On Classification of Noiseless and Noisy Pattern Orientations

SIMC Endeavor Team 04
Hans Gabriel D. De Vera
Benjamin L. Jacob
Davis Nicholo A. Magpantay

Republic of the Philippines
Department of Science and Technology
Philippine Science High School – Main Campus
Agham Rd., Diliman, Quezon City, Philippines

Presentation Outline

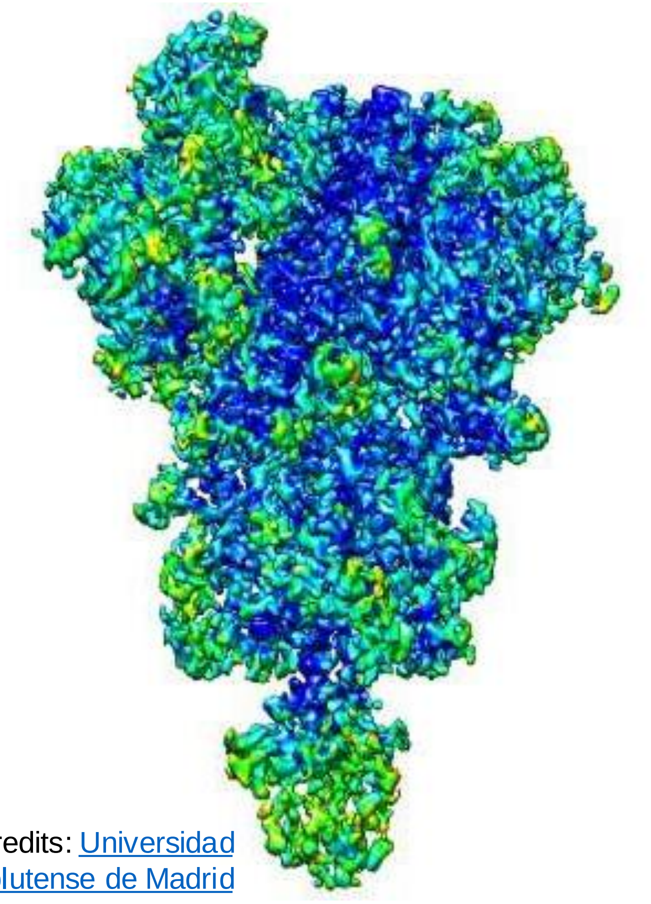
- Preamble / Motivation
- Noiseless Data (Tasks 1-3)
- Noisy Data (Task 4)
- Likelihood Models (Task 5)
- Noisy Sparse Data (Task 6)
- The Master Image (Task 7)

Part 0: Preamble

Motivation

Understanding involves seeing. This is especially true for complex biological systems, such as proteins.

Techniques such as single particle imaging (SPI) can achieve this by computationally reconstructing structures from several measurements.



Credits: Universidad Complutense de Madrid

A Toy Model

SPI involves reconstructing 3D structures from 2D images in varying orientations.

Here we explore the simpler case of reconstructing 2D images from differently oriented 2D patterns.

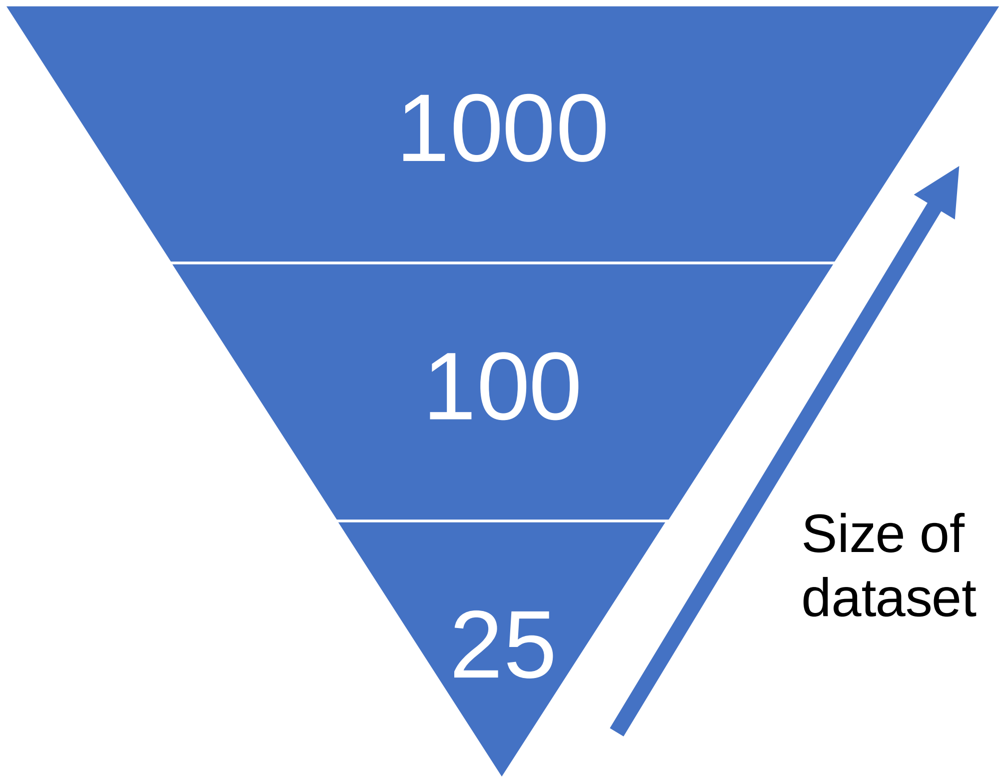
Part 1:
On Noiseless Data

Tasks 1-3: Noiseless Images

- In an idealized world, the image data we collect would be **noiseless**.
- This makes it **easy to classify in what orientation** each measurement is (i.e., measurements of the same orientation would have the same pixel values)

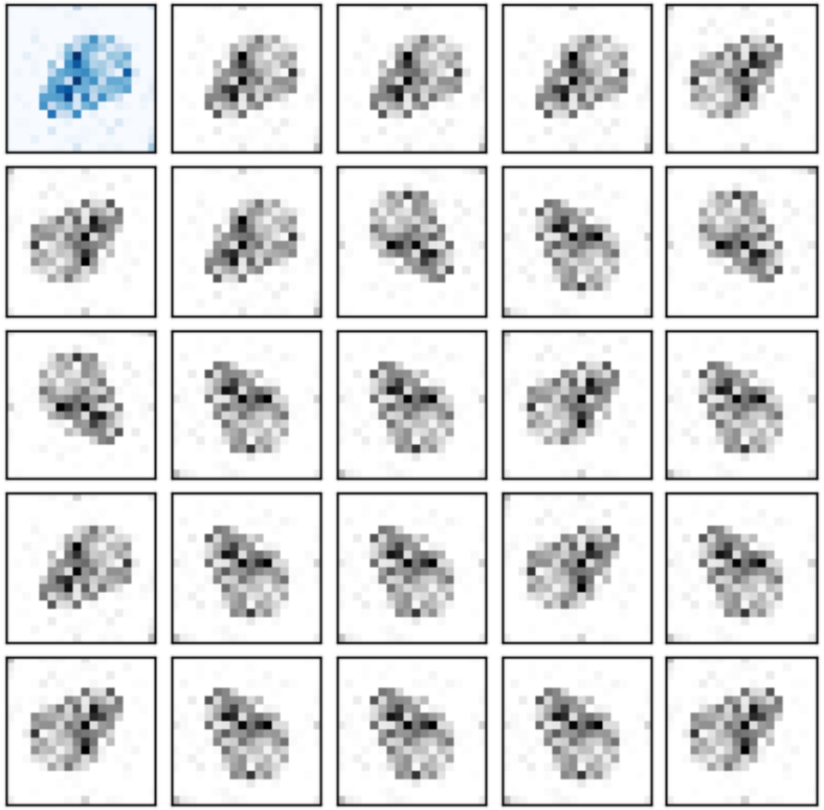
Profiling Tasks 1-3

- Tasks 1 to 3 all deal with noiseless pattern data.
- The magnitude of the dataset size increases.
- **We were able to create one algorithm that can classify patterns into 4 orientation classes for all of Tasks 1 to 3.**



Task 1

25 images (2D)



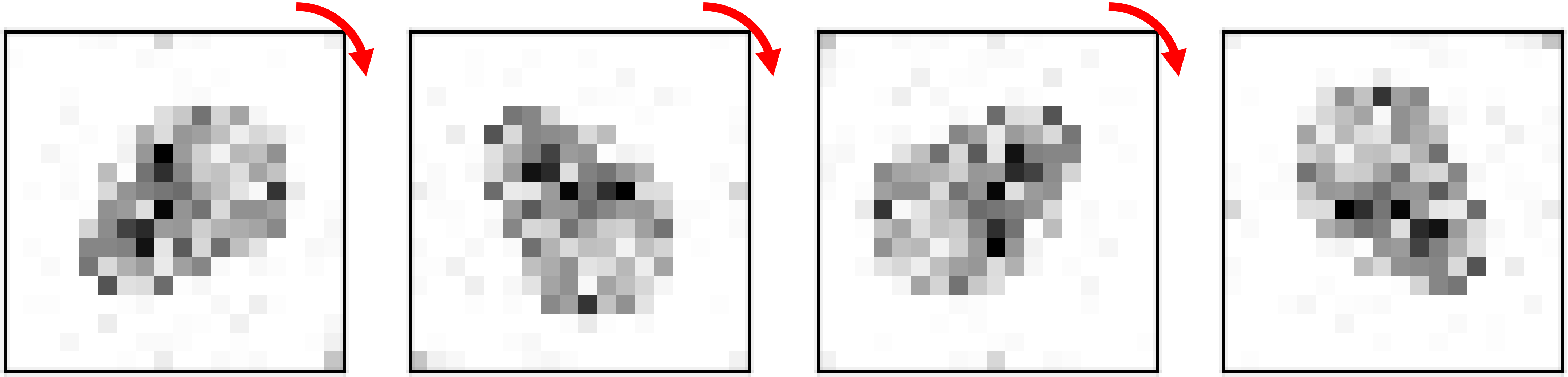
(from problem statement)

Classified into orientation classes

0° 90° 180° 270°

Task 1(a)

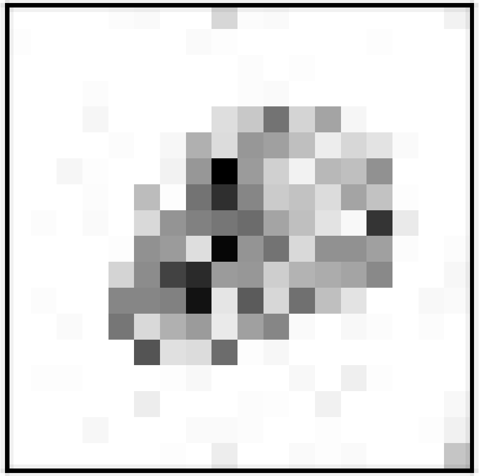
- We produced the rotations of the reference image.



