A Non-Negative Matrix Decomposition Framework for Testing Noise Robustness

Haixu Liu
The University of Sydney
hliu2490@uni.sydney.edu.au

Zerui Tao
The University of Sydney
ztao0063@uni.sydney.edu.au

Naihui Zhang
The University of Sydney
nzha3727@uni.sydney.edu.au

Sixing Liu
The University of Sydney
sliu2285@uni.sydney.edu.au

Abstract

This paper illustrated the robustness of Non-negative Matrix Factorization algorithms, taking Salt and Pepper Noise, Block Occlusion Noise, and Gaussian Noise into account respectively. To compare the performance and robustness of Traditional NMF, L_1-NMF and $L_{2,1}$-NMF, these algorithms were tested on the ORL dataset and the Extended YaleB dataset. In this paper, Section 1 contains a short introduction to this report, and Section 2 introduces some previous work about NMF algorithms. We explained the theoretical method of this study in Section 3. Then we conducted experiments and the results are shown in Section 4. Finally, there is a short conclusion in Section 5 and some suggestions are also proposed here. In addition, there is a link and introduction of the code used in this research in Appendix A.

1 Introduction

Non-negative Matrix Factorization (NMF) is a linear method used for data dimension reduction and feature extraction by decomposing a data matrix into two low-order metrics, which was initially introduced by Lee and Seung.[1] It requires that the elements of all matrices involved must be non-negative. Compared with other matrix factorization methods such as Principal Component Analysis (PCA) and Singular Value Decomposition (SVD), the non-negative constraint makes NMF more interpretable and provides physical significance for real-world applications.[2] Taking the field of computer vision as an example, NMF ensures the characteristics of features and weights in the image decomposition process are non-negative, which matches the fact that image pixel values cannot be negative and makes decomposition results more stable.

Based on the basic characteristics of NMF, we will validate the performance of the NMF algorithm on the ORL dataset and the YaleB dataset respectively in this study and explore its robustness when image data is heavily contaminated by noise. Considering that NMF has a wide range of applications, such as collaborative filtering in recommendation algorithms[3], topic modeling in natural language processing[4], gene expression data analysis[5], and sound separation in audio signal processing[6], which is not limited to computer vision, the discussion focusing on noise interference in NMF is very important. In theory, this research may promote further understanding and development of the NMF algorithms; on the other hand, it will provide a fundamental starting point for NMF research considering noise and lays the foundation for its potential expansion into more fields of application.

According to Yu-Xiong Wang and Yu-Jin Zhang, NMF algorithms can be devided into four categories: Basic NMF (BNMF), Constrained NMF (CNMF), Structured NMF (SNMF) and Gen-

eralized NMF (GNMF).[7] In this study, we will focus on BNMF (which only satisfies the non-negativity constraint) and CNMF (which contains some additional constraints as regularization). More specifically, we will mainly compare the performance of traditional NMF algorithm and $L_{2,1}$-Norm Based NMF in the contexts of three different types of noise: Salt and Pepper Noise, Block Occlusion Noise and Gaussion Noise. Meanwhile, we will also consider L_1-Norm Based NMF as an extension content.

2 RELATED WORK

Referring to the review of previous NMF algorithms in the truncated Cauchy NMF[8] by Guan et al., published in TPAMI in 2019, we selected and reviewed some representative NMF algorithms.

2.1 Traditional NMF(1999)

Traditional NMF[9], also known as L_2-Norm MUR NMF, adopts the multiplicative update rule (MUR[9]) and assumes that noise follows a Gaussian distribution and derives the following squared L_2-Norm based objective function: $min_{W\geq 0,H\geq 0}\|V-WH\|_F^2$, where $\|X\|_F=\sqrt{\sum_{ij}X_{ij}^2}$ signifies the matrix Frobenius norm. Due to the favorable mathematical properties of the squared L_2-Norm and the efficiency of MUR, the classical NMF method has been incorporated into the well-known machine learning library Scikit-learn, with extensions that include SVD initialization and the coordinate descent parameter update method. However, NMF and its extensions lack robustness, as the L_2-Norm is highly noise-sensitive.

2.2 Hypersurface Cost Based NMF(2006)

Hamza and Brady[10] introduced a hypersurface cost based NMF (HCNMF) by minimizing the summation of hypersurface costs of errors, i.e., $\min_{W\geq 0,H\geq 0}\left\{\sum_{ij}\delta((V-WH)_{ij})\right\}$, where $\delta(x)=\sqrt{1+x^2}-1$ is the hypersurface cost function. Hamza and Brady [10] highlighted that the hypersurface cost function is differentiable and has a bounded influence function, enabling HCNMF to be efficiently solved using the projected gradient method. In terms of performance, while HCNMF demonstrates strong robustness against outliers, the use of Armijo's rule-based line search results in high time complexity, which limits its applicability in large-scale scenarios.

2.3 L_1-Norm Based NMF(2008)

Many researchers have studied algorithms based on the L_1-norm for NMF. Kong et al. [11] derived an L_1-NMF algorithm through the computation algorithm used in $L_{2,1}$-NMF. For better numerical stability of the algorithm, $|(V-WH)_{ij}|$ is replaced by $\left((V-WH)_{ij}^2+\epsilon^2\right)^{1/2}$, where ϵ is set to a very small number.It uses MUR to optimize parameters, making it highly efficient in computation, but it lacks robustness in practice. Additionally, Lam [12] assumed that noise is independent and identically distributed from Laplace distribution and proposed L_1-NMF as follows $min_{W\geq 0,H\geq 0}\|V-WH\|_1$, where $\|X\|_1=\sum_{ij}|X_{ij}|$ and $|\cdot|$ signifies the absolute value function. Due to the non-smooth nature of the L_1-norm based loss function, the optimization algorithm is not scalable for large-scale datasets. Moreover, Manhattan NMF [13] resolves the issue of the non-smoothness in the L_1-NMF loss function by approximating it with a smooth function and minimizing it using Nesterov's method.

2.4 L_1-Norm Regularized Robust NMF(2011)

To improve the robustness of L_1-NMF, Zhang et al.[14] assumed that the dataset contains both Laplace distributed noise and Gaussian distributed noise and proposed an L_1-norm regularized Robust NMF(RNMF-L_1) as follows: $\min_{W\geq 0,H\geq 0,S}\{\|V-WH-S\|_F^2+\lambda\|S\|_1\}$, where λ is a positive constant that trades off the sparsity of S. According to Guan [8], similar to L_1-NMF, RNMF-L_1 is also less sensitive to outliers than NMF, but they are both non-robust to large numbers of outliers because the L_1-minimization model has a low breakdown point. And, it is important to fine the optimal $\mathrm{parameter}\ \lambda$.

2.5 $L_{2,1}$-Norm Based NMF(2011)

As NMF primarily sums the squared L_2-norm of the errors, large errors tend to dominate the objective function, making NMF less robust. However, methods based on the L_1-norm often cannot be directly solved using MUR, resulting in high time complexity. Kong et al.[11] proposed the $L_2, 1$-norm based NMF ($L_{2,1}$-NMF) which minimizes the $L_2, 1$-norm of the error matrix, i.e., $\min_{W\geq 0, H\geq 0} \|V - WH\|_{2,1}$, where the $L_{2,1}$-norm is defined as $\|E\|_{2,1} = \sum_{j=1}^{n} \|E_{\cdot j}\|_2$. And a set of weighted multiplicative update rules has been provided. Compared to NMF, $L_{2,1}$-NMF offers improved robustness with only a modest increase in computational time, as both methods utilize the multiplicative update rule.

2.6 Correntropy lnduced Metric Based NMF(2012)

Correntropy-Induced Metric-based NMF (CIMNMF) is the half-quadratic algorithm for optimizing robust NMF [15]. CIM-NMF measures the approximation errors by using Correntropy-Induced Metric(CIM) [16], i.e., $\min_{W\geq 0, H\geq 0} \sum_{i=1}^{m} \sum_{j=1}^{n} \rho((V - WH)_{ij}, \delta)$, where $\rho(x, \delta) = 1 - \frac{1}{\sqrt{2\pi}\delta} e^{-\frac{x^2}{2\delta^2}}$. Because the energy function $\rho(x, \delta)$ increases slowly as the error increases, CIM-NMF is insensitive to outliers. However, CIM-NMF still has certain drawbacks. Firstly, it is highly sensitive to the choice of kernel bandwidth parameters and the type of noise. Secondly, since CIM-NMF requires the computation of nonlinear similarity measures (correntropy) for each sample, the computational cost escalates rapidly on large-scale datasets, leading to inefficiencies.

2.7 Truncated CauchyNMF (2019)

Truncated CauchyNMF is considered the latest state-of-the-art NMF algorithm, inspired by CauchyNMF. Its objective function is given by:

$$\sum_{ij} g\left(\left(\frac{(V - WH)_{ij}}{\gamma}\right)^2\right), \quad \text{where } g(x) = \begin{cases} \ln(1 + x), & 0 \leq x \leq \sigma \\ \ln(1 + \sigma), & x > \sigma \end{cases},$$

and γ is a truncation parameter, and σ is the scale parameter. It is derived from the proposed Truncated Cauchy loss which can model both modearte and extreme outliers [8]. It iteratively detects outliers by the robust statistics on the magnitude of errors and obtains the optima for each factor in each iteration round by solving the weighted non-negative least squares (WNLS) problems [8].

3 Methods

3.1 Pre-Processing

During the pre-processing, in addition to conventional image downsampling operations, we consider deleting images with the suffix "Ambient.pgm" to reduce the impact of irrelevant images on clustering results and increase the credibility of metrics. Moreover, normalization of the image is considered necessary, as the range of Gaussian noise typically has a mean of 0, with the maximum standard deviation controlled within 0.25. By normalizing images, the pixel values will shift from $(0, 255)$ to the $(0, 1)$ interval, which is similar to the range of noise values. In this case, the impact of noise on the image will not be approximated as 0 due to a significant difference from the range of pixel values, which ensures that our research on the noise robustness of NMF algorithms is meaningful. It is worth mentioning that the pixel values after adding Gaussian noise may exceed the $(0, 1)$ interval, so the np.clip method will be used to readjust the pixel values back.

