# Semidefinite Programming

## Applications in approximating NP-Complete problems & Matrix Completetion

Dimitri Lopez  Xinshi Wang  Jenny Gao

Rensselaer Polytechnic Institute

April 18, 2023

# Presentation Overview

# Motivation for Semidefinite Programming

- Linear Programming is a common constrained optimization technique with uses in:
  - Math
  - Computer Science
  - Economics
  - Business
- Wide applicability combined with fast runtimes makes linear programming quite popular

# Motivation for Semidefinite Programming

- Linear Programming is a common constrained optimization technique with uses in:
  - Math
  - Computer Science
  - Economics
  - Business
- Wide applicability combined with fast runtimes makes linear programming quite popular

---

- Semidefinite programming (SDP) expands upon the ideas of linear programming
  - SDP can solve everything that linear programming can and more
  - Widely used in combinatorial optimzation problems
  - Many NP-Hard problems can be approximated well with SDP
- Both linear programming and semidefinite programming are convex problems

# Formalization of Linear Programming

- Suppose that we have control over a set of variables: $\vec{x} = [x_1, x_2, \ldots, x_n]^\top$

# Formalization of Linear Programming

- Suppose that we have control over a set of variables: $\vec{x} = [x_1, x_2, \ldots, x_n]^\top$
- Furthermore we often constraint $\vec{x}$ to be non-negative: $\vec{x} \geq 0$.

# Formalization of Linear Programming

- Suppose that we have control over a set of variables: $\vec{x} = [x_1, x_2, \ldots, x_n]^\top$
- Furthermore we often constraint $\vec{x}$ to be non-negative: $\vec{x} \geq 0$.
- Where for each of these $n$ variables we have an associated coefficient $\vec{c} = [c_1, c_2, \ldots, c_n]^\top$

# Formalization of Linear Programming

- Suppose that we have control over a set of variables: $\vec{x} = [x_1, x_2, \ldots, x_n]^\top$
- Furthermore we often constraint $\vec{x}$ to be non-negative: $\vec{x} \geq 0$.
- Where for each of these $n$ variables we have an associated coefficient $\vec{c} = [c_1, c_2, \ldots, c_n]^\top$
- In the end we want to find the optimal value for the following equation:

$$\min \vec{c} \cdot \vec{x}$$

# Formalization of Linear Programming

$$\min \vec{c} \cdot \vec{x}$$

- The problem is quite easy as is. What if each of these $n$ variables corresponds to how much of a given product that we want to buy?
  - This would add additional constraints to what values $\vec{x}$ could take on.

# Formalization of Linear Programming

$$\min \vec{c} \cdot \vec{x}$$

- The problem is quite easy as is. What if each of these $n$ variables corresponds to how much of a given product that we want to buy?
  - This would add additional constraints to what values $\vec{x}$ could take on.
- Each variable must satisfy some constraint, we can express this as:

$$\vec{a}_1 \cdot \vec{x} \geq b_1$$
$$\vec{a}_2 \cdot \vec{x} \geq b_2$$
$$\vdots$$
$$\vec{a}_n \cdot \vec{x} \geq b_n$$

# Formalization of Linear Programming

$$\min \vec{c} \cdot \vec{x}$$

- The problem is quite easy as is. What if each of these $n$ variables corresponds to how much of a given product that we want to buy?
  - This would add additional constraints to what values $\vec{x}$ could take on.
- Each variable must satisfy some constraint, we can express this as:

$$\vec{a}_1 \cdot \vec{x} \geq b_1$$
$$\vec{a}_2 \cdot \vec{x} \geq b_2$$
$$\vdots$$
$$\vec{a}_n \cdot \vec{x} \geq b_n$$

- We can gather this into a single expression using a matrix

$$\mathbf{A}\vec{x} \geq \vec{b}$$

# Formalization of Linear Programming

$$\min \vec{c} \cdot \vec{x}$$

- The problem is quite easy as is. What if each of these $n$ variables corresponds to how much of a given product that we want to buy?
  - This would add additional constraints to what values $\vec{x}$ could take on.
- Each variable must satisfy some constraint, we can express this as:

$$\vec{a}_1 \cdot \vec{x} \geq b_1$$
$$\vec{a}_2 \cdot \vec{x} \geq b_2$$
$$\vdots$$
$$\vec{a}_n \cdot \vec{x} \geq b_n$$

