**Using Generative Methods for Acoustic Data Preprocessing and Dereverberation**

**This project was submitted as part of the course:**

**"Deep Generative Models"**

**Authors: Moshe Beutel (037580792)**

**Idan Cohen (208749481)**

**Theoretical Background**

In this project we aim to use generative learning methods to approach the task of acoustic dereverberation. The dereverberation we wish to achieve is not of audio recordings, but of an input feature called Relative-Transfer-Function (RTF). This input feature is of great use for Idan’s thesis as it can be used via methods of dimensionality reduction to describe the geometrical configuration of our problem, and explicitly, learn the geometrical shape of the room based on acoustical measurements only.

**Our Input Feature - The Relative Transfer Function**

Assume we have a signal measured in two microphones: תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

Where is the source signal, are the acoustic impulse responses relating the source and each of the microphones, and are the noise signals, independent of the source.

In the case where the measurement noise is relatively low, we can estimate the RTF of the two microphones as follows:

Where are the DFT of the acoustic impulse responses, is the cross spectral density of the microphones, and is the auto-spectral density of the reference mic - .

This input feature, which is defined as the quotient of acoustic transfer functions, is widely used in acoustic research, as it captures the entire propagation of each of the signals in space, including each of their reverberations.

If we speak of the most simple case where there is no reverberation, then the transfer functions are .  
 Therefore the RTF is simply a complex exponential with linear slope:

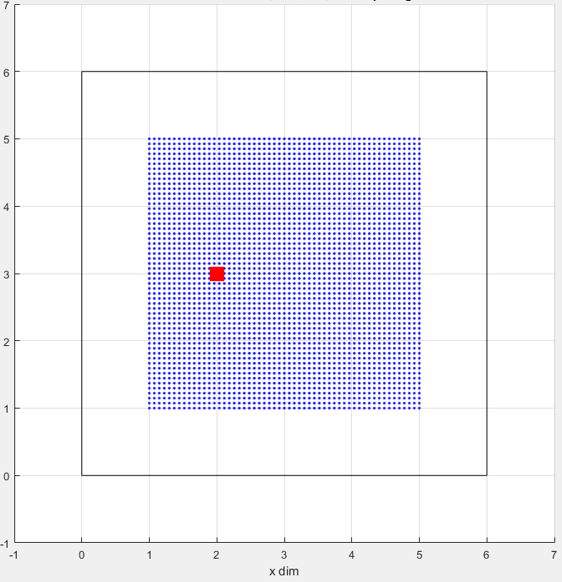
In this case, we use the phase of the **RTF** as input, which is simply a linear line.  
 We work with frequency resolution of , and get a result of linear line of size frequency bins.

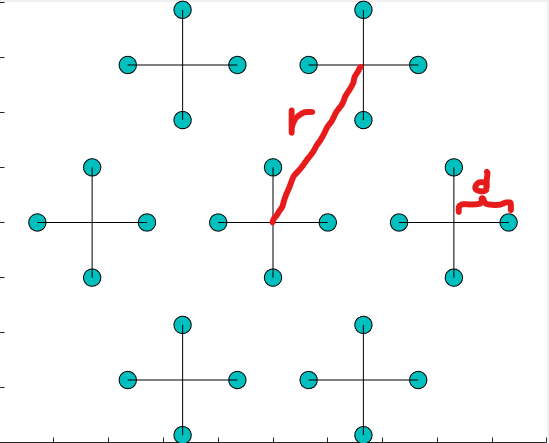
**Our Problem Setting**

In order to use the RTF-Phase features to reconstruct the space, we use the [Local Conformal Autoencoder](https://arxiv.org/abs/2004.07234) model which was suggested by Peterfreund E. and Lindenbaum O.

This method of work requires us to sample our space in a structure called “burst” – a group of adjacent locations. In our case, our bursts will have the configuration shown in the left image below. Each burst is constructed of 7 points, located in a circular shape. At each of these points we will sample the horizontal-RTF-Phase, and the vertical- RTF-Phase, thus we have two pairs of opposing microphones that look like crosses. We can control the parameters ‘r’ and ‘d’ of this configuration, but through this project they will be constant,  
d = 1cm, r = 3.3cm.

The sampling is carried out in a square grid of size [4m, 4m] with a resolution of 55 points in x and y dimensions, inside a square room of dimensions [6m, 6m, 2.4m], resulting in 3025 bursts.  
A speech source is located at the point [2, 3, 1.2], which is 1 meter above the plane on which  
the sampling grid is located.  
Note that the data is synthetic, and generated using the [gpuRIR-generator](https://github.com/DavidDiazGuerra/gpuRIR).







Looking at a single burst, our data samples look as follows:



We have 7 lines for 7 locations in the burst, where each line is a concatenation of 2 linear lines, as there are two RTF-Phases at each point – the horizontal and vertical.  
Therefore, a dataset is made of samples of shape [7, 2\*129] = [7, 258], and we have 3025 such samples.

Using the Local-Conformal-Autoencoder mentioned above, we can reduce the dimension of the 258-dimensional data into a 2-dimensional embedding that represents the problem,  
and we currently have the ability to perfectly reconstruct the geometry of the room in the **non – reverberated** case:

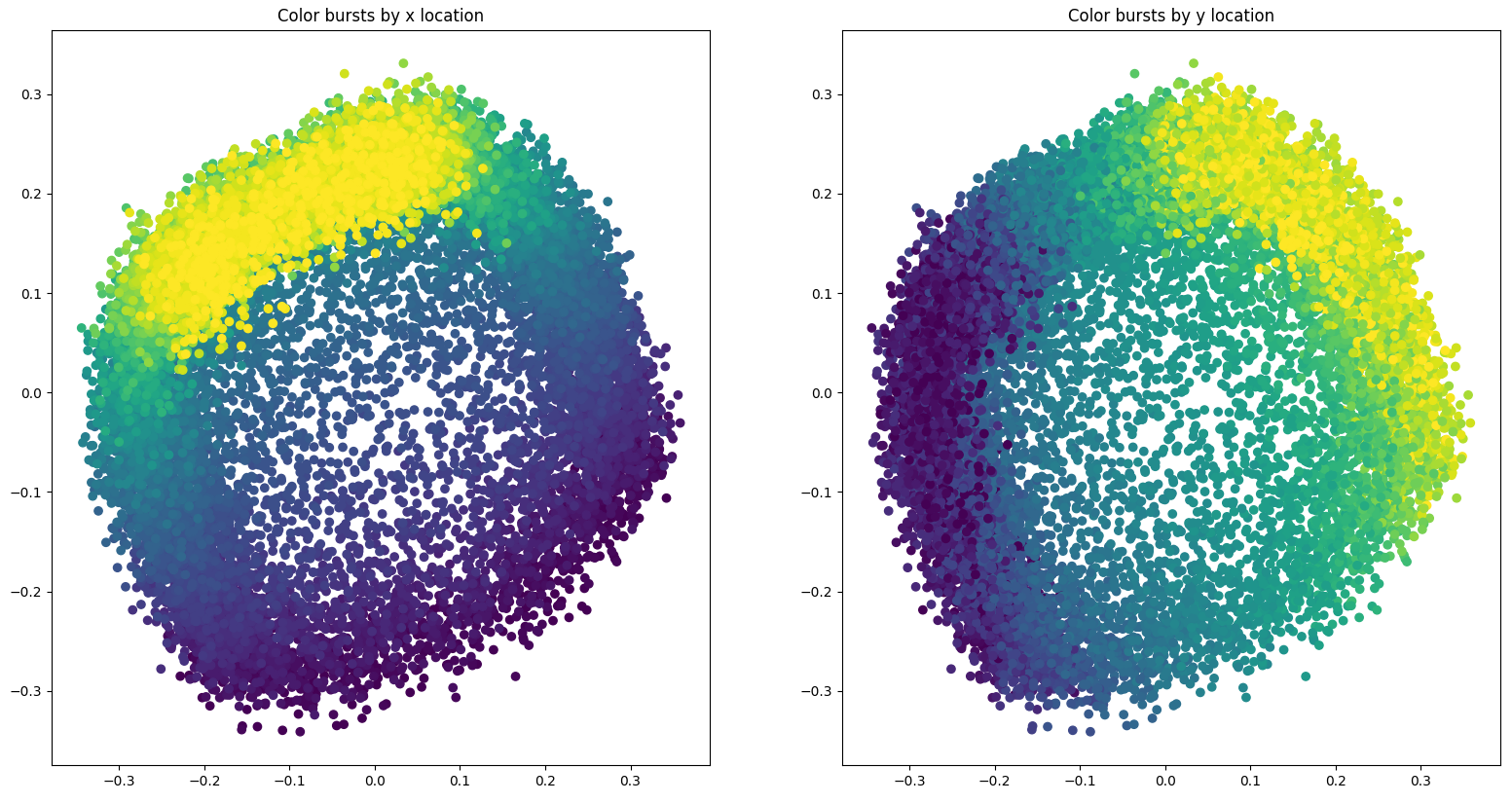


Notice that this is an unsupervised problem, but we can color the embedding after the dimension reduction as we know the true labels. We color the labels both by ‘x’ labels and ‘y’ labels, as it helps to understand the model’s performance.

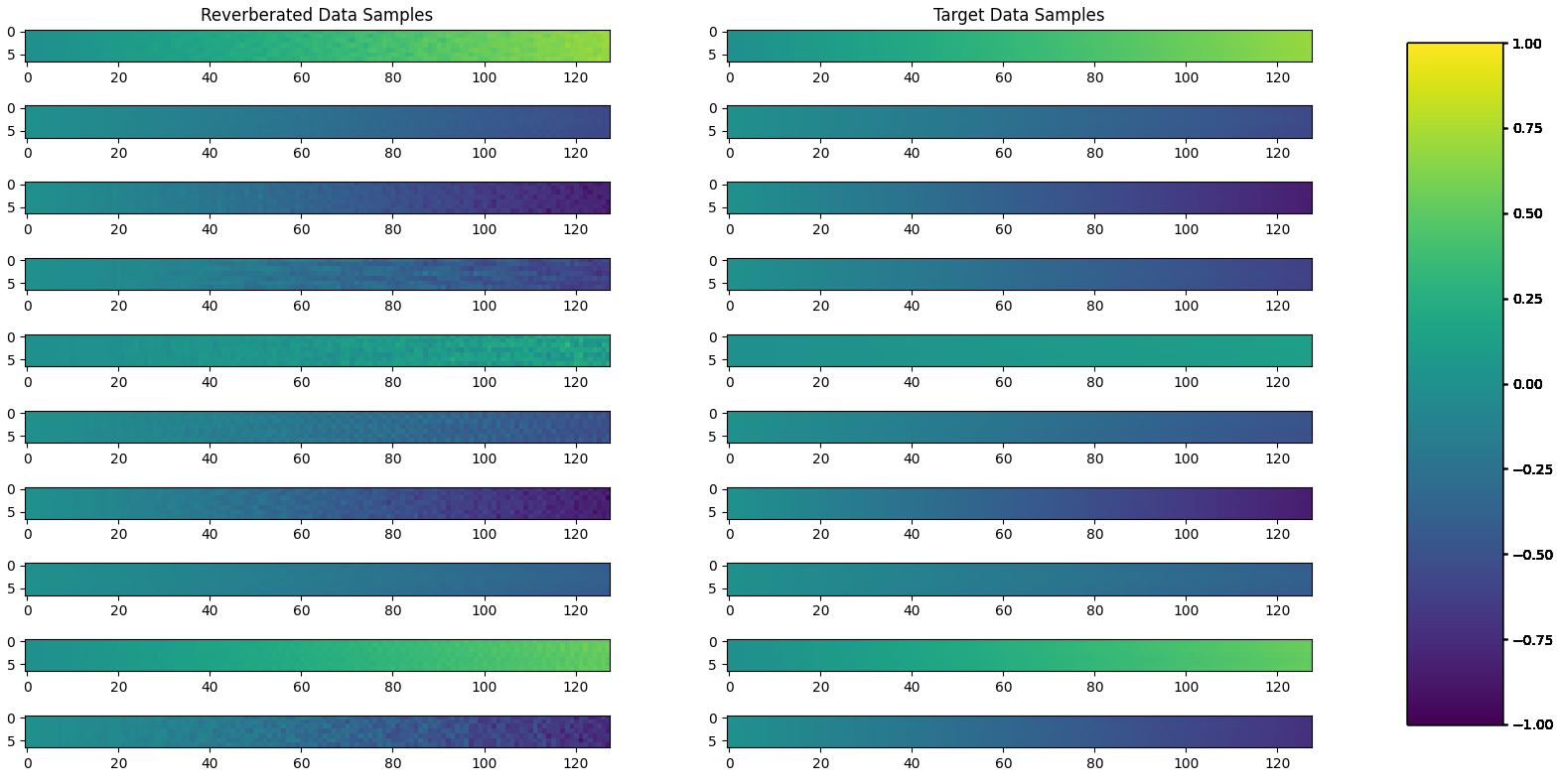
The next step is to add reverberation. This time the RTF-Phase is no longer linear, as more propagations of the signal arrive at the microphones at different times.  
If we look at the horizontal RTF-Phase of some burst, we get:  
If we compare it to the ground-true RTF-Phase -