3.2 Traditional NMF Optimisation

Given a nonnegative matrix V of dimensions $n \times m$ and a desired rank $k \leq \min(m, n)$, the aim of NMF is to find two nonnegative matrices W of dimensions $n \times k$ and H of dimensions $k \times m$, such that the difference between the original data V and the approximation by the product of WH is minimized.[17]

3.2.1 NMF Objective Function

To quantify the difference between V and WH, a squared L_2-Norm-Based cost function (Frobenius norm) is defined. To find the optimal W and H, the objective function should be minimized as shown in Eq. 1:

$$\min_{W \in \mathcal{W}, H \in \mathcal{H}} \|V - WH\|_F^2 \tag{1}$$

where

$$\mathcal{W} = \mathbb{R}_+^{d \times k}, \quad \mathcal{H} = \mathbb{R}_+^{k \times n}$$

and

$$\|V - WH\|_F^2 = \text{Tr}((V - WH)^\top (V - WH))$$

3.2.2 NMF Optimization Steps

The NMF optimization problem is defined as Eq. 2:

$$\min_{W \in \mathbb{R}^{n \times k}, H \in \mathbb{R}^{k \times m}} \|V - WH\|_F^2 \quad \text{s.t.} \quad W, H \geq 0 \tag{2}$$

Using the formula $\|X\|^2 = \text{Tr}(X^\top X)$, we can express the Frobenius norm as:

$$\|V - WH\|_F^2 = \text{Tr}((V - WH)^\top (V - WH)) \tag{3}$$

This expression can be further expanded as:

$$\text{Tr}((V - WH)^\top (V - WH)) = \text{Tr}(V^\top V - V^\top WH - H^\top W^\top V + H^\top W^\top WH) \tag{4}$$

Using the property $\text{Tr}(A + B) = \text{Tr}(A) + \text{Tr}(B)$, we obtain Eq. 5:

$$\|V - WH\|_F^2 = \text{Tr}(V^\top V) - \text{Tr}(V^\top WH) - \text{Tr}(H^\top W^\top V) + \text{Tr}(H^\top W^\top WH) \tag{5}$$

Minimizing $\|V - WH\|_F^2$ is challenging due to the nonnegativity constraints and the non-convexity of the objective function when considering both W and H matrices together. [1] Considering the convexity would guarantee that any local minimum is a global minimum, when $\|V - WH\|_F^2$ is treated as a function of either W and H alone, the problem becomes convex and easy to solve. However, the joint optimization over both matrices does not satisfy this property, making the problem complex to find local minima.

To address the non-convexity challenges, an iterative algorithm Multiplicative Update Rules (MUR) can be applied. The MUR iteratively updates W and H by fixing one while optimizing the other one, thereby gradually reducing the approximation error $\|V - WH\|_F^2$. Additionally, while the MUR does not guarantee a global minimum due to its non-convex nature, it can effectively find a good local minimum.[18]

To optimize the problem using MUR, a block coordinate descent scheme is implemented to optimize for H first while keeping W fixed, and then optimize for W while keeping H fixed:

$$H \leftarrow H - \eta_H \cdot \nabla_H \|V - WH\|_F^2 \tag{6}$$

$$W \leftarrow W - \eta_W \cdot \nabla_W \|V - WH\|_F^2 \tag{7}$$

where η_W and η_H are the learning rates for W and H, respectively.

To find the derivatives of the objective function with respect to W and H, formulas $\nabla_A \text{Tr}(AB) = B^\top$, $\nabla_A \text{Tr}(A^\top B) = B$ and the trace's cyclic property $\text{Tr}(ABC) = \text{Tr}(CAB)$ is applied.

Therefore, given Eq. 5, we can calculate the derivative of each term with respect to W as follows:

$$\nabla_W \text{Tr}(V^\top V) = 0$$

$$-\nabla_W \text{Tr}(V^\top WH) = -\nabla_W \text{Tr}(WHV^\top) = -(HV^\top)^\top = -VH^\top$$

$$-\nabla_W \text{Tr}(H^\top W^\top V) = -\nabla_W \text{Tr}(W^\top V H^\top) = -VH^\top$$

$$\nabla_W \text{Tr}(H^\top W^\top WH) = \nabla_W \text{Tr}(WHH^\top W^\top) = W((HH^\top)^\top + HH^\top) = 2WHH^\top \tag{8}$$

Summing up each term's derivative, we obtain the Eq. 9:

$$\nabla_H \|V - WH\|_F^2 = -2W^\top V + 2W^\top WH \tag{9}$$

Similarly, the derivatives with respect to H of can be computed as above, shown as Eq. 10:

$$\nabla_W \|V - WH\|_F^2 = -2VH^\top + 2WHH^\top \tag{10}$$

Using the above derivatives, the update rules for W and H can incorporate the constant 2 into the learning rate.

Firstly, based on Eq. 9, updating H while fixing W:

$$H \leftarrow H + \eta_H \cdot (W^\top V - W^\top WH) \tag{11}$$

Then, based on Eq.10, updating W while fixing H:

$$W \leftarrow W + \eta_W \cdot (VH^\top - WHH^\top) \tag{12}$$

where the new learning rates η_W and η_H include the factor of 2 from the original derivatives in Eq. 6 and Eq. 7.

Lee and Seung proposed using adaptive learning rates to avoid subtraction in the update rules, and thus there were no negative elements.[17] The main idea is to set the step size parameters to such values with the result that the additive update rules become multiplicative update rules, enforcing the nonnegativity constraints. More precisely, setting the learning rates as:

$$\eta_W = \frac{W}{WHH^\top} \quad \text{and} \quad \eta_H = \frac{H}{W^\top WH} \tag{13}$$

For a certain number of iterations $\ell = 0, 1, 2, \ldots, L$ for $L \in \mathbb{N}$, define nonnegative matrices $W^{(0)}$ and $H^{(0)}$ by some random initialization, and the update rule for H becomes Eq. 14:

$$H_{rn}^{(\ell+1)} = H_{rn}^{(\ell)} + \frac{H_{rn}^{(\ell)}}{(W^\top WH^{(\ell)})_{rn}}((W^\top V)_{rn} - (W^\top WH^{(\ell)})_{rn}) = H_{rn}^{(\ell)} \cdot \frac{(W^\top V)_{rn}}{(W^\top WH^{(\ell)})_{rn}} \tag{14}$$

Similarly, the update rule for W converts to Eq. 15:

$$W_{kr}^{(\ell+1)} = W_{kr}^{(\ell)} + \frac{W_{kr}^{(\ell)}}{(WHH^\top)_{kr}} \cdot \left((VH^\top)_{kr} - (WHH^\top)_{kr}\right) = W_{kr}^{(\ell)} \cdot \frac{(VH^T)_{kr}}{(W^{(\ell)}HH^T)_{kr}} \tag{15}$$

3.3 $L_{2,1}$-Norm NMF

The traditional NMF is widely used in sparsity problems and feature extraction but is prone to noises and outliers. A few outliers with large errors easily dominate the objective function due to the Frobenius norm objective function of $\|V - WH\|_F^2$. [18] Moreover, in the traditional NMF, both matrices W and H are unknown, making the impact of outliers much more complicated. [11] Thus a robust version of NMF is crucial and a robust formulation of NMF using $L_{2,1}$-norm loss function has been established.

3.3.1 Cost Function of $L_{2,1}$-Norm NMF

The loss function of $L_{2,1}$-norm is defined as:

$$\|V - WH\|_{2,1} = \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{p} (V - WH)_j^2} = \sum_{i=1}^{n} \|v_i - Wh_i\| \tag{16}$$

In this robust formulation, the error for each data point is $\|v_i - Wh_i\|$, which is not squared. As a result, when the outliers cause large errors, it does not dominate the objective function. To find the optimal W and H, the objective function should be minimized as shown in Eq. 17:

$$\min_{W,H} \|V - WH\|_{2,1} \quad \text{s.t.} \quad W \geq 0,\ H \geq 0 \tag{17}$$

3.3.2 $L_{2,1}$-Norm NMF Optimization Steps

The $L_{2,1}$-Norm NMF optimization problem is defined as:

$$\min_{W,H} \|V - WH\|_{2,1} \quad \text{s.t.} \quad W \geq 0,\ H \geq 0 \tag{18}$$

To facilitate the optimization of the $L_{2,1}$-norm, a diagonal matrix D is introduced [11], where each diagonal entry D_{ii} is the inverse of the L_2 norm of the corresponding i-th data point's residual row $r_i = v_i - Wh_i$:

$$D_{ii} = \frac{1}{\|r_i\|_2} \tag{19}$$

With the use of diagonal matrix D, the $L_{2,1}$-Norm NMF algorithm update rule follows:

firstly, updating H while fixing W:

$$H_{ki} \longleftarrow H_{ki} \frac{(W^T V H)_{ki}}{(W^T W H D)_{ki}}, \tag{20}$$

then, updating W while fixing H:

$$W_{jk} \longleftarrow W_{jk} \frac{(V D H^T)_{jk}}{(W H D H^T)_{jk}} \tag{21}$$

It can be shown that under this updating rule, the objective function of Eq.17 decreases monotonically and the converged solution for both W and H satisfies the KKT condition. [11] Moreover, Not only the $L_{2,1}$-Norm NMF algorithm is effective with simplicity, but also its computational cost is almost the same as the traditional NMF.

3.4 L_1-Norm NMF

To address the sensitivity of the outliers, the L_1-Norm NMF has also been introduced, which is a robust variant of the NMF. The L_1-Norm NMF focuses on minimizing the absolute differences between the original matrix and the approximated product of the factor matrices[11]. As introduced in Section 2.3, there are multiple implementation methods of the L_1-Norm NMF, and we chose the one based on MUR by Kong et al[11]. This method has the highest computational efficiency in terms of iterations, but since the objective function is non-smooth, gradient-based update algorithms may affect its inherent robustness. Here, we choose this method merely for comparison with the $L_{2,1}$-Norm NMF algorithm mentioned in the same paper.

3.4.1 Cost Function of L_1-Norm NMF

Given $V \in \mathbb{R}^{n \times p}$ is the input data matrix and $W \in \mathbb{R}^{n \times k}, G \in \mathbb{H}^{k \times p}$ are the non-negative matrices, the most robust formulation of the L_1-NMF cost function is Eq. 22:

$$\|V - WH\|_1 = \sum_{i=1}^{n} \sum_{j=1}^{p} |(V - WH)_{ji}| \tag{22}$$

However, for better numerical stability of the algorithm, $|(V - WH)_{ij}|$ is replaced by $\left((V - WH)_{ij}^2 + \epsilon^2\right)^{1/2}$, where ϵ is set to a very small number. [11] As a result, the objective function can be written as Eq. 23:

$$J_1 = \|V - WH\|_1 = \sum_{i=1}^{n}\sum_{j=1}^{p}\left((V - WH)_{ij}^2 + \epsilon^2\right)^{1/2} \tag{23}$$

3.4.2 L_1-Norm NMF Optimization Steps

The optimization problem of L_1-NMF can be formulated as 24:

$$\min_{W,H} J_1(W, H) \quad \text{s.t.} \quad W \geq 0,\ H \geq 0. \tag{24}$$

The iteratively updating algorithm for Eq. 24 is shown as:

$$W_{jk} \longleftarrow \frac{[X \circ FH^T]_{jk}}{[(WH) \circ FH^T]_{jk}}, \tag{25}$$

$$H_{ki} \longleftarrow \frac{[W^T X \circ F]_{ki}}{[W^T(WH) \circ F]_{ki}}, \tag{26}$$

where F is a matrix given by $F_{ij} = \left((V - WH)_{ij}^2 + \epsilon^2\right)^{-1/2}$ and $\circ$ is the element-wise product between two matrices. According to the definition in the original paper, the Hadamard product has a higher multiplication priority than regular matrix multiplication in the update rules [11].

3.5 Noise Choice and Analysis

3.5.1 Salt and Pepper Noise

This type of noise is also known as Impulse Valued Noise or Data Drop Noise. Salt and Pepper Noise alters certain pixel values in an image, where the affected pixels are replaced with either the maximum possible value, 255 (Salt noise), or the minimum value, 0 (Pepper noise). Randomly modifying pixel values generate the noise throughout the image. There is one parameter to adjust Salt and Pepper noise: prob. The prob parameter controls the noise level, with prob = 0.1 meaning that 20% of the image's pixel values are modified as the noise is added in equal proportions of Salt and Pepper (10% for each), where Salt noise sets pixels to white, and Pepper noise sets pixels to black. Figure 1 shows salt and pepper noise applied to an image with different prob values.

