Combining Autoencoders and Classifiers for Latent Space Optimization

Gregory Nothnagel

May 2, 2024

1 Abstract

Certain applications of image generation as well as technical fields such as drug design that require input optimization over a large number of input variables benefit from techniques to generate samples that meet specific requirements. Current systems for accomplishing these tasks often require the creation of many new models for slight variations of the same task, and their architectures are far from simple. We propose a method that combines simple autoencoders and classifiers, and targets a set of specific requirements by optimizing the latent space instead of the input space. The autoencoder and classifier are trained separately, and then the decoder is appended to the front of the classifier. Optimization on the front latent space layer can then be performed, as opposed to optimizing the input layer. Once optimization is complete, the intermediate input layer holds the generated sample. This paper aims to improve on traditional input optimization techniques which nearly always lead to noisy or unclear outputs, and produce recognizable outputs comparable to Generative Adversarial Networks (GANs), and Variational Autoencoders (VAEs). The paper compares LSO with the existing methods of simple input optimization, GANs, and VAEs through experiments generating handwritten digits. Results show that LSO wildly outperforms simple input optimization in generating identifiable digits, although it falls notably short of GAN and VAE quality. The study suggests that LSO still holds promise as an extension of GANs and VAEs, and could benefit from further research and experimentation with latent space regularization.

2 Introduction

The purpose of this paper is to demonstrate the effectiveness and simplicity of latent space optimization. Various disciplines require the use of AI to generate outputs that matches specific requirements. The work in this paper would be useful for creative professionals and manufacturing institutions, as well as any discipline which requires advanced optimization techniques over a large number of input variables, such as drug design. Researchers may be interested in this

work, since the architecture proposed is modular and can be easily built upon. Currently, the most popular techniques for image generation don't involve output specification, but instead rely on creating separate models for each "class" of the output data. CGANs allow an extent of output specification, but this method as well as all other methods discussed in this paper become infeasible if one desires to generate many diverse and specific outputs, and on top of that, the attempt to enhance generative functionality in these models further complicates their architectures to a high degree, requiring a lot of fine tuning, and making them hard to adapt to one's own research. This problem can be solved by using a form of basic input optimization that operates on the latent space of a generative decoder instead of the raw input of a dataset. This solution is simple and modular, making it easy for future researchers to expand upon. Most current research, like research of the many variations of the GAN architecture, involves expanding on an already complicated architecture, which make it seem worthwhile to look at the relatively simple architecture of the latent space optimization as another possible foundation to build on.

3 Related Work

3.1 Application of Variational AutoEncoder (VAE) Model and Image Processing Approaches in Game Design

The general approach to regularizing a latent space is to normalize the activations of each layer during training or to use some form of weight decay. This paper was able to achieve single latent space which could be divided into clusters that could be randomly sampled in order to create outputs that closely match the underlying distribution of the distinct numbers 2, 6, and 7, but their approach assumes that the desired features of a generation will be part of a cluster that exists in the training dataset. This assumption is correct for generating handwritten digits, but the same technique could not be applied to domains which require generating outputs that don't fit neatly into classes, such as generating specific combinations of facial features. This paper's approach uses advanced statistical inference and Bayesian reasoning to push an autoencoder towards creating a regularized latent space which can be used for generative purposes. By compressing the dataset using a VAE, then clustering images based on their position in the latent space, this paper succeeds in generating novel samples from those clusters. But again, this solution can only generate features that exist in clusters in the dataset. If the desired features aren't well represented together in the dataset, such as blue eyes and brown hair in the case of face image generation, there will be no cluster for those features and this VAE clustering model won't be able to generate it, even though such a face should presumably exist in the latent space.