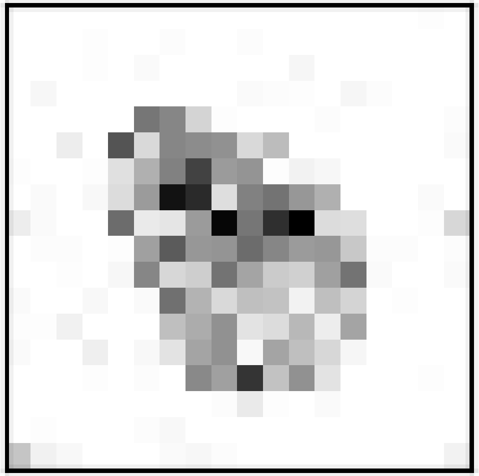
0° 90° 180° 270°

Task 1(b)

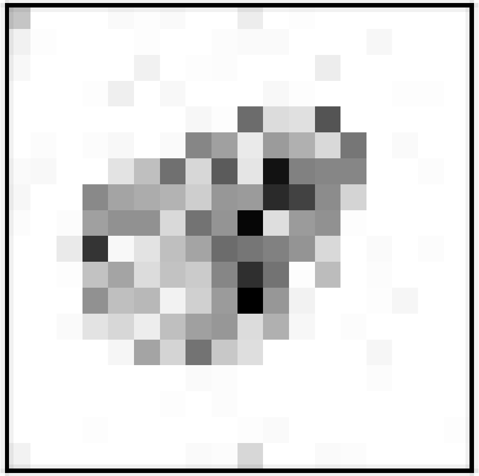
- We produced the rotations of the reference image.



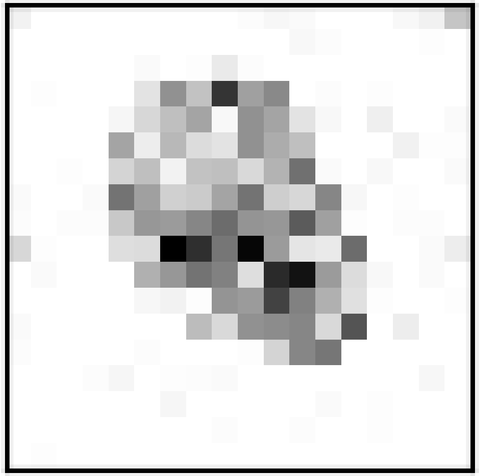
0°
6 patterns



90°
10 patterns

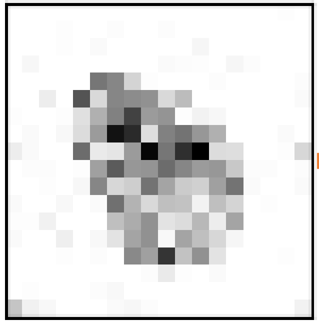


180°
6 patterns



270°
3 patterns

Task 1(c)



Image

$$H = \sum a_i \cdot 2^i \pmod p$$

Prime Hash Function ($p = 10^9 + 7$)

458621916

Hashed Value

0°: 240650399
90°: 458621916
180°: 967870281
270°: 747319434

Hash rotations of the reference image

Compare hashes of each pattern to reference hashes

Task 1(c)

Given

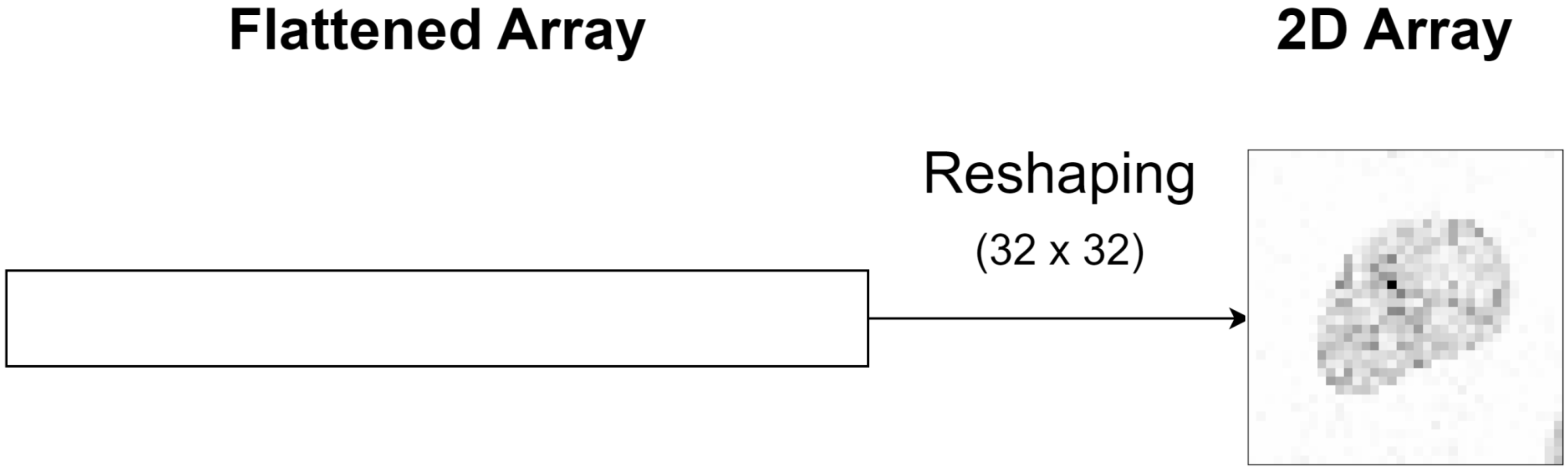
- We are given A = 25 patterns and B = 625 pixels in the image.

Time Complexity

- The time complexity of hashing is O(B).
- The time complexity of all comparisons is O(AB).
- The number of operations is $\sim 10^1 \times 10^2 = 10^3$.
- Non-hashing algorithms to compare patterns to reference images could probably be used (i.e., naïve searching or visual comparison).

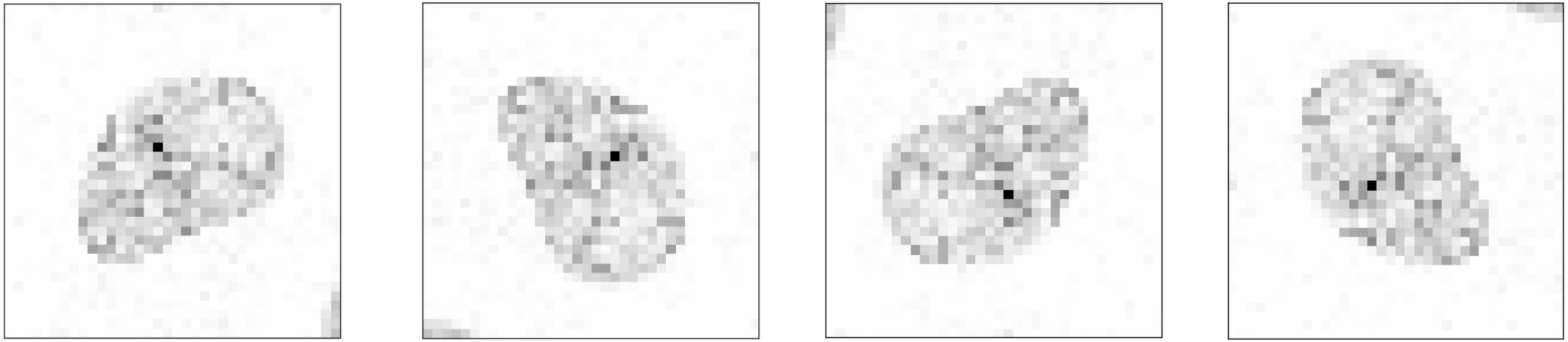
Task 2(a)

- The data was provided in a flattened format.
- We first rendered the reference image by reshaping a 1024-long flattened array into a 32 x 32 array.



Task 2(b)

- We produced the rotations of the reference image and counted the number of such images:



0°
22 patterns

90°
32 patterns

180°
21 patterns

270°
25 patterns

Task 2(c)

- We use the same methodology to compare the images.

Hash rotations of the reference image → Compare pattern hashes to reference hashes

- However, we perform slightly more operations (with a magnitude change of $\sim 10^3$ to $\sim 10^4$). Yet, our algorithm remains resilient to this change in magnitude.