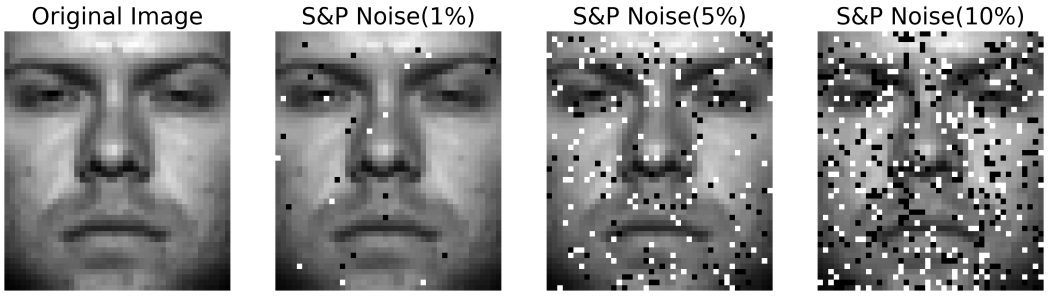


Figure 1: Example of Salt and Pepper Noise.

3.5.2 White Block Noise

The removal of contiguous segments of an object due to occlusion is a challenging problem in computer vision as large numbers of outliers with large magnitudes must be ignored to learn a clean subspace. In this experiment, we utilize contiguous occlusion to simulate extreme outliers. Specifically, we randomly position a $b \times b$-sized block on each face image of the dataset and fill the

pixel values in these blocks with 255. Figure 2 shows block occlusion noise applied to an image with different block size. To study the influence of outliers, the value of block size b is specified as 10, 12 or 14 in this research, and the number of the blocks will be set as 1 or 2.

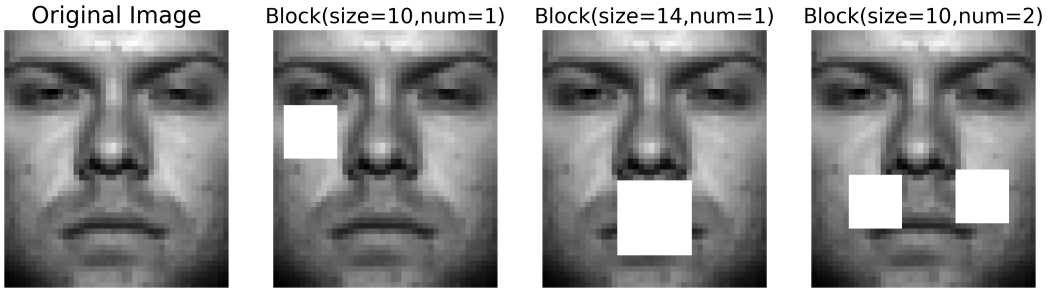


Figure 2: Example of White Block Noise.

3.5.3 Gaussian Noise

Gaussian noise is commonly found in electronic devices, particularly in amplifiers and detectors, which is typically caused by irregular fluctuations in electronic signals. It causes pixel values in an image to randomly deviate from their original gray values, making the image more blurry or grainy. AS for the influence of parameters, the higher standard deviation σ corresponds to more pronounced fluctuations in grayscale values. Figure 3 shows Gaussian Noise applied to an image with different standard variations.

Due to the normalized nature of the Gaussian noise curve, noise values are primarily concentrated around the mean μ (which is set as 0 in this study). As a result, the majority of the modified pixel values in an image will experience relatively small perturbations, with approximately 70% to 90% of them remaining within a limited range around the original value. This gives Gaussian noise a relatively smooth effect, as opposed to Salt and Pepper noise which generates more extreme deviations.

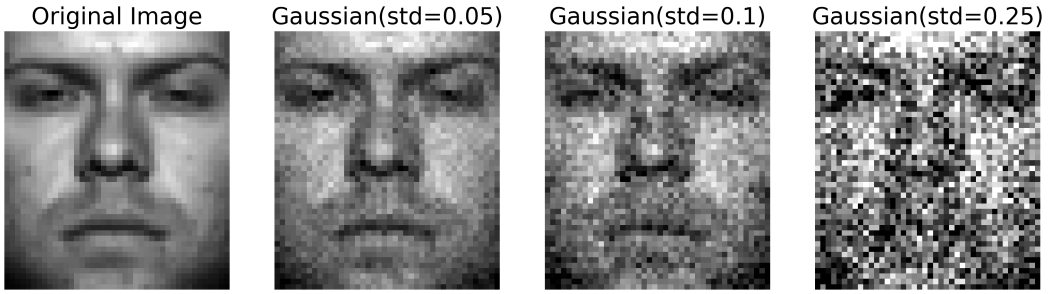


Figure 3: Example of Gaussian Noise.

3.5.4 Robustness Analysis

Although Non-negative Matrix Factorization (NMF) and its variants have already been widely used for matrix decomposition problems, the robustness of different NMF approaches to noise is still a research focus as it can vary significantly. In this section, we will analyze the traditional NMF, $L_{2,1}$-NMF and L_1-NMF methods from the perspective of their robustness to noise by a sample-weighted procedure interpretations.

In NMF, we aim to decompose a non-negative matrix $V \in \mathbb{R}_+^{m\times n}$ into the product of two non-negative matrices $W \in \mathbb{R}_+^{m\times r}$ and $H \in \mathbb{R}_+^{r\times n}$, such that $V \approx WH$. The objective function of the NMF problem is usually the minimization of the squared Frobenius norm between V and WH,

which is defined as:

$$F(WH) = \frac{1}{2}\sum_{i,j}(V_{ij} - (WH)_{ij})^2 \tag{27}$$

A scalar variable can be introduced as $t \in \mathbb{R}$ and the function $f(t)$ can be defined as following accordingly[8]:

$$f(t) = F(tWH) = \frac{1}{2}\sum_{i,j}(V_{ij} - t(WH)_{ij})^2 \tag{28}$$

By taking the derivative $f'(t)$, we can verify the equivalence of the NMF problem, which is to find a pair of matrices W and H such that $f'(1) = 0$. Here, $f'(t)$ represents the derivative of the objective function.

Taking the derivative of $f(t)$ with respect to t, we obtain:

$$f'(t) = -\sum_{i,j}(V_{ij} - t(WH)_{ij})(WH)_{ij} \tag{29}$$

Setting $t = 1$, we get:

$$f'(1) = -\sum_{i,j}(V_{ij} - (WH)_{ij})(WH)_{ij} \tag{30}$$

The NMF problem is equivalent to finding a pair of matrices W and H such that $f'(1) = 0$, leading to the condition:

$$\sum_{i,j}(V_{ij} - (WH)_{ij})(WH)_{ij} = 0 \tag{31}$$

Based on this, we define the contribution function for each element (i, j) as:

$$c(V_{ij}, WH) = (V_{ij} - (WH)_{ij})(WH)_{ij} \tag{32}$$

This function reflects the contribution of the element (i, j) to the overall optimization process. The error function is defined as:

$$e(V_{ij}, WH) = |V_{ij} - (WH)_{ij}| \tag{33}$$

This represents the noise or deviation at the (i, j)-th entry of V. We use $c(V_{ij}, WH)$ as the contribution basis because NMF aims to find W and H such that the sum of contributions across all elements is zero, i.e.,

$$\sum_{i,j} c(V_{ij}, WH) = 0 \tag{34}$$

This process is sensitive to noise, making it a robust method to handle errors. The contribution function $c(V_{ij}, WH)$ is designed to incorporate the residual $(V_{ij} - (WH)_{ij})$ and the predicted value $(WH)_{ij}$, which allows the optimization procedure to account for noise when updating the matrices. Hence, the NMF optimization can be viewed as a contribution-weighted procedure in which each element is weighted by the level of noise present in the data.

L_2 - Norm NMF The objective function for traditional NMF is defined as:

$$F(WH) = \|V - WH\|_F^2 = \sum_{i,j}(V_{ij} - (WH)_{ij})^2 \tag{35}$$

The derivative with respect to this objective function is given by:

$$f'(1) = \sum_{i,j} 2c(V_{ij}, WH) = \sum_{i,j} 2(V_{ij} - (WH)_{ij})(WH)_{ij} \tag{36}$$

The traditional NMF objective function uses the squared error, which makes it sensitive to large deviations or outliers. If there is noise in the data V, especially in the form of outliers, the squared loss function magnifies the impact of these large errors because the loss grows quadratically with the magnitude of the error. Therefore, standard NMF tends to be less robust to noise, as large deviations in V will disproportionately affect the solution, leading to overfitting of the noise.

$L_{2,1}$-Norm NMF $L_{2,1}$-Norm NMF modifies the objective function to minimize the $L_{2,1}$-norm of the difference between V and WH. The $L_{2,1}$-norm is defined as the sum of the Euclidean norms of the rows (or columns) of a matrix. Here, the definition of the objective function is:

$$F(WH) = \|V - WH\|_{2,1} = \sum_j \left(\sum_i (V_{ij} - (WH)_{ij})^2 \right)^{1/2} \tag{37}$$

In this expression, the inner summation over i calculates the Euclidean norm of the j-th column of $V - WH$, and the outer summation over j sums these norms.

To analyze the robustness of $L_{2,1}$-Norm NMF to noise, we need to compute the derivative of $f(t) = F(tWH)$ with respect to t.

First, we define:

$$f(t) = \sum_j \left(\sum_i (V_{ij} - t(WH)_{ij})^2 \right)^{1/2} \tag{38}$$

Taking the derivative with respect to t, we get:

$$f'(t) = -\sum_j \left(\sum_i (V_{ij} - t(WH)_{ij})^2 \right)^{-1/2} \sum_i (V_{ij} - t(WH)_{ij}) (WH)_{ij} \tag{39}$$

When $t = 1$, the derivative becomes:

$$f'(1) = -\sum_j \frac{1}{e_j} \sum_i c(V_{ij}, WH) \tag{40}$$

The $L_{2,1}$-Norm NMF method effectively treats the residuals for each column as a whole and penalizes the overall magnitude of the errors between groups. The $L_{2,1}$-norm grows linearly with the Euclidean norm of the residual vector in each column, which helps reduce the impact of individual outliers. By grouping residuals, $L_{2,1}$-Norm NMF is robust to outliers or noisy entries in the rows or columns of the data matrix V. In the derivative $f'(1)$, the contribution of each column is weighted by the inverse of its residual's Euclidean norm, which reduces the influence of columns with large residuals (i.e., noisy or outlier columns) on the overall derivative.

