GAN-UNET: Non-negative Matrix Factorization Using Deep Neural Networks

Sihan Wei
Department of Computer Science & Engineering
University of Minnesota, Twin Cities
Minneapolis, MN, 55414
wei00114@umn.edu

Abstract

We introduce an end-to-end deep neural network framework GAN-UNET for non-negative matrix factorization (NMF). As a generative model, we use ReLU in the generator to inherently guarantee the non-negativity of basis vectors and weights. We test our model on different datasets to assess the performance in terms of reconstruction error, compared with many other classic algorithms. Such experiments do demonstrate that our method has a generalization ability under different scenarios. Further, since the generator itself is an autoencoder, which is also a common method in solving NMF problems, an ablation study is conducted to discover the benefits and necessity of each module in our model. In the end, we discuss the ability of symmetry breaking of our model.

1 Introduction

Non-negative matrix factorization (NMF) has been widely applied in astronomy, computer vision, audio signal processing, etc. In this project, we intend to utilize a deep neural network to build a generalized NMF solver by considering NMF as an inverse problem. Generally, an NMF problem is stated as follows. Given a non-negative matrix $\mathbf{Y} \in \mathbb{R}^{n\times m}$ and a non-negative integer $k \leq \min(n, m)$, the NMF problem requires to find two non-negative matrices $\mathbf{H} \in \mathbb{R}^{n\times k}$ and $\mathbf{W} \in \mathbb{R}^{k\times m}$ that minimize the loss function

$$\min_{\mathbf{H},\mathbf{W}} \mathcal{L}\left(\mathbf{H},\mathbf{W}\right) = \min_{\mathbf{H},\mathbf{W}} \|\mathbf{Y} - \mathbf{HW}\|_F,$$

where the product $\mathbf{HW}$ is an approximate factorization of $\mathbf{Y}$ of rank at most k.

We consider the NMF problem as an inverse problem where the forward process is $\mathbf{Y} = f(\mathbf{Y}) = \mathbf{HW}$ and the inverse process is to retrieve a pair of matrices, $(\mathbf{H},\mathbf{W}) = g(\mathbf{Y}) = \arg\min_{\mathbf{H},\mathbf{W}} \|\mathbf{Y} - \mathbf{HW}\|_F$, such that $\mathbf{H} \geq 0, \mathbf{W} \geq 0$. Our project aims to using deep neural network to learn the function of $g(\cdot)$.

In general, the inverse problem aims to analyze the process of generating causal factors from a set of observations. For example, given $y = f(x) = x^2$, we expect to get its inverse function $x = g(y)$. However, multiple x's are mapping to y, which is called symmetry. It leads to that the inverse function $g(\cdot)$ maybe very oscillation and hence is hard for deploying the end-to-end deep learning approach. Similarly, in NMF problem, the symmetry exists. Given a pair $(\mathbf{H},\mathbf{W})$ that suffices $\min \|\mathbf{Y} - \mathbf{HW}\|_F$, for any non-negative invertible matrix $\mathbf{B}$, pair $(\mathbf{H}',\mathbf{W}') = (\mathbf{HB},\mathbf{B^{-1}W})$ is still a solution of this problem. Due to multiple possibilities of $\mathbf{B}$, it is hard to naively use a deep neural network to train the decomposition model. The key to solving this problem is to break the symmetry. In our project, we will use a deep neural network to solve the symmetry problem implicitly.

Our Contributions: In this paper, we propose **GAN-UNET**, an end-to-end framework, to solve the NMF problem. We test on different datasets to assess the performance of our model in terms of reconstruct error, compared with many other algorithms. An ablation study is also conducted to discover the benefits and necessity of each module in our model.

2 Related Work and Limitations

In 1999, Lee and Seung [1] found that NMF could extract meaningful features from facial images. They also find that Multiplicative Update (MU) rules are a good compromise between speed and ease of implementation for solving NMF problems. Alternating Non-negative Least Squares (ANLS) [2] and Hierarchical Alternating Least Squares (HALS) [3] are also two classic algorithms in this field.