Task 3(a)

- The tasks revealed that the image has dimensions 33 x 33 pixels, since the total number of pixels in the flattened image is 1089 = 33^2 pixels. This produces an image of a horse:

Task 3(b)

- We produced the rotations of the reference image and counted the number of such images:

0°
254 patterns

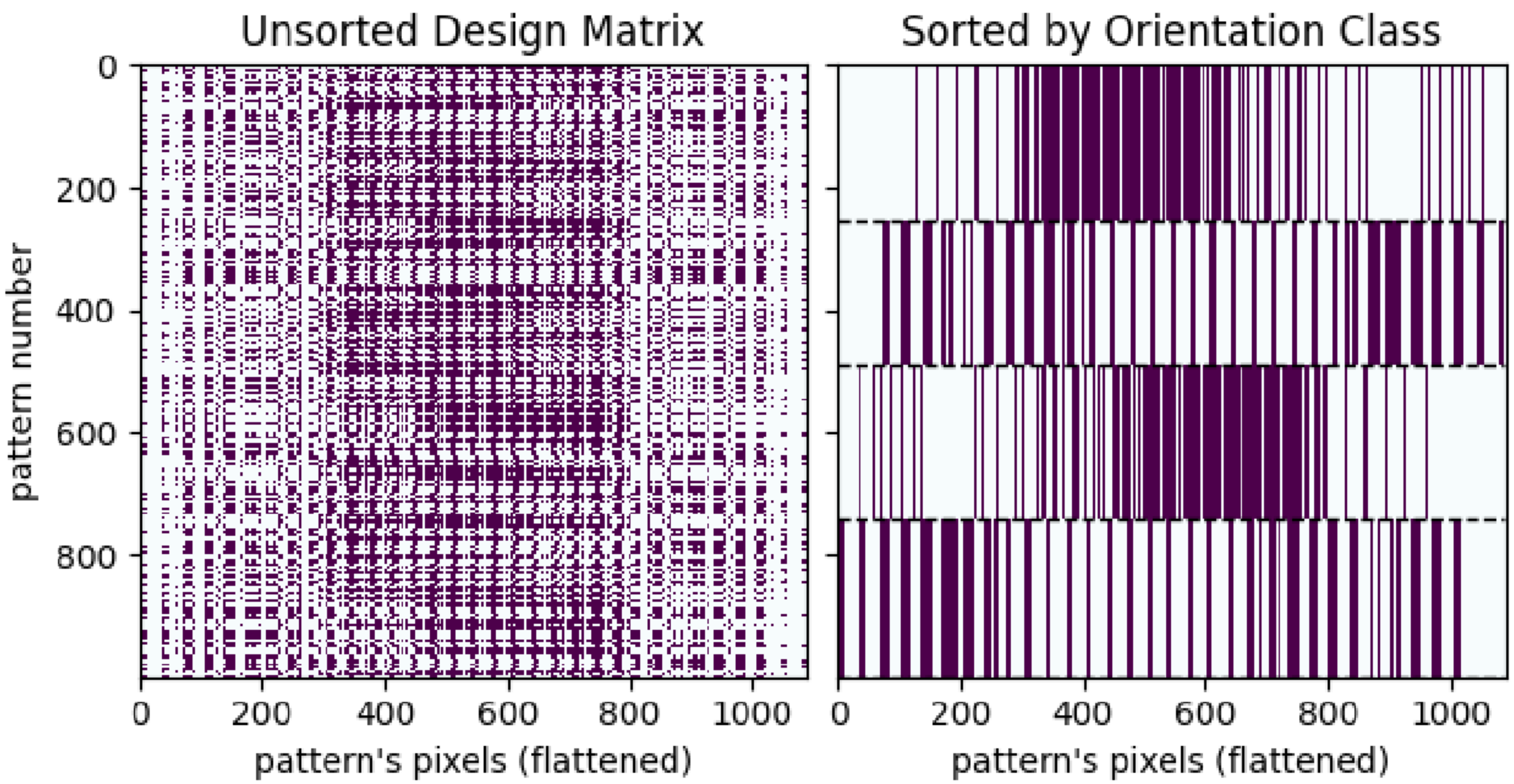
90°
236 patterns

180°
250 patterns

270°
260 patterns

Task 3(c)

- We also show that we have classified the images based on the sorted design matrices:



Task 3(d)

- We use the same methodology to compare the images.

Hash rotations of the reference image → Compare pattern hashes to reference hashes

- However, we perform slightly more operations (with a magnitude change of $\sim 10^4$ to $\sim 10^6$). Yet, our algorithm remains resilient to this change in magnitude.

Part 2:
On Noisy Data

Tasks 4-7: Noisy Images

- **Noise:** A simple comparison of each image to the representative images would no longer work.
- **Larger Datasets:** The number of combinations of the orientations can get drastically higher (as high as $10^{60,000}$).
- **Signal-to-Noise Ratios:** The cause of the difference between the two images is no longer obvious (i.e., noise or orientation). There is also no way to visually confirm the orientations.

Task 4 (Introduction of Noise)

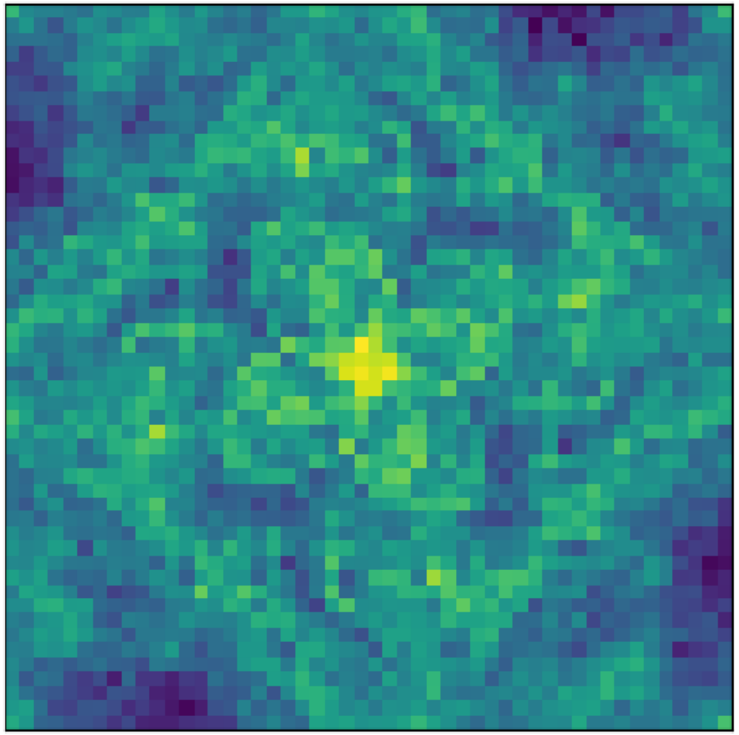
- The dataset for Task 4 is similar to the dataset for Task 3, with a total of A = 1,000 images and B = 50 x 50 = 2,500 pixels. However, noise is introduced in the dataset, which makes hashing useless.

Task 4(a)

- Using native **numpy** implementations, we take the sum of all the values in the B = 50 x 50 pattern (sum of the values in each row of the design matrix), and take its average over A = 1,000 patterns. This average is equal to **1251.047**.

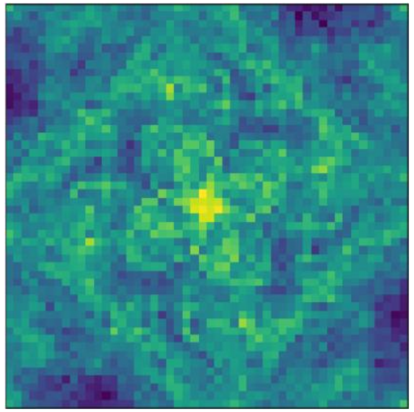
Task 4(b)

- Rendering the average value of each dataset, we arrive at this figure (the coloring scheme does not matter much):

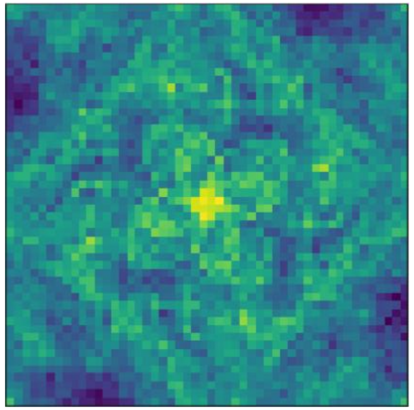


Task 4(b)

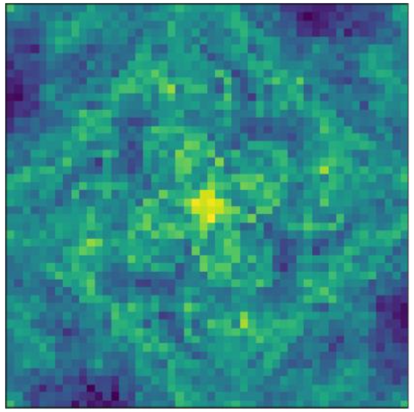
- This figure has an approximate 4-fold symmetry, which motivates us to use $K = 4$ means for our clustering algorithm:



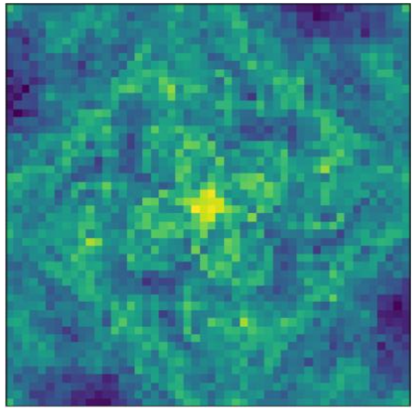
Original Image



90°



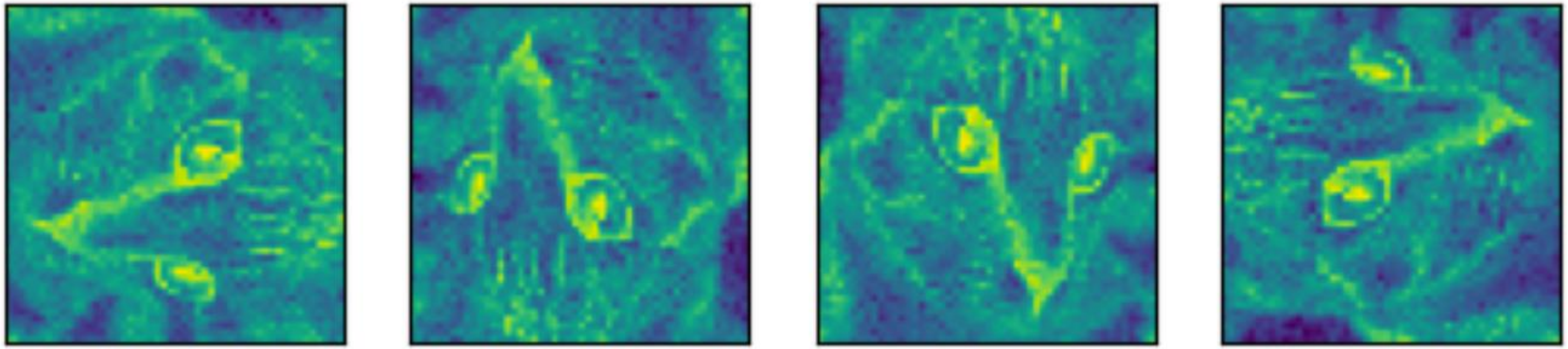
180°



270°

Task 4(c)

- After implementing the K-means clustering algorithm (K = 4), we arrive at the following images separated by orientation:



- This looks like a cat. Quite cute, actually.