- We can gather this into a single expression using a matrix

$$\mathbf{A}\vec{x} \geq \vec{b}$$

# Standard Form for a Linear Program

$$\begin{aligned} \text{minimize} \quad & \vec{c} \cdot \vec{x} \\ \text{subject to} \quad & \mathbf{A}\vec{x} \geq \vec{b} \\ & \vec{x} \geq 0 \end{aligned}$$

# Standard Form for a Linear Program

$$\begin{aligned} \text{minimize} \quad & \vec{c} \cdot \vec{x} \\ \text{subject to} \quad & \mathbf{A}\vec{x} \geq \vec{b} \\ & \vec{x} \geq 0 \end{aligned}$$

- Where $\vec{x}$ are the variables that we have control over
  - Example: Each variable of $\vec{x}$ represents how much of a certain product to purchase

# Standard Form for a Linear Program

$$\begin{aligned} \text{minimize} \quad & \vec{c} \cdot \vec{x} \\ \text{subject to} \quad & \mathbf{A}\vec{x} \geq \vec{b} \\ & \vec{x} \geq 0 \end{aligned}$$

- Where $\vec{x}$ are the variables that we have control over
  - Example: Each variable of $\vec{x}$ represents how much of a certain product to purchase
- $\vec{c}$ is a set of corresponding coefficients for the variables of $\vec{x}$
  - Example: Each variable of $\vec{c}$ corresponds to how much each product costs

# Standard Form for a Linear Program

$$\begin{aligned} \text{minimize} \quad & \vec{c} \cdot \vec{x} \\ \text{subject to} \quad & \mathbf{A}\vec{x} \geq \vec{b} \\ & \vec{x} \geq 0 \end{aligned}$$

- Where $\vec{x}$ are the variables that we have control over
  - Example: Each variable of $\vec{x}$ represents how much of a certain product to purchase
- $\vec{c}$ is a set of corresponding coefficients for the variables of $\vec{x}$
  - Example: Each variable of $\vec{c}$ corresponds to how much each product costs
- $\min \vec{c} \cdot \vec{x}$ is the function we are trying to optimize
  - Example: We want to minimize the cost of buying our products

# Standard Form for a Linear Program

$$\begin{aligned} \text{minimize} \quad & \vec{c} \cdot \vec{x} \\ \text{subject to} \quad & \mathbf{A}\vec{x} \geq \vec{b} \\ & \vec{x} \geq 0 \end{aligned}$$

- Where $\vec{x}$ are the variables that we have control over
  - Example: Each variable of $\vec{x}$ represents how much of a certain product to purchase
- $\vec{c}$ is a set of corresponding coefficients for the variables of $\vec{x}$
  - Example: Each variable of $\vec{c}$ corresponds to how much each product costs
- $\min \vec{c} \cdot \vec{x}$ is the function we are trying to optimize
  - Example: We want to minimize the cost of buying our products
- $\mathbf{A}\vec{x} \geq \vec{b}$ constraints on our solutions. Defines the feasible region.
  - Example: This could represent the minimum amount of products that we must buy

# Standard Form for a Linear Program

$$\begin{aligned} \text{minimize} \quad & \vec{c} \cdot \vec{x} \\ \text{subject to} \quad & \mathbf{A}\vec{x} \geq \vec{b} \\ & \vec{x} \geq 0 \end{aligned}$$

- Where $\vec{x}$ are the variables that we have control over
  - Example: Each variable of $\vec{x}$ represents how much of a certain product to purchase
- $\vec{c}$ is a set of corresponding coefficients for the variables of $\vec{x}$
  - Example: Each variable of $\vec{c}$ corresponds to how much each product costs
- $\min \vec{c} \cdot \vec{x}$ is the function we are trying to optimize
  - Example: We want to minimize the cost of buying our products
- $\mathbf{A}\vec{x} \geq \vec{b}$ constraints on our solutions. Defines the feasible region.
  - Example: This could represent the minimum amount of products that we must buy