We can see that the reverberated sample is very similar but has a pattern that was introduced due to the reverberations.   
As we try to reconstruct using reverberated data, with a reverberation time of ,  
the model is struggling to capture the true nature of the problem:



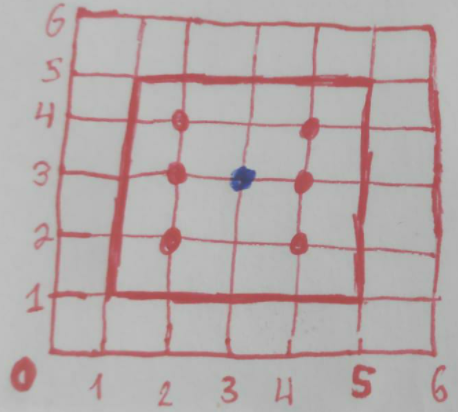
The reason the model doesn’t work for reverberated data is that the mathematical theory of Local-Conformal-Autoencoder requires that the input data is smooth, and indeed it was experimentally seen to be very sensitive to non-smooth data. Reverberation, as can be seen in the images, introduces grainy patterns that completely ruin the model’s learning ability.

We would like to use generative models to clean reverberated data, as shown below:

The idea is to use generative models and to treat our data as images. It is true that a more standard way to approach the problem would be to use standard methods of signal processing to achieve dereverberation, but we thought that it was a nice idea for a project in generative models. A key idea in the work is that a burst of samples is made of points that were sampled in close proximity and should be very similar, thus we can use correlative information between the lines to perform smoothing of the reverberation pattern.

**Methodology**

We create a few datasets – each one with a source located in a different location in the room (red points in the image below). That way we would create more variability in the reverberation patterns, to help the model’s generalization ability. Finally, we will test the model’s performance over a dataset whose source is located in the middle of the room (blue point in the image below).



Having 6 training datasets made of 3025 bursts, each one separated into 2 RTF-Phases (vertical and horizontal), our training data consists of 36,300 images of size [7,129].

We will want to achieve the best spatial reconstruction using the Local Conformal Autoencoder model.

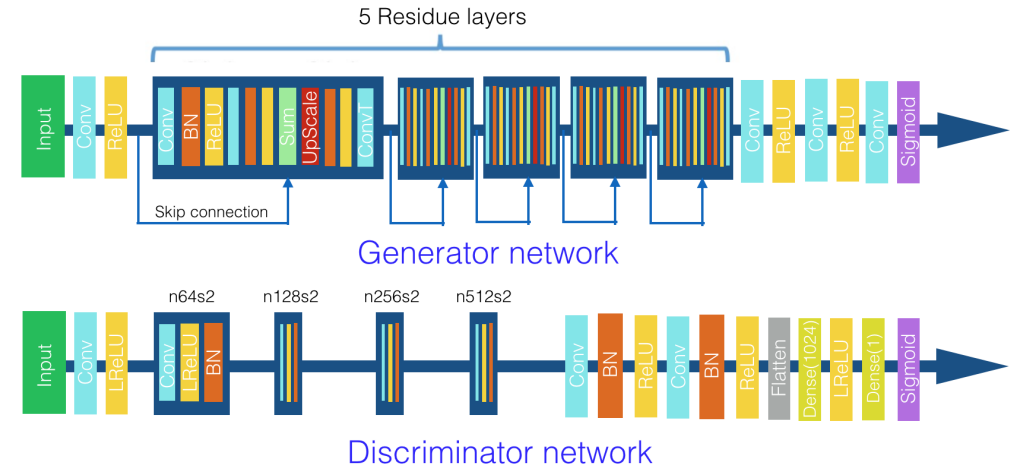
We will implement and test two generative models that could be helpful in the task of generative image denoising:

1. [DCGANs for image super-resolution, denoising and debluring](http://stanford.edu/class/ee367/Winter2017/yan_wang_ee367_win17_report.pdf) – A straightforward residual-deep convolutional network with content loss and adversarial loss.
2. [Pix2Pix](https://arxiv.org/abs/1611.07004) – We implement the well-known architecture in a way that allows us to perform denoising.