L_1-Norm NMF In contrast, L_1-Norm NMF modifies the objective function to minimize the L_1-norm of the difference between V and WH. The objective function is given by:

$$F(WH) = \|V - WH\|_1 = \sum_{i,j} |V_{ij} - (WH)_{ij}| \tag{41}$$

The derivative of this function is:

$$f'(1) = \sum_{i,j} \frac{1}{|V_{ij} - (WH)_{ij}|} c(V_{ij}, WH) = \sum_{i,j} \frac{(V_{ij} - (WH)_{ij})(WH)_{ij}}{|V_{ij} - (WH)_{ij}|} \tag{42}$$

The L_1-NMF method uses the absolute value of the residuals instead of their squared values. This means that the objective function grows linearly with the magnitude of the error, making it less sensitive to large deviations or outliers. As a result, L_1-NMF is more robust to noise, particularly when the noise is characterized by outliers. The influence of large deviations in V is mitigated, as the linear growth in the loss function does not overly penalize large errors. L_1-NMF is thus more appropriate for scenarios where the data contains outliers or noise, as it tends to produce solutions that are less influenced by those noisy entries.

4 Experiments and Discussion

4.1 Dataset Description

In this report, the experiments were conducted on the ORL Dataset of Faces and the Extended YaleB dataset given by teaching team of COURSE COMP5329 at USYD.

The ORL dataset contains 400 face images of 40 distinct subjects (10 photos for each subjects), which were divided into 40 subdirectories according to the objects, each containing 10 images of the corresponding objects. These photos were taken in different real-life environments, causing differences in lighting, facial expressions, and facial details(with or without glasses). All images provided in this dataset were cropped and resized to 92×112 pixels, which were further processed in the pre-processing stage to a size of 30×37 pixels to reduce the computational complexity. The grayscale value of each pixel ranges from 0 to 255.

Another dataset is the Extended YaleB dataset, which contains 2414 images of 38 subjects in total. These images involve 9 poses and 64 illumination conditions. Similar to the ORL dataset, the images are aligned manually and cropped to 168×192 and then downsampled to 42×48 during pre-processing.

4.2 Evaluation Metrics

For all experiments, three evaluation metrics were implemented to compare the performance and robustness of the different NMF algorithms:

- **Relative Reconstruction Errors (RRE):** RRE measures the difference between the reconstructed data and the original data. Let X denote the contaminated data, and $\hat{X}$ denote the clean data. Meanwhile, let D and R denote the factorization results on X, the relative reconstruction errors can be described as below:

$$RRE = \frac{\|\hat{X} - DR\|_F}{\|\hat{X}\|_F} \tag{43}$$

- **Average Accuracy:** As the images belong to different classes, a K-means clustering algorithm was performed on the reconstructed matrix with the **number of clusters** equal to the **number of classes**. Each sample was assigned a cluster label, and then the average accuracy can be denoted as below:

$$ACC(Y, Y_{pred}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{Y_{pred}(i) = Y(i)\} \tag{44}$$

- **Normalized Mutual Information (NMI):** NMI can measure how similar two clusters are(a higher value indicates a greater degree of similarity) [19], which can be defined by:

$$NMI(Y, Y_{pred}) = \frac{2 \cdot I(Y, Y_{pred})}{H(Y) + H(Y_{pred})} \tag{45}$$

  where $I(\cdot, \cdot)$ is mutual information and $H(\cdot)$ is entropy.

To ensure a rigorous evaluation, all experiments were repeated 5 times by randomly sampling 90% of the data from the entire dataset. Thus, averaging the performance metrics across the 5 subsets can provide a more reliable estimate of each algorithm's performance. It is particularly noteworthy that we found a difference in the RRE calculated after NMF reconstruction for images with grayscale ranges of 0-1 (after normalization) and 0-255 (the range in the provided dataset). The RRE for the latter is an order of magnitude larger than the former. Although we normalized the images during preprocessing, in order to reflect the algorithm's true performance on the original dataset, we calculated the RRE between the inverse-normalized reconstructed image and the original image without normalization. However, some published papers, due to using datasets with inconsistent grayscale ranges, uniformly used the normalized images to calculate the RRE. As a result, the values reported in those papers are smaller than the results we obtained, but this does not affect the model's performance in terms of NMI and ACC.

4.3 Comparison Experiments

Our comparative experiments focus on comparing different models under various conditions of feature selection and noise types. For each model, we conducted five experiments, with 90% of the samples randomly selected from the full dataset for training in each experiment. The average of the metrics calculated from these experiments is used as the basis for the final analysis.

4.3.1 Parameter Analysis

We analyze the influence of the number of components on the performance of NMF algorithms.The results are shown in fig 4, Table 1 and Table 2. The full result is in appendix B, C and F.

In both the ORL and YaleB datasets, both the $L_{2,1}$ and the L_2 methods consistently outperform L_1 methods, particularly in terms of clustering accuracy and NMI. While the $L2$ method shows moderate improvement with increased components, it falls short compared to $L_{2,1}$. Moreover, the L_1 method, which emphasizes sparsity, performs poorly across both datasets, indicating that heavy sparsity may not be ideal for these tasks.

In the ORL dataset, the $L_{2,1}$ method consistently outperforms the L_2 and L_1 methods regarding clustering accuracy and NMI. This indicates that the $L_{2,1}$ norm effectively captures the data structure with high robustness. However, in the YaleB dataset, the L_2 and $L_{2,1}$ methods show very similar performance when using 50 components. Both obtain comparable clustering accuracy and NMI, but L_2 slightly outperforms $L_{2,1}$ in NMI by a small margin. Overall, $L_{2,1}$ can strike the best balance between complexity and performance. Overall, $L_{2,1}$ strikes the best balance between complexity and performance.

The overall performance of NMF models improved as the **n_components** increases, which is an expected result. The more we increase the number of components, the more information is retained.

Method	N Components	Avg Loss	Avg RRE	Avg Accuracy	Avg NMI
L_2	5	9872.22	2.90	0.0861	0.0826
L_2	20	7592.29	2.24	0.2010	0.2719
L_2	35	6618.98	1.96	0.2421	0.3266
L_2	50	**6039.80**	**1.80**	**0.2591**	**0.3444**
L_1	5	66656.67	3.59	**0.2172**	**0.4123**
L_1	20	66052.22	3.55	0.1956	0.3862
L_1	35	65242.75	3.51	0.1983	0.3906
L_1	50	**64732.06**	**3.49**	0.1944	0.3923
$L_{2,1}$	5	3200.99	2.63	0.5367	0.7429
$L_{2,1}$	20	2394.25	1.97	0.7006	0.8317
$L_{2,1}$	35	2086.78	1.72	**0.7344**	**0.8573**
$L_{2,1}$	50	**1898.24**	**1.57**	0.7272	0.8491

Table 1: ORL Dataset: Comparison of methods with selected components (5, 20, 35, 50)

Method	N Components	Avg Loss	Avg RRE	Avg Accuracy	Avg NMI
L_2	5	25974.33	2.90	0.0826	0.0803
L_2	20	15268.76	2.21	0.2053	0.2816
L_2	35	11823.40	1.95	0.2401	0.3210
L_2	50	**9813.15**	**1.78**	**0.2633**	**0.3493**
L_1	5	344479.86	3.36	**0.0919**	**0.0971**
L_1	20	300640.37	2.98	0.0850	0.0822
L_1	35	299102.69	2.96	0.0836	0.0799
L_1	50	**293695.59**	**2.91**	0.0822	0.0801
$L_{2,1}$	5	9872.22	2.90	0.0861	0.0826
$L_{2,1}$	20	7592.29	2.24	0.2010	0.2719
$L_{2,1}$	35	6618.98	1.96	0.2421	0.3266
$L_{2,1}$	50	**6039.80**	**1.80**	**0.2591**	**0.3444**

Table 2: YaleB Dataset: Comparison of methods with selected components (5, 20, 35, 50)

The variance results are shown in the boxplots in Figure 4 based on five experimental runs. This can highlight the variability and effectiveness of the L_1, L_2, and $L_{2,1}$ methods across different numbers of components.

Regarding accuracy and NMI, L_1-NMF exhibits the highest fluctuation, especially when the number of components is small. The variance is large, indicating instability, and although it decreases as the number of components increases, the overall accuracy and NMI remain low. In contrast, L_2-NMF and L_{21}-NMF show more stable performance as the number of components increases, with significantly lower variance. L_{21}-NMF, in particular, demonstrates higher accuracy and NMI with smaller fluctuations.

For loss and reconstruction error (RRE), L_1-NMF has a relatively small variance, but its RRE values remain high. On the other hand, both L_2-NMF and L_{21}-NMF have minimal variance, and their loss and RRE values decrease steadily as the number of components increases, reflecting better reconstruction ability and stability, with L_{21}-NMF performing slightly better overall.

In summary, L_{21}-NMF demonstrates the best overall stability and robustness, with consistently low variance across all metrics and superior performance. L_2-NMF also performs well, with good stability and reconstruction capabilities. L_1-NMF, however, exhibits significantly higher variance and poorer performance in terms of accuracy, RRE, and NMI, making it the least robust method of the three.

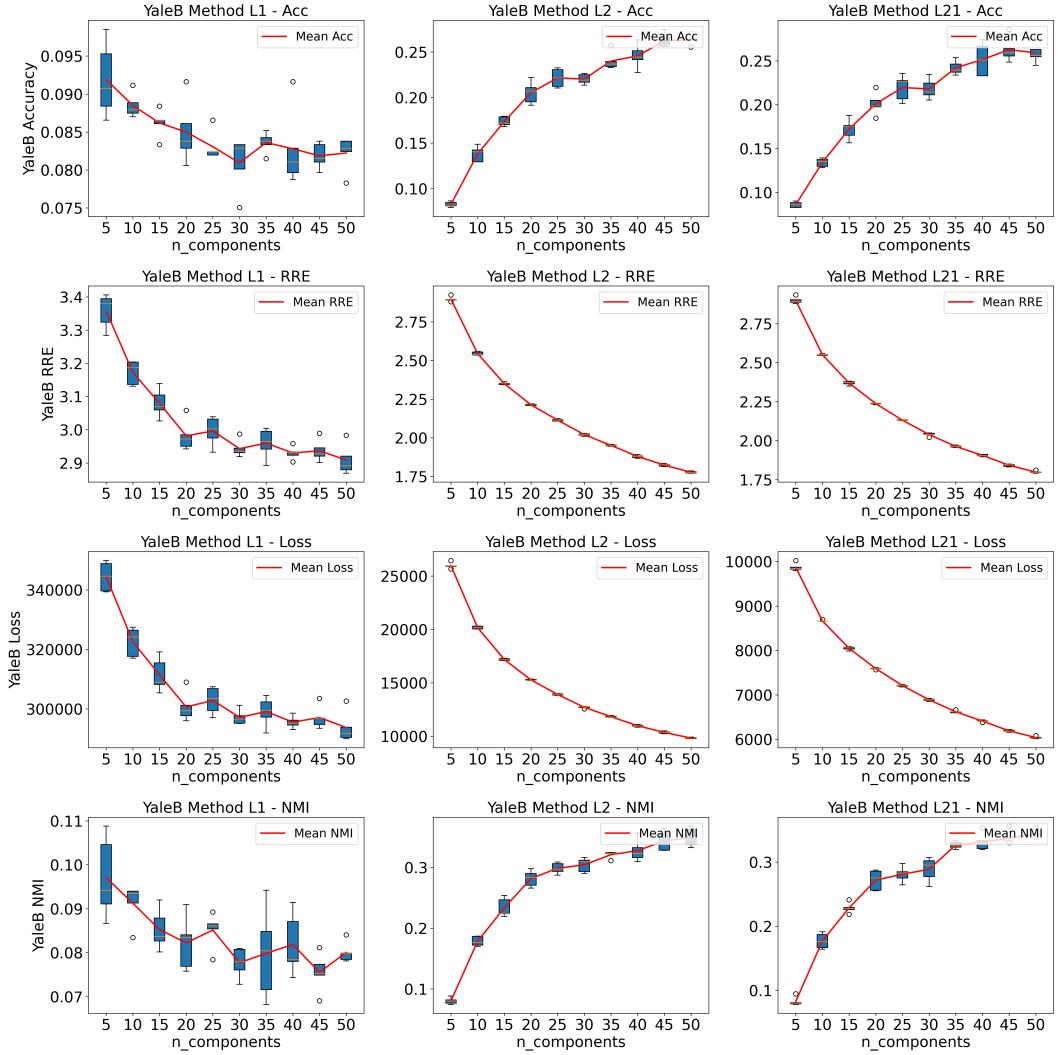


Figure 4: Result for n_components in YaleB Dataset

4.3.2 Noise Comparison

In this section, we conduct experiments on the three types of noise mentioned earlier. The parameter settings for Salt and Pepper noise are [0.01, 0.02, 0.05, 0.1, 0.15, 0.2, 0.25], for Gaussian noise are [0.01, 0.02, 0.05, 0.1, 0.15, 0.2, 0.25], and for block occlusion are [10, 12, 14] for block size and [1, 2] for the number of blocks. The experimental results for YaleB dataset are shown in Figures 5-8, and the detailed tabular results and figure result for ORL dataset can be found in Appendix D and Appendix E.