# Linear Programming to Semidefinite Programming

- Semidefinite programming (SDP) takes the concept that linear programming has with vectors and generalizes it to matrices.
- The $\langle\rangle_{\mathrm{F}}$ operator is the Frobenius inner product which is the sum of element wise multiplication on vectors: $\langle \mathbf{C}, \mathbf{X} \rangle_{\mathrm{F}} = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j} x_{i,j} = trace(\mathbf{C}^{\top} \mathbf{X})$
- $\mathbf{X} \succeq 0$ means that $\mathbf{X}$ is positive semi-definite (PSD)

---

| Linear Programming | Semidefinite Programming |
|---|---|
| $\vec{x} \in \mathbb{R}^n$ | $\mathbf{X} \in \mathbb{R}^{n \times n}$ |
| $\vec{x} \geq 0$ | $\mathbf{X} \succeq 0$ |
| $\vec{c} \in \mathbb{R}^n$ | $\mathbf{C} \in \mathbb{R}^{n \times n}$ |
| $\min \vec{c} \cdot \vec{x}$ | $\min \langle \mathbf{C}, \mathbf{X} \rangle_{\mathrm{F}}$ |
| $\mathbf{A} \in \mathbb{R}^{n \times n}, \vec{b} \in \mathbb{R}^n$ | $\mathbf{A}_i \in \mathbb{R}^{n \times n}, \vec{b} \in \mathbb{R}^m$ |
| $\mathbf{A}\vec{x} \geq \vec{b}$ | $\langle \mathbf{A}_i, \mathbf{X} \rangle_{\mathrm{F}} = b_i : i = 1, \ldots, m$ |

# Linear Programming to Semidefinite Programming

- A linear program is defined as:

$$\begin{array}{lll} \text{minimize} & \vec{c} \cdot \vec{x} & \\ \text{subject to} & \mathbf{A}\vec{x} & \geq \vec{b} \\ & \vec{x} & \geq 0 \end{array}$$

- A semidefinite program is defined as:

$$\begin{array}{lll} \min & \langle \mathbf{C}, \mathbf{X} \rangle_{\mathrm{F}} & \\ \text{subject to} & \langle \mathbf{A}_i, \mathbf{X} \rangle_{\mathrm{F}} \geq b_i & i = 1, \ldots, m \\ & \mathbf{X} \succeq 0 & \end{array}$$

# Semidefinite Programming Duality

- It is important to note the dual of an SDP problem which is:

$$\begin{array}{ll} \max & \sum_{i=1}^{m} z_i b_i \\ \text{such that} & \sum_{i=1}^{m} z_i \mathbf{A}_i + \mathbf{S} = \mathbf{C} \\ & \mathbf{S} \succeq 0 \end{array}$$

# Semidefinite Programming Duality

- It is important to note the dual of an SDP problem which is:

$$\begin{array}{ll} \max & \sum_{i=1}^{m} z_i b_i \\ \text{such that} & \sum_{i=1}^{m} z_i \mathbf{A}_i + \mathbf{S} = \mathbf{C} \\ & \mathbf{S} \succeq 0 \end{array}$$

- We are trying to find a set of scalars $z_1, z_2, \ldots, z_m$
- Where our objective function is $\vec{z} \cdot \vec{b}$
- We also satisfy the constraint $\sum_{i=1}^{m} z_i \mathbf{A}_i + \mathbf{S} = \mathbf{C}$ where $\mathbf{A}_i$ and $\mathbf{C}$ are from before.