Task 4(d)

Implement *K*-means clustering algorithm → Set the following parameters: *K* = 4 and *N* = 10 initializations → Set initialization method to kmeans++

KMeans module from **scikit-learn** library

Task 4(d)

- Here are the time complexities for the two tasks:

Algorithm	Time Complexity
K-means clustering	$O(NKI \times AB)$
Hashing algorithm	$O(AB)$

A = number of images
B = number of pixels per image
N = number of initializations
K = number of clusters
I = number of iterations

- The time is **linear with the size of the dataset**.
- Keeping the **hyperparameters *N*, *K*, and *I* constant**, the time complexity of *K*-means is **asymptotically optimal**, which performs about the same as hashing.

Part 4:
Likelihood Models

Task 5(a) (Success with Noise Distributions)

- We number the pixels from left to right and top to bottom:

$\boldsymbol{\mu}$	$\beta\lambda$	λ	$\beta\lambda$	$\beta\lambda$
$\boldsymbol{K}$	1	0	0	0

$$\mathscr{L}(r = 90^\circ \,|\, K_{(4)}, \mu_{(4)}) = \mathbf{Pr}(k_1 = 1|\beta\lambda) \cdot \mathbf{Pr}(k_2 = 0|\lambda) \cdot \mathbf{Pr}(k_3 = 0|\beta\lambda) \cdot \mathbf{Pr}(k_4 = 0|\beta\lambda)$$

$$= \boxed{\beta\lambda(1-\lambda)(1-\beta\lambda)^2}$$

Task 5(b)

- We use the expression from Task 5(a). Substituting the aligned likelihood from the Problem Statement and the misalignment likelihood from Task 5(a), the likelihood ratio is:

$$\frac{\mathcal{L}(r = 0^\circ | K_{(4)}, \mu_{(4)})}{\mathcal{L}(r = 90^\circ | K_{(4)}, \mu_{(4)})} = \frac{\lambda(1 - \beta\lambda)^3}{\beta\lambda(1 - \lambda)(1 - \beta\lambda)^2} = \boxed{\frac{1 - \beta\lambda}{\beta(1 - \lambda)}}$$

Task 5(c)

- We use the likelihood ratio from Task 5(b):

$$\frac{1-\beta\lambda}{\beta(1-\lambda)} = 1$$

$$1-\beta\lambda = \beta(1-\lambda)$$

$$\boxed{\beta = 1}$$

- When $\beta = 1$, all pixels have the same value.

Task 5(d)

aligned λ	λ	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$
measured K	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$$\frac{\mathscr{L}(r=0^\circ|K_{(16)},\mu_{(16)})}{\mathscr{L}(r=90^\circ|K_{(16)},\mu_{(16)})} = \frac{\boxed{\mathbf{Pr}(k=1|\lambda)}\cdot\boxed{\mathbf{Pr}(k=0|\beta\lambda)^{15}}}{\mathbf{Pr}(k=1|\beta\lambda)\cdot\mathbf{Pr}(k=0|\lambda)\cdot\mathbf{Pr}(k=0|\beta\lambda)^{14}}$$

$$= \frac{\lambda(1-\beta\lambda)^{15}}{\beta\lambda(1-\lambda)(1-\beta\lambda)^{14}} = \boxed{\frac{1-\beta\lambda}{\beta(1-\lambda)}}$$

Task 5(d)

misaligned λ	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	λ	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$
measured K	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$$\frac{\mathscr{L}(r=0^\circ|K_{(16)},\mu_{(16)})}{\mathscr{L}(r=90^\circ|K_{(16)},\mu_{(16)})} = \frac{\mathbf{Pr}(k=1|\lambda)\cdot\mathbf{Pr}(k=0|\beta\lambda)^{15}}{\boxed{\mathbf{Pr}(k=1|\beta\lambda)}\cdot\boxed{\mathbf{Pr}(k=0|\lambda)}\cdot\boxed{\mathbf{Pr}(k=0|\beta\lambda)^{14}}}$$

$$= \frac{\lambda(1-\beta\lambda)^{15}}{\beta\lambda(1-\lambda)(1-\beta\lambda)^{14}} = \boxed{\frac{1-\beta\lambda}{\beta(1-\lambda)}}$$

Task 5(d)

aligned λ	λ	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$
misaligned λ	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	λ	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$
measured K	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$$\frac{\mathscr{L}(r = 0^\circ | K_{(16)}, \mu_{(16)})}{\mathscr{L}(r = 90^\circ | K_{(16)}, \mu_{(16)})} = \frac{\mathbf{Pr}(k = 1|\lambda) \cdot \mathbf{Pr}(k = 0|\beta\lambda)^{15}}{\mathbf{Pr}(k = 1|\beta\lambda) \cdot \mathbf{Pr}(k = 0|\lambda) \cdot \mathbf{Pr}(k = 0|\beta\lambda)^{14}}$$

$$= \frac{\lambda(1 - \beta\lambda)^{15}}{\beta\lambda(1 - \lambda)(1 - \beta\lambda)^{14}} = \boxed{\frac{1 - \beta\lambda}{\beta(1 - \lambda)}}$$

Task 5(e)

- The alignment table is:

master	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0
0° (aligned)	λ	$\beta\lambda$	λ	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	λ	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$
90° (misaligned)	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	λ	$\beta\lambda$	λ	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	$\beta\lambda$	λ	$\beta\lambda$	$\beta\lambda$

- The likelihood ratio is:

$$\frac{\mathscr{L}(\text{aligned})}{\mathscr{L}(\text{misaligned})} = \frac{\mathbf{Pr}(k=1|\lambda)^3 \cdot \mathbf{Pr}(k=0|\beta\lambda)^{13}}{\mathbf{Pr}(k=0|\lambda)^3 \cdot \mathbf{Pr}(k=1|\beta\lambda)^3 \cdot \mathbf{Pr}(k=0|\beta\lambda)^{10}}$$

$$= \frac{\lambda^3(1-\beta\lambda)^{13}}{(1-\lambda)^3(\beta\lambda)^3(1-\beta\lambda)^{10}} = \boxed{\frac{(1-\beta\lambda)^3}{\beta^3(1-\lambda)^3}} \qquad \text{(Eq. 2.4)}$$

Task 5(f)

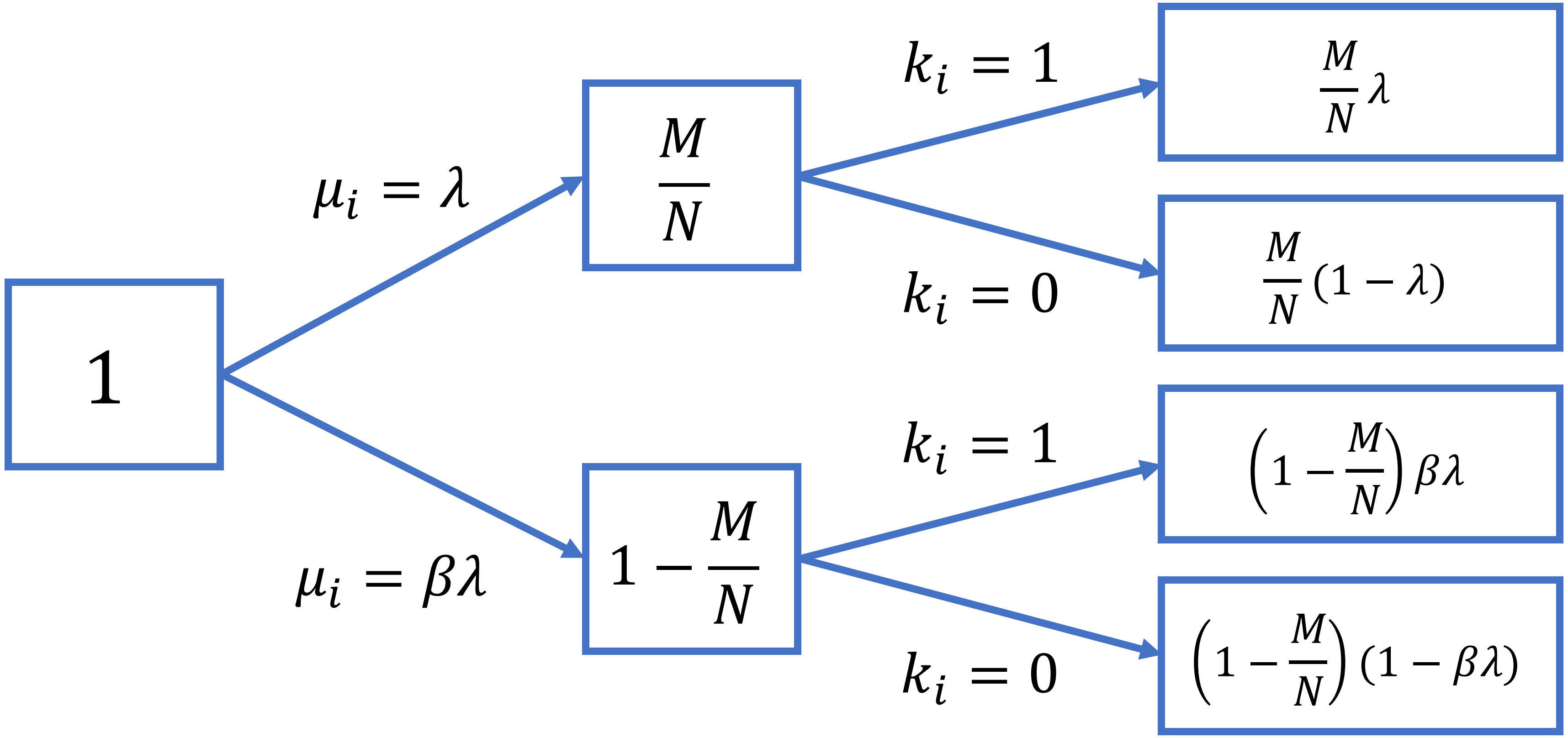
- We define a function $f(\mu_i, k_i)$ as in the problem statement:

$$f(\mu_i, k_i) = k_i \ln \mu_i + (1 - k_i) \ln(1 - \mu_i)$$

- Here are the pertinent values of $f(\mu_i, k_i)$:

μ_i	k_i	$f(\mu_i, k_i)$
λ	1	$\ln \lambda$
λ	0	$\ln(1-\lambda)$
$\beta\lambda$	1	$\ln(\beta\lambda)$
$\beta\lambda$	0	$\ln(1-\beta\lambda)$

Task 5(f)


