Investigating Compressive Sensing for Multimedia

William Fu

1 Introduction

1.1 Motivation

With the rise of scalable and increasingly deep convolutional neural networks in 2010, there has also arisen an increased need for efficient video analytics systems. Currently, video streams are optimized through codecs such as H.265 and VP8 for high throughput and optimized for human perception. State of the art video analytic systems, such as NVidia's TensorRT and AWS Firehose, use codecs optimized for the human viewing experience. However, as modern analytics platforms leverage more and more video streams, the physical bandwidth limitations play a large role in the scalability of the system. As video sources become cheaper to manufacture, the bottleneck for aggregate video analytics becomes the amount of video state being sent through the network.[1]. Therefore, there is an increasing demand for video codecs that leverage the large resources of commercial servers to compensate for low-powered edge devices and low-bandwidth first-hop connections. Currently, state of the art video encoders such as H265 and VP8 save bandwidth through lossy codecs[1], but provide fast encoding and decoding routines. Recently, compressed sensing has emerged as a method for low-bandwidth data collection in applications where data collection is expensive. For example, compressed sensing has been applied to MRI machines where there are physical limitations on the sampling bandwidth[2]. Since it provides lossless transmission and fast encoding routines, but expensive decoding, compressed sensing seems promising as a new video compression system for increasingly powerful servers running video analytics.

1.2 Overview

I investigated the feasibility of compressive sensing to lower bandwidth constraints for streaming video analytics. First, I explored the basis pursuit reconstruction algorithm for a random sensing matrix, and compared the compression rate to current state of the art video codecs. Then, I explored the feasibility of data-driven approaches for learning the sensing matrix and analyzed the improvements that would provide. The code is available here.

2 Compressed Sensing with Basis Pursuit

2.1 Background

Compressed sensing involves recovering a sparse signal $\mathbf{x} \in \mathbb{R}^N$ by undersampling only $M < N$ measurements. The measurements $\mathbf{y} \in \mathbb{R}^M$ can be created by multiplying a sensing matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$, such that $\mathbf{y} = \mathbf{A}\mathbf{x}$. To retrieve the original sparse $\mathbf{x}$, we can use the method of basis pursuit introduced by Candes et al.[3]. Instead of using the intractible estimation problem with the L_0 norm, we will use the L_1 norm for recovering our original signal.

$$\hat{\mathbf{x}} = \text{argmin}_{\mathbf{x}} \|\mathbf{x}\|_1 \qquad \text{s.t. } \mathbf{A}\mathbf{x} = \mathbf{y}$$

For the purposes of analysis, we will assume that our true signal is distributed i.i.d from some distribution $x_i^0 \sim P_0(X;\mu)$, where μ is an arbitrary distribution. We also assume that our sensing matrix is distributed as a Gaussian.

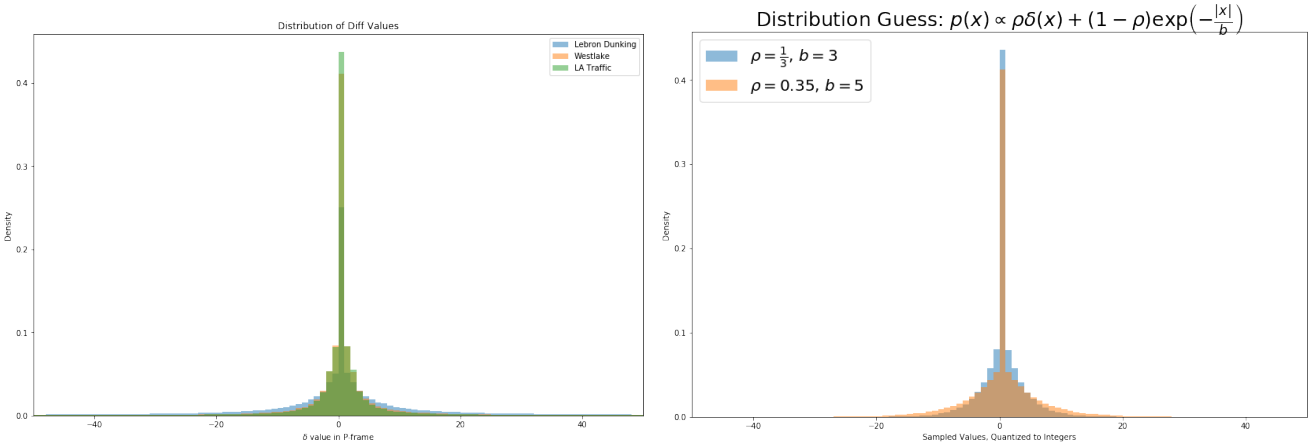
$$P_0(x_i^0) = (1-\rho)\delta(x_i^0) + \rho\mu \qquad\qquad A_{ij} \sim \mathcal{N}\left(0, \frac{1}{N}\right)$$

Then, we can use linear programming on the above optimization problem, which gives us $\hat{\mathbf{x}}$, our estimation for the original sparse signal $\mathbf{x}$.

[1] See the NVidia DeepStream talk for more insight on how much network bandwidth is required for real-time inferencing on multiple cameras. https://devblogs.nvidia.com/accelerate-video-analytics-deepstream-2/

2.2 Video Compression

Video has traditionally been stored through the difference between frames rather than frame-by-frame, since most part of the video do not change per new frame. In state of the art video codecs such as H.265, videos frames are typically classified into KeyFrames, P-Frames, and B-Frames. P-Frames are stored using the previous frame as reference; a over-simplified explanation would just be storing the differences between successive frames to encode the video. Since most of the video is encoded using P-Frames, I focused my analysis on the P-Frames. First, I explored the distribution of P-Frame values for a few natural videos. The distribution visibly follows a Laplacian distribution, which can be seen in Figure 1.