The model's input will be only half of the data, either the horizontal RTF-Phase, or the vertical RTF-Phase of a single burst, an image of size [7,128] and it should return a dereverberated image of the same size.  
As the pixels in the image represent phase, they take values in the range ,  
which will be normalized to the range by simple division.

**Denoising DCGAN**

We implement the following architecture shown in “DCGANs for image super-resolution, denoising and deblurring”,Qiaojing Yan,Wei Wang :



Note that we did not include the upscaling layers, as they were used in the article to perform super resolution. Note that the generator takes a [7, 128] image as input and outputs an image of the same size. The final ‘sigmoid’ layer at the generator was replaced by a ‘Tanh’ layer as our range of values is .

The Generator loss function we minimize:

is composed of the content loss:

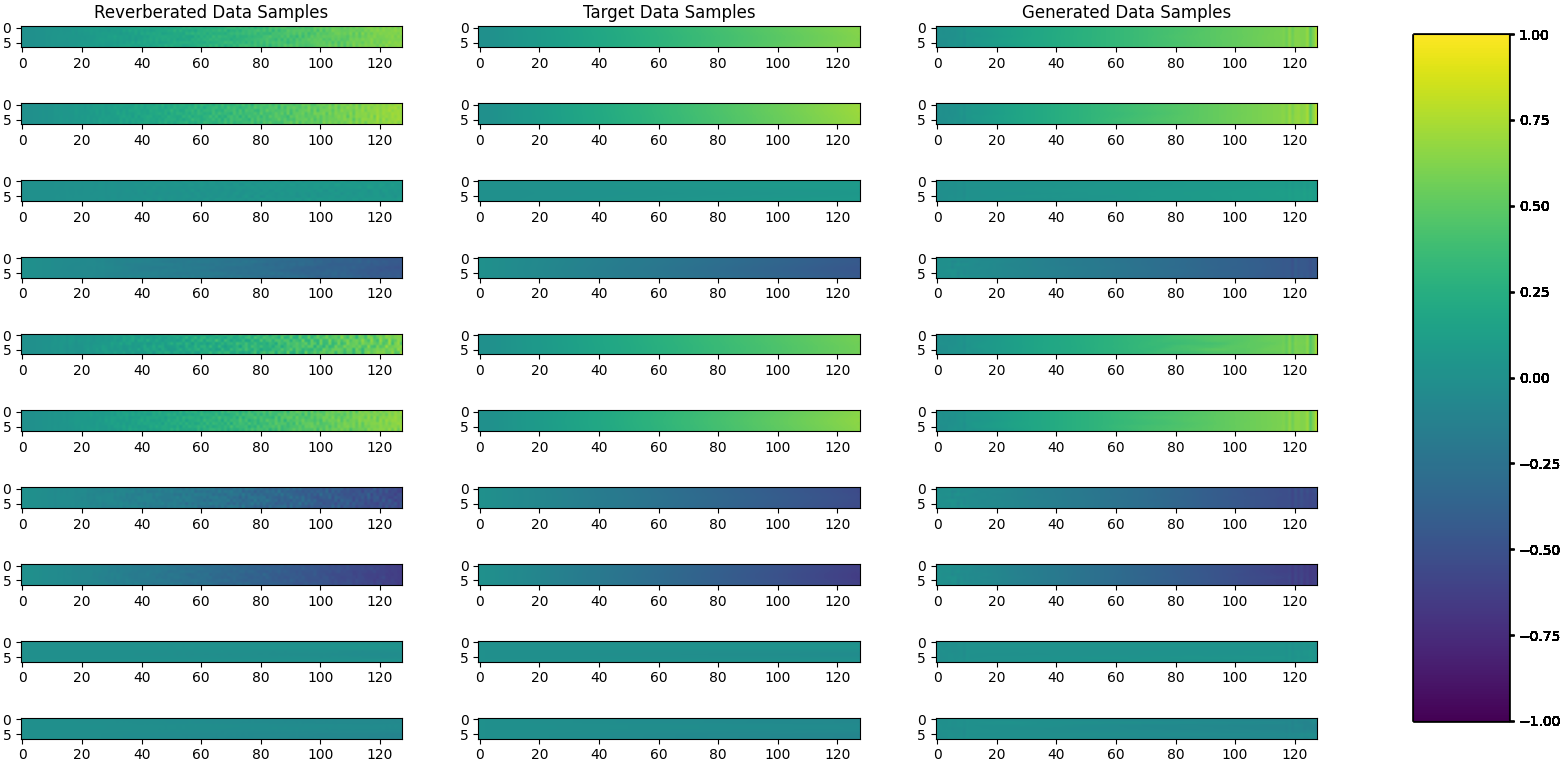
between a reverberated image and its clean target,

and by the regular adversarial loss:

The discriminator loss function is the adversarial loss:

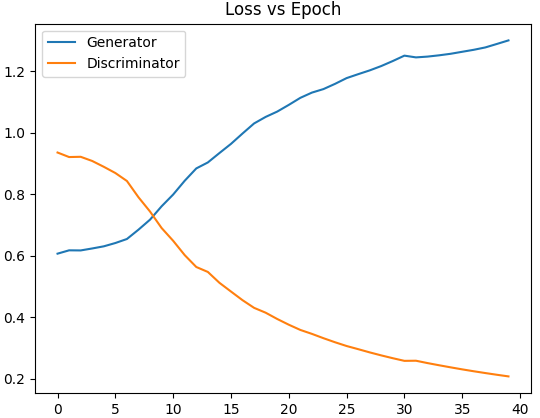
A key point in our implementation was understanding that our data is not a normal image – there is no importance to the order of the rows in a burst.  
Any permutation of the row order represents the same data sample, and thus if we take a regular [3,3] kernel for the convolution layers, we give importance to the order in which rows of the data are ordered in the input.  
Our solution was to work with kernels of size [7,3], where we first pad the row dimension using circular padding of 6 rows. That way, filters that are learned have to see the data in all of the rows, without influence of local relations between adjacent rows.  
We use this workflow both in the denoising-DCGAN and in the Pix2Pix models.

