Image Encryption with Sparse Autoencoders

December 9, 2022

Deep Learning for Computer Vision COMS W 4995 006

December 9, 2022

Howard Yong (hy2724), Shivansh Srivastava (ss5945)

GitHub: https://github.com/ShivanshSrivastava1/Image-Encryption-with-Sparse-Autoencoders

Presentation: https://drive.google.com/file/d/1x75lrqTcSuNGuk1K-6cIQuzTM8mUAiZg/view?usp=sharing

1 Introduction (hy2724)

Network-based machine learning algorithms are extremely powerful. Especially in the computer vision field, various different methods are continually being developed to enhance computer capabilities to classify and contextualize objects in our world. One key strength of network-based algorithms is its ability to approximate non-linear behaviors. These patterns are learned by feeding the model massive amounts of training data. The more training (and testing) data available, the better the chance the model has at seeing all the different nuances in the underlying distribution of whateever activity it is being applied to.

However, with vision problems particularly, privacy issues are often encountered when collecting datasets. Faces, actions and behaviors that are personally identifiable and private are stored often without consumer realization. In the event of a data breach or some adversarial attack to the host of such images, thousands or millions of individuals faces and other data tied to them can circulate without their control. Nonetheless, for innovation to continually grow and achieve state of the art performance, researchers still require such data to be available.

There are current methods of encryption of such images so that humans cannot understand images but the network model can still be trained on these encrypted images. This is a growing feel in recent years. Early strategies included shuffling pixels around, directly encrypting images, or applying Gaussian or Poisson noise to the image. In recent years, it has been demonstrated that these defenses can be overcome with adversarial neural networks. Thus, the ideal solution would successfully conceal these images from adversaries, while allowing trustworthy parties (e.g. users who own the pictures with their faces) to reconstruct them. Because end-users need an easy way to obtain their hidden images, the ideal solution must also be compact enough so that speed and efficiency are unaffected. For instance, federal level databases storing citizenship information and faces may need to share specific images of individuals with different parties. The solution we propose is to introduce a method of sharing only encrypted, codified versions of images rather than images directly. Thus, only parties with the corresponding key can access and reconstruct the image appropriately.

2 Related Work (ss5945, hy2724)

Image recognition has always been a core challenge for computer vision since the field's inception. Among the various breakthroughs, Yann LeCun's LeNet holds the honor of pioneering the first feasible convolutional neural networks. Though LeNet's design was simplistic, its clever use of 5 layers composed of 5x5 convolutions and 2x2 maxpooling paved the way for machine digit recognition. Indeed, the combination of two dimensional convolutions and maxpooling served as the main inspiration for the encoder that was trained in this study.

At the same time, the prior literature has shown that while massive convolutional neural networks might be able to accomplish the goal of privatization, they are ultimately not suitable for end-users on a larger scale. Sohrab Ferdowsi et al.'s research from 2020 was instrumental for showcasing one of the first realizable attempts at compressing and searching visual data in large-scale systems. In his paper "Learning to compress and search visual data in large-scale systems", Ferdowsi et al. proposed a novel, practical framework design for securely sharing images. Ferdowsi's findings not only demonstrate that compactness is necessary at scale, but also representations can be stored and shared publicly without compromising on privacy protection. Although the inception of the idea was of our own, the contributions made by Ferdowsi et al. 2020 provided a strong foundation for our model.

A key component of the practical framework proposed by Ferdowsi was ambiguating representations before sharing with the public domain. In order to do so, Ferdowsi et al proposed a unique top-k activation function. Effectively, in between the encoder and decoder stages, the support of the latent representation vector space is collected. This is done so to avoid large intermediary matrices with intractable complexities and extremely high risks of overfitting. By breaking up the convolutional features and passing them to a much smaller fully-connected linear layer, the final network can learn multiple features of each image in a non-strenuous manner.

In the same vein, this study also drew inspiration from Twewari et al.'s work "MoFA: Model-based Deep Convolutional Face Autoencoder for Unsupervised Monocular Reconstruction" from 2017. Tewari et al's goals sowed the seeds for the model from this study's first iteration. Tewari et al. demonstrated the viability of a deep auto encoder for facial reconstruction via implementing well-known models. Namely, the researchers relied on the pre-trained models AlexNet and VGG-Face as the encoders.