(a) Actual video delta distribution

(b) Simulated sparse Laplacian distribution

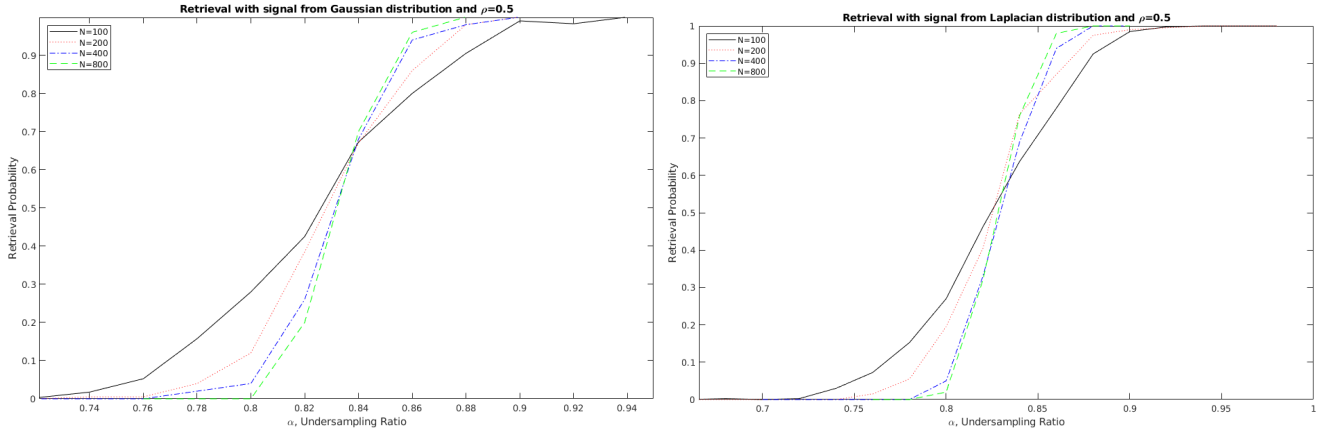
Figure 1. Distribution of deltas between video frames. The video used are all real-life shots taken from different settings. The video settings are a basketball game broadcast, a dashcam video, and a busy pedestrian street. 1000 frames in 1080p were taken from each video source to generate the histogram.

Therefore, we will analyze the basis pursuit compressed sensing problem assuming that the data is sparse and comes from a Laplacian distribution. I have tried to approach the problem with the replica method; however, it was difficult to analyze the saddle point equation as the Laplacian complicates the algebra. We see that the replica method predicts that the phase transition be the same for the sparse Laplacian signals and sparse Gaussian signal case. **Some intermediate work is shown in the Appendix, where most of the steps follow the derivation by Kabashima et al.**[4]. I also performed numeric simulations of the predicted phase transition.

2.3 Simulations

First, I performed some simulations for the retrieval rates using generated data. For each of the simulations, I fixed the parameter ρ and varied the undersampling rate of our sensing matrix $\alpha = M/N$. Then, I plotted the resulting successful retrieval probability. The results are described in 2.

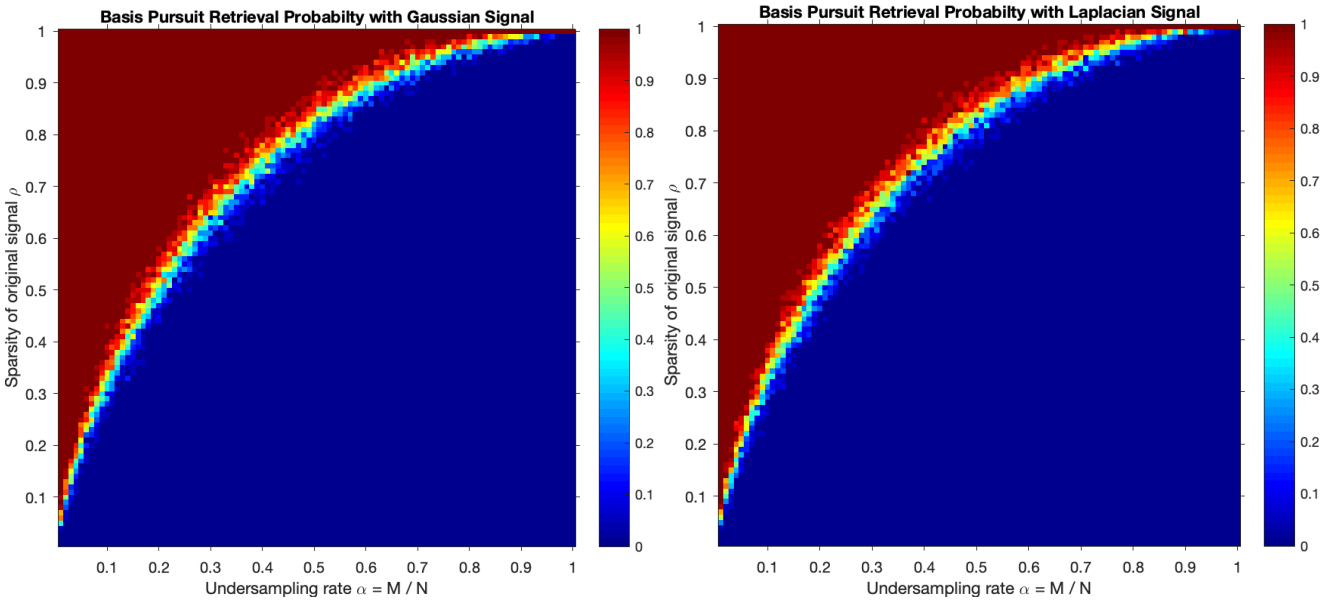
More simulations were done to determine the phase transition line for Gaussian and Laplacian signals. The Gaussian signal was analyzed in class, and I extended the simulation to work for sparse signals sampled from a Laplacian. Scanning through values of ρ and α and plotting the heatmap of retrieval probabilities, we see that the phase transition is very similar in both the Gaussian and Laplacian case. The phase transitions are described in figure 3.