The earliest reference, to the best of our knowledge, using neural networks to implement NMF also comes from the work of Lee and Seung [4], where they refer to NMF as "conic coding". However, their work does not gain much attention until the rise of deep learning over the past decade. Ehsan et al. [5] demonstrate a deep learning autoencoder network with constraints on non-negative hidden encodings and weights. Although the main purpose of their paper is not to implement NMF, they do show the potential of generative models, such as autoencoder, in improving the interpretability and performance of deep neural networks.

The non-negativity constraint of NMF could be obtained easily with a nonlinear activation layer (e.g., Sigmoid, ReLU) in neural network implementations, while obtaining non-negative basis vectors could be more complex. Lemme et al [6, 7] introduce an autoencoder-based model using an asymmetric regularization term to penalize negative weights so that they could enforce non-negative basis vectors. But their approach in general does not guarantee the non-negativity because the resulting weights may still very likely contain negative values. Chorowski and Zurada [8] present a similar approach using "project gradient descent" where they set the negative weights to zero in each iteration. However, such method of setting negative weights to zero seems to be arbitrary and unreasonable, and may end with bad reconstruction performance in practice. In [9], Alaa et al. propose using autoencoders to implement NMF with what they call "exponential gradient descent" update rule which smoothly maintains the non-negativity of the basis vectors during training. They experiment their method in clustering applications while it is not that convincible whether their model works well in reconstructing images. Flenner and Hunter [10] introduce a deep NMF framework capable of producing an interpretable hierarchical classification of many types of data. However, they did not discuss how to handle non-unique solutions (i.e., symmetry breaking) of NMF problems. However, these neural network approaches list above are not generalized enough. In other words, for different input matrix/image groups, we need to train different models. In 2020, Li et al. try to build a generalized model by using conditional Generative Adversarial Networks hybridized with classic method [11]. They use a least square solver to get $\mathbf{W}$ and continue using the HALS iterations to ensure the non-negativity of $\mathbf{W}$. Due to the additional process, it is still not generalized enough.

Recently, Tayal et al. [12] pay attention to the symmetry breaking in solving the generalized phase retrieval problem using DNN. They break the symmetry by carefully selecting training points to satisfy the following three properties: connected, representative, and smallest. In this work, we intend to use a similar idea to deal with a different type of symmetry in the NMF problem. Then we will reach our ultimate goal: build a powerful end-to-end NMF solver using data after breaking symmetry.

3 Approach

Before introducing our method, we firstly address three challenges in our project:

- **Infinite large amount of samples for training**: ideally, we will use many pairs of data like $(\mathbf{Y}, (\mathbf{H}, \mathbf{W}))$ to train our DNN. However, if the expected DNN model can perform well on any arbitrary inputs $\mathbf{Y}$, the sample size for training will be infinitely large. To counter this issue, we narrow down the ability of our model to one specific application scenario where the input matrix $\mathbf{Y}$ is an intrinsically low rank, such as image analysis.

- **No ground truth of $(\mathbf{H}, \mathbf{W})$ given an input $\mathbf{Y}$**: in a specific application scenario, there is no the ground truth for the matrix $\mathbf{Y}$. To find the exact NMF of matrix $\mathbf{Y}$ is NP-Hard in general. Thus, there is no training data $(\mathbf{H}, \mathbf{W})$ to training the inverse function $g(\cdot)$.
- **Actively breaking symmetry may be a issue**: suppose we have large enough samples and ground truth data to train the model, the next step is breaking the symmetry. We observed two kinds of symmetries: permutation and scaling. However, to keep training samples consistent as [12] did, we should first identify one connected subspace in the matrix space. Due to the two challenges listed above, we cannot actively break the symmetry now.