# Semidefinite Programming Duality

- It is important to note the dual of an SDP problem which is:

$$\begin{array}{ll} \max & \sum_{i=1}^{m} z_i b_i \\ \text{such that} & \sum_{i=1}^{m} z_i \mathbf{A}_i + \mathbf{S} = \mathbf{C} \\ & \mathbf{S} \succeq 0 \end{array}$$

- We are trying to find a set of scalars $z_1, z_2, \ldots, z_m$
- Where our objective function is $\vec{z} \cdot \vec{b}$
- We also satisfy the constraint $\sum_{i=1}^{m} z_i \mathbf{A}_i + \mathbf{S} = \mathbf{C}$ where $\mathbf{A}_i$ and $\mathbf{C}$ are from before.
- We know that $\mathbf{S} \succeq 0$ which allows us to get the more intuitive:

$$\mathbf{C} - \sum_{i=1}^{m} z_i \mathbf{A}_i \succeq 0$$

- Pulling it all together:

$$\begin{array}{ll} \max & \sum_{i=1}^{m} z_i b_i \\ \text{such that} & \mathbf{C} - \sum_{i=1}^{m} z_i \mathbf{A}_i \succeq 0 \end{array}$$

# Semidefinite Programming Runtime

- SDPs can be solved in polynomial time which makes them qutie useful.
- One algorithm to solve them is Alizadeh's interior point method which runs in:

$$\tilde{O}(n^{3.5})$$

# Reviewing TSP

The Traveling Salesman Problem (TSP) is an optimization problem in which the objective is to find the shortest possible route for a salesman to visit a given set of cities, passing through each city exactly once, and returning to the starting city. It is a well-known NP-hard problem.

# Semidefinite Programming Methods for the Symmetric Traveling Salesman Problem , 1999

Let $C \in \mathbb{R}^{n \times n}$ denote the matrix of edge costs. Let $J$ denote the all-ones matrix, and $e$ denote the all-ones vector.

$$
\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\text{trace}(CX) \\
\text{subject to} \quad & Xe = 2e \\
& X_{ii} = 0, \quad i = 1, \ldots, n \\
& 0 \leq X_{ij} \leq 1, \quad i, j = 1, \ldots, n \\
& 2I - X + (2 - 2\cos\left(\frac{2\pi}{n}\right))(J - I) \succeq 0 \\
& X \text{ is a real, symmetric } n \times n \text{ matrix.}
\end{aligned}
$$

X is a fractional adjacency matrix, meaning for $e = \{i, j\}$, $x_{ij} = x_{ji}$ is the proportion of edge $e$ used.

# Integrality Gap And Running Time

| # Of Nodes | SDP Time | BF Time | SDP Objective Value | BF Objective Value | Integrity Gap | Time Ratio |
|---|---|---|---|---|---|---|
| 10 | 0.7101 | 0.0156 | 53224.4854 | 53228.3976 | 0.9999 | 45.519 |
| 15 | 0.6776 | 0.8224 | 65753.5934 | 67299.5625 | 0.9770 | 0.8239 |
| 20 | 1.2271 | 97.2059 | 69558.9865 | 76199.4928 | 0.9129 | 0.0126 |
| 21 | 1.3689 | 266.7778 | 73969.6527 | 77373.6362 | 0.9560 | 0.0051 |
| 22 | 5.4774 | 657.7847 | 66459.7265 | 68245.9576 | 0.9738 | 0.0083 |