**Denoising DCGAN - Results**

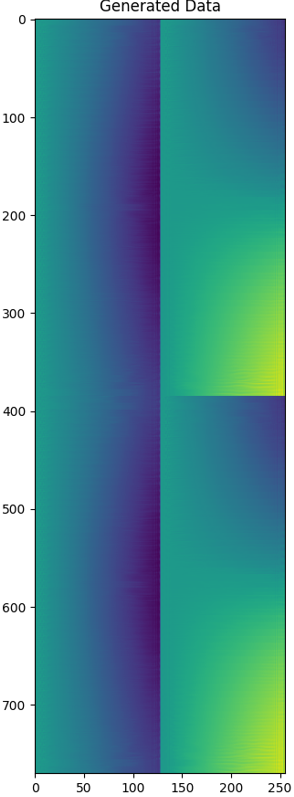
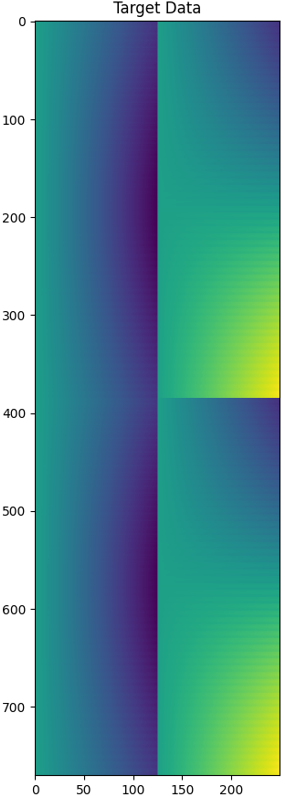
Our best model was achieved upon reaching epoch 40, we get a relative good denoising as shown below:

The denoising seems pretty good if we disregard some artifacts at the right side of the generated samples that weren’t fully removed. We were unable to remove these artifacts using the Denoising-DCGAN, but were able to remove them using the Pix2Pix model.

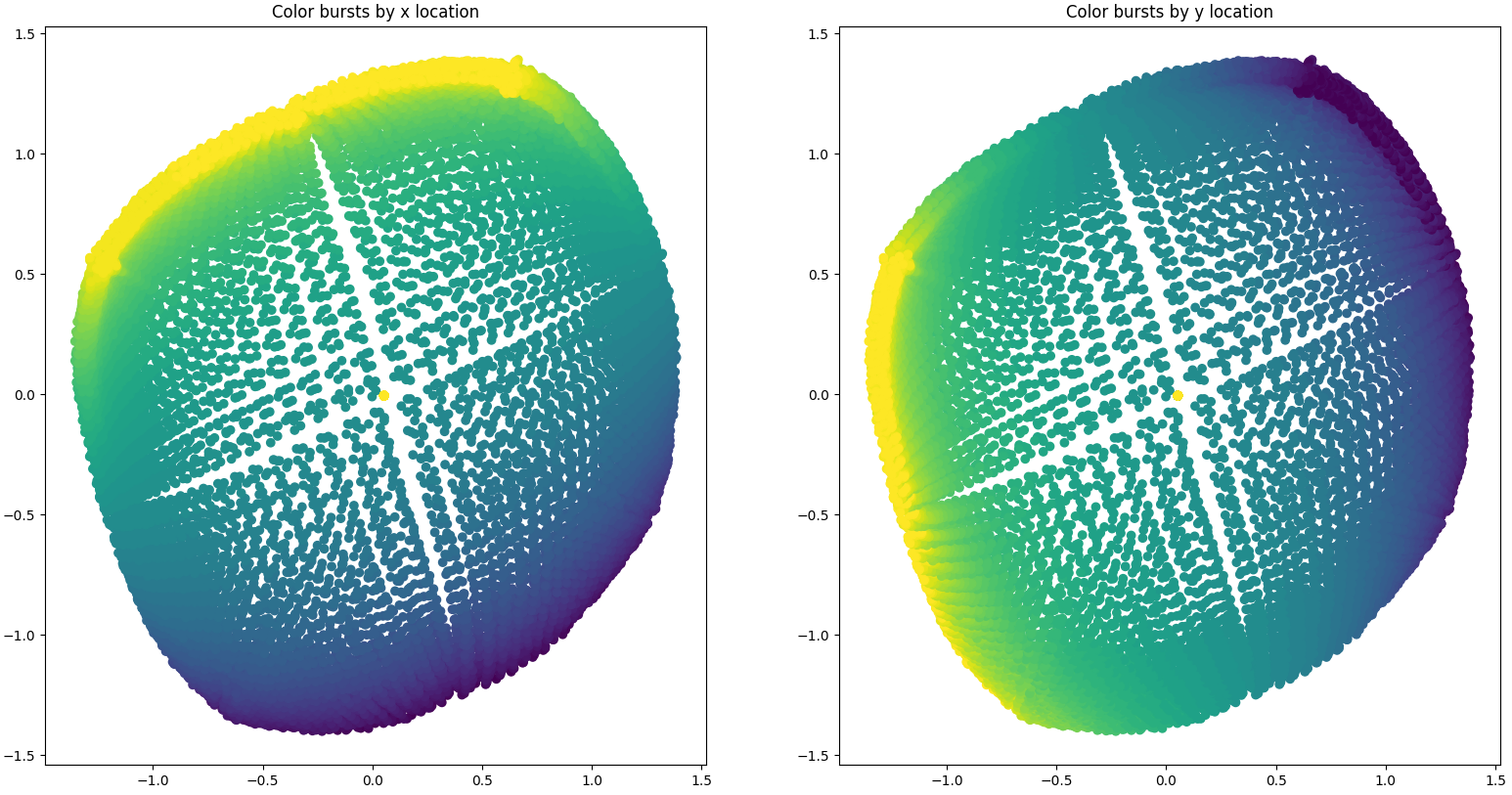
The learning curve:



To further visualize the denoising, we show the generated data vs the target data of the first two columns in the sampling grid (concatenated vertically)



The denoising seems good, but now we need to check if the Local-Conformal-Autoencoder can reconstruct the space:

Note that this is the best achieved result using the Denoising-DCGAN model.  
 