1
$\mu_i = \lambda$
$\mu_i = \beta\lambda$
$\frac{M}{N}$
$1 - \frac{M}{N}$
$k_i = 1$
$k_i = 0$
$k_i = 1$
$k_i = 0$
$\frac{M}{N}\lambda$
$\frac{M}{N}(1-\lambda)$
$\left(1-\frac{M}{N}\right)\beta\lambda$
$\left(1-\frac{M}{N}\right)(1-\beta\lambda)$

Task 5(f)

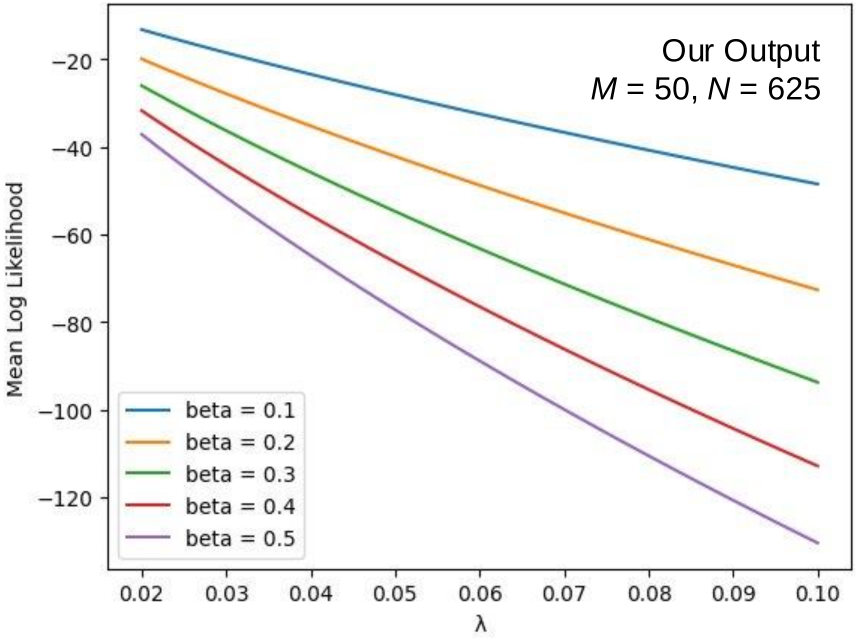
- The expected value of the log-likelihood is:

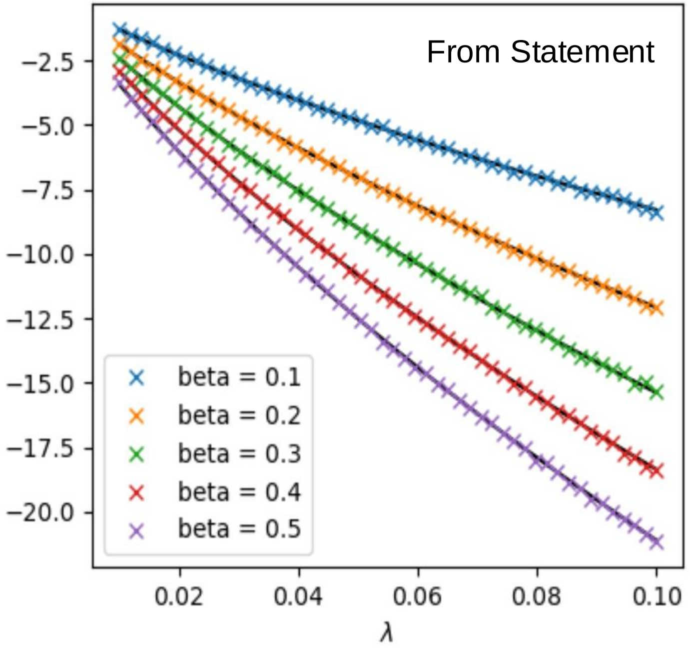
$$\begin{aligned}\mathbf{E}(\text{likelihood}) &= \mathbf{E}\left(\sum f(\mu_i, k_i)\right) = N \cdot \mathbf{E}(f(\mu_i, k_i)) \\ &= N\left[\frac{M}{N}\lambda \cdot \ln\lambda + \frac{M}{N}(1-\lambda)\cdot\ln(1-\lambda) + \left(1-\frac{M}{N}\right)\beta\lambda\cdot\ln(\beta\lambda) + \left(1-\frac{M}{N}\right)(1-\beta\lambda)\cdot\ln(1-\beta\lambda)\right] \\ &= \boxed{M(\lambda\ln\lambda + (1-\lambda)\ln(1-\lambda)) + (N-M)(\beta\lambda\ln(\beta\lambda) + (1-\beta\lambda)\ln(1-\beta\lambda)}\end{aligned}$$

- This looks similar to the cross-entropy of a system.

Task 5(f)

$$\mathbf{E}(\text{likelihood}) = \mathbf{E}\left(\sum f(\mu_i, k_i)\right) = N \cdot \mathbf{E}(f(\mu_i, k_i))$$
$$= N\left[\frac{M}{N}\lambda \cdot \ln\lambda + \frac{M}{N}(1-\lambda)\cdot\ln(1-\lambda) + \left(1-\frac{M}{N}\right)\beta\lambda\cdot\ln(\beta\lambda) + \left(1-\frac{M}{N}\right)(1-\beta\lambda)\cdot\ln(1-\beta\lambda)\right]$$
$$= \boxed{M(\lambda\ln\lambda + (1-\lambda)\ln(1-\lambda)) + (N-M)(\beta\lambda\ln(\beta\lambda) + (1-\beta\lambda)\ln(1-\beta\lambda))}$$

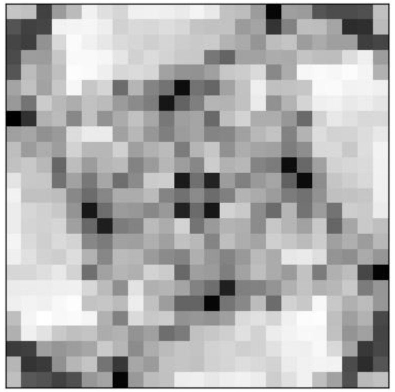




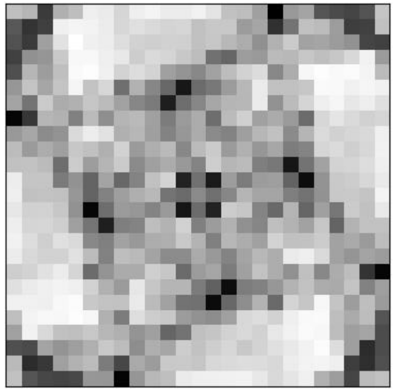
Part 5:
Noisy Sparse Data

Task 6 (Initial Observations)

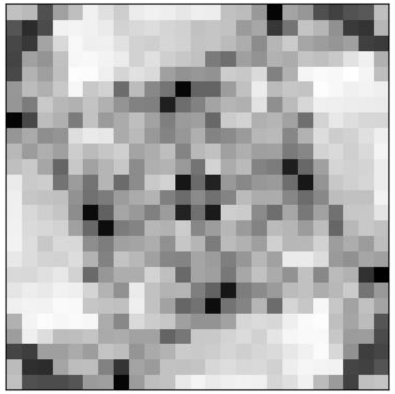
- The mean image for Task 6 has four-fold symmetry:



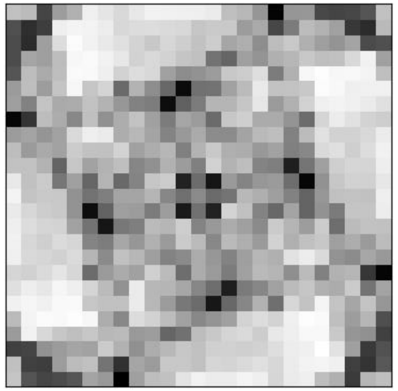
Original Image



90°



180°



270°

Task 6 (Problems with Dense Representation)

- For Task 6, there are A = 65,535 images with each image having a size B = 25 x 25 = 625 pixels.
- **Quick Computation:** How much memory would it take to store all images using 64-bit floating point numbers?

$$65{,}535 \times 625 \times 64 \text{ bits} \approx \mathbf{312.5\ Mb}$$

- This uses up a lot of memory. Also, K-means runs multiple iterations of cluster and mean assignments. Naïve K-means implementations will take too long to run on the dataset.

Task 6 (Observations about the Data)

- The average number of activated pixels per image is 23.745, which is a lot smaller than the number of pixels per image (625).
- Majority of the pixels are set to a value of 0, so the activated pixels (i.e., pixels with a value of 1) are sparsely distributed within the image.

Task 6 (Sparse Representation)

- **Sparse matrices** work with sparse data (i.e., arrays with only a few nonzero values).

Array

	0	1	2	3
0	0	0	2	0
1	3	0	0	0
2	0	0	0	0
3	0	0	1	0
4	0	0	0	0
5	0	0	0	0

→

Sparse Matrix

(Coordinate Form)

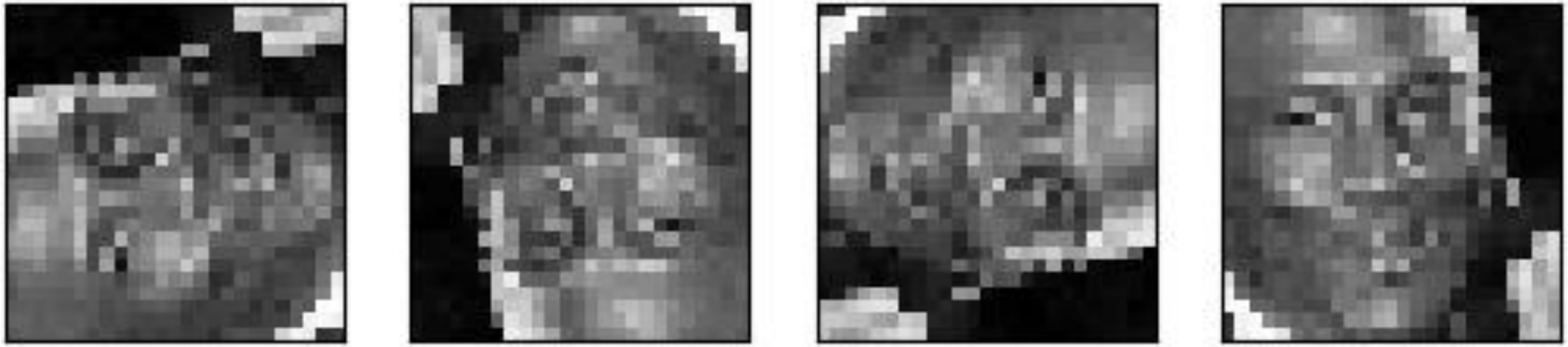
row number	column number	value
0	2	2
1	0	3
3	2	1

Task 6 (Sparse Representation)

- Sparse matrices have lower memory requirements:
  - Normal Array: O(L), where L – elements in the array.
  - Sparse Matrix: O(L'), where L' – nonzero elements.
- If $L' \ll L$, sparse matrices use significantly less memory.
- For Task 6, L = 40,959,375, while L' = 1,556,153. Since $L' \ll L$, sparse matrices use significantly less memory for Task 6.
- We used K-means clustering with sparse matrices.