To encounter the three challenges, we select images as our application and take advantage of the conditional Generative Adversarial Networks (cGAN) which is also used in [11]. We learn a generator to implicitly generate $\mathbf{H}$ and $\mathbf{W}$ in a latent space and use the forward process, i.e. $\langle \mathbf{H}, \mathbf{W} \rangle$, to give reconstructed images, $\hat{\mathbf{Y}}$. The discriminator tries to distinguish the reconstructed images from the source images, which give an implicit regularization to our reconstructed images.

3.1 Loss Function

The loss function of cGAN is used in the project, which is

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_x[\log(1 - D(x, G(x))], \tag{1}$$

Where G tries to minimize this objective against an adversarial D that tries to maximize it, furthermore, an L1 reconstruction loss is added to the generator because we want the reconstructed image not only to be close to the distribution of source images but also to be the same as the input source images. We use L1 distance because it tends to maintain high-frequency details in images and produce more sparse basis vectors, $\mathbf{H}$.

$$\mathcal{L}_{rec}(G) = \mathbb{E}_{x,y}\left[\|y - G(x)\|_1\right]. \tag{2}$$

Thus, the final objective of generator is

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{rec}(G). \tag{3}$$

Unlike [13], We do not add random noise as input for the generator because the expectation for the generator is to learn a better mapping from x to x itself, which is essentially an autoencoder.

3.2 Network Architectures

We adapted the network design idea from those in [13]. Fig. 1 shows our overall model architecture. The generator will output reconstructed images. When training the discriminator, the reconstructed images will be concatenated to the source images as the input of discriminator. The detailed design is described below.

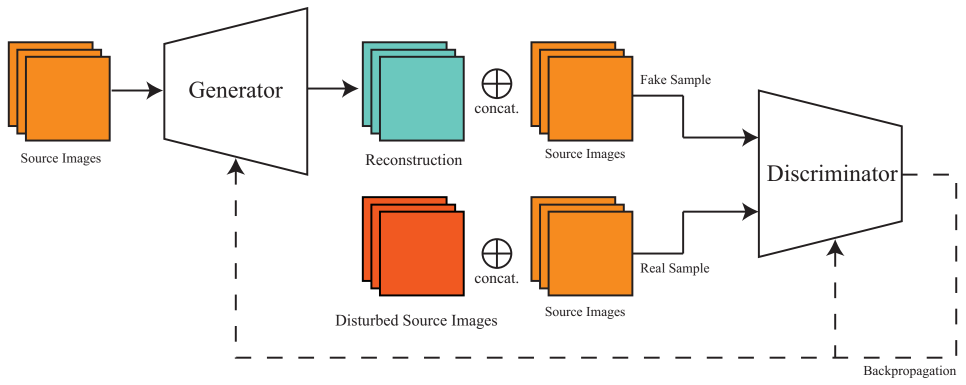


Figure 1: Architecture of Our Model

3.2.1 Generator

U-Net [14] is a contemporary architecture nowadays because the skip connection allows the model to choose the depth of the network itself, which gives it a good expressiveness. As Fig. 2 shows, we utilize U-net architecture as part of our generator to generate the basis component, $\mathbf{H}$, and we further modified the architecture to generate component importance matrix, $\mathbf{W}$. By going through the forward process, $\langle \mathbf{H}, \mathbf{W} \rangle$ in the NMF problem, the generator would output the reconstructed images. To ensure the non-negativity of $\mathbf{H}$ and $\mathbf{W}$, we use ReLU as the last layer of the output.