We can see that training the model over the generated data is somewhat better than before denoising. The reconstruction model, however, mainly captures the variability in the middle of the data – where the gradients are much stronger. That property of the gradients being strong in the middle, close to the speech source, is characteristic also of our original data, but as it seems, it didn’t allow the generative denoising model to capture the nature of the data.

As the distribution that characterizes the data is different in space, we get a generative model that can’t capture the little variability in the edges.

**Pix2Pix**

In “*Image-to-Image Translation with Conditional Adversarial Networks*”, Phillip Isola,Jun-Yan Zhu, Tinghui Zhou and Alexei A. Efros describe a conditional GAN that learns a mapping from an input image to an output image and a loss function to train this mapping. The conditioning on the mapping is not application specific (inpainting, reconstruction from edges etc.) and is only learned from the training data pairs .

The objective function is composed of a conditional GAN loss and L1 loss:

where:

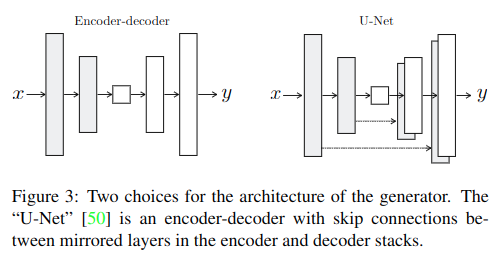
and

In practice, the generator maximizes instead of

minimizing and the discriminator loss is divided to slow down its learning.

**Pix2Pix Architecture**

***U-Net based Generator***



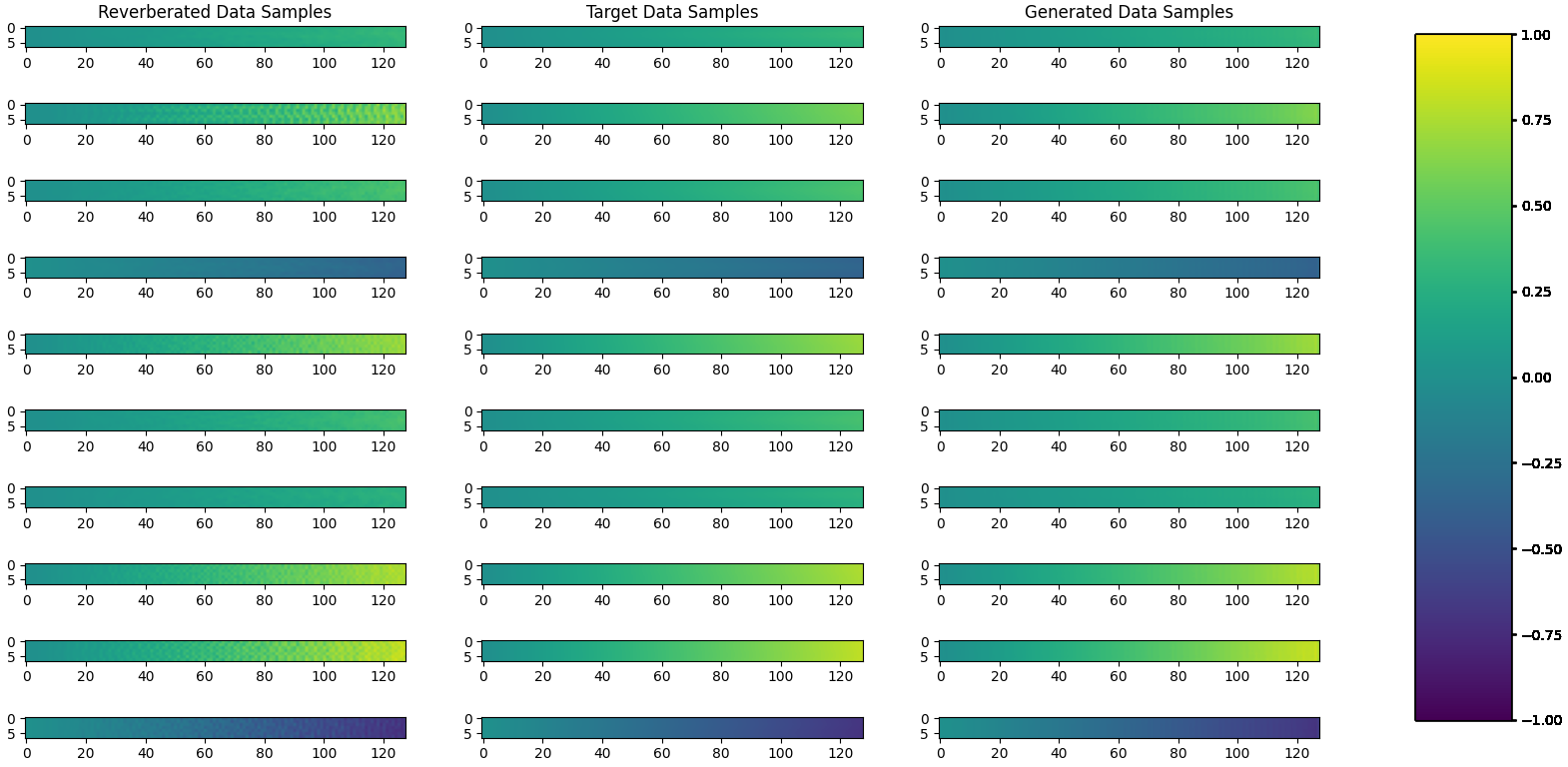
Many image-to-image translation uses Encoder-Decoder based architectures for the generator. This kind of architecture enables a detailed image as an input and a detailed image as an output while the bottleneck encapsulates enough data to generate the output image.