3 Methods

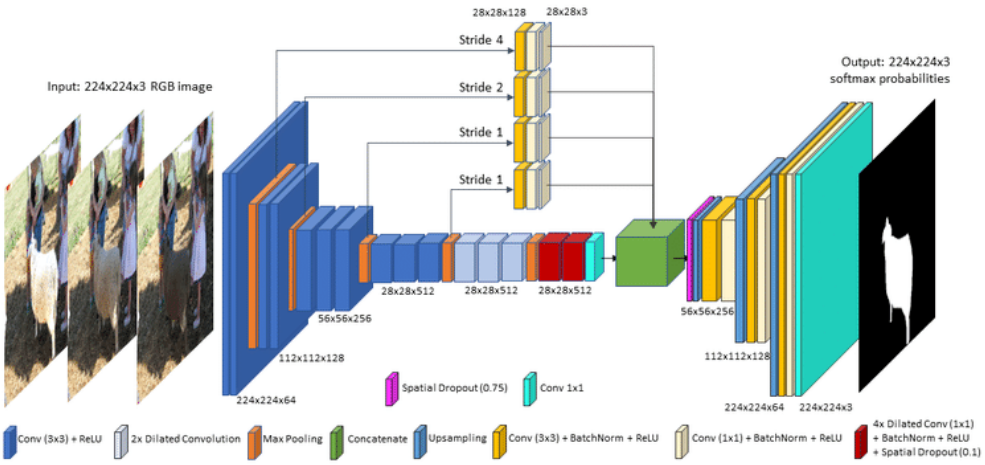
3.1 Proposed Idea (hy2724, ss5945)

In class, we discussed the power of autoencoders and their ability to denoise images. We also briefly touched on the topic of diffusion models and introduced the idea of injecting noise into the latent space of autoencoders to enhance its denoising capability while making the training procedure more computationally efficient. This inspired us to explore latent representations of images for encryption techniques. Our idea is to build an autoencoder with stacked linear layers such that multiple codecs of an image would be generated. The network would be trained in a way to allow control over reconstruction fidelity so that security can be regulated. The technique of implementing such a training procedure was motivated by Ferdowsi et al. 2020. Each of these codes would be sparsified, latent vectors that when passed to the trained decoder can be reconstructed appropriately if the correct key is provided. The index of the correct code would act as the reconstruction key, which

would then be given to the corresponding trustworthy party. Note that this key can be much more easily encrypted than an entire image. In order to formulate a compact solution, the model's training heavily relied on sparsity. Specifically, sparsity was injected into the latent representations by applying a special activation function that collected the top-k nonzero values. The added sparsity allowed control over the resolution of the reconstructed images. Otherwise, if a given code wants to be private, locally sampled noise would be applied to inject noise to prevent image reconstruction, since the decoder was trained on sparsity but not to handle noise.

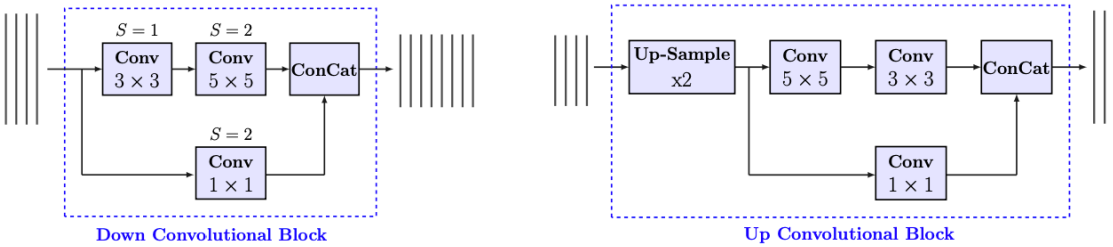
3.2 Autoencoder Architectures (hy2724)