(a) Retrieval rates for Gaussian Signal

(b) Retrieval rates for Laplacian Signal

Figure 2. Retrieval rates for a sparse Gaussian and Laplacian signal for different values of N, the size of the original signal. We see that the phase transition sharpens for larger values of N. The simulation was run with a sparsity value of $\rho = 0.5$, 800 trials per (ρ, α) pair.



(a) Original signal x distributed as sparse Gaussian

(b) Original signal x distributed as sparse Laplacian

Figure 3. Phase transitions for basis pursuit simulations. Retrieval probability is plotted for different values of sparsity ρ and undersampling α. Each value of (ρ, α) was run for 20 trials. Signals were generated randomly from a sparse Gaussian $\mathcal{N}(0,1)$ and a sparse Laplacian with parameter $b = 1$. The phase transitions are essentially identical up to noise, as confirmed by taking the difference between the two phase transition plots. A retrieval probably is considered a success if the MSE between $\hat{x}$ and x is less than $\varepsilon = 0.001$.

2.4 Results

We can reference the phase transition diagrams 3 to determine the maximum undersampling rate of basis pursuit for general video streams. We assume video deltas are distributed as sparse laplacian signals. We have seen from collections of various real-world videos that the sparsity ratio is around $\rho \in [0.6, 0.7]$, which gives us undersampling rates of $\alpha = [0.26, 0.37]$. This corresponds to a range of $2.70\times - 3.85\times$ for lossless compression for average videos. For comparison, current state of the art, widely used video codecs like `x.264` currently have around $2.60\times$ lossless compression[5]. From the phase transition diagram, we see that compressed sensing with basis pursuit already outperforms classical hand-designed video codecs. In the next section, we will learn from the input data to improve our basis pursuit results and increase the compression ratio even more.

3 Data Driven Approaches

3.1 Background

Recently, a lot of work has been done on compression through data-driven approaches. A large focus is to use deep neural networks to "learn" the manifold of natural images and map it to a lower-dimensional space. Essentially, we are creating a sparser representation of images and learning a good basis for compression. However, many current techniques use an encoder-decoder architecture, with multiple non-linear layers in the encoding stage. Researchers from Google have proposed deep recurrent neural networks and applying it to patches in images to capture relationship betweens blocks[6]. Work has also been done on using LSTMs and deep architectures to capture relationships between successive frames of images[7]. Unconventional deep neural network architectures have also been proposed and shown to produce good results, such as GAN based compression with the generator acting as the decompression box[8][9]. However, all of these models require deep encoding layers, making it computationally infeasible for edge devices (sensors, cameras, etc.) to use these compression schemes. We want to be able to use data-driven approaches to improve compression while maintaining the simplicity of encoding in our L1 minimization scheme. Mousavi et al. has worked on jointly training a fully connected layer as the encoder and a neural network as the decoder[10][11]. Since the fully connected layer is equivalent to a sensing matrix in compressed sensing, I decided to use a similar architecture to train a sensing matrix and analyze its performance.

3.2 Goal

Neural network based compression schemes do not provide near-perfect reconstruction, since the loss criteria used is MSE loss for computational feasibility. Therefore, I wanted to find a way to combine the near-perfect reconstruction of basis pursuit methods and have a data-driven approach that can learn the space of natural images. I explored using encoder-decoder based approaches to "learn" an optimal sensing matrix for our compressed sensing problem.

3.3 Experiments

My main goal was to use a data-driven approach to train a sensing matrix, and determine whether using a learned sensing matrix would outperform choosing a random sensing matrix in a basis-pursuit compressed sensing setup. In order to learn a sensing matrix, I used an encoder-decoder type architecture, with the encoding layer as a sensing matrix and the decoder as a fully convolutional neural network. I tried many different hyperparameters and layer combinations. The main hyperparameter set and layer architecture is described in 4. I ran the tests on the MNIST dataset[12], a collection of grayscale handwritten numbers. The neural network first uses a sensing matrix (fully connected layer) to map the MNIST image from $28 \times 28 \rightarrow K \times K$, then uses fully convolutional layers to recover the original image. Then, after training multiple neural networks with different undersampling rates α, I evaluated the decoded results of the neural network and used the basis pursuit algorithm with the "learned" sensing matrices.