# Visualization

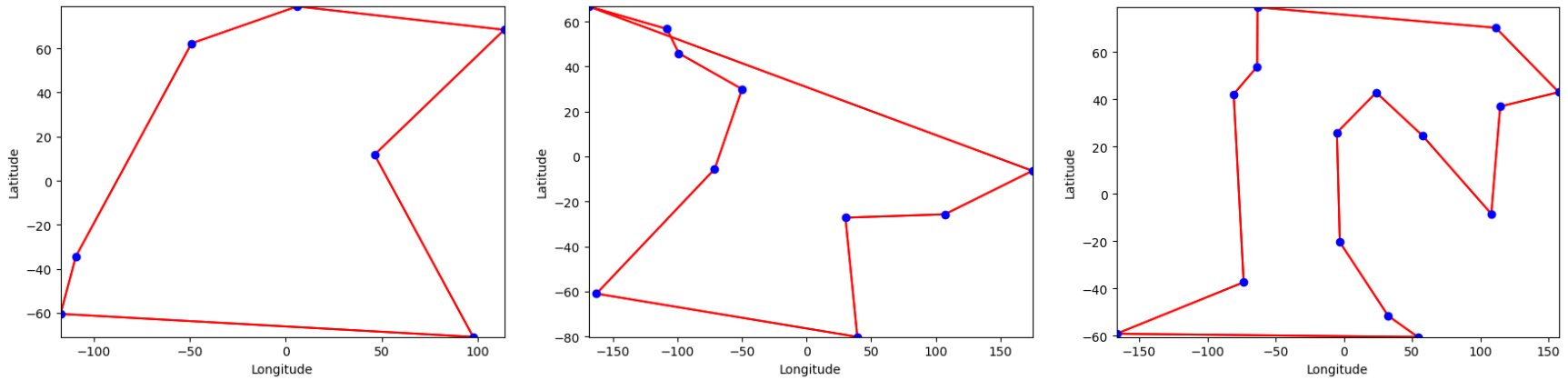



Figure: reasonable solution

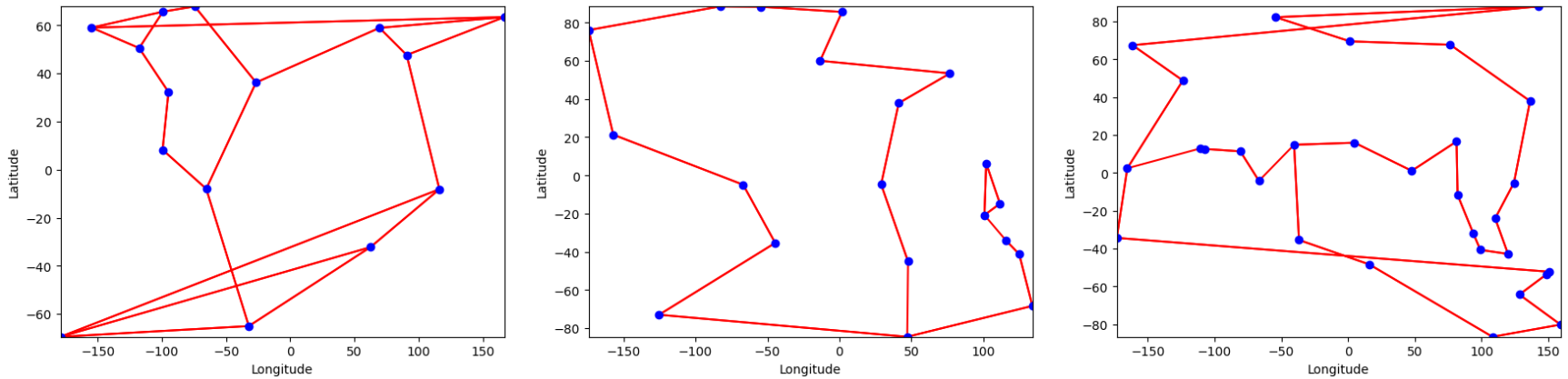



Figure: unreasonable solution

# Low rank matrices

Given an incomplete matrix, can we recover the missing values?

| | | | | |
|---|---|---|---|---|
| | | -1 | | |
| | | | 1 | |
| 1 | 1 | -1 | 1 | -1 |
| 1 | | | | -1 |
| | | -1 | | |

| | | | | |
|---|---|---|---|---|
| 1 | 1 | -1 | 1 | -1 |
| 1 | 1 | -1 | 1 | -1 |
| 1 | 1 | -1 | 1 | -1 |
| 1 | 1 | -1 | 1 | -1 |
| 1 | 1 | -1 | 1 | -1 |

# Low rank matrices

**Yes!**

Given:

- The matrix is low rank*
- We have enough sample data

Note: This does not apply to *all* low-rank matrices. But most.

# Low rank matrices

**Yes!**

Given:

- The matrix is low rank*
- We have enough sample data

Note: This does not apply to *all* low-rank matrices. But most.

Why low-rank matrices?