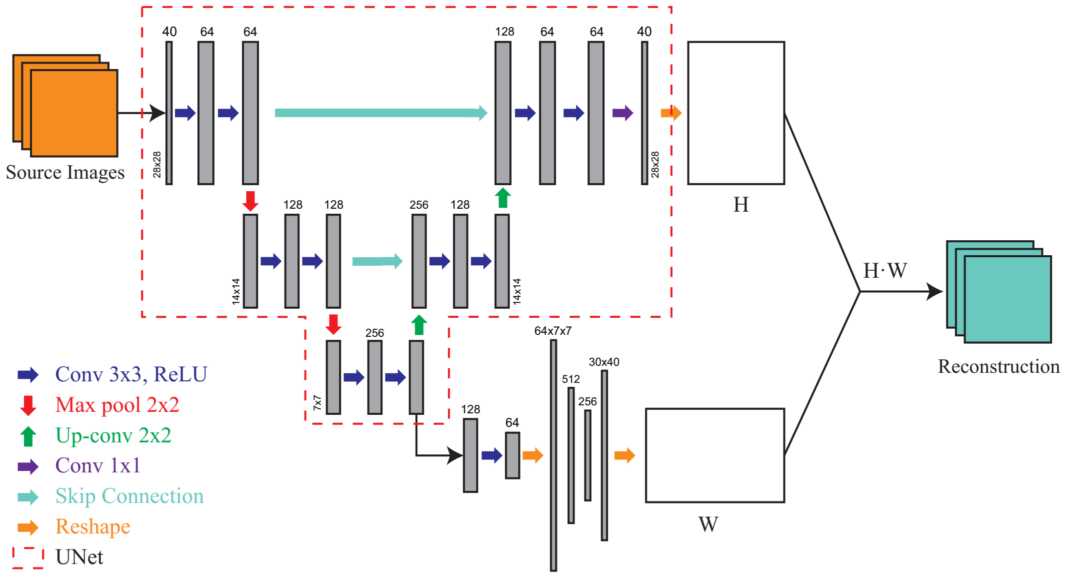


Figure 2: Generator

3.2.2 Discriminator with PCA Disturbing

Similar to [11], the discriminator takes both the reconstructed images and the source images as input. They are concatenated together and fed into the network. Thus, the discriminator takes $2m$ channels of images as input and classify real or fake using feature maps calculated by convolutional layers. The detailed architecture is shown in Fig. 3.

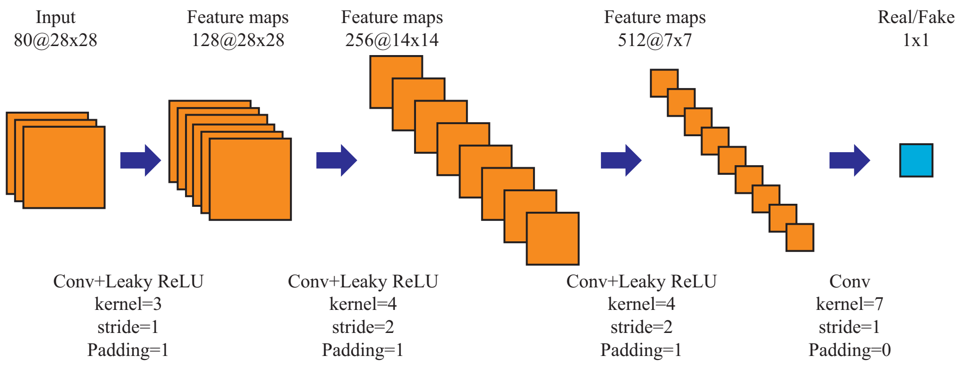


Figure 3: Discriminator

Similarly, when training discriminator on real samples, we disturb the source images by using principal component analysis (PCA) reconstruction. The reason for putting disturbing information on source images is that input a pair of identical images to the discriminator is too easy for it to find out whether it is real or fake. Additionally, images reconstructed by PCA are perceptually similar to those reconstructed by NMF, which is also useful guidance for the generator. To further weaken the discriminator's ability, we randomly label real or fake when training the real samples.

4 Validation Methodologies and Results

In this section, we are going to explore the reconstruction and generalization capabilities of our model. To illustrate that each part of our model is beneficial and essential, some ablation experiments are also conducted.