3.2 Generative adversarial network based synthetic data training model for lightweight convolutional neural networks

The common approach to improving classifier network accuracy is by expanding the training dataset in some way, generally by using preprocessing or simply including more training data. Using their GAN-ST model, this paper was able to train their classifier to an accuracy close to the accuracy of a classifier trained on the original dataset. When allowed to use a mix of original dataset and generated training data, the trained classifier outperformed a classifier trained on the original dataset, and was more generalizable. But this paper assumed that very few, discrete classes need to be represented in the training data, only the numbers 0 and 8, which is of course true for the MNIST database. But this assumption is incorrect in cases where the data that one wishes to generate cannot be split neatly into classes, such as wishing to create a molecular structure with certain scalar features. The paper used DCGAN and CGAN to create synthetic training data, which they then used to train a classifier. It used two separately trained GAN models, a DCGAN and a CGAN, with the intent of covering different parts of the data distribution of the real data, resulting in a more diverse set of training data for the classifier. While this approach succeeds in improving the quality of the classifier, it does not focus on generating images with different specific sets of properties using the same model, and requires different generative models to be trained for the numbers 0 and 8.

3.3 Sample-Efficient Optimization in the Latent Space of Deep Generative Models via Weighted Retraining

In latent space optimization, most approaches, such as the one in this paper, treat training the generative model as a separate affair from the optimization task. This paper's approach achieved significantly better chemical design optimization results as compared to other forms of weighted retraining and other methods in the literature of the chemical design task. This paper states that "all LSO methods known to us employ some sort of measure to restrict the optimization to near or within the feasible region. This means that LSO should be treated as a bounded optimization problem", but this assumption may be incorrect with a fully regularized latent space. What this paper describes ("it has been widely observed that optimizing outside of the feasible region tends to give poor results, yielding samples that are low-quality, or even invalid") is consistent with the digit image outputs of our version of LSO (figs. 4, 5). But we strongly suspect that a fully regularized latent space would turn the whole latent space into a feasible region, leading to higher quality results. Their approach to coupling generative model training with the optimization task is to assign a weight to each data point, such that the accuracy of the generator for some data points affects the loss function more than that of other data points. Their methodology is to periodically retrain the generative model during the optimization procedure, increasing the weights of data points that are high scoring

in the objective function and decreasing the weights of those with low scores. While this solution is useful for creating a generator that can create outputs that match specifications, it has essentially the same problem that occurs with GANs, as a new model must be created and periodically retrained for every new set of required specifications.

4 Approach

All of the methods referenced in this paper are unable to generate outputs that satisfy specific and uncommon requirements without requiring multiple models. This paper proposes a method that trains an autoencoder and classifier separately, and uses the decoder as a generator that can be appended to the classifier. Then, the technique of input optimization can be applied at the latent space layer by picking a random point in the latent space and optimizing the latent space to produce the desired requirements. The intermediate input space is then the result of the generation. This approach is based on realizing and removing the flaws of simple input optimization. With simple input optimization, the classifier does not know how to classify images that are outside of the feasible region of the input space, so with a random starting point a local minimum is typically reached too quickly and an incredibly noisy image results. But the idea behind this technique is that, with an ideally regularized latent space, every point in the latent space lies within the feasible region. In other words, every point in the latent space decodes into a recognizable number. Since the classifier knows how to handle every point produced by the latent space, this makes for smoother gradient descent with less local minima, which makes optimization feasible. The closest thing we have seen to this exact approach is the application of another gradient during GAN training [1] to try to guide the optimization of the latent space towards having the desired properties. This is one approach that is commonly referred to as Latent Space Optimization (LSO), but it differs from our approach in that the work happens on the latent space during training, instead of afterwards as a variation of input optimization, and still maintains the weakness of needing new models for new requirements. In theory, our approach should work because the latent space should have far fewer "meaningless" infeasible points than the input space, which are points that the classifier classifies unpredictably. This should make the latent space's error function less noisy and thus have more infrequent local minima, making optimization towards the target requirements smoother. We used C++ to create and combine the simple autoencoders and classifier because we were trying to make a model that to our knowledge may not be possible to create easily using other machine learning libraries, and because we appreciated the degree of control that building from scratch allows. Our LSO approach does not require the creation of new algorithms, but existing algorithms that are used include backpropagation and gradient descent to train neural networks of different structures, and input optimization to optimize the latent space (algs. 1, 2, 3). No new mathematical formulas were created in this paper, but new models were created (figs. 3, 4, 5).