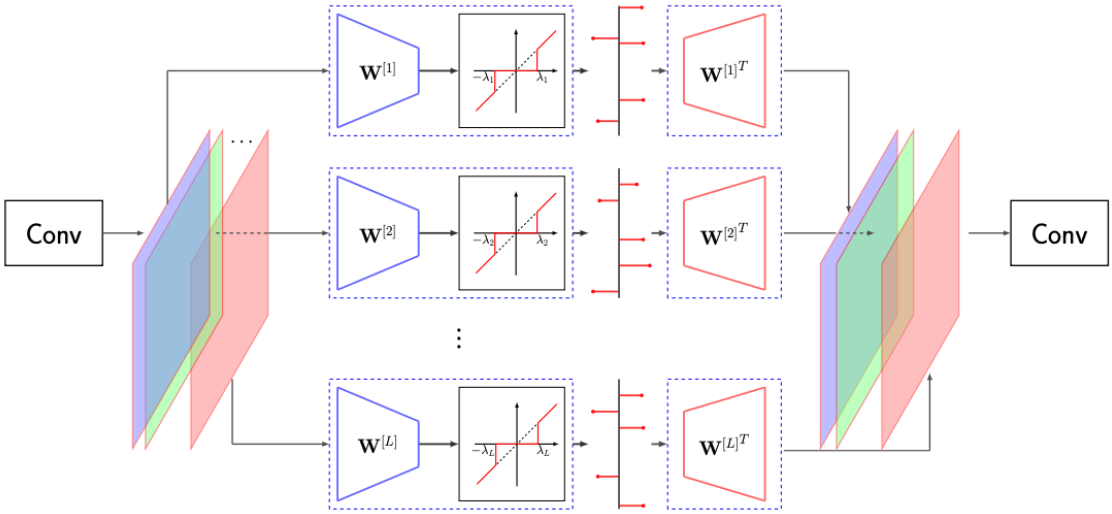
Some design specifications for the model architecture required that it bottlenecks the input dimension space. Additionally, it must have some sparsity in the representation space such that reconstruction resolution can be controlled. To achieve these, we explored 2 different architectures and 3 iterations of training. The first training iteration implemented a deep autoencoder architecture inspired directly from VGG-16. We effectively utilized VGG-16 as the encoder and replicated it on the decoder end. Although the following diagram (Reference 9) doesn't exactly reflect the architecture (becausee we had adjusted the latent space representation), it captures the general idea.



In the next 2 iterations of training, the high level architecture from Ferdowsi et al. 2020 (Reference 4) was implemented. For the second training iteration, we adjusted the dimensions of the layers and shapes of the filters in the downsampling and upsampling blocks, making them uniformly `(3x3)` as VGG-16 had implemented it. Initially, the implementation in Ferdowsi et al. 2020 had `[40,40,40,40,40,10]`. However, in our implementation we modeled our filter sizes off of VGG-16 and changed them to `[64,128,256,512,12]`, with the last dimension for bottlenecking and to align with the number of groups in the stacked linear layers. Additionally, we maintained uniform strides whereas the original implementation had alternating scaling factors between 1 and 2. Lastly, in our final training iteration, we adopted the same kernel sizes as implemented in Ferdowsi et al. 2020. However, we still reduced the number of blocks from 6 to 5 in both the encoder and decoder. The principle change in iteration 3 was that we inspected the architecture of the decoder and noticed

that it was not symmetrical to the encoder. We hypothesized that for the performance to be optimal and proposed idea to be practical, we would want a decoder that matches the reflection of the encoder. Thus, we injected an additional convolutional layer in the upsampling blocks and adopted the forward pass. See a figure of an example of the downsampling block, upsampling block, and stacked linear layers, respectively below. Figures taken from Reference 4.





4 Experiments (hy2724)

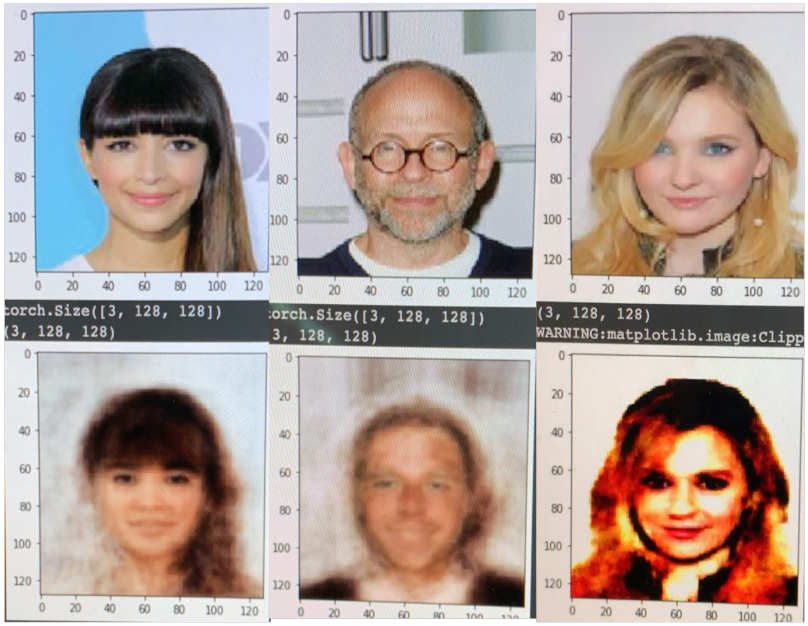
Because this project focused on facial recognition, the CelebFaces Attributes (CelebA) Dataset was used. Though the complete dataset contains over 200,000 images of popular celebrities, a random subset of 20,000 images were selected for this study. The typical 80-10-10 ratio was used for splitting the data into training, validation, and testing sets, as this ratio is in line with the guidelines for the homework assignments in this class. We trained for 10 epochs.