Under Block noise, L_{21}-NMF maintains high accuracy levels and NMI across all block size and number combinations, with stable RRE performance. L_2-NMF followed closely, with loss and RRE slightly increasing as noise intensified, but it still outperformed L_1-NMF. L_1-NMF again showed the worst performance, with high loss and almost no improvement in accuracy.

Under Salt & Pepper noise, L_{21}-NMF demonstrated the strongest robustness, especially at lower noise levels. As the noise proportion increased, although there was a slight decrease in accuracy and NMI, its overall performance remained superior to both L_2-NMF and L_1-NMF, highlighting its greater resilience to noise. L_2-NMF also performed steadily, with loss and RRE increasing gradually as the noise intensified. In contrast, L_1-NMF showed the weakest performance, with significantly higher loss values and almost no improvement in accuracy.

In the Gaussian noise experiments, both L_{21}-NMF and L_2-NMF managed to handle noise effectively, with loss and RRE increasing linearly as the noise level rose. They also showed strong stability in terms of accuracy and NMI. In comparison, L_1-NMF experienced a significant rise in loss, with accuracy and NMI being notably lower than the other two algorithms, making it the weakest performer.

In summary, L_{21}-NMF exhibited the best robustness across all types of noise, particularly in Salt & Pepper and Block noise environments, where its stability was especially apparent. L_2-NMF also performed well, particularly under Gaussian noise. L_1-NMF, however, underperformed across all noise types, with significantly higher loss, RRE, and lower accuracy and NMI compared to the other two algorithms.

The fig 8 illustrates the image reconstruction performance of three NMF algorithms. The L_{21}-Norm NMF consistently demonstrates superior performance across all noise types, particularly in preserving facial details and minimizing artifacts. Its reconstructions are clearer and exhibit fewer noise artifacts compared to the other methods, especially when dealing with challenging noise like Block and Salt & Pepper.

On the other hand, L_2-Norm NMF performs reasonably well but introduces more blurring and noise, particularly in Gaussian and Block noise scenarios. L_1-Norm NMF shows the weakest performance overall, struggling to effectively remove noise, resulting in significant remaining artifacts and lower reconstruction quality. This highlights L_{21}-Norm NMF as the most robust method for noisy image reconstruction among the three.

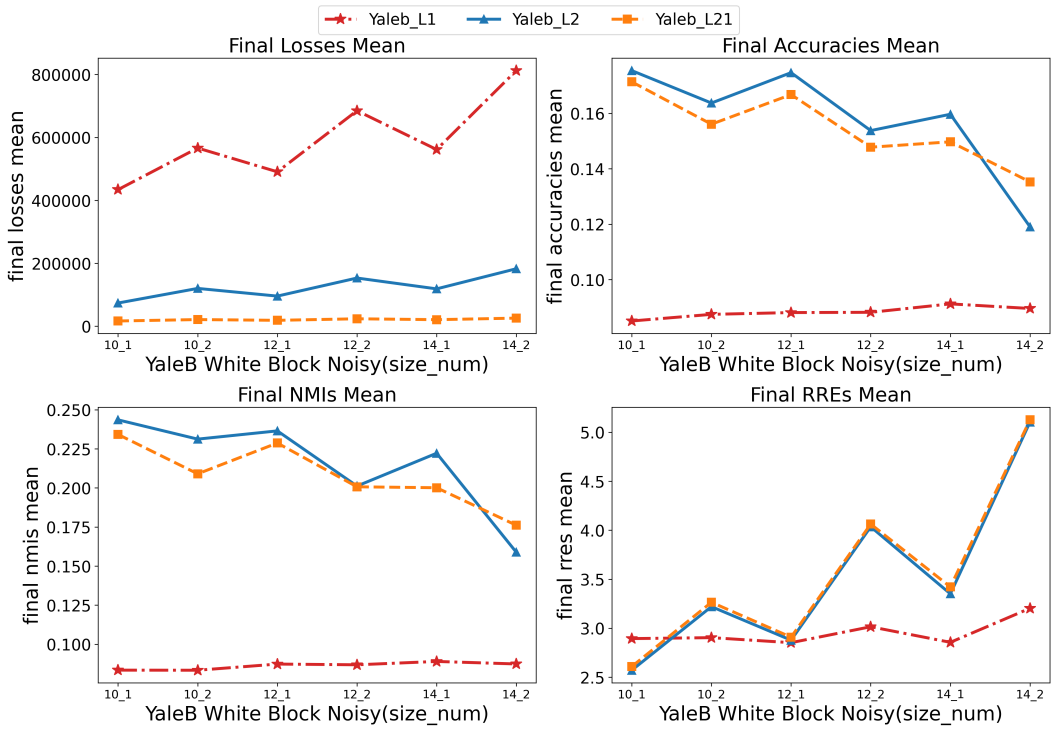


Figure 5: Result of White Block Noise.

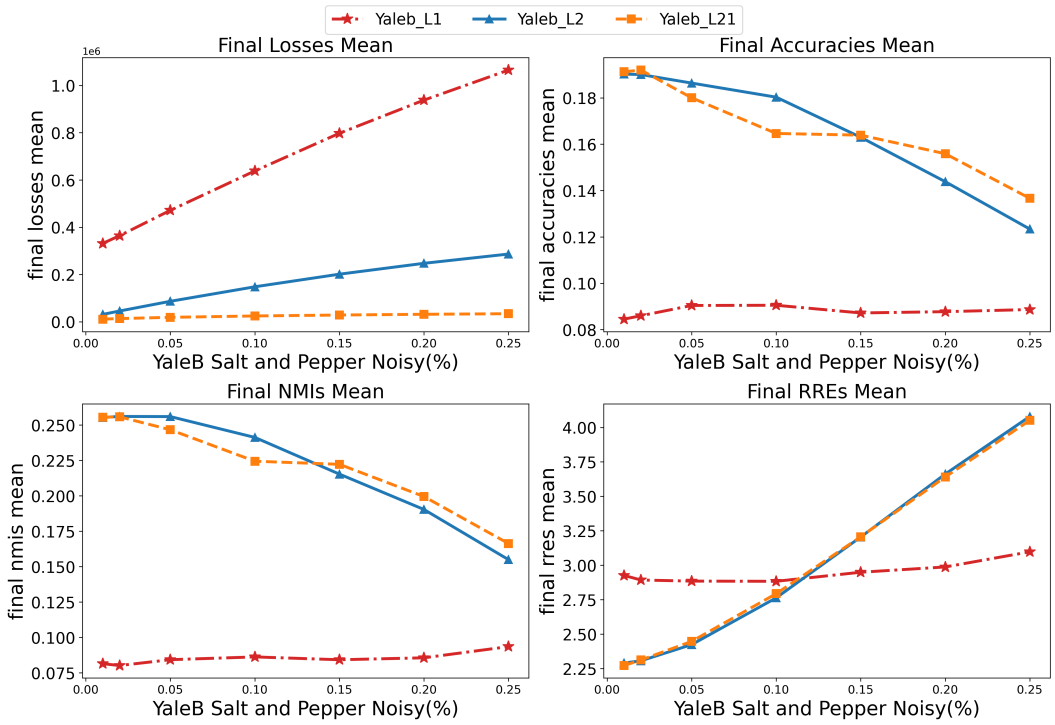


Figure 6: Result of S&P Noise.

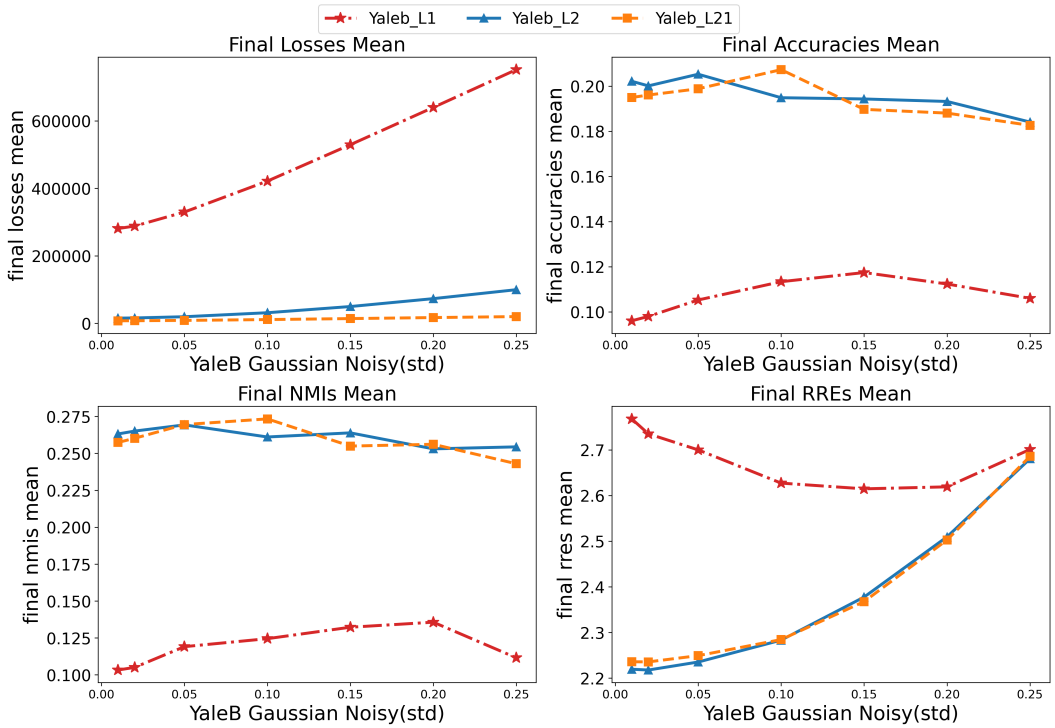


Figure 7: Result of Gaussian Noise.