Algorithm 1 Backpropagation for Nodes

for each node N in the network **do**
    Initialize the derivative value ND of N to zero
    for each node M in the following layer **do**
        Add (weight of N w/ respect to M) * (derivative of M w/ respect to total error function) to ND
    end for
    Adjust the ND using the activation function of N
end for

Algorithm 2 Backpropagation for Parameters

Since each node in all but the last layer of the network has only one bias parameter, let's call it NB
for each node N in all but the last layer of the network **do**
    Set NB to derivative value of N
    for each node M in the following layer **do**
        Let NWM be the weight of N with respect to M
        Set the derivative NWM to M * (derivative of M with respect to total error function)
    end for
end for

Algorithm 3 Gradient Descent

Let V be the set of all values in the network that are going to be changed (this could be parameters in the case of training, or first layer node values in the case of input optimization)
Let D be the set of derivatives corresponding to V
Let C be the desired change of the error function
Compute the squared sum S of the numbers in D
for each parameter or node in V **do**
    Add to it its corresponding derivative in D multiplied by C/S
end for

Figure 1: A simple autoencoder that encodes and decodes 14x14 pixel handwritten digit images.

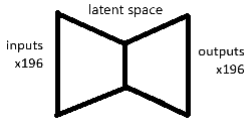


Figure 2: A simple classifier of handwritten digits using one-hot encoding.

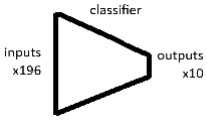
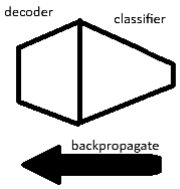


Figure 3: Combining the decoder and classifier. This new model now operates like any normal neural network.



We expect this LSO model architecture to perform better than simple input optimization.

5 Experiments

We expect the new approach to perform better than simple input optimization at generating recognizable numbers. We will compare the generated images of this algorithm with those of simple input optimization as well as those of GAN and VAE networks. We will compare our method of LSO with simple input optimization to show that it is a viable extension of and significant improvement on simple input optimization. We will also compare our method of LSO with GAN and VAE outputs to see how LSO compares to the current standard generative models. The hypotheses we will test is that LSO is superior to simple input optimization, and that LSO using a simple autoencoder's decoder produces images with the same quality as GANs and VAEs. We assume that as many attempts as needed are allowed to the LSO to generate an image to the specifications. This seems reasonable because LSO typically requires very few retries before finding a local minimum that is low enough to meet our threshold of .01 total squared error. We will test these hypotheses by generating handwritten digit images with two LSO models, LSO-1 that has a latent space of size 5 and LSO-2 with a latent space of size 20. These tests are simulations, since there is no feasible way to test this algorithm with image generation physically. 100 images were generated with each model. The hardware to train both models was an Asus Zenbook laptop running the C++ training and generating program which utilizes the above algorithms [algo. 1, 2, 3]. We will analyze the generations of the LSO-1 and LSO-2 models and compare them with GAN and VAE handwritten digit image generations. The frequency of generation success with each of the LSO models were recorded, and the accuracy of these frequencies is dependent on the number of test images generated (100 in our case).

5.1 Results Summary

LSO-1 (figure 4) had 5 latent space nodes and was trained on 1,000 samples from the MNIST database of handwritten digits, and reached an Mean Squared Error (MSE) of .0239 on the 1,000 training samples. 64% of LSO-1 generations succeeded to meet the MSE requirement of .001. LSO-2 (figure 5) had 20 latent space nodes and was trained on 10,000 examples from the same database, and reached an Mean Squared Error (MSE) of .0203 on the 10,000 training samples. 82% of LSO-2 generations succeeded to meet the MSE requirement of .001. Both LSO models used the same classifier.