Task 6 (Results)

- After applying the *K*-means clustering algorithm on the Task 6 dataset, we obtain the following cluster means:



- This might be Michael Jordan? We are also not sure.

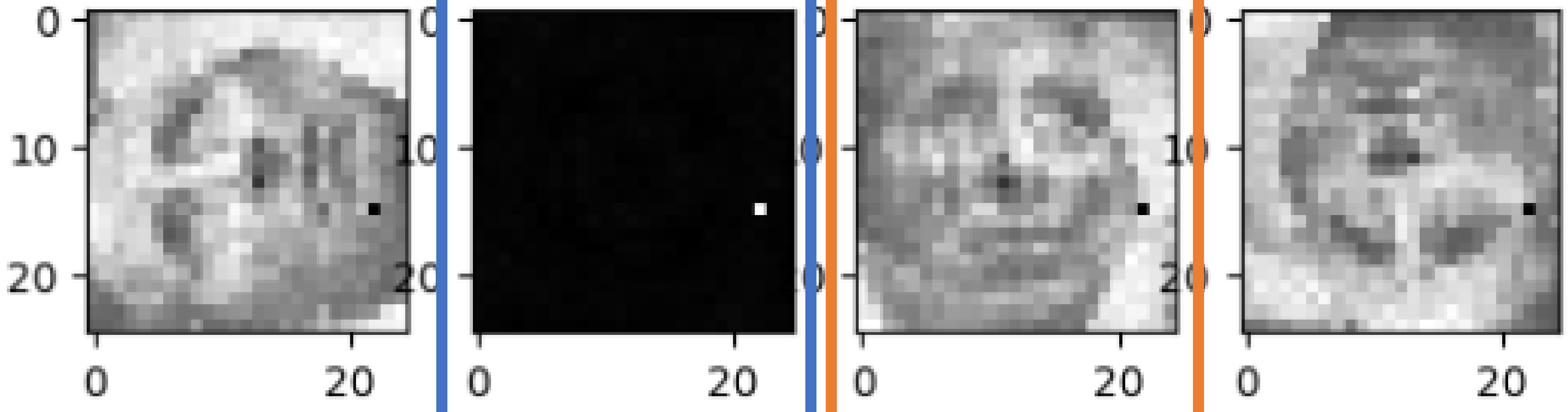
Part 6: The Master Image

Task 7 (Problems with Dense Representations$^{2.0}$)

- In the dataset, there are a total of A = 100,000 images, each with a size B = 25 x 25 = 625 pixels.
- The dataset is composed of **task7a** (with parameters x_i indicating the image number, $0 \leq x_i \leq A - 1$) and **task7b** (with parameters y_i indicating the pixel number in the design matrix, $0 \leq y_i \leq B - 1$).

Task 7 (Problems with Dense Representations$^{2.0}$)

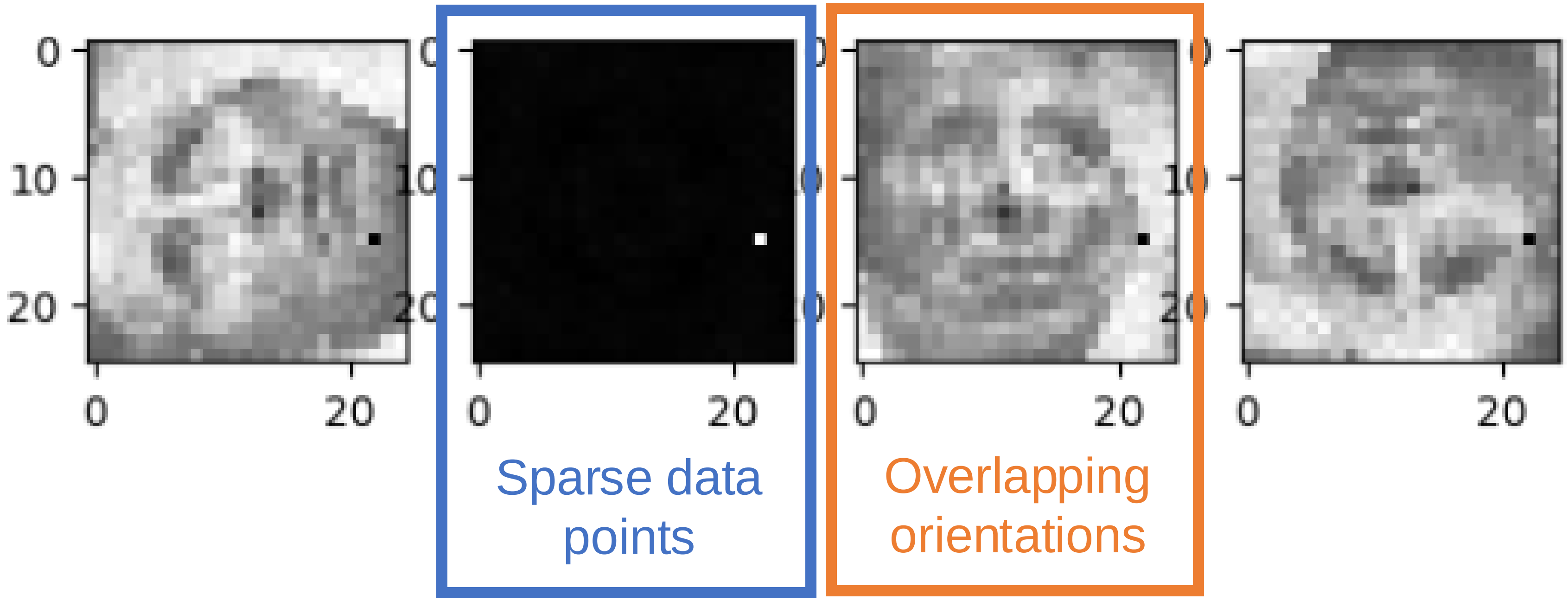
- The implementation of *K*-means clustering algorithm in Task 6 came with a lot of failures:



Sparse data points

Overlapping orientations

Task 7 (Problems with Dense Representations$^{2.0}$)



- These are both characterized by low entropy (same colors).
- This is a problem with initial mean selection.

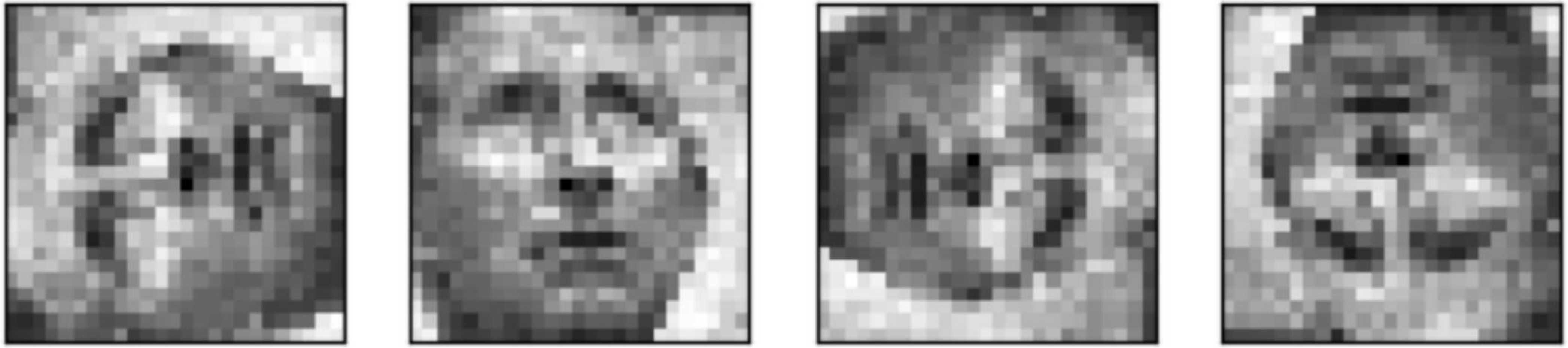
Task 7 (Problems with Dense Representations$^{2.0}$)

- We decided to characterize the “messiness” or the entropy of the function using a modified Shannon entropy:

$$S(\vec{p}) = -\sum \left[p_i \log_2 p_i + (1 - p_i) \log_2 p_i\right]$$

Task 7(a)

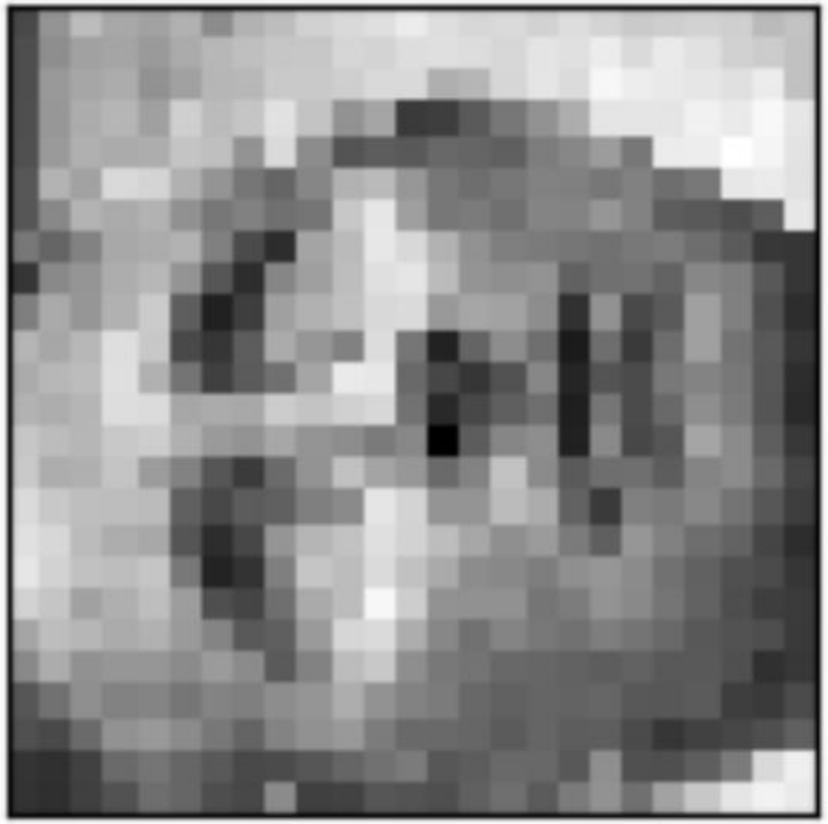
- The oriented images that result from a modified version of the *K*-means algorithm are as follows:



- Might be Vladimir Putin? Boris Johnson? King Charles? James Bond? We don’t know, but it does look like a person.