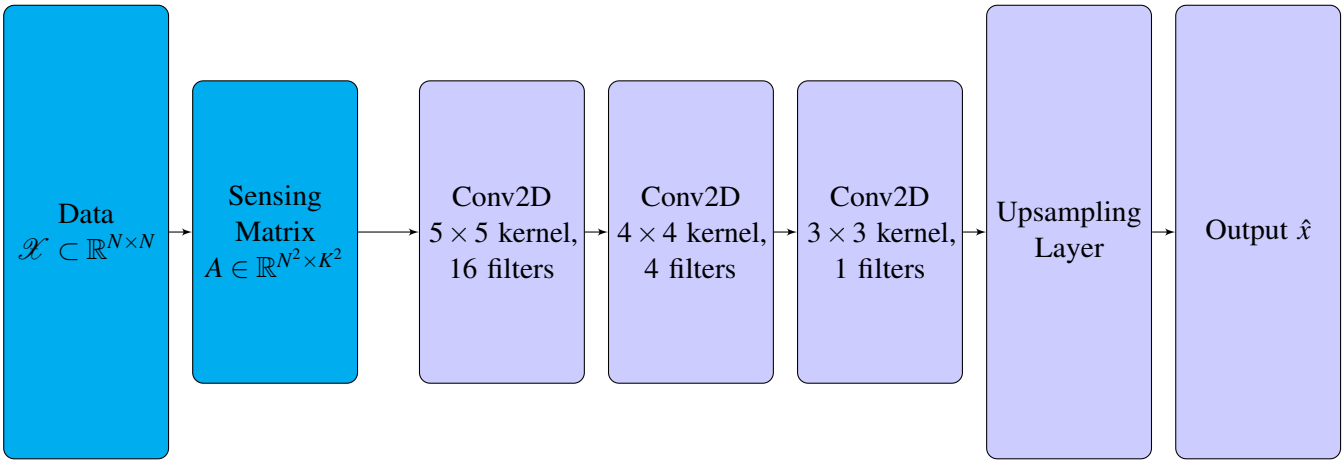


Figure 4. Neural Network architecture used for training the sensing matrix. The network uses a sensing matrix as the encoder and uses fully convolution layers as a decoder, upsampling to the original resolution, and applying L1 loss for gradient updates. The network was optimized with ADAM. Each neural network took around 3 hours to train on a NVIDIA GTX 1080 Ti GPU. The neural network was trained on MNIST (input data 28×28) for the sensing matrices $K = 4 \rightarrow 24$, which took around 2 days to train. The cyan colors denote the layers corresponding to the encoding, and the light purple color decoding.

3.4 Neural Network Results

First, we can visualize the SNR and outputs of our encoder-decoder neural network to visually see if the neural network is learning a viable encoding. Some examples of the neural network outputs are displayed in Figure 5. We can visually see that the neural network does seem to produce distinguishable outputs at an undersampling rate of around $\alpha = 0.18$. To evaluate quality of the decoded images, the SNR is plotted for the test set in Figure 6.

3.5 Basis Pursuit Results

After training the neural networks, I took the trained sensing matrices and used that as the A in basis pursuit. Experimentally, this visibly increases the performance of the basis pursuit algorithm. On the MNIST test set with random choice of A, the phase transition seems to be centered around $\alpha = 0.5$; however, with the learned choice of A, the phase transition seems to be centered around 0.41, a significant improvement. This seems to be generally the case on other datasets of natural images that I tested on, such as CIFAR-100. However, I was not able to train enough fully convolutional encoder-decoder networks to show the retrieval probability against undersampling rate α due to computational constraints (each CIFAR-100 network took multiple hours to train).

4 Further Work

4.1 Next steps

Due to time constraints, I was not able to do everything I wanted to in the project proposal. My next steps would have been to train 1 layer encoder-decoder neural networks for optimizing the matrix A and analytically come up with phase transitions for basis-pursuit based on a learned A matrix. I also would also continue analyzing the basis pursuit algorithm for a Laplacian x fully with the replica method. The data seems like it should match the Gaussian version very closely, but the absolute values in the Laplacian distribution have made the saddle point equations difficult to solve.

My next steps are also to apply basis pursuit with a learned A matrix to actual videos instead of just the MNIST dataset and compare the file sizes to basic entropy encoding, to see if compressed sensing is feasible for real-world