Over the course of this study, 3 iterations and 2 architectures were needed to obtain the final model. The first iteration drew heavy inspiration from the VGG16. In this iteration, the encoder relied on 5 convolutional blocks that implemented the same architecture as VGG16. These 5 blocks were followed by a stacked linear layer with an activation function that collected the k support of the vector—it kept the elements with the k largest magnitudes as appropriate tensor dimensions and zeroed the remainder.

The decoder was remarkably similar to its sibling. Most notably, the decoder's architecture was symmetric to the encoder's, but in reverse. Another defining trait was the decoder's inclusion of an UpsamplingBilinear2d layer.

This first iteration was trained using k = 128 for sparsity, binary cross entropy loss with a sigmoid,

and the Adam optimizer with an initial learning rate of 0.002 and a weight decay of 0 for 10 epochs. However, this model was unable to learn features well. Results of reconstructed images during validation at the final epoch of training 3,000-5,000 images are shown below.

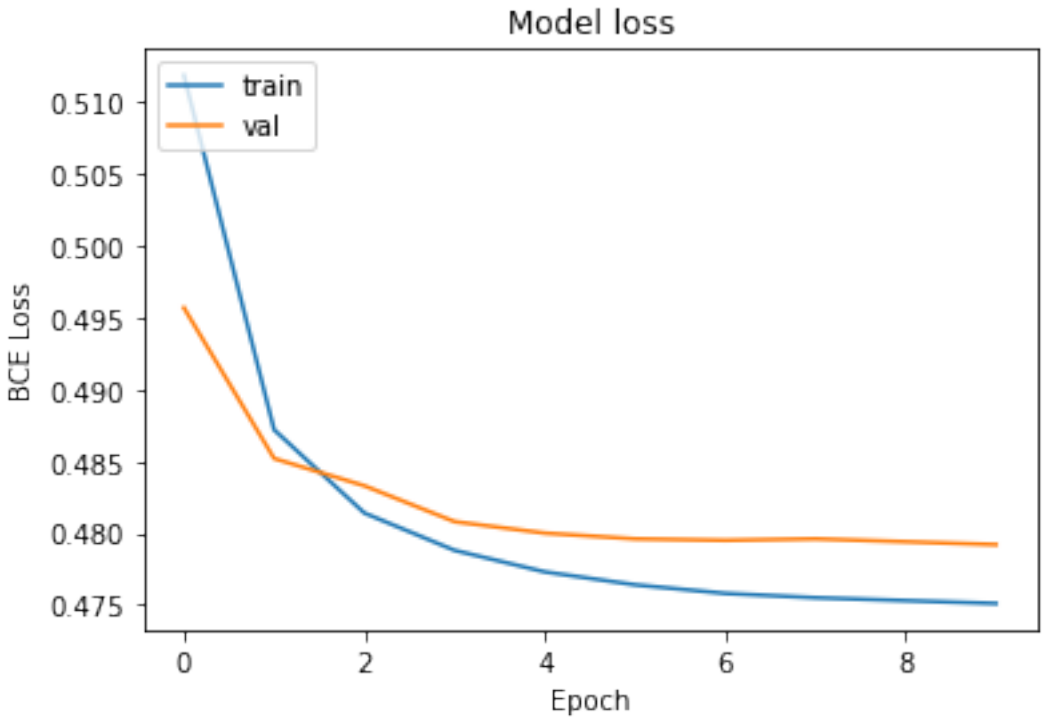


From the disappointing results, it could be reasonably inferred that the exact VGG16 architecture was a mismatch for the top-k activation function. Iteration 2 strived to rectify this conflict by looking to the prior literature. Ultimately, Ferdowsi et al.'s work from 2020 provided the primary motivation for the successful redesigns in the second iteration. While the encoder still used 5 convolutional blocks and the top-k activation for enforcing sparsity, each of the 5 blocks now had 2 or 3 Conv2d layers and a MaxPool2d layer. Moreover, rather than keeping all kernel sizes fixed at 3 for each of these Conv2d layers, the kernel sizes now scaled up from 1 to 3 to 5. Finally, the Conv2d layers were concatenated instead of remaining in sequential order.