4.1 Setup

Datasets: We did experiments on three different datasets. Part of the raw data is shown in Fig. 4. MNIST [15] is a database of handwritten digits, containing 60,000 train examples and 10,000 test examples. Each example is a 28x28 grayscale image and classified into one of ten classes, i.e. number 0 to 9. Fashion [16] is a dataset of Zalando's, containing 60,000 train examples and 10,000 test examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. Face [17] is a human face dataset taken by AT&T Laboratories Cambridge. There are 10 different 64x64 images of each of 40 distinct people. We take 320 images as the training set and the remaining 80 as the test set.

Preprocessing: We first flatten all images in the dataset into column vectors and combine every 40 of them as a group. The input of our model $\mathbf{Y} = \{\mathbf{Y}_1, \mathbf{Y}_2, \cdots, \mathbf{Y}_{40}\} \in \mathbb{R}^{(m \times n) \times 40}$ where (m, n) is the size of each raw image. In this way, $\mathbf{H}$ can be regarded as the basis vectors and $\mathbf{W}$ are linear weights. In the process of creating a group, we keep the class balance. For example, for the MNIST dataset, we pick four images from each class, while for the face dataset, only select one image from each class.

Baselines: We use Alternating Non-negative Least Squares (ANLS), Multiplicative Update (MU), and Hierarchical Alternating Least Squares (HALS) as our baselines.

Metrics: We use the reconstruction error, which is defined by Eq. 4 to evaluate the ability of the model quantitatively. We also take advantage of the image to describe the reconstruction result qualitatively.

$$\epsilon\left(\widehat{\mathbf{Y}}, \mathbf{Y}\right) = \frac{1}{40} \sum_{i=1}^{40} \left\| \widehat{\mathbf{Y}}_i - \mathbf{Y}_i \right\|_F \tag{4}$$

Where $\widehat{\mathbf{Y}} = \mathbf{HW}$.

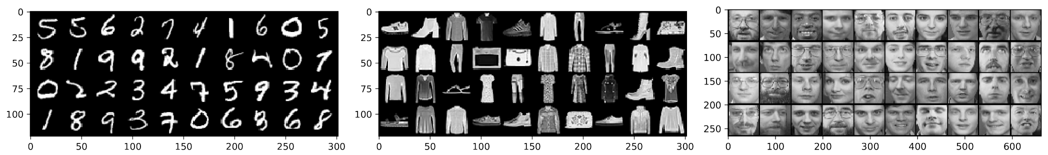
4.2 Reconstruction Capability

The reconstruction errors of ANLS, MU, HALS, GAN-UNET on three datasets are shown in Tab. 1. Experiments show that our generative model can perform NMF. Although the accuracy of our model on the training set is not as good as the classic methods, the comparison among them on the test set indicates that our model has stronger generalization ability (we will further explore it in Section 4.3). The test performance on Face dataset is not good because the size of dataset is small. Thus, the training data space is under-sampling which may lead to an overfitting.

Table 1: The reconstruction error of ANLS, MU, HALS, GAN-UNET on three datasets.

	ANLS		MU		HALS		GAN-UNET	
	train	test	train	test	train	test	train	test
MNIST	**1.533**	9.794	1.801	9.850	3.265	9.417	3.847	**3.933**
Fashion	**1.265**	11.189	2.063	10.991	1.717	11.077	3.622	**4.122**
Face	**1.778**	12.147	2.994	11.709	1.894	12.083	3.305	**9.911**

Fig 4 is the visualization of a data group. The decomposition results of GAN-UNET and HALS in this group are shown in Fig. 5 and Fig. 6, which shows both of them get meaningful decomposition. From the decomposition result of MNIST and Fashion dataset, we can see that they both have some blurred images and our model is comparable to HALS. From the decomposition result of Face dataset, we can also see that there are more black parts of the face, i.e. the features are more prominent, which means GAN-UNET learns a more sparsity result.



(a) MNIST (b) Fashion (c) Face

Figure 4: A group of raw data.