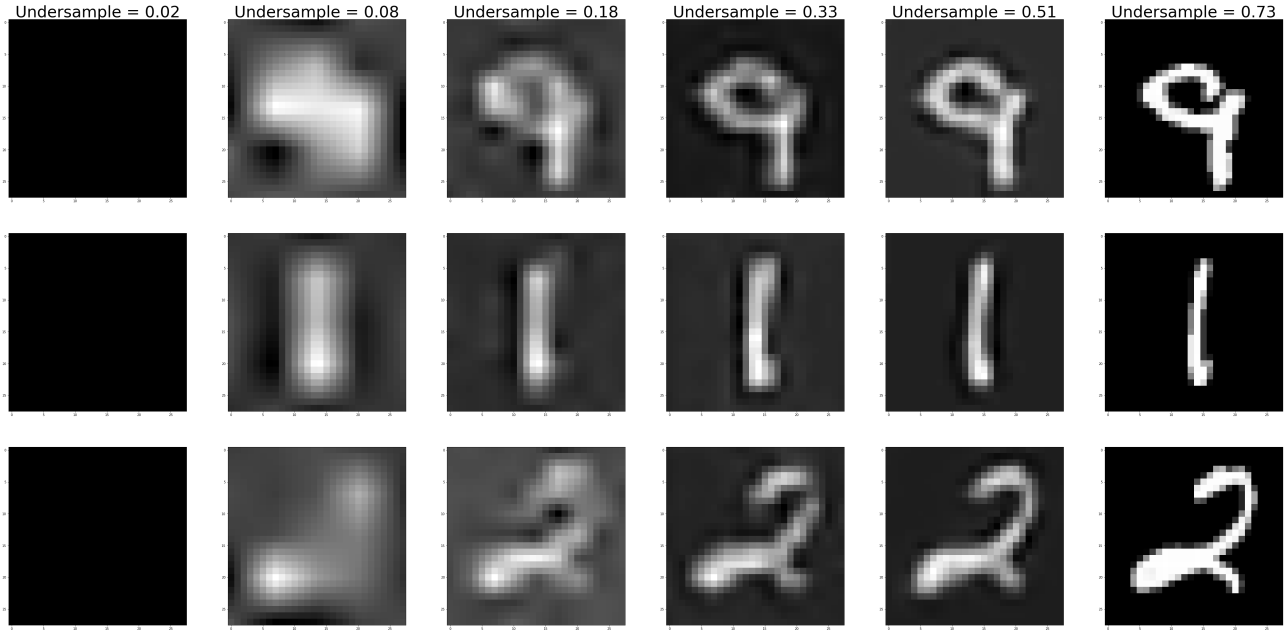


Figure 5. Decoded results from the fully convolution neural network. The network architecture uses a learned sensing matrix encoder and fully convolution decoder, which are jointly optimized during the training process. Results shown are from the MNIST test dataset, which are not seen during the optimization process. The undersampling ratio $\alpha = M/N$ for the columns are $0.02, 0.08, 0.18, 0.33, 0.51, 0.73$ respectively. Unsurprisingly, the quality of reconstruction increases as the α increases, with the reconstructed image being identifiable at around $\alpha = 0.10$. Since the neural network is trained with MSE loss, we cannot directly use retrieval probability and use SNR to see how well the encoder-decoder setup does.

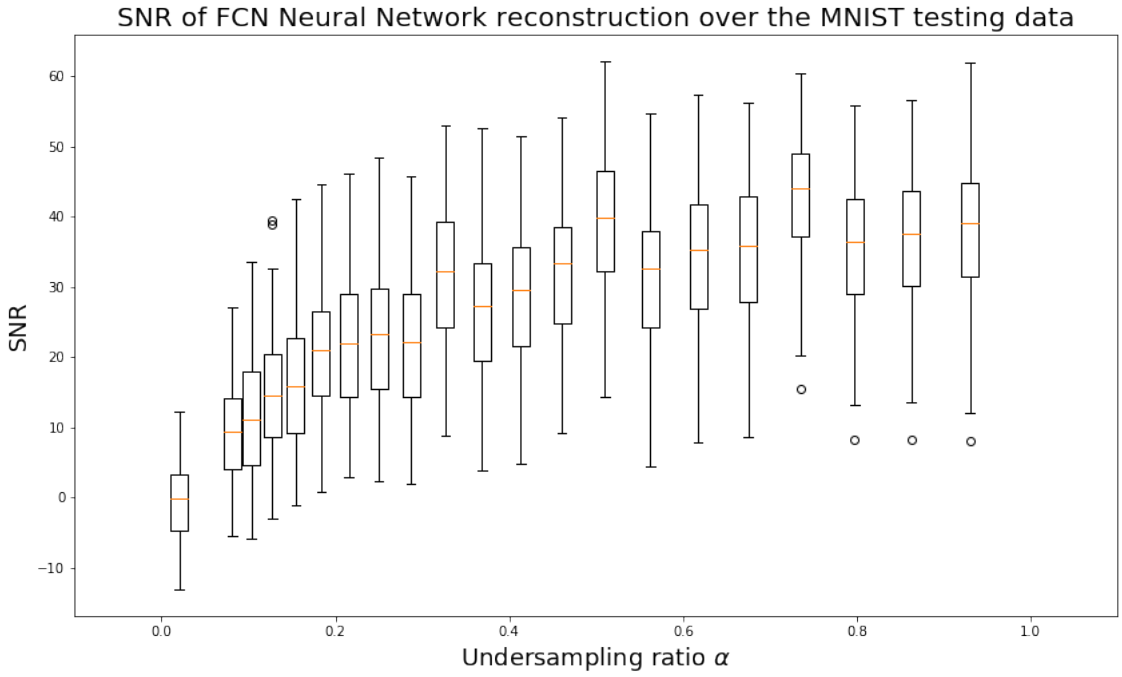


Figure 6. SNR results from the trained neural networks. The results are collected using 1000 randomly chosen images from the MNIST test set.