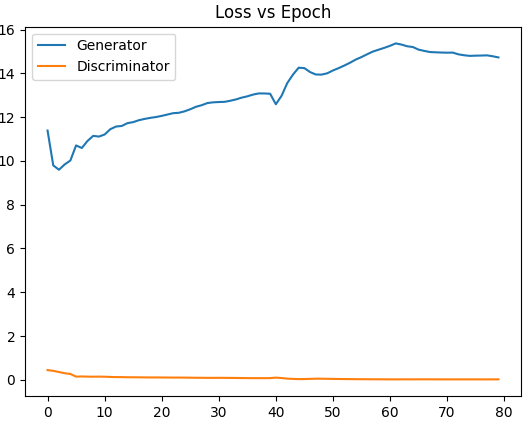
But in many image-to-image scenarios the input and output image share data that need not be transformed at all and we want to allow this kind of data to skip the bottleneck. For this end a U-Net architecture is used where skip connections between equal level layers between down-sample and up-sample enable unchanged data to pass through.

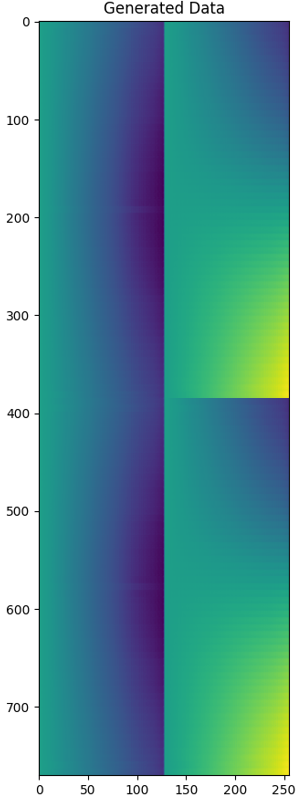
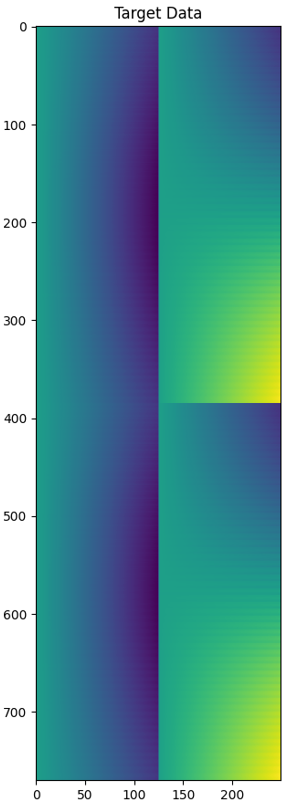
***PatchGAN Classifier Discriminator***

The discriminator used in pix2pix is called PatchGAN because it penalizes only at patch level – high level true\fake judgment. This is done using the L1 term that forces low-frequency correctness. The discriminator looks at N\*N patches and decides for each patch whether it is true or fake. N may be much smaller than image size. As a result PatchGAN models the image as a Markov Random Field (MRF) . The smaller N is PatchGAN runs faster and can be applied to large images.