# Why is this useful

1. **Netflix** has an incomplete set of user preferences based off their past watch history. Can they use this information to recommend new movies?
2. **Recommendation Engine:** The netflix problem can be extended to general recommendation engines where a vendor knows some of the user preferences.
3. **Images:** We will give an example of recovering a corrupted image using matrix completetion

# Relaxing Matrix Completetion to SDP

Suppose we have a low rank matrix $\mathbf{M}$. We have a set of location $\Omega$ describing our sampling. That is, if $(i, j) \in \Omega$, we observe entry $M_{ij}$. Given $\mathbf{M}$ is low rank, it seems resonable that we would like to solve the following optimization problem

# Relaxing Matrix Completetion to SDP

Suppose we have a low rank matrix $\mathbf{M}$. We have a set of location $\Omega$ describing our sampling. That is, if $(i, j) \in \Omega$, we observe entry $M_{ij}$. Given $\mathbf{M}$ is low rank, it seems resonable that we would like to solve the following optimization problem

$$\begin{aligned} \text{minimize} \quad & \text{rank}(\mathbf{X}) \\ \text{subject to} \quad & X_{ij} = M_{ij} \quad (i, j) \in \Omega \\ & \mathbf{X} \in \mathbb{R}^{n \times n} \end{aligned}$$

# Relaxing Matrix Completetion to SDP

Suppose we have a low rank matrix $\mathbf{M}$. We have a set of location $\Omega$ describing our sampling. That is, if $(i, j) \in \Omega$, we observe entry $M_{ij}$. Given $\mathbf{M}$ is low rank, it seems resonable that we would like to solve the following optimization problem

$$\begin{aligned} \text{minimize} \quad & \text{rank}(\mathbf{X}) \\ \text{subject to} \quad & X_{ij} = M_{ij} \quad (i, j) \in \Omega \\ & \mathbf{X} \in \mathbb{R}^{n \times n} \end{aligned}$$

**But...**

Rank is not a convex. This turns out to be an NP-Hard Problem.

# Introduce the nuclear norm

## Nuclear Norm

The nuclear norm is a close approximation of the rank.

The nuclear norm of a matrix $\mathbf{X}$ is defined as the sum of the eigenvalues.

$$\|\mathbf{X}\|_* = \sum_{k=1}^{n} \sigma_k \mathbf{X}$$

# Introduce the nuclear norm

## Nuclear Norm

The nuclear norm is a close approximation of the rank.

The nuclear norm of a matrix $\mathbf{X}$ is defined as the sum of the eigenvalues.

$$\|\mathbf{X}\|_* = \sum_{k=1}^{n} \sigma_k \mathbf{X}$$

For a symmetric positive semi-definite (SPSD) matricies, the nuclear norm is equal to the trace.

# A better relaxation

What if our matrix is not SPSD

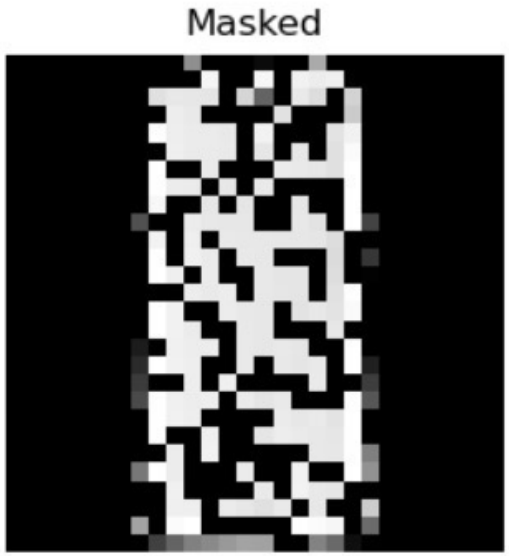
- We introduce two matricies $\mathbf{W}_1$ and $\mathbf{W}_2$

# A better relaxation

$$
\begin{aligned}
\text{minimize} \quad & \operatorname{trace}(\mathbf{W}_1) + \operatorname{trace}(\mathbf{W}_2) \\
\text{subject to} \quad & X_{ij} = M_{ij} \quad (i,j) \in \Omega \\
& \begin{bmatrix} \mathbf{W}_1 & \mathbf{X} \\ \mathbf{X}^\top & \mathbf{W}_2 \end{bmatrix} \succeq 0
\end{aligned}
$$