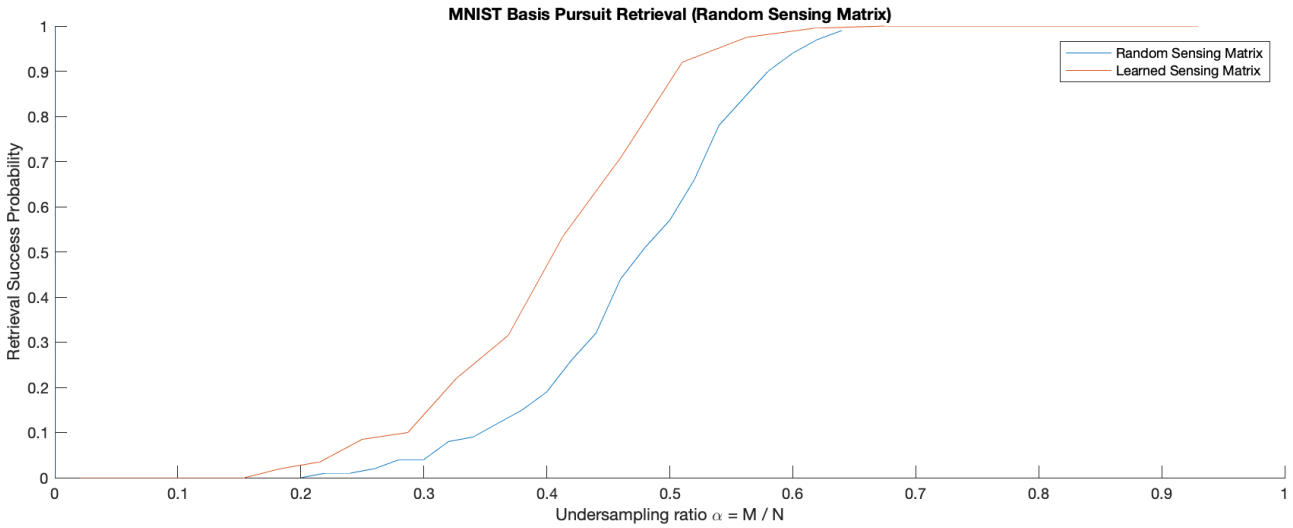


Figure 7. Retrieval probability plotted against undersampling ratio α of the sensing matrix. Retrieval was done using basis pursuit using MATLAB's Linear Programming optimizer, on 1000 randomly chosen MNIST test set images. The basis pursuit algorithm was first done using a random choice of A, and then done with a fixed A that was learned through the neural network. We considered a $\hat{x}$ as recovered if the MSE loss between x and $\hat{x}$ was less than a small error term $\varepsilon = 0.001$.

applications. The main roadblock was the GPU training time required to train encoder-decoder networks - since we use MSE loss on whole images, the networks take much longer to train.

4.2 Challenges

The main challenges were getting the neural networks to train properly. Tuning and training neural networks with full-resolution output caused a lot of GPU memory errors. Furthermore, I was unable to implement D-AMP, basis pursuit, or LASSO in python due to the linear programming solvers not being able to solve the equivalent LP problem. Therefore, I had to pick up MATLAB for all of the non-neural network parts of the course project.

4.3 Acknowledgements

I would like to thank Professor Lu and Oussama Dhifallah for the excellent lectures and sections, especially for someone new to these types of methods. The topics covered during class have been invaluable toward this project, and I am glad I had the opportunity to take the class! Also I would like to thank John Alex Keszler for helping me get starting with MATLAB.

5 Conclusion

Video sources and computational resources have grown steadily in the past decade. Along with the success of convolutional neural networks for computer vision, there have been an increased demand in accurate and reliable video streaming for limitless real-world applications but little growth in network bandwidth infrastructure. As a result, a significant number of applications are bandwidth-limited rather than limited by computational power. Current video codecs reduce bandwidth, but are usually lossy. Compressed sensing is promising in that it takes low-computational power for the sender, reduces bandwidth usage, and offers near perfect reconstruction. We have shown that combining data learning approaches with neural networks can dramatically improve the compression ratio for the basis pursuit reconstruction scheme. Furthermore, we have shown that the compression ratios from compressed sensing outperform hand-designed video codecs on lossless settings, which highlights the potential for compressed sensing to be used in future lossless video streaming applications.

References

1. Grois, D., Marpe, D., Mulayoff, A., Itzhaky, B. & Hadar, O. Performance comparison of h.265/mpeg-hevc, vp9, and h.264/mpeg-avc encoders. In *2013 Picture Coding Symposium (PCS)*, 394–397, DOI: 10.1109/PCS.2013.6737766 (2013).
2. Lustig, M., Donoho, D. & Pauly, J. M. Sparse mri: The application of compressed sensing for rapid mr imaging. *Magn. Reson. Medicine* **58**, 1182–1195, DOI: 10.1002/mrm.21391. https://onlinelibrary.wiley.com/doi/pdf/10.1002/mrm.21391.
3. Candes, E., Romberg, J. & Tao, T. Stable signal recovery from incomplete and inaccurate measurements (2005).
4. Kabashima, Y., Wadayama, T. & Tanaka, T. A typical reconstruction limit for compressed sensing based on l p -norm minimization. *J. Stat. Mech. Theory Exp.* **2009**, L09003 (2009).
5. Takamura, S. Lossless video coding.
6. Toderici, G. *et al.* Full resolution image compression with recurrent neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5435–5443, DOI: 10.1109/CVPR.2017.577 (2017).
7. Palangi, H., Ward, R. & Deng, L. Distributed compressive sensing: A deep learning approach. *IEEE Transactions on Signal Process.* **64**, 4504–4518, DOI: 10.1109/TSP.2016.2557301 (2016).
8. Santurkar, S., Budden, D. & Shavit, N. Generative compression. *2018 Pict. Coding Symp. (PCS)* 258–262 (2018).
9. Liu, Z. *et al.* Deepn-jpeg: A deep neural network favorable jpeg-based image compression framework. *CoRR* **abs/1803.05788** (2018). 1803.05788.
10. Mousavi, A., Dasarathy, G. & Baraniuk, R. G. DeepCodec: Adaptive Sensing and Recovery via Deep Convolutional Neural Networks. *arXiv e-prints* arXiv:1707.03386 (2017). 1707.03386.
11. Mousavi, A. & Baraniuk, R. G. Learning to Invert: Signal Recovery via Deep Convolutional Networks. *arXiv e-prints* arXiv:1701.03891 (2017). 1701.03891.
12. Lecun, Y. & Cortes, C. The MNIST database of handwritten digits. .