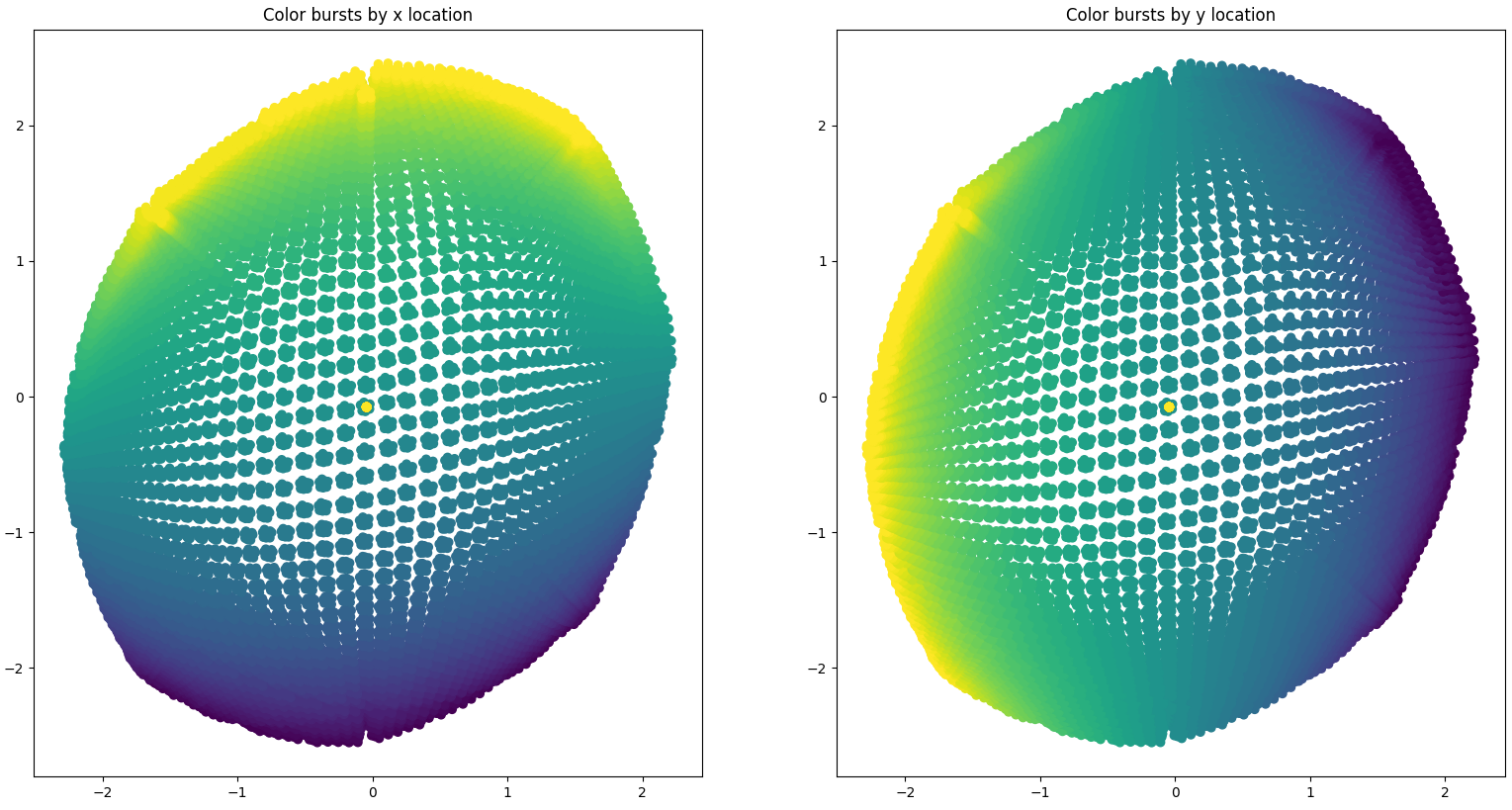
**Pix2Pix – Results**

We tried training a bunch of Pix2Pix models, mainly with different values of the “lambda\_recon” parameters that are in charge of weighting the adversarial loss and the reconstruction loss, and found no major difference between the results.  
Using the Pix2Pix model, the denoising looks better. We trained it for 80 epochs to reach the following results:

The learning curve:

Same as before we visualize the denoising by showing the generated data vs the target data of the first two columns in the sampling grid (concatenated vertically)

This time as well, the denoising looks good, even better than using the Denoising-DCGAN.  
We can barely see the difference between generated data and the target data.  
We will need, however, to see the spatial reconstruction over the generated data.



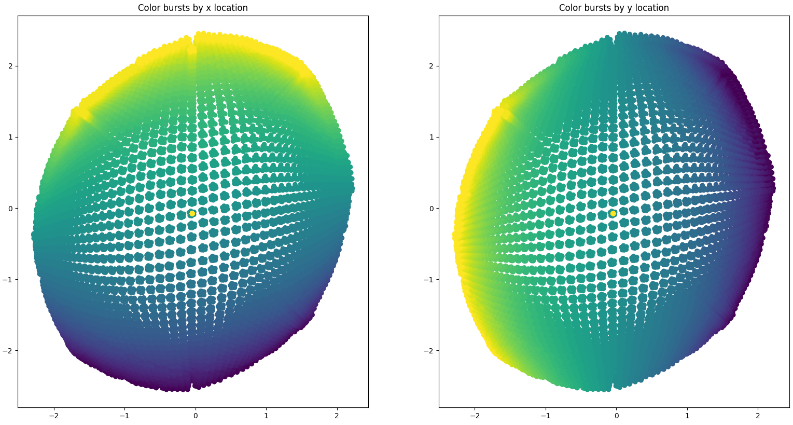
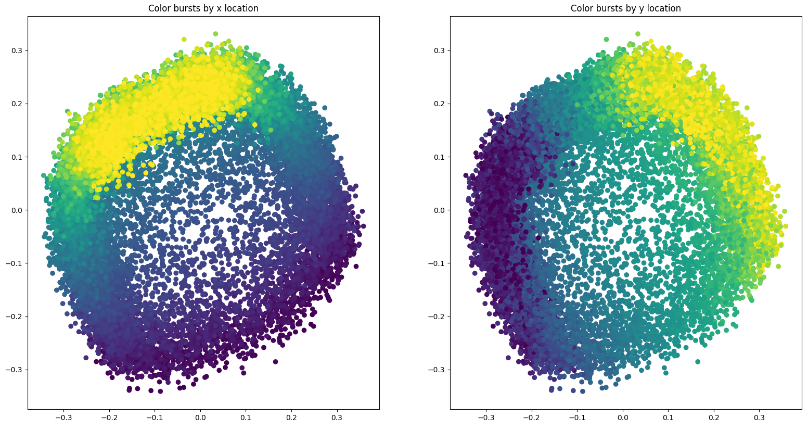
Same as before, the model is able to get much better spatial reconstruction using the cleaned data, than it could get with the reverberated data.  
We don’t see a major difference in the result of the Pix2Pix model compared to the Denoising-DCGAN model.

It appears here as well, that due to the nature of the data being very variant in the middle while having very subtle changes in the edges, we can’t build a generative model that catches this behavior perfectly.

**Conclusions**

In this project we used two generative models to ‘denoise’ patterns of reverberation from RTF-Phase input features, in hope to improve our ability to reconstruct the spatial configuration in which acoustic measurements were conducted.

We did manage to improve results by a significant factor using both Denoising-DCGAN, and Pix2Pix model, but did not manage to reach the desired result of perfect spatial reconstruction.



As explained before, we believe that the inability of the generative model to clean the data in the edges as well as in the middle is due to the model being unable to capture the nature of data that has different variability in different locations.  
Being close to the speech source (around the middle) is tied with bursts (image samples) that have rows with slopes that are more variable, whereas being far from the source (in the edges) is tied with bursts whose rows’ slopes are almost constant.  
The Local-Conformal-Autoencoder that reconstructs the 2D embedding needs to be able to perfectly distinguish the relations between the slopes of different rows in the data, but the generative model generates cleaned data that is much less distinguishable in the edges.

It would be more standard to approach this problem using signal processing techniques, but we had a great time figuring out how we could use generative models to achieve denoising,  
and get to implement and train them!