Task 7(a)

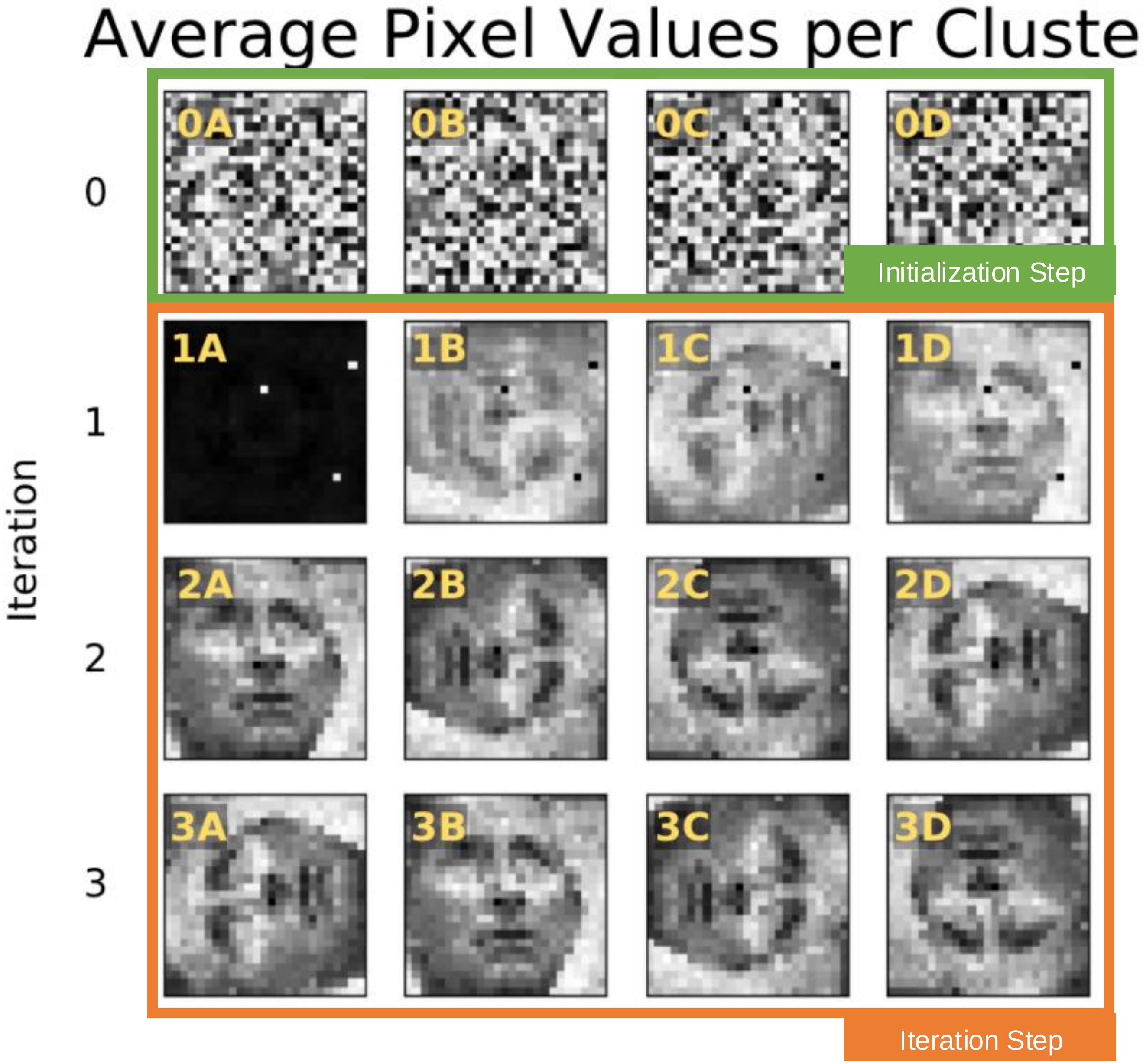
- We implemented a system that determines the orientation of each sample image, and takes the average of the aligned images to get a "correct" master image. *(Note: We assumed that the master image is orientation-agnostic.)*



"Correct" Master Image

Task 7(b)

- There are two steps to the implementation of our modified *K*-means algorithm.



Task 7(b)

- There are two steps to our modified *K*-means algorithm. The first step is the mean initialization step:

Generate a random matrix $\boldsymbol{\mu}_0'$ with elements uniformly distributed between 0 and 1

→

Guess that the means of the clusters are the matrix rotations $\boldsymbol{\mu}_{0(0)}, \ldots, \boldsymbol{\mu}_{0(270)}$ of the random matrix.

Task 7(b)

- The second step is the mean iteration step. **For i = 1, 2, …, *T*, where *T* is the number of iterations steps, do the following:**

Implement the *K*-means algorithm using the rotation matrices $\boldsymbol{\mu}_{(i-1)(0)}, \ldots, \boldsymbol{\mu}_{(i-1)(270)}$ as the initial means.

→ Get four candidate master images $\boldsymbol{M}_{(i)(1)}, \ldots, \boldsymbol{M}_{(i)(4)}$ as a result of the *K*-means algorithm

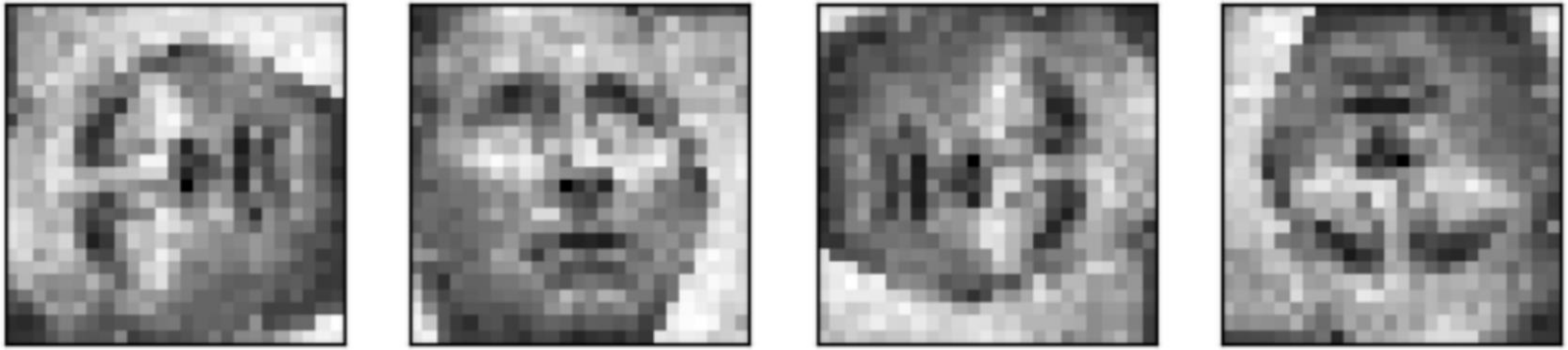
→ Find the candidate image $\boldsymbol{\mu}_i'$ among $\boldsymbol{M}_{i(1)}, \ldots, \boldsymbol{M}_{i(4)}$ with the highest **entropy**.

→ Set the rotations of $\boldsymbol{\mu}_i'$: $\boldsymbol{\mu}_{(i)(0)}, \ldots, \boldsymbol{\mu}_{(i)(270)}$ to be the new means for the *K*-means algorithm