6 Appendix

6.1 Basis Pursuit Replica Method

We can use the replica method to analyze the basis pursuit reconstruction when our signal is Laplacian. Suppose that we have our original signal $\mathbf{x} \in \mathbb{R}^N$ and sensing matrix $A \in \mathbb{R}^{N \times M}$, distributed as:

$$x_i^0 \sim (1-\rho)\delta(x_i^0) + \frac{\rho}{2b}\exp\left(-\frac{|x|}{b}\right) \qquad A_{i,j} \sim \mathcal{N}\left(0, \frac{1}{N}\right)$$

Then, we want to analyze the performance of the basis pursuit recovery given measurements $y = Ax$.

$$\hat{\mathbf{x}} = \text{argmin}_{\mathbf{x}} \|\mathbf{x}\|_1 \qquad \text{s.t. } \mathbf{Ax} = \mathbf{y}$$

We can analyze this using the replica method. Similar to the derivation of the lecture notes and work by Kabashima et al.[4], the partition function of our system is

$$Z(\beta; \mathbf{A}, \mathbf{x}^0) = \int e^{-\beta\|\mathbf{x}\|_1} \delta(A\mathbf{x}^0 - A\mathbf{x}^i)\{d\mathbf{x}^i\}$$

with the free energy

$$f(\beta) = -\frac{1}{\beta}\lim_{N\to\infty} \mathbb{E}_{\mathbf{x^0},\mathbf{A}} \frac{1}{N} \log Z(\beta; \mathbf{A}, \mathbf{x}^0)$$

Now, we consider n replicas of the system. We then analyze the free energy for the limiting behavior $N \to \infty$ and with undersampling ratio $\alpha = M/N$ fixed. Then, the free energy is determined by a saddle point equation.

$$\frac{\partial}{\partial\{m,r,q,p\}}\left[-I(m,r,q,p)+\alpha\log g(m,r,q,p)\right]=0$$

$$g(m,q,r,p)=(r-q)^{-n/2}(2\pi)^{-n/2}\left(1+\frac{n(q-2m+p)}{r-q}\right)^{-1/2}$$

$$I(m,r,q,p)=\sup_{\hat{m},\hat{r},\hat{q},\hat{p}}\left(\frac{n(n-1)}{2}\hat{q}q+n\hat{m}m+\hat{p}p+n\hat{r}r-\log\mathbb{E}_{x^0,z}e^{\hat{p}(x^0)^2}Z^n\right)$$

$$Z=\int\exp\left(-\frac{1}{2}(\hat{q}-2\hat{r})x^2+(z\sqrt{\hat{q}}+\hat{m}x^0)x-\beta|x|\right)dx$$

We can follow the derivations done on Homework 4. The steps diverge when we solve for the value of m for our saddle point equation. In Homework 4, we made the following substitutions:

$$\chi=\beta(r-q)\qquad\hat{\chi}=\frac{\hat{q}}{\beta^2}\qquad v=\frac{\chi}{\alpha}\qquad u=\frac{\chi\sqrt{\hat{\chi}}}{\alpha}$$

Then, considering the limiting case $\beta\to\infty$, we have that

$$x=f(x_0+uz,v)\qquad\text{where } f(a,b)=(a-\text{sign}(a)\cdot b)^{\mathbf{1}(|a|>b)}$$

Then, the update equation of χ will be

$$\chi=\frac{u(1-\rho)}{\sqrt{\hat{\chi}}}\text{erfc}\left(\frac{v}{\sqrt{2}u}\right)+\frac{\rho}{\sqrt{\hat{\chi}}}\mathbb{E}_{z,x^*}f(x^*+uz,v)\qquad\text{where } x^*\sim\text{Laplace}(0,1)$$

The second term is what changes when we change it to a Laplacian instead of having our sparse signal come from a Gaussian. Computing the expectation for the second term, we have

$$\mathbb{E}_{z,x^*}f(x^*+uz,v)=2u\left(e^{\frac{u^2}{2}-v}\left(\text{erfc}\left(\frac{u^2-v}{\sqrt{2}u}\right)-e^{2v}\text{erfc}\left(\frac{u^2+v}{\sqrt{2}u}\right)\right)+2\text{erfc}\left(\frac{v}{\sqrt{2}u}\right)\right)$$

As reference, our original expectation for the second term when we have a sparse Gaussian signal was much simpler.

$$\text{Gaussian Result}=u\left(\frac{v}{\sqrt{2u^2+2}}\right)$$

Now, we have single powers of u, which complicates our expression later. Plugging this equation back into our expression for χ,

$$\chi=\frac{1-\rho}{\sqrt{\hat{\chi}}}\text{erfc}\left(\frac{v}{\sqrt{2}u}\right)+\frac{2u\rho}{\sqrt{\hat{\chi}}}\left(e^{\frac{u^2}{2}-v}\left(\text{erfc}\left(\frac{u^2-v}{\sqrt{2}u}\right)-e^{2v}\text{erfc}\left(\frac{u^2+v}{\sqrt{2}u}\right)\right)+2\text{erfc}\left(\frac{v}{\sqrt{2}u}\right)\right)$$

Now, we can do the same for the other parameters in our saddle point equation. Again we see that the Laplacian makes the expression much more complicated than the Gaussian.