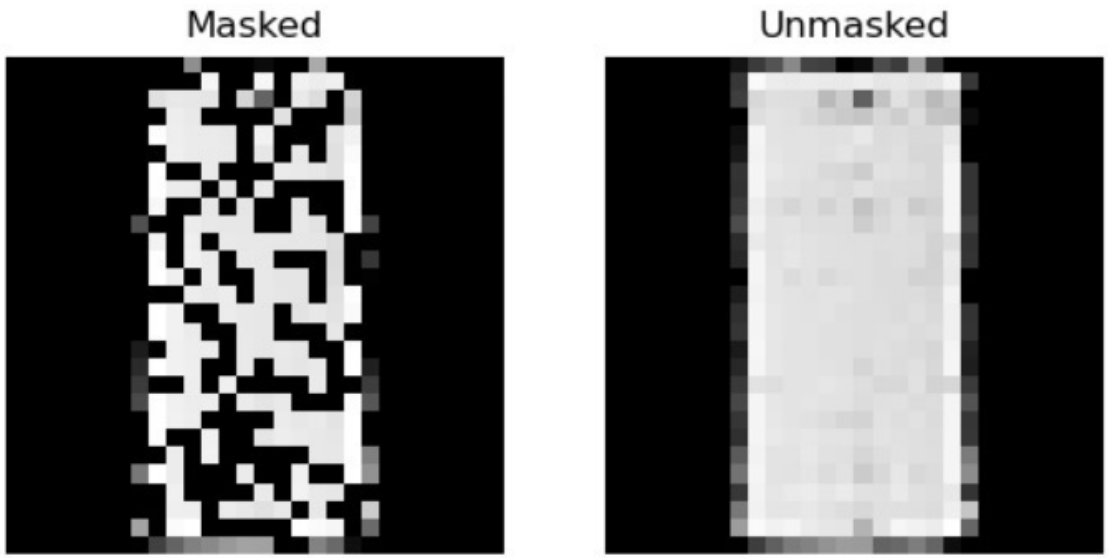
# Fashion-MNIST

55% of data



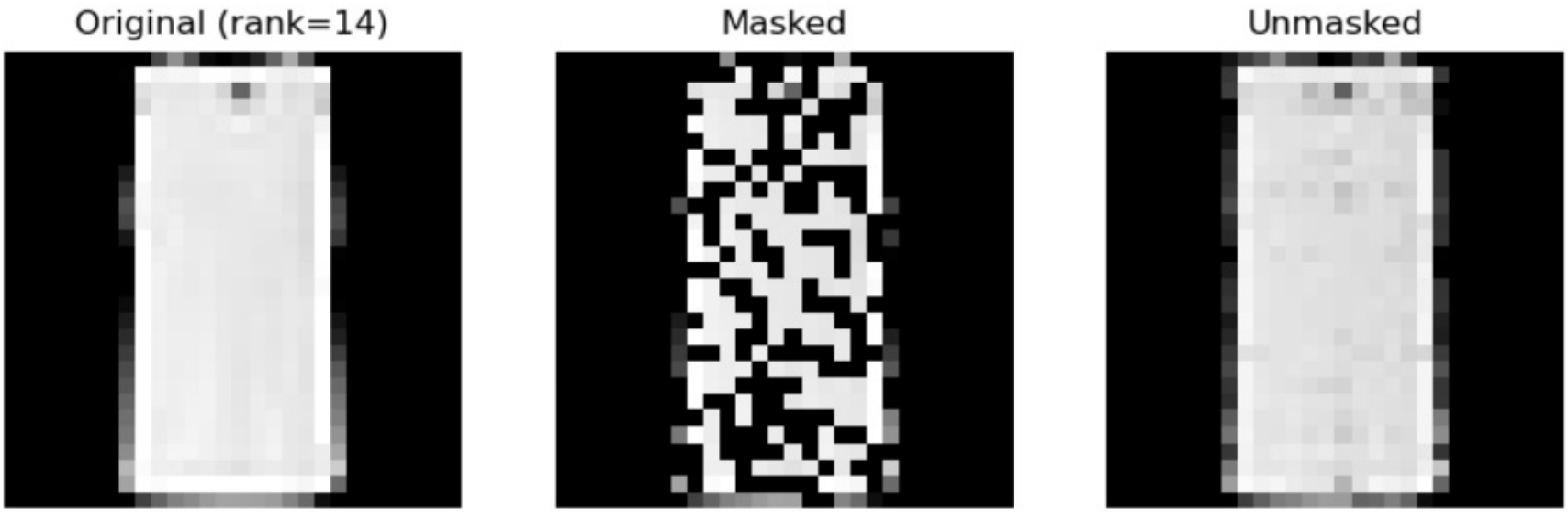
# Fashion-MNIST

55% of data



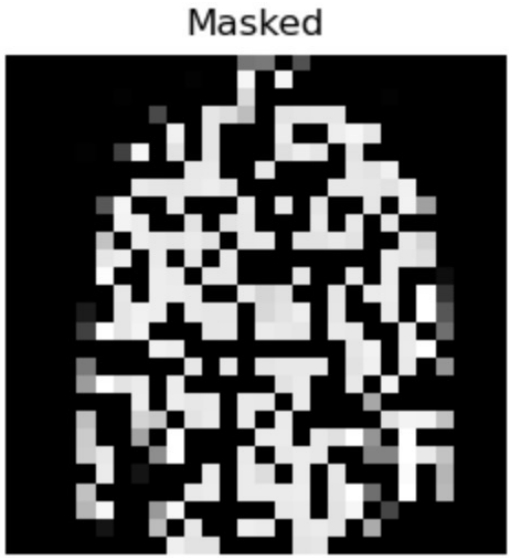
# Fashion-MNIST

55% of data



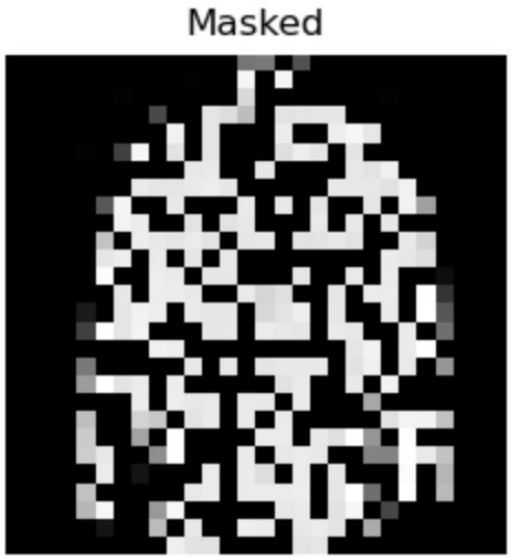
# Fashion-MNIST

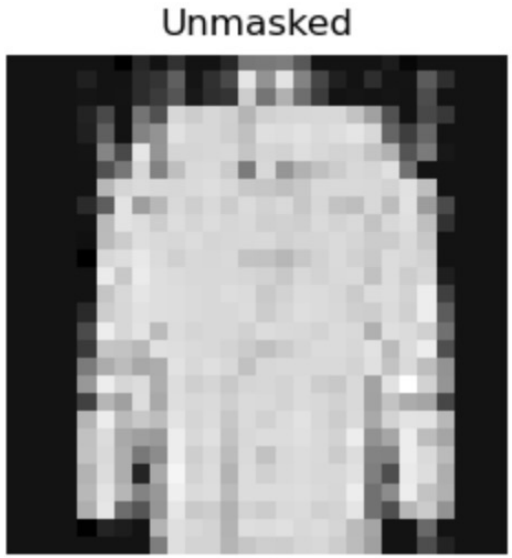
50% of data



# Fashion-MNIST

50% of data



# Fashion-MNIST

50% of data

# Citing References

An example of the `\cite` command to cite within the presentation:

This statement requires citation [Smith, 2022, Kennedy, 2023].

# References



John Smith (2022)
Publication title
*Journal Name* 12(3), 45 – 678.


Annabelle Kennedy (2023)
Publication title
*Journal Name* 12(3), 45 – 678.

# Acknowledgements

**Smith Lab**

- Alice Smith
- Devon Brown

**Cook Lab**

- Margaret
- Jennifer
- Yuan

**Funding**

- British Royal Navy
- Norwegian Government