The structure of the decoder remained a reverse image of the encoder. The kernel sizes for the decoder's upsampling block layers were changed to scale down from 5 to 3 to 1, further distancing the model from VGG16's architecture. The Conv2d layers in the decoder were also concatenated to uphold symmetry.

The second iteration was again trained with k = 128 for sparsity, binary cross entropy loss with a sigmoid, and the Adam optimizer with an initial learning rate of 0.002 and a weight decay of 0 for 10 epochs.

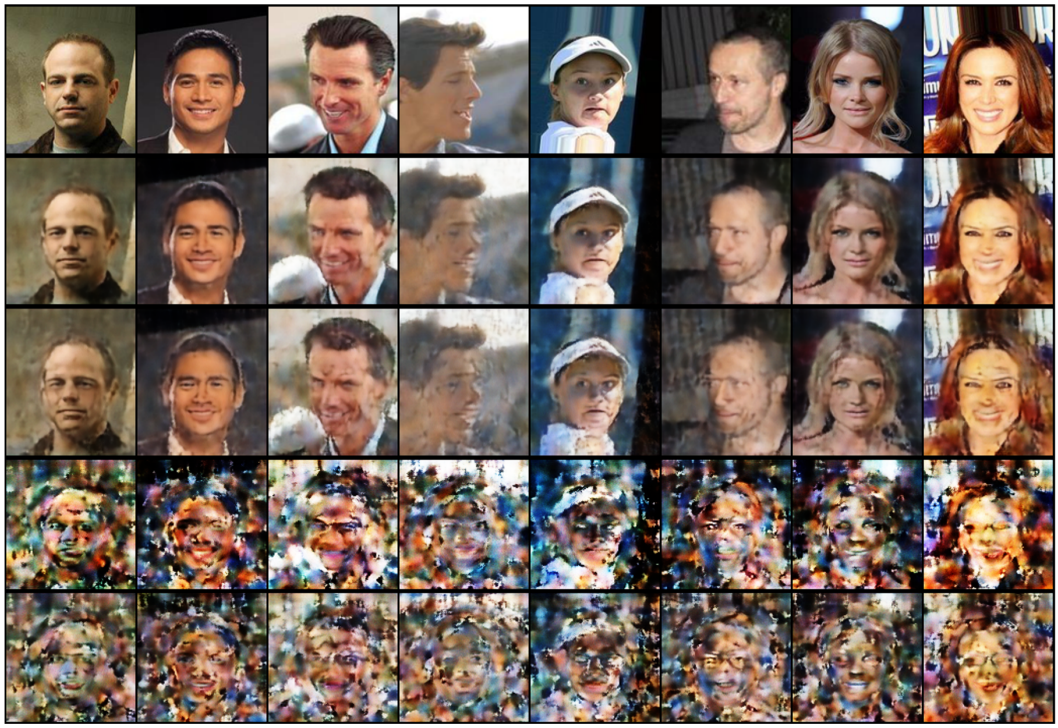
Training and validation loss from the experiments in the final iteration of this study are shown below as these were the most promising results. It can be seen that training progress was satisfactory as both training and validation losses continually decreased until eventually leveling out. Following the model loss curve are examples of some reconstructed images from Epoch 1, Epoch 6, and Epoch 10, respectively. The top row shows the input image and the bottom shows the reconstructed image or model output.



Next, we also needed to test the model against out-of-sample testing data. The 2 principle experimental results we wanted to verify were that the model effectively controls resolution quality upon reconstruction and the latent representation learned is robust to reconstruction if desired. The results are shown in the following figure. For the first result, we simply enumerated samples in the test dataloader. We provide results for reconstructing with same sparsity level as during training (k=128) and half (k=64). The results are satisfactory as the image resolution visibly diminishes with half the information given. These results are shown in rows 2 and 3 in the following figure.

We also inspected the model's robustness to reconstruction since our proposed idea is image encryption capability. To reiterate, the key idea of our proposal is to output sparse codes drawn from the latent space of the autoencoder. In practice, if this idea were deployed then multiple codecs would be produced and the key would be the correct code shaerd. All other codecs would

have some type of sampled noise applied to obscure the reconstruction process. In this test, we leveraged a pre-built module for estimating the distirbution of pixel values to draw noise from. These results demonstrate our model's reconstruction of encrypted codes of images. The 4th row shows the result of reconstruction after obscuring the code. The 5th row demonstrates the model robustness to adversial attacks. In this result, if an adversary were provided 50% of the information available and was responsible for randomly permuting the latent space to identify the remaining 50%, it would fail to reconstruct a discernible person.