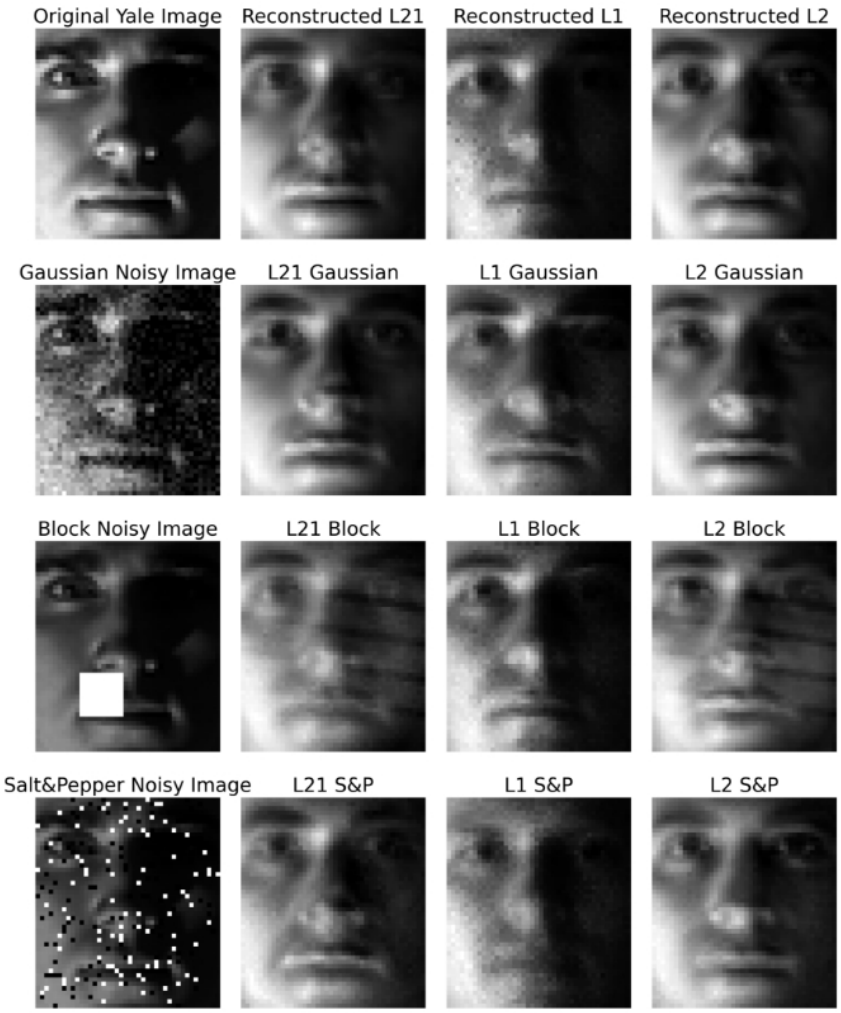


Figure 8: Reconstruction Result

5 Conclusion and Future Work

5.1 Conclusion

In this project, we introduced and implemented three distinct Non-negative Matrix Factorization (NMF) algorithms. To evaluate their performance, we introduced several types of noise into the sample images from the ORL and YaleB datasets and assessed how these algorithms handled varying noise levels.

As the number of components (n) increases, the overall performance of the NMF models improves, which is an expected outcome. At the same time, the variance of L_1-NMF is significantly higher than that of L_2-NMF and L_{21}-NMF.

On the other hand, L_{21}-NMF demonstrated the greatest robustness across all types of noise, particularly in environments with Salt & Pepper and Block noise, where its stability was most evident. L_2-NMF also performed well, especially under Gaussian noise. In contrast, L_1-NMF consistently underperformed across all noise types, exhibiting significantly higher loss, RRE, and lower accuracy and NMI compared to the other two algorithms.

In conclusion, L_{21}-NMF is the most robust model among the three when dealing with noisy data.

5.2 Future Work

In this study, although we have discussed in detail the theoretical derivation of NMF, L_1-NMF, and $L_{2,1}$-NMF, as well as their model performance on the ORL and YaleB datasets, there is still room for further exploration of the performance of NMF algorithms in the context of noise. Here we point out four possible directions for future work:

- The implementation of L_1-NMF in this research did not reach the expected level, which may be due to the use of the MUR update rules. We referred to the approaches in some articles and further attempted to improve by using SVD for parameter initialization, as well as optimizing with methods like the Nesterov method and coordinate descent.
- Attempt to conduct experiments on more datasets and improve noise levels, and discuss the robustness of NMF algorithms in more extreme scenarios.
- Incorporate more variants of NMF models into consideration and compare the robustness of different models on a broader scale.
- Discuss the robustness of different NMF algorithms under other types of noise such as Poisson noise, Brownian noise, impulse noise, etc

References

[1] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *nature*, 401(6755):788–791, 1999.

[2] G. Casalino, N. Del Buono, and C. Mencar. *Nonnegative Matrix Factorizations for Intelligent Data Analysis*, pages 49–74. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[3] Mehdi Hosseinzadeh Aghdam, Morteza Analoui, and Peyman Kabiri. Collaborative filtering using non-negative matrix factorisation. *Journal of Information Science*, 43(4):567–579, 2017.

[4] Da Kuang, Jaegul Choo, and Haesun Park. *Nonnegative Matrix Factorization for Interactive Topic Modeling and Document Clustering*, pages 215–243. Springer International Publishing, Cham, 2015.

[5] Attila Frigyesi and Mattias Höglund. Non-negative matrix factorization for the analysis of complex gene expression data: identification of clinically relevant tumor subtypes. *Cancer informatics*, 6:CIN–S606, 2008.

[6] Mikkel Nørgaard Schmidt. Single-channel source separation using non-negative matrix factorization. 2009.

[7] Yu-Xiong Wang and Yu-Jin Zhang. Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1336–1353, 2013.

[8] Naiyang Guan, Tongliang Liu, Yangmuzi Zhang, Dacheng Tao, and Larry S Davis. Truncated cauchy non-negative matrix factorization. *IEEE Transactions on pattern analysis and machine intelligence*, 41(1):246–259, 2017.

[9] D.D. Lee and H.S. Seung. Algorithms for non-negative matrix factorization. In *Proc. Int. Conf. Neural Inf. Process. Syst.*, pages 556–562, 2001.

[10] A. B. Hamza and D. J. Brady. Reconstruction of reflectance spectra using robust non-negative matrix factorization. *IEEE Trans. Signal Process.*, 54(9):3637–3642, September 2006.

[11] Deguang Kong, Chris Ding, and Heng Huang. Robust nonnegative matrix factorization using l21-norm. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 673–682. Association for Computing Machinery, 2011.

[12] E.Y. Lam. Non-negative matrix factorization for images with laplacian noise. In *Proc. IEEE Asia Pacific Conf. Circuits Syst.*, pages 798–801. IEEE, 2008.

[13] N. Guan, D. Tao, Z. Luo, and J. Shawe-Taylor. Mahnmf: Manhattan non-negative matrix factorization. *arXiv*, 2012. arXiv:1207.3438v1.

[14] L. Zhang, Z. Chen, M. Zheng, and X. He. Robust non-negative matrix factorization. *Frontiers Elect. Electron. Eng. China*, 6(2):192–200, February 2011.

[15] Liang Du, Xuan Li, and Yi-Dong Shen. Robust nonnegative matrix factorization via half-quadratic minimization. In *2012 IEEE 12th International Conference on Data Mining*, pages 201–210. IEEE, 2012.

[16] W. Liu, P.P. Pokharel, and J.C. Principe. Correntropy: Properties and applications in non-gaussian signal processing. *IEEE Trans. Signal Processing*, 55(11):5286–5298, Nov. 2007.

[17] Daniel Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13, 2000.

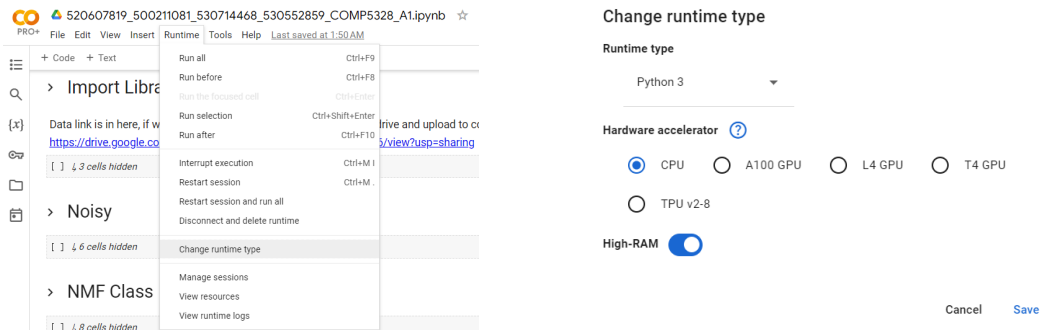
[18] Meinard Müller. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer, Cham, Switzerland, 2015.

[19] Pablo A Estévez, Michel Tesmer, Claudio A Perez, and Jacek M Zurada. Normalized mutual information feature selection. *IEEE Transactions on neural networks*, 20(2):189–201, 2009.

A Appendix A. Code and Running Instruct

The code link of this report is below: Colab Notebook Link

After opening the link, please change the runtime type to 'CPU (High RAM)'.



(a) Click Runtime type

(b) Example of White Block Noise 2.

Figure 9: Choose CPU High RAM

Then, start running the code from the first block sequentially. If the data download fails, you can manually download the data from the link provided in the notebook and upload it to the Content directory.

The dataset can be downloaded from the following link: Download Dataset

The experimental results table can be downloaded from the following link: Download Experimental Results Table

If you only want to check the experimental results rather than verifying if the experiment code runs successfully, please skip the "Experiment" Block and directly run the "Analytics and Figure" Block. Running the full experiment takes a long time, and the results have already been collected in a CSV archive, which are visualized in the "Analytics and Figure" Block.

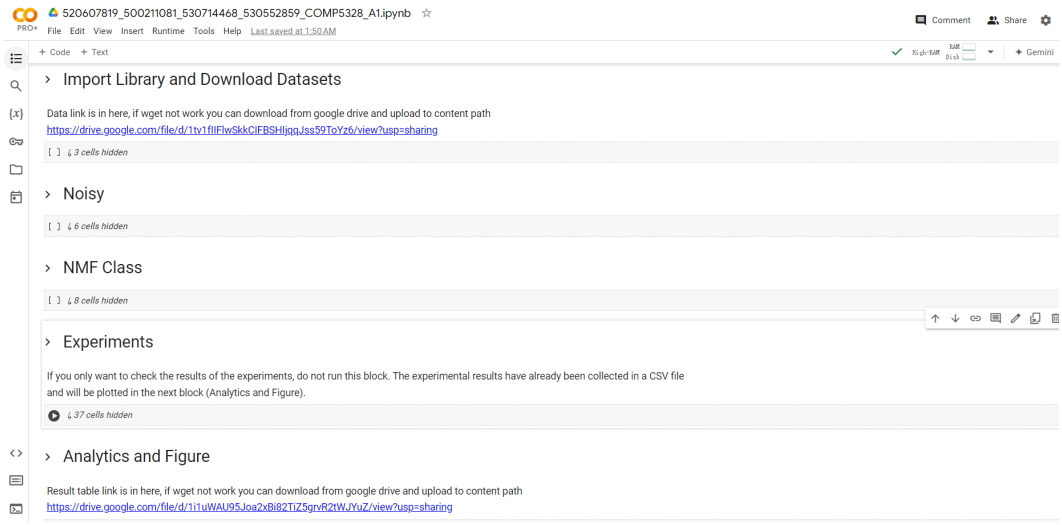


Figure 10: Example of White Block Noise.

