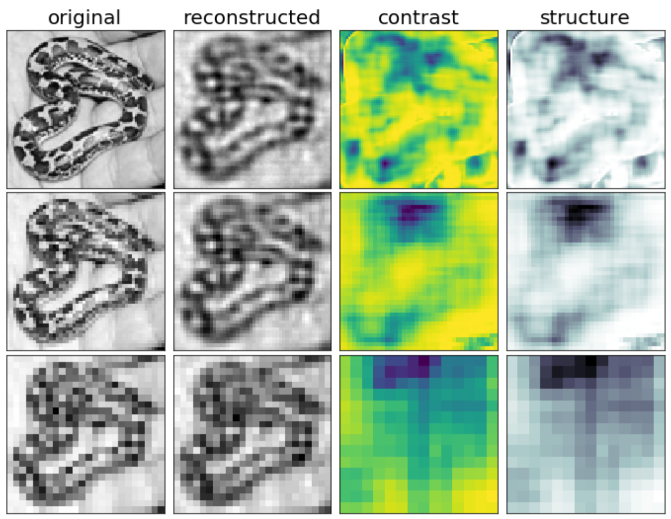

Fig. 7. Contrast and structure results at the three resolutions used in the MS-SSIM calculation.

The last part of the data to obtain was the luminance at the lowest resolution [2]. Recall that the average of the contrast and structure maps at each resolution are multiplied with the average luminance at the lowest resolution to obtain the last MS-SSIM score. Figure 8 (following page) shows some examples of the luminance pixel by pixel score for some low resolution images. Now we have all the data needed to do the MS-SSIM calculation. This data is also what is used to train the constituent pieces of the MS-SSIM network, where the target is the total MS-SSIM score for the image pair.

3) Architecture: The architecture for the MS-SSIM net is essentially a number of SSIM nets in parallel. Because it would be impossible for the net to learn the task from just the single MS-SSIM score presented at the end, we must train the net using the component parts of the MS-SSIM calculation. We chose to train a network to do $contrast{*}structure$ and another network to do $luminance$. These networks with then be placed in parallel, one for each resolution, and take as input the same features as the SSIM net (original image, reconstructed image, original image squared, reconstructed image squared). Then we can simply take the average output from each net and multiply them together to get the MS-SSIM score. This can then be

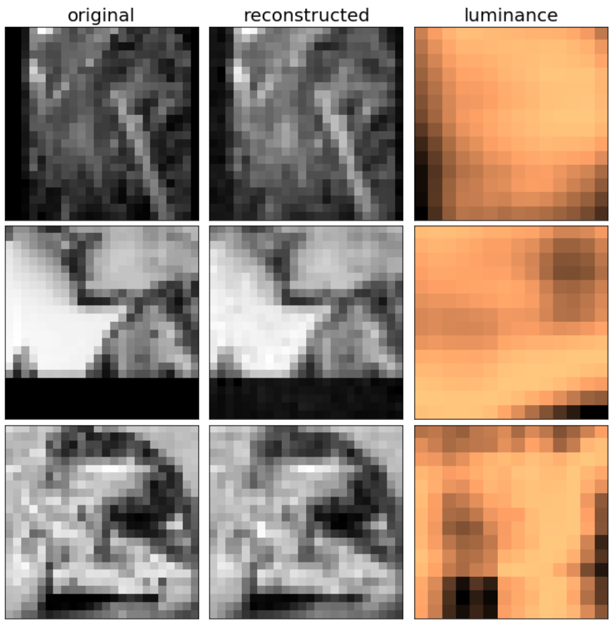

Fig. 8. Luminance calculation on the lowest resolution version of the image.

compared the the analytic result. Figure 9 shows what the MS-SSIM net looks like in its entirety. Recall that each of the nets (4 total - 3 for $contrast * structure$ and one for $luminance$), has the same architecture as the full SSIM net.

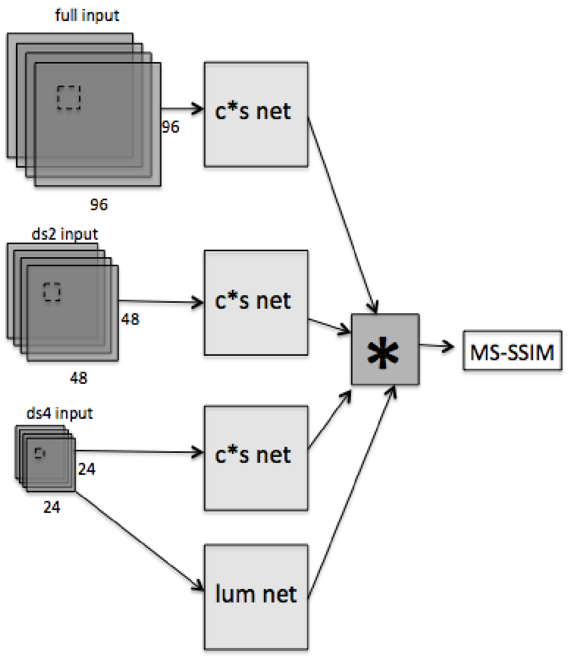

Fig. 9. MS-SSIM net in all entirety. Ds2 and Ds4 refer to down sample by a factor of 2 and factor of 4 respectively.