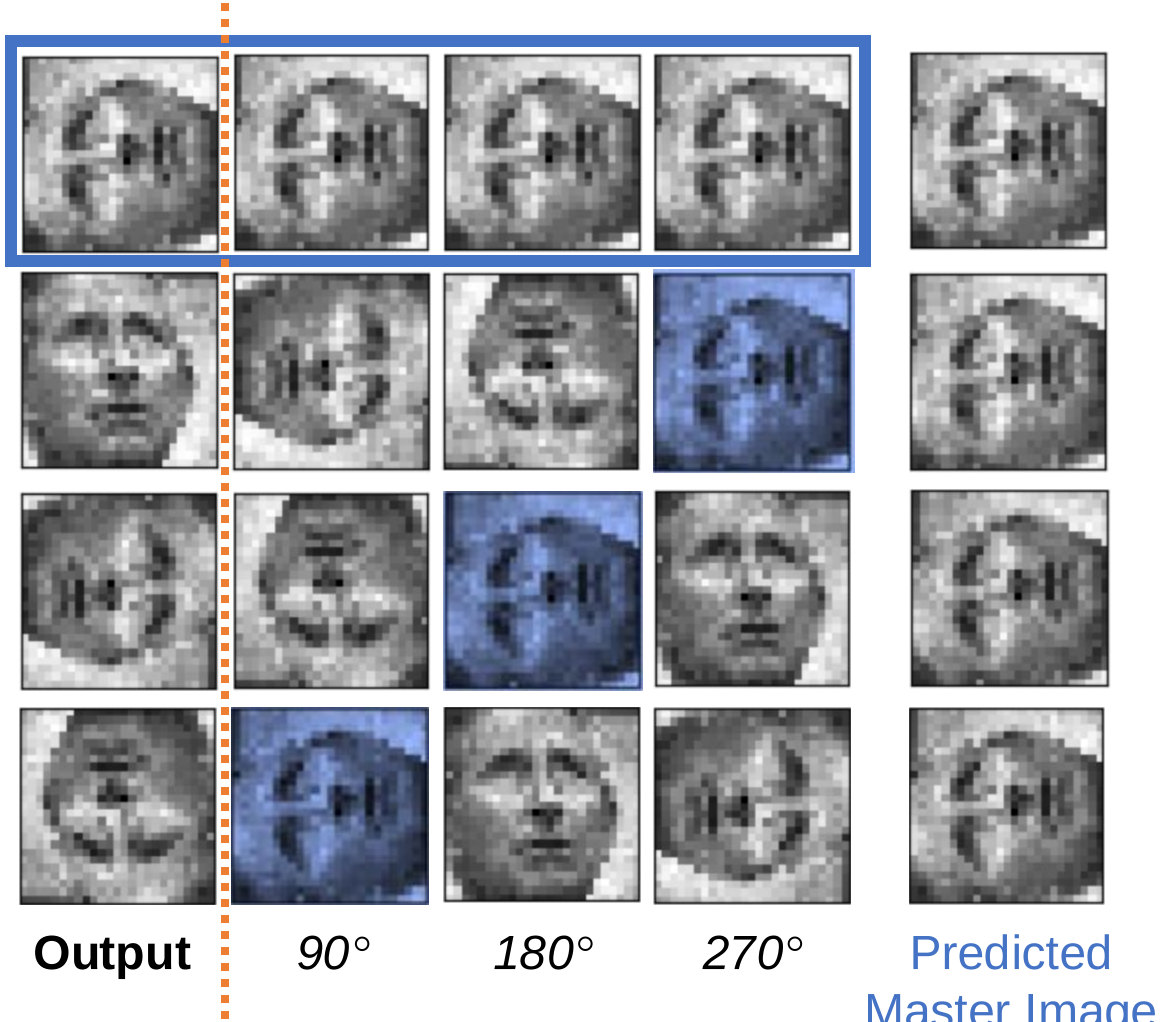
- After *T* iterations, the master images $\boldsymbol{M}_{(T)(1)}, \ldots, \boldsymbol{M}_{(T)(4)}$ produce our candidate images.

Task 7(b)

- We attempt to extract a master image from each of the orientations that was extracted after *T* iterations of our modified algorithm.

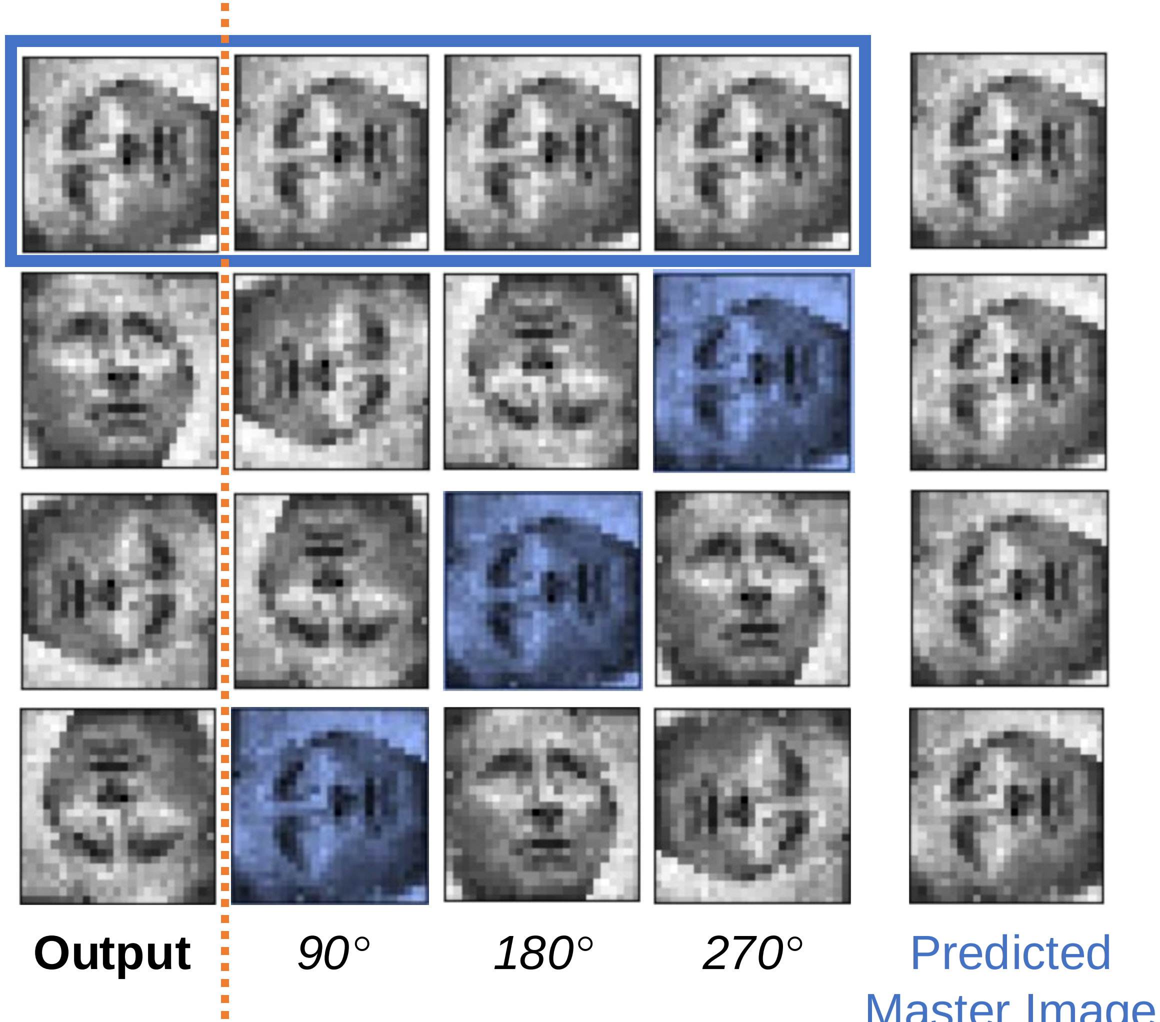


Task 7(b)



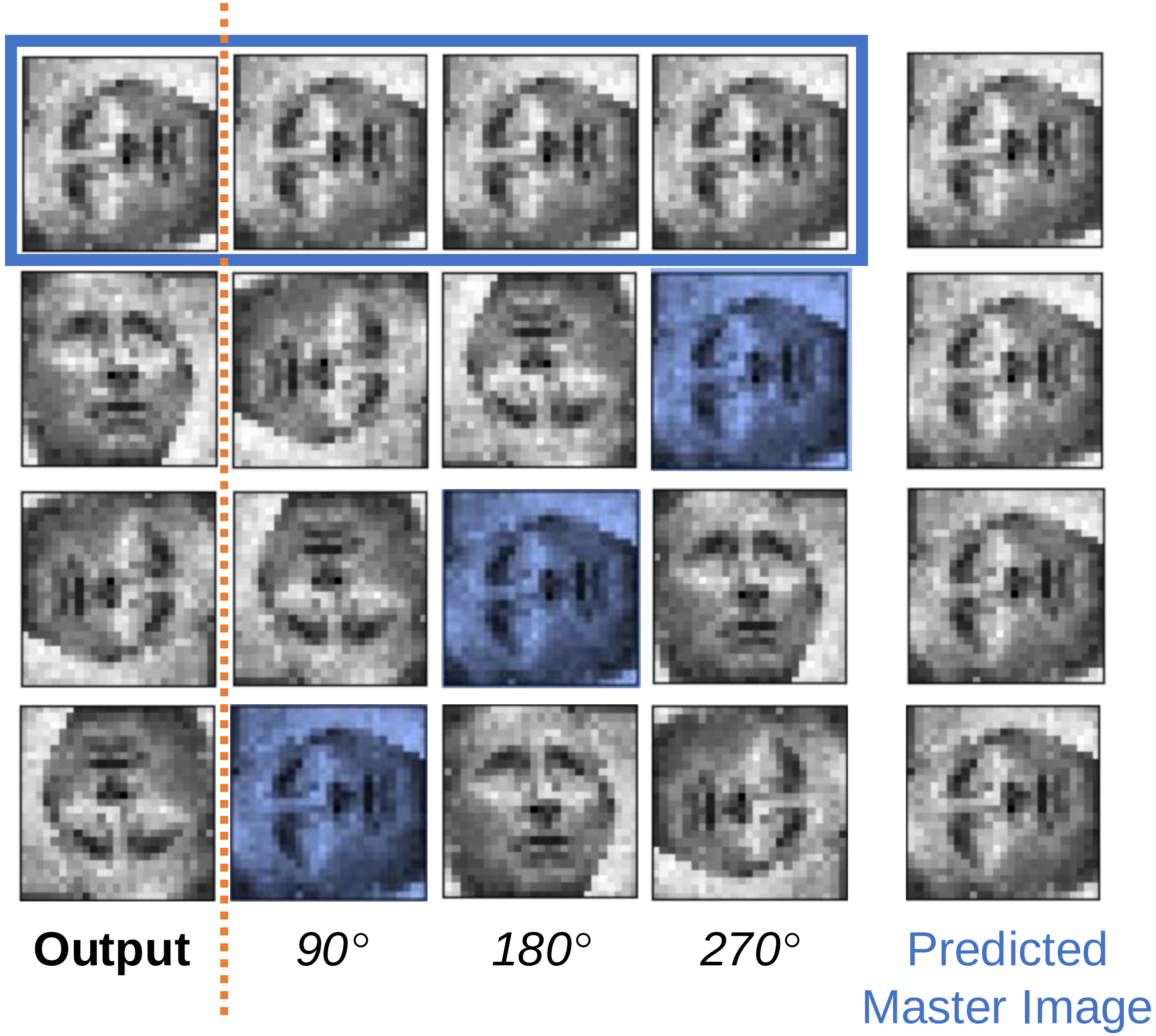
- First, we guess that the master image is in the orientation of the first master image $\boldsymbol{M}_{(T)(1)}$ since K-means clustering cannot determine what an “upright” face looks like (i.e., master images are orientation-agnostic).

Task 7(b)



- Second, for each of the master images $\boldsymbol{M}_{(T)(2)}, \ldots, \boldsymbol{M}_{(T)(4)}$, we determine the orientation with the closest similarity (cosine similarity) to master image $\boldsymbol{M}_{(T)(1)}$. The closest image is highlighted in blue.

Task 7(b)