B Appendix B. Results for Rank in ORL Dataset

Method	n_components	avg_final_loss	avg_final_rres	avg_final_accuracy	avg_final_nmi
L2	5	9872.223	2.8985	0.0861	0.0826
L2	10	8664.860	2.5505	0.1342	0.1772
L2	15	8046.502	2.3688	0.1723	0.2285
L2	20	7592.293	2.2371	0.2010	0.2719
L2	25	7208.493	2.1308	0.2195	0.2810
L2	30	6889.273	2.0390	0.2180	0.2888
L2	35	6618.981	1.9624	0.2421	0.3266
L2	40	6410.488	1.9054	0.2515	0.3313
L2	45	6190.910	1.8417	0.2627	0.3371
L2	50	6039.801	1.7980	0.2591	0.3444
L1	5	66656.675	3.5870	0.2172	0.4123
L1	10	67052.642	3.5905	0.2022	0.4034
L1	15	66508.968	3.5688	0.2056	0.4030
L1	20	66052.219	3.5466	0.1956	0.3862
L1	25	65666.677	3.5322	0.2033	0.4016
L1	30	65402.256	3.5173	0.2089	0.4043
L1	35	65242.751	3.5133	0.1983	0.3906
L1	40	64993.455	3.5011	0.1967	0.3836
L1	45	64904.941	3.4976	0.1972	0.3914
L1	50	64732.061	3.4890	0.1944	0.3923
L21	5	3201.000	2.6265	0.5367	0.7429
L21	10	2778.083	2.2693	0.6372	0.8070
L21	15	2550.354	2.0862	0.6917	0.8322
L21	20	2394.248	1.9657	0.7006	0.8317
L21	25	2266.013	1.8614	0.7300	0.8495
L21	30	2169.083	1.7840	0.7306	0.8526
L21	35	2086.777	1.7190	0.7344	0.8573
L21	40	2016.159	1.6611	0.7150	0.8482
L21	45	1950.549	1.6102	0.7267	0.8468
L21	50	1898.244	1.5669	0.7272	0.8491

Table 3: Comparison of different methods across different n_components

C Appendix C. Results for Rank in YaleB Dataset

Method	n_components	avg_final_loss	avg_final_rres	avg_final_accuracy	avg_final_nmi
L2	5	25974.33	2.8965	0.0826	0.0803
L2	10	20168.56	2.5443	0.1380	0.1784
L2	15	17176.86	2.3500	0.1737	0.2335
L2	20	15268.76	2.2133	0.2053	0.2816
L2	25	13896.95	2.1144	0.2215	0.2986
L2	30	12699.47	2.0204	0.2203	0.3042
L2	35	11823.40	1.9509	0.2401	0.3210
L2	40	10970.98	1.8791	0.2457	0.3275
L2	45	10351.54	1.8234	0.2622	0.3438
L2	50	9813.15	1.7770	0.2633	0.3493
L1	5	344479.86	3.3578	0.0919	0.0971
L1	10	322566.39	3.1728	0.0885	0.0912
L1	15	311446.37	3.0798	0.0862	0.0853
L1	20	300640.37	2.9814	0.0850	0.0822
L1	25	302844.71	2.9961	0.0831	0.0851
L1	30	297128.80	2.9429	0.0809	0.0777
L1	35	299102.69	2.9596	0.0836	0.0799
L1	40	295493.28	2.9298	0.0828	0.0819
L1	45	297107.61	2.9365	0.0819	0.0755
L1	50	293695.59	2.9086	0.0822	0.0801
L21	5	9872.22	2.8985	0.0861	0.0826
L21	10	8664.86	2.5505	0.1342	0.1772
L21	15	8046.50	2.3688	0.1723	0.2285
L21	20	7592.29	2.2371	0.2010	0.2719
L21	25	7208.49	2.1308	0.2195	0.2810
L21	30	6889.27	2.0390	0.2180	0.2888
L21	35	6618.98	1.9624	0.2421	0.3266
L21	40	6410.49	1.9054	0.2515	0.3313
L21	45	6190.91	1.8417	0.2627	0.3371
L21	50	6039.80	1.7980	0.2591	0.3444

Table 4: Comparison of different methods across different n_components

D Appendix D. Results for Different Noisy Type

Experiment	Method	ORL			YaleB		
		RRE	ACC	NMI	RRE	ACC	NMI
S&P (p=0.01)	L_1-norm	3.25	0.29	0.49	2.93	0.08	0.08
	$L_{2,1}$-norm	2.01	0.70	0.84	2.29	0.19	0.26
	L_2-norm	2.02	0.70	0.83	2.29	0.19	0.26
S&P (p=0.02)	L_1-norm	3.07	0.35	0.55	2.89	0.09	0.08
	$L_{2,1}$-norm	2.04	0.71	0.84	2.31	0.20	0.26
	L_2-norm	2.04	0.70	0.83	2.31	0.20	0.26
S&P (p=0.05)	L_1-norm	2.81	0.43	0.64	2.88	0.09	0.08
	$L_{2,1}$-norm	2.08	0.69	0.83	2.44	0.19	0.25
	L_2-norm	2.08	0.69	0.83	2.44	0.19	0.25
S&P (p=0.10)	L_1-norm	3.19	0.31	0.51	2.88	0.09	0.09
	$L_{2,1}$-norm	2.22	0.70	0.84	2.35	0.21	0.26
	L_2-norm	2.22	0.70	0.84	2.35	0.20	0.25
S&P (p=0.15)	L_1-norm	3.06	0.37	0.57	2.95	0.09	0.08
	$L_{2,1}$-norm	2.36	0.69	0.83	2.47	0.20	0.23
	L_2-norm	2.56	0.65	0.78	2.47	0.19	0.23
S&P (p=0.20)	L_1-norm	3.40	0.25	0.45	2.99	0.09	0.09
	$L_{2,1}$-norm	2.61	0.65	0.80	3.64	0.16	0.22
	L_2-norm	2.76	0.67	0.80	4.05	0.13	**0.16**
S&P (p=0.25)	L_1-norm	3.51	0.26	0.45	3.09	0.09	0.09
	$L_{2,1}$-norm	2.58	0.69	0.83	3.42	0.16	0.22
	L_2-norm	2.58	0.68	0.80	3.42	0.16	0.22

Table 5: Noise Experiments on ORL and YaleB Datasets (Salt & Pepper Noise)

Experiment	Method	ORL			YaleB		
		RRE	ACC	NMI	RRE	ACC	NMI
Gaussian (std=0.01)	L_1-norm	2.47	0.61	0.78	2.77	0.10	0.10
	$L_{2,1}$-norm	1.96	0.70	0.84	2.22	0.20	0.26
	L_2-norm	1.96	0.70	0.84	2.22	0.20	0.26
Gaussian (std=0.02)	L_1-norm	2.43	0.62	0.78	2.74	0.10	0.10
	$L_{2,1}$-norm	1.97	0.70	0.84	2.22	0.20	0.26
	L_2-norm	1.97	0.70	0.84	2.22	0.20	0.26
Gaussian (std=0.05)	L_1-norm	2.35	0.60	0.78	2.70	0.11	0.12
	$L_{2,1}$-norm	1.98	0.69	0.83	2.23	0.21	0.27
	L_2-norm	1.98	0.69	0.83	2.23	0.21	0.27
Gaussian (std=0.10)	L_1-norm	2.32	0.63	0.79	2.62	0.11	0.12
	$L_{2,1}$-norm	2.00	0.70	0.83	2.35	0.21	0.26
	L_2-norm	2.02	0.70	0.83	2.44	0.20	0.25
Gaussian (std=0.15)	L_1-norm	2.30	0.62	0.79	2.61	0.11	0.11
	$L_{2,1}$-norm	2.02	0.71	0.82	2.51	0.19	0.25
	L_2-norm	2.11	0.71	0.82	2.68	0.19	0.25
Gaussian (std=0.20)	L_1-norm	2.30	0.62	0.79	2.61	0.11	0.11
	$L_{2,1}$-norm	2.05	0.71	0.82	2.55	0.19	0.25
	L_2-norm	2.05	0.71	0.82	2.68	0.19	0.25
Gaussian (std=0.25)	L_1-norm	2.68	0.52	0.69	2.68	0.11	0.11
	$L_{2,1}$-norm	2.11	0.71	0.82	2.51	0.19	0.25
	L_2-norm	2.11	0.71	0.82	2.68	0.19	0.25

Table 6: Noise Experiments on ORL and YaleB Datasets (Gaussian Noise)

Experiment	Method	ORL			YaleB		
		RRE	ACC	NMI	RRE	ACC	NMI
Block (block_size=10, num_blocks=1)	L_1-norm	3.31	0.28	0.49	2.89	0.09	0.08
	$L_{2,1}$-norm	2.30	0.69	0.83	2.57	0.17	0.23
	L_2-norm	2.30	0.69	0.83	2.57	0.17	0.23
Block (block_size=10, num_blocks=2)	L_1-norm	3.66	0.24	0.44	2.83	0.09	0.08
	$L_{2,1}$-norm	2.83	0.68	0.83	2.83	0.17	0.23
	L_2-norm	2.83	0.68	0.83	2.83	0.17	0.23
Block (block_size=12, num_blocks=1)	L_1-norm	3.52	0.26	0.45	3.01	0.09	0.09
	$L_{2,1}$-norm	2.61	0.68	0.80	3.42	0.16	0.22
	L_2-norm	2.61	0.68	0.80	3.42	0.16	0.22
Block (block_size=12, num_blocks=2)	L_1-norm	4.35	0.20	0.38	3.01	0.09	0.09
	$L_{2,1}$-norm	3.41	0.65	0.80	4.06	0.15	0.20
	L_2-norm	3.40	0.64	0.80	4.05	0.15	0.20
Block (block_size=14, num_blocks=1)	L_1-norm	3.98	0.20	0.38	3.01	0.09	0.09
	$L_{2,1}$-norm	3.98	0.70	0.80	4.06	0.20	0.23
	L_2-norm	3.97	0.70	0.80	4.06	0.19	0.22
Block (block_size=14, num_blocks=2)	L_1-norm	4.34	0.20	0.38	3.01	0.09	0.09
	$L_{2,1}$-norm	3.41	0.65	0.80	4.06	0.15	0.20
	L_2-norm	3.40	0.64	0.80	4.05	0.15	0.20

Table 7: Noise Experiments on ORL and YaleB Datasets (Block Occlusion)