4) Results: The $contrast * structure$ and $luminance$ nets were trained in the same manner as the SSIM net. Figure 10 shows the error rates as a function of epoch ($luminance$[1] is the top plot and $contrast * structure$ on bottom).

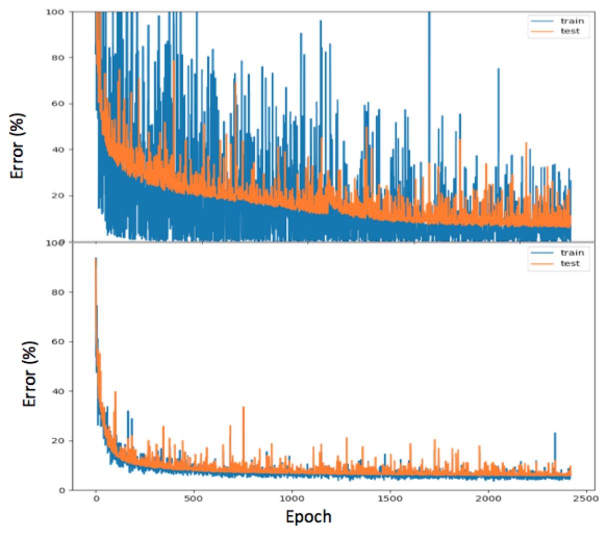

Fig. 10. Error rates for parts of MS-SSIM net. $Luminance$ net is on the top and $contrast * structure$ is on the bottom.

Both of these error rates are normalized by dividing by the variance of the target. Both networks end up in the 10% error range, though it is clear that the luminance net was much noisier in the learning process. We believe this is because the luminance calculation can be very constant over an entire image, so for every patch of that image the network is just learning to predict the constant, and then it suffers when it hits the next image. This is speculation though, and we are not too sure what is causing this. Regardless, it is comforting that the error rates achieved are so low. Now all the pieces of the MS-SSIM net are in place and we are ready to use the human perception data.

III. Next Steps

The next steps of this project are clear and exciting. Now that we have the MS-SSIM net in place, it is time to incorporate the human judgment data to see if we can perform better than the analytic functions, the goal of this entire venture. The human judgment data comes in this form: a triplet of images, one original and two lossy reconstructions, and the selection that the human made as to which was a better representation of the original image. Figure 11 (next page) shows a sample of the human data.

The goal is to beat out regular SSIM and MS-SSIM. We tested the predictive accuracy of SSIM on the data set and the image with the higher average SSIM score predicted the human

[1]Note that the luminance net was actually trained on full resolution images instead of reduced dimension images. This is done because at such a low dimension there are not enough windows to train accurately. When part of the MS-SSIM net it will be used on the low resolution version still.

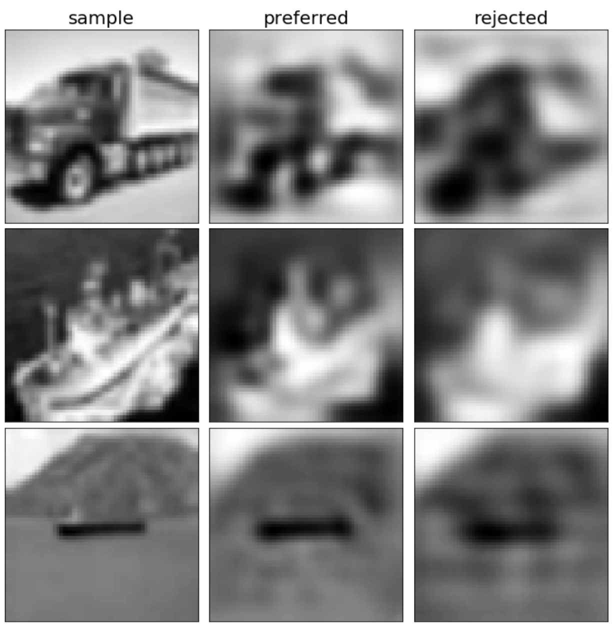

Fig. 11. Human perception data. The images are low resolution, but high enough to distinguish. The left image is the original, the middle is the one that was chosen to be a better representation and the left is the rejected image.

choice with 85% accuracy, which is pretty good. But we are hoping that the fine tuned MS-SSIM net with fine tuning using a subset of the human data for training will be able to match this score and beat it.

IV. Conclusion

The purpose of this work it to provide a model which given input images can predict how well a human like a reconstruction or which of two they prefer. This was inspired by the fact that most of the state of the art image evaluation metrics are simply analytic functions that perhaps don't perfectly predict human perception. Using deep learning to extend these models to incorporate human judgments is how we are trying to close the gap on prediction of human preference. Stay tuned for more results coming soon, and thanks for reading.

Acknowledgment

The author would like to thank Dr. Mike Mozer for his constant support and guidance.

References

[1] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, Image quality assessment: From error measurement to structural similarity, *IEEE Trans. Image Processing*, vol. 13., Jan 2004.

[2] Z. Wang, E. P. Simoncelli, A. C. Bovik, Multi-Scale Structural Similarity for Image Quality Assessment, *Proceedings of Signals, Systems, and Computers*, 2:1398-1402, Nov 2003.