6 Analysis

The hypothesis that LSO outperforms simple input optimization is supported by the data qualitatively, since simple input optimization yields noisy images

Figure 4: LSO-1 generations (5-node latent space) (0 on left, 8 on right)

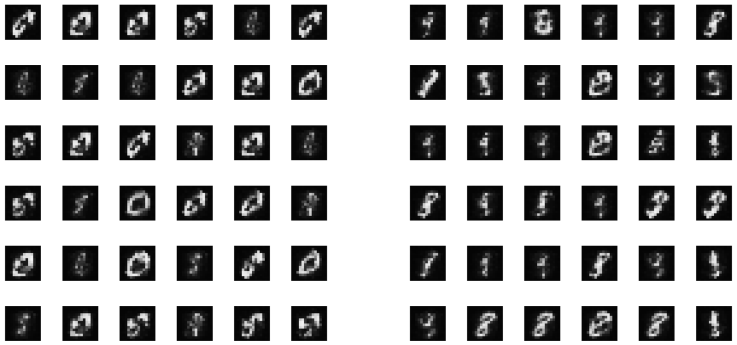


Figure 5: LSO-2 generations (20-node latent space) (0 on left, 8 on right)

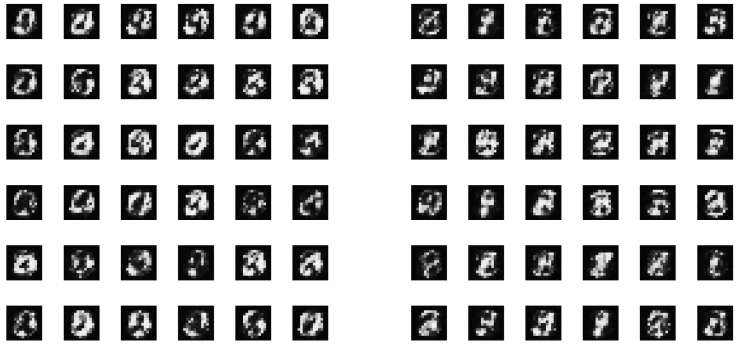


Figure 6: Figure a depicts an input optimization generation of a certain number whose quality is so poor that the author of this paper can't remember what it was supposed to be. Figure b is an LSO-2 generation of 8. Figure c is a GAN generation of 8 [3]. It should be known that figures a and b were trained to the same MSE threshold of .01

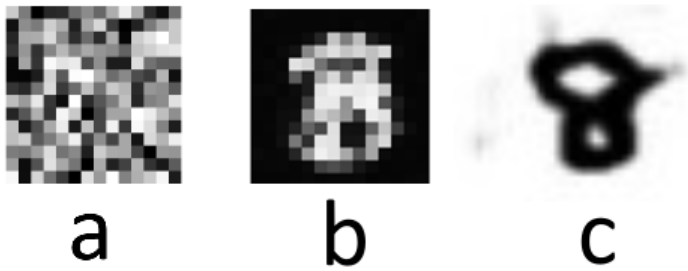
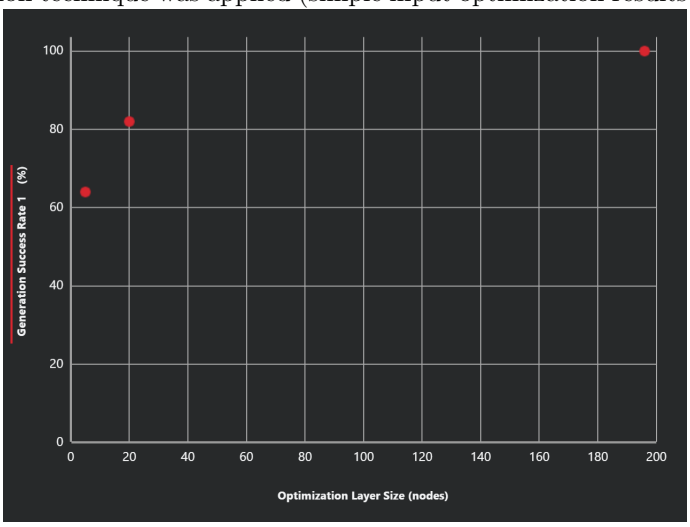