E Appendix E. Figure Results for ORL dataset

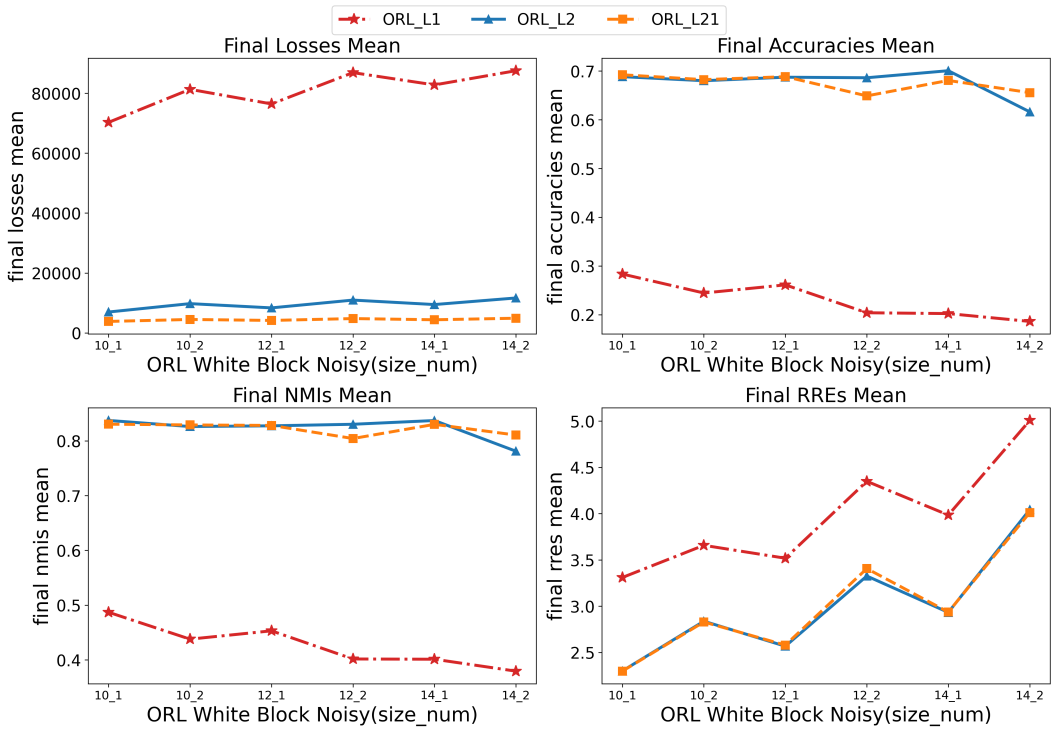


Figure 11: Example of White Block Noise.

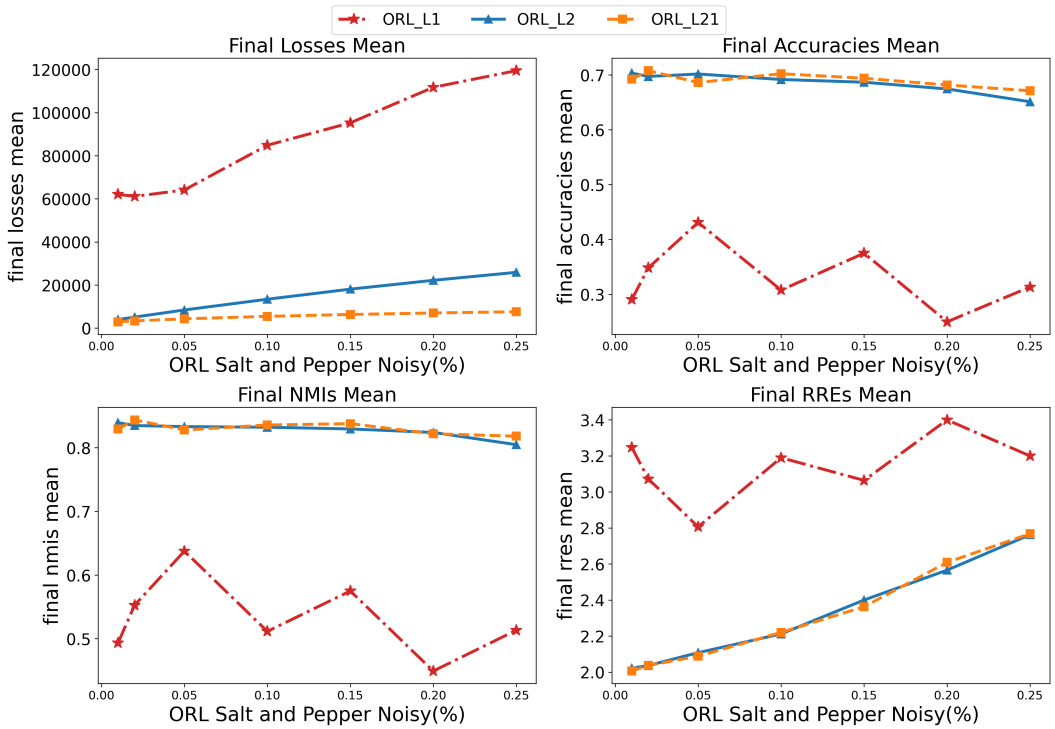


Figure 12: Example of S&P Noise.

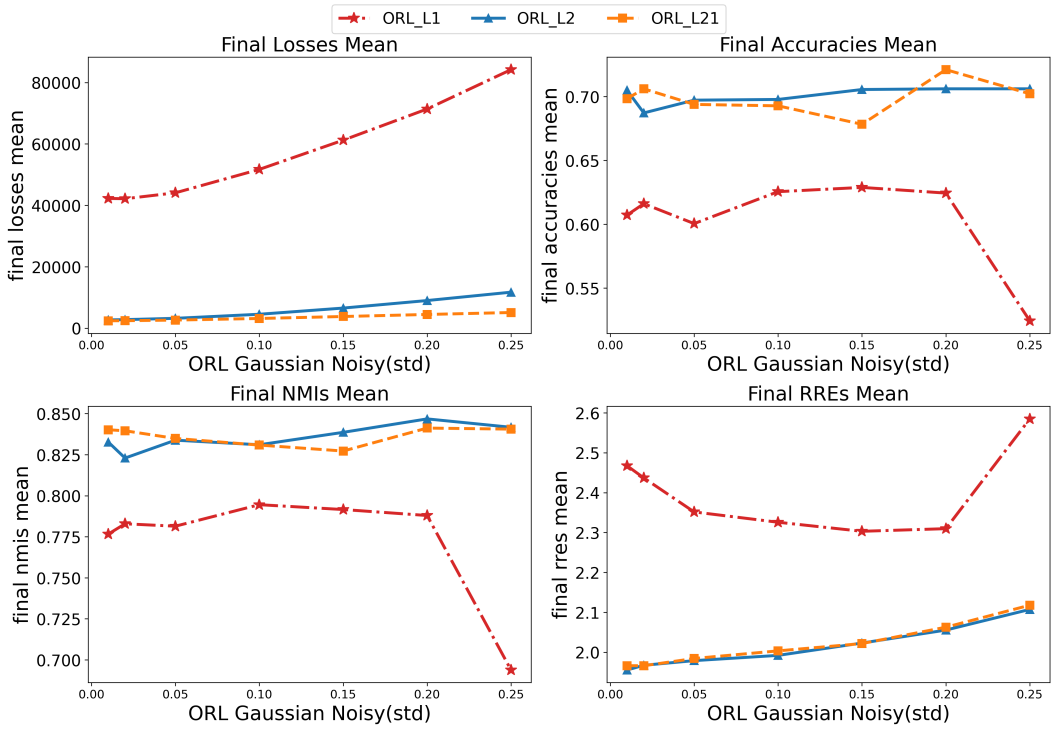


Figure 13: Example of Gaussian Noise.

F Appendix F. Figure Result for rank in ORL dataset

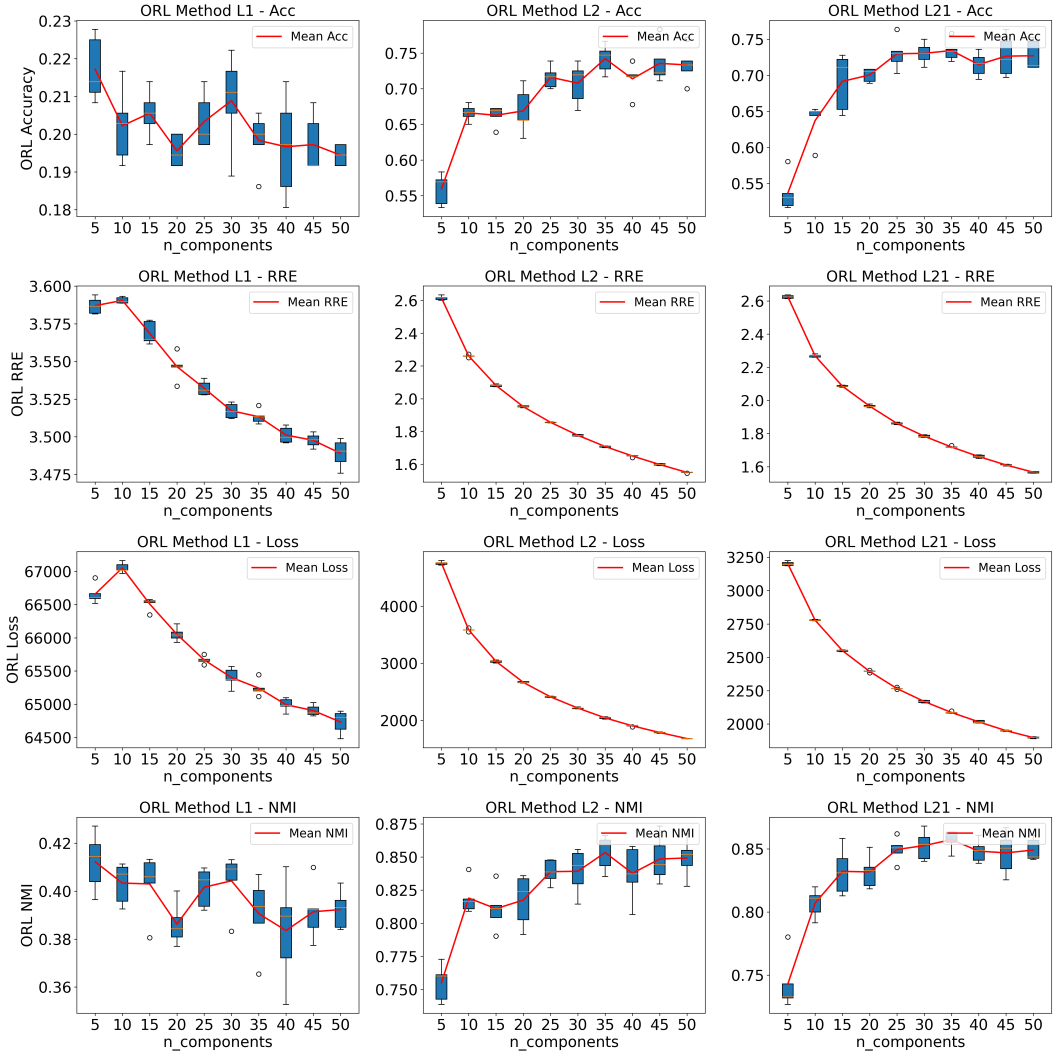


Figure 14: Result for n_components in ORL dataset