Lastly, we wanted to explore methods to quantify the quality of our model outputs. There are several metrics utilized for measuring quality of images, but the 2 standard measures we computed were Peak Signal Noise Ratio (PSNR) and Structural Similarity Index (SSIM). PSNR is a measurement that inspects the quality of image compression with respect to the maximum possible power available. State of the art results generally range from 30-50 decibels (it is measured on the logarithmic scale). SSIM measures the similarity between images by inspecting the distribution of values for each pixel and thus ranges from 0-1. The metrics aligned with our visual inspection. Directionally, the quality of reconstruction diminishes on both indices as less information is provided. For both scenarios of obscured code and adversarial attack, the reconstruction quality significantly declined. The average PSNR and SSIM for test data are summarized in the following table.

Metric	Output, Mid resolution (k=128)	Output, Less resolution (k=64)	Sample noise distortion	Adversary random guess
Avg. PSNR	24.344	16.801	12.817	11.937
Avg. SSIM	0.846	0.644	0.301	0.247

5 Conclusions (ss5945)

In general, the refined model from the second architecture, third iteration was successful. Given k values of 128 and 64, the model was able to reconstruct images with PSNR values of 24.344 and 16.801 respectively. The disparity between these scores is attributed to the fidelity of the images; as outlined in the goals of the study, modifying the value of k for sparsity not only facilitated the creation of a more compact solution, but it also served as an easy controller for the resolutions of the final images. Regardless, both of the reconstructed images boasted better PSNR and SSIM scores than the sample noise distortion and sample adversarial attacks, proving the effectivess of this model.

6 Future Work (hy2724, ss5945)

There were definite challenges encountered during this project. Nonethelss, our results were satisfactory despite only training for 10 epochs and on 1/10 of the available dataset. Some learnings taken away were to train for more time and epochs as well as more data. Given the size of the model and number of parameters, it requires more variance in the input dataset to be able to learn and generalize well. Additionally, we would not have leveraged VGG-16 filter sizes and dimensions anymore. The VGG-16 architecture was designed for a massive classification activity of identifying up to 1000 classes. It is generally claimed that there are 40 features to learn for human face reconstruction. This leads us to potentially believe that our model architecture, given its efficient success in training iteration 3, could generalize better in the context of privatizing a wider domain of images if trained appropriately.

However, despite these auspicious results, there is still room for improvement, as prominent state of the art PSNR scores are usually between 30 to 50. In order to improve this metric, further experiments could be done with the model's architecture. For example, instead of relying on a modified version of VGG16, perhaps the model could do better with some repurposed parts from VGG16's sibling, AlexNet. As AlexNet consists of only 8 layers, our model could be further simplified with even fewer parameters. Because our study was often hindered by computational limits with Google Colab's GPU, a less complex model could potentially have an easier time training on more samples from the CelebFaces Attributes Dataset. In turn, this could bolster our results and PSNR scores.

7 References (hy2724, ss5945)

1. Cai et al. (2022) Learning to Generate Realistic Noisy Images via Pixel-level Noise-aware Adversarial Training. https://arxiv.org/abs/2204.02844
2. Cha et al. (2019) GAN2GAN: Generative Noise Learning for Blind Denoising with Single Noisy Images. https://arxiv.org/abs/1905.10488
3. Ferdowsi, S. (2019) Learning to compress and search visual data in large-scale systems. https://arxiv.org/abs/1901.08437
4. Ferdowsi et. al (2020) Privacy-Preserving Image Sharing via Sparsifying Layers on Convolutional Groups. https://doi.org/10.48550/arXiv.2002.01469
5. LeCun et al. (1998) Gradient-based learning applied to document recognition. http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf
6. Razeghi et. al (2017) Privacy Preserving Identification Using Sparse Approximation with Ambiguization. https://doi.org/10.48550/arXiv.1709.10297

7. Sirichotedumrong et al. (2020) A GAN-Based Image Transformation Scheme for Privacy-Preserving Deep Neural Networks. https://arxiv.org/pdf/2006.01342.pdf
8. Shi et al. (2022) Adversarial Masking for Self-Supervised Learning. https://arxiv.org/abs/2201.13100
9. Tewari et. al (2017) MoFA: Model-based Deep Convolutional Face Autoencoder for Unsupervised Monocular Reconstruction. https://doi.org/10.48550/arXiv.1703.10580

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```