$$\begin{aligned}m&=\rho\mathbb{E}_{x^*,z}x^*f(x^*+uz,v)\\&=2\rho e^{-\frac{v^2}{u^2}-\frac{u^2}{2}-v}\left(e^{\frac{v^2}{u^2}+\frac{u^2}{2}}\left(4e^v\text{erfc}\left(\frac{v}{\sqrt{2}u}\right)-e^{\frac{u^2}{2}}\left(u^2-v-2\right)\text{erfc}\left(\frac{u^2-v}{\sqrt{2}u}\right)\right)+\left(u^2+v-2\right)e^{\frac{\left(a^2+v\right)^2}{u^2}}\text{erfc}\left(\frac{u^2+v}{\sqrt{2}u}\right)\right)\end{aligned}$$

$$
\begin{aligned}
r &= (1-\rho)\mathbb{E}_z(f(uz,v))^2 + \rho\mathbb{E}_{x^*,z}(f(x^*+uz,v))^2 \\
&= (1-\rho)\left(u^2+v^2\right)\operatorname{erfc}\left(\frac{v}{\sqrt{2}u}\right) - \sqrt{\frac{2}{\pi}}uve^{-\frac{v^2}{2u^2}} + \\
&\quad 2\rho e^{-\frac{v^2}{u^2}-\frac{u^2}{2}-v}\left(e^{\frac{v^2}{u^2}+\frac{u^2}{2}}\left(4e^v\operatorname{erfc}\left(\frac{v}{\sqrt{2}u}\right) - e^{\frac{u^2}{2}}\left(u^2-v-2\right)\operatorname{erfc}\left(\frac{u^2-v}{\sqrt{2}u}\right)\right) + \left(u^2+v-2\right)e^{\frac{\left(a^2+v\right)^2}{u^2}}\operatorname{erfc}\left(\frac{u^2+v}{\sqrt{2}u}\right)\right)
\end{aligned}
$$

Now we have everything we need for the update rules of $\hat{\chi}$. With $\chi \to 0$, we have $r = q$, and the update equation is

$$
\begin{aligned}
\hat{\chi} &= \frac{\alpha}{\chi^2}(q-2m+\rho) \\
&= \lim_{\chi\to 0}\frac{\alpha}{\chi^2}(r-2m+\rho)
\end{aligned}
$$

Since we are taking $\chi \to 0$, we can plug in the analytic results we get for r and m. Then, we take the Taylor expansions and most of the terms cancel out because $\chi \to 0$ (These steps were done in Mathematica). Note that u has a factor of χ and a factor of $\hat{\chi}^{1/2}$, and v has a factor of χ. Therefore, we separate everything to terms with factors of χ.

$$
\begin{aligned}
\hat{\chi} &= v^2\operatorname{erfc}\left(\frac{v}{\sqrt{2}u}\right) - \sqrt{\frac{2}{\pi}}uve^{-\frac{v^2}{2u^2}} + \frac{\rho u^2}{\alpha v^2} + \frac{\rho}{\alpha} + \lim_{\chi\to 0}(O(\chi)f(\chi)) \qquad \text{where } f \text{ is some function of } \chi \\
&= v^2\operatorname{erfc}\left(\frac{v}{\sqrt{2}u}\right) - \sqrt{\frac{2}{\pi}}uve^{-\frac{v^2}{2u^2}} + \frac{\rho u^2}{\alpha v^2} + \frac{\rho}{\alpha}
\end{aligned}
$$

However, this is just the same self-consistent equation for $\hat{\chi}$ for the Gaussian case in the homework! All of the extra terms from the Laplacian that contributed $\frac{u^2 \pm v}{u}$ had factors of χ, which disappears in the limit $\chi \to 0$. Therefore, the replica method predicts that we should get approximately the same phase transition line as in the Gaussian case (My algebra may have been incorrect - I did the Taylor expansion and then cancelled out terms containing χ by hand).

6.2 Basis Pursuit Phase Transitions

Since the Gaussian and Laplacian versions of the basis pursuit problem seemed so similar, I ran other tests to rule out programming error. In this test, I generated the original sparse signal using a generalized beta distribution. The first numeric simulation was using uniform values in the range $[-1,1]$. The results are described in Figure 8. The phase transition shape is actually slightly different than the Gaussian and the Laplacian case.

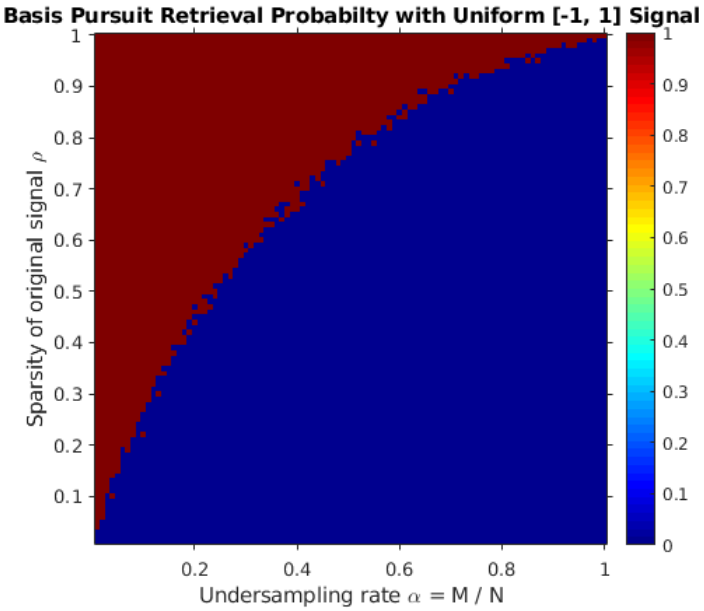


Figure 8. The same experiment with the same parameters as the other basis pursuit phase transition plots. However, instead of being sampled from a sparse Gaussian or Laplacian, it is sampled from a sparse uniform distribution $U(-1,1)$. Surprisingly, the phase transition for the same parameters is much harsher - retrieval either works for all or none.