Figure 7: Percent success rate of generation vs size of layer at which input optimization technique was applied (simple input optimization results included)



that, though they meet the .01 squared error threshold, have only a vague appearance of having been altered in some way. Meanwhile, a solid portion of LSO generations are identifiable as the number they are supposed to represent, and they are much less noisy than the input optimization generations. However, the hypothesis that LSO using a simple autoencoder's decoder would perform as well as GAN and VAE was not supported by the data, because the GAN and VAE generations appear to have significantly higher quality. LSO-1 was less frequently able to find a local minimum that met the threshold (64% of generations met the MSE threshold, as opposed to 82% for LSO-2). This is surprising because we expected the lower-dimensional latent space to be more regular and have fewer "meaningless" local minima. But perhaps 5 dimensions was simply not enough to capture the underlying distribution of numbers to a degree at which the .01 threshold could be met. Interestingly enough, simple input optimization almost never fails to meet the same .001 MSE threshold, implying that dimension reduction leads to worse performance in latent space optimization. It is unclear how or if this would change if the latent space of the decoder was more regularized. Using LSO with the decoder with the smaller latent space led to a lower success rate of generations whose local minima were able to meet the threshold of .001 MSE (fig. 7). A factor that was not measured that affected the data was the MSE of the classifier. While the MSE for the classifier is normally recorded, the scores for the classifier model used in LSO-1 and LSO-2 was accidentally overwritten, though it was likely very low for it to be thought fit for use in these models. The effects of this are definitely insignificant in comparing LSO-1 and LSO-2 since they both use that same classifier, and the effects of this in comparing LSO to GAN is very likely insignificant, since the

MSE for the classifier was most likely very low. Another relevant experiment that would add more important data would be generating numbers with LSO using the generator of a GAN or VAE, instead of the decoder of a simple autoencoder. This may reveal that LSO produces high quality images comparable to GAN when a GAN decoder is used, or it may reveal that the quality of LSO is determined by more than the sum of its decoder and classifier parts. Latent space irregularity is a relevant phenomenon that is not apparent when viewing static produced images, and only somewhat apparent when viewing the latent space walk that leads to the final generated image, in that there are clearly long periods of nearly vanishing gradients where little change occurs, as well as large jumps where the image and MSE change drastically towards a higher quality. This qualitative perception of a vanishing gradient (and the general knowledge that autoencoders generally yield irregular latent spaces) indicates the presence of latent space irregularity, even though there is sadly no data in these experiments to support that claim.

7 Conclusion

Various disciplines benefit from AI that is able to generate samples that match specific requirements. We hypothesized that LSO would perform better at generating recognizable handwritten digits than simple input optimization, and that LSO would perform as well as GANs and VAEs. We found through testing that LSO can generate handwritten digits with a degree of fidelity higher than that of simple input optimization, but that LSO is still not at the level of GAN of VAE generations. Such a substantial improvement over simple input optimization suggests that LSO could be an area for future research, and these results are certainly generalizable in that LSO is likely to outperform simple input optimization at generative tasks in other domains as well. Since latent space quality appears to be the limiting factor of LSO, the generalization that "GANs and VAEs perform better than LSO with a simple autoencoder" should hold true for any domain in which a simple autoencoder is led during training to create a comparatively irregular latent space. The experiments in this paper can be easily extended by using decoders from other state-of-the-art architectures like GANs.

8 References

[1] L. Sanchez-Lengeling and A. Aspuru-Guzik, "Inverse molecular design using machine learning: Generative models for matter engineering," Science, vol. 361, no. 6400, pp. 360-365, 2018.

[2] W. L. Mak et al., "Application of Variational AutoEncoder (VAE) Model and Image Processing Approaches in Game Design," Sensors, vol. 23, no. 7, p. 3457, Mar. 2023. [Online]. Available: https://doi.org/10.3390/s23073457.

[3] I. Hussain and S. Kumar, "Generative adversarial network based synthetic

data training model for lightweight convolutional neural networks," Multimedia Tools and Applications, pp. 1-23, May 20, 2023, doi: 10.1007/s11042-023-15747-6.

[4] A. Tripp, E. Daxberger, and J. M. Hernández-Lobato, "Sample-Efficient Optimization in the Latent Space of Deep Generative Models via Weighted Retraining," arXiv preprint, arXiv:2006.09191 [cs.LG], June 2020.