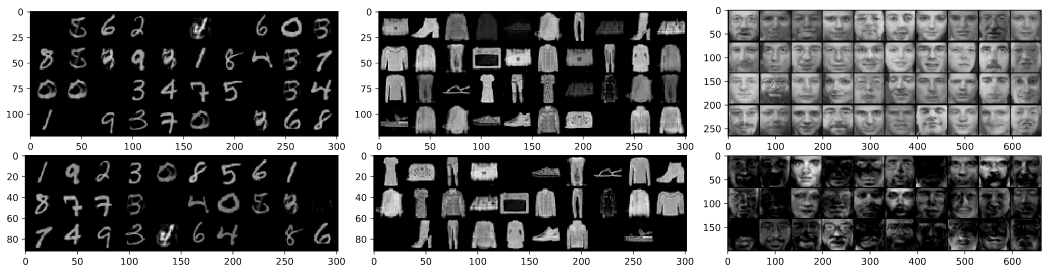


Figure 5: The decomposition results obtained by GAN-UNET. The first row is the reconstruction result of the corresponding dataset. The second row is the basis $\mathbf{H}$ of the corresponding dataset.

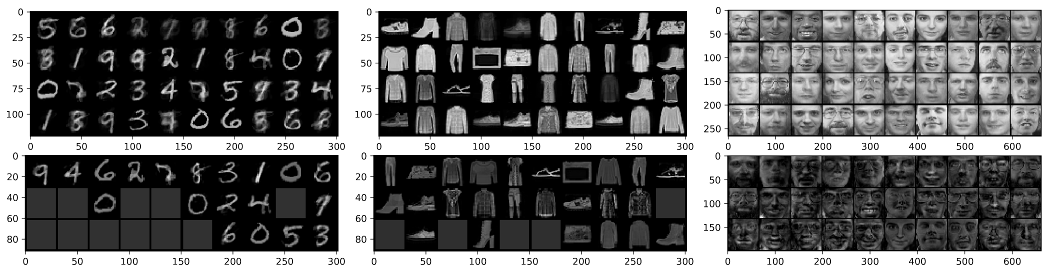
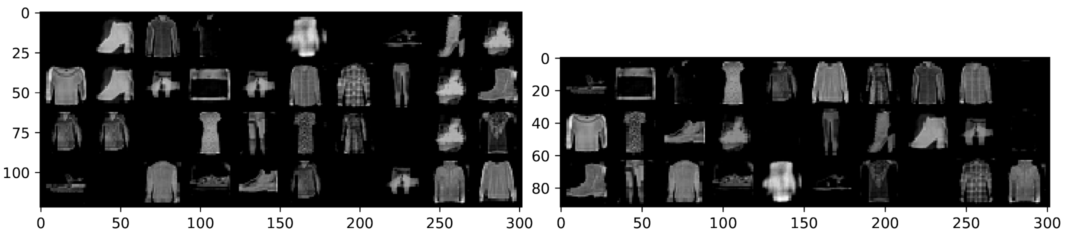


Figure 6: The decomposition results obtained by HALS. The first row is the reconstruction result of the corresponding dataset. The second row is the basis $\mathbf{H}$ of the corresponding dataset.

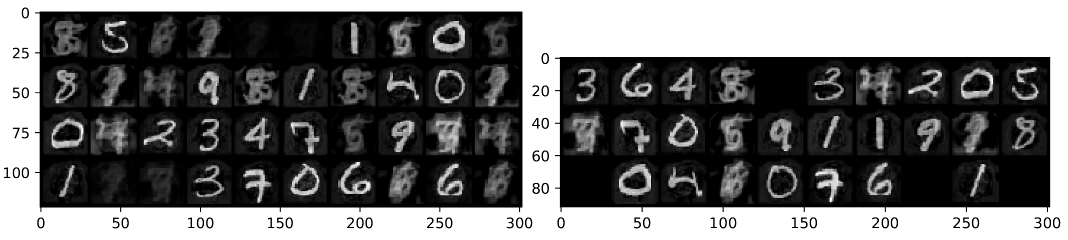
4.3 Generalization Capability

To illustrate the generalization capability of our model,we try to decompose different datasets, as shown in Fig. 7 and 8. From these figures, we can find that our model can still decompose and get some useful features.



(a) The reconstruction result (b) Generated basis $\mathbf{H}$

Figure 7: Train on MNIST, test on Fashion.



(a) The reconstruction result (b) Generated basis $\mathbf{H}$

Figure 8: Train on Fashion, test on MNIST.

4.4 Ablation Study: Conditional GAN

We further explore the effectiveness of cGAN in our method. Without the adversarial discriminator, we consider the generator as an autoencoder. To ensure the consistent experiment setting, we use the same L1 reconstruction loss as the optimization objective and test both on MNIST dataset. The quantitative reconstruction comparison is shown in Tab. 2 and the qualitative visualization is shown in Fig. 9.

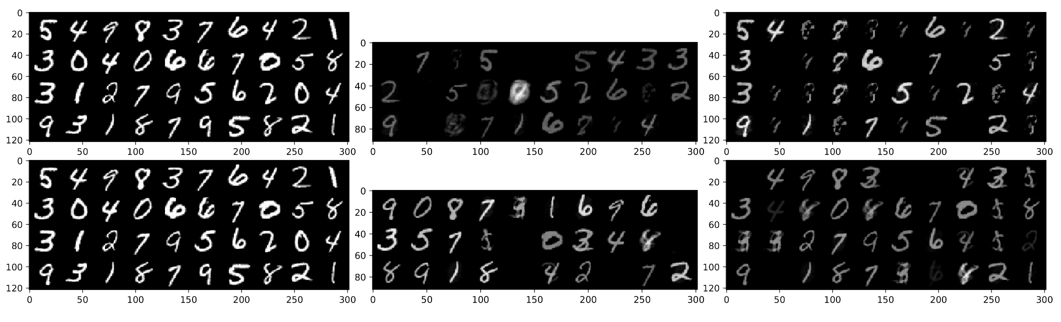


Figure 9: The qualitative comparison between UNET and GAN-UNET. The first row is the result of UNET. The second row is the result of GAN-UNET. The first column is the input data. The second column is the decomposed basis components, $\mathbf{H}$. The third column is the reconstruction result of the corresponding dataset.

Table 2: The reconstruction error of UNET and GAN-UNET on MNIST datasets.

	train	test
UNET	5.371	5.374
GAN-UNET	**3.847**	**3.933**

From the results, we can see our GAN-UNET model has a better perceptual reconstruction and lower reconstruction error. Thus, we can tell the conditional GAN is helping the generator to give a better reconstruction and generate high quality basis, $\mathbf{H}$.

5 Discussion and Conclusion

5.1 Implicit Symmetry Breaking

As discussed in 1, the inverse problem may suffer from many-to-one mapping in forward process. In our method, we use neural network to implicitly break the symmetry. To validate the effectiveness, several basis components regarding to different input image groups are shown as in Fig. 10 where the Face dataset is used, the input $\mathbf{Y} \in \mathbb{R}^{4096\times 10}$, and $k = 10$.

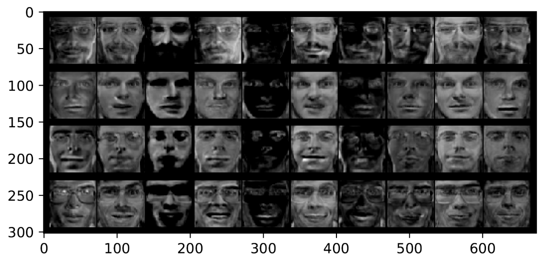


Figure 10: 4 visualized basis components regarding to 4 different input image groups

From the visualization, we can see there is a significant perceptual consistency among different basis. For example, all of the fifth column features are about eyes and mouth. Additionally, the intensity of components are roughly the same crossing the rows. Therefore, we can tell the neural network is helping to break the symmetry in NMF problem. Due to limited time, we do not have a rigours proof here. So, we leave it as a discussion.

5.2 Convex Optimization Differential Programming Attempted

Inspired by [11] and [18], we also tried to use `cvxpylayer` to solve a non-negative least square problem to get $\mathbf{W}$ instead of using fully connected layers. However, the open-sourced `cvxpylayer` library is still at a primitive stage. It does not support GPU and multi-threading acceleration. We show a preliminary result here which runs only 20 epochs as in Fig. 11.

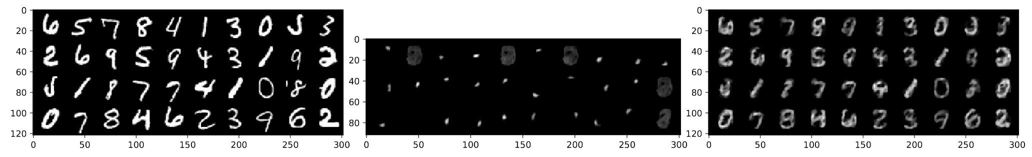


Figure 11: The qualitative visualization of `cvxpylayer` method. The first column is the input data. The second column is the decomposed basis components, $\mathbf{H}$. The third column is the reconstruction result of the corresponding dataset.

In the result, we can see the learned basis, $\mathbf{H}$ is more sparse. This is a totally expected result of NMF. Compared with classic methods, such as HALS, the reconstructed images are more perceptually meaningful. Due to the low efficiency and limited time, no further experiments are conducted on this model.

5.3 Conclusion

In this paper, we gain some insights into solving the NMF problem using generative models. We explore the generalization capability and reconstruction performance of neural networks under different scenarios. We also dive into the symmetry breaking from the perspective of inverse problems, and try convex optimization differential programming. It is likely that the true potential of our approach has yet to be reached.

Acknowledgment

We thank Prof. Ju Sun for providing advice on the project.

References

[1] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[2] H. Kim and H. Park, "Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method," *SIAM journal on matrix analysis and applications*, vol. 30, no. 2, pp. 713–730, 2008.

[3] A. Cichocki and A.-H. Phan, "Fast local algorithms for large scale nonnegative matrix and tensor factorizations," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 92, no. 3, pp. 708–721, 2009.

[4] D. D. Lee and H. S. Seung, "Unsupervised learning by convex and conic coding," in *Advances in neural information processing systems*, pp. 515–521, 1997.

[5] E. Hosseini-Asl, J. M. Zurada, and O. Nasraoui, "Deep learning of part-based representation of data using sparse autoencoders with nonnegativity constraints," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 12, pp. 2486–2498, 2015.

[6] A. Lemme, R. F. Reinhart, and J. J. Steil, "Efficient online learning of a non-negative sparse autoencoder.," in *ESANN*, Citeseer, 2010.

[7] A. Lemme, R. F. Reinhart, and J. J. Steil, "Online learning and generalization of parts-based image representations by non-negative sparse autoencoders," *Neural Networks*, vol. 33, pp. 194–203, 2012.

[8] J. Chorowski and J. M. Zurada, "Learning understandable neural networks with nonnegative weight constraints," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 1, pp. 62–69, 2014.

[9] A. El Khatib, S. Huang, A. Ghodsi, and F. Karray, "Nonnegative matrix factorization using autoencoders and exponentiated gradient descent," in *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2018.

[10] J. Flenner and B. Hunter, "A deep non-negative matrix factorization neural network," 2017.

[11] D. Li, Z. Gao, X.-P. Zhang, G. Zhai, and X. Yang, "Generative adversarial networks for non-negative matrix factorization in temporal psycho-visual modulation," *Digital Signal Processing*, vol. 100, p. 102681, 2020.

[12] K. Tayal, C.-H. Lai, V. Kumar, and J. Sun, "Inverse problems, deep learning, and symmetry breaking," 2020.

[13] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[15] Y. LeCun and C. Cortes, "MNIST handwritten digit database." http://yann.lecun.com/exdb/mnist/, 2010.

[16] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[18] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter, "Differentiable convex optimization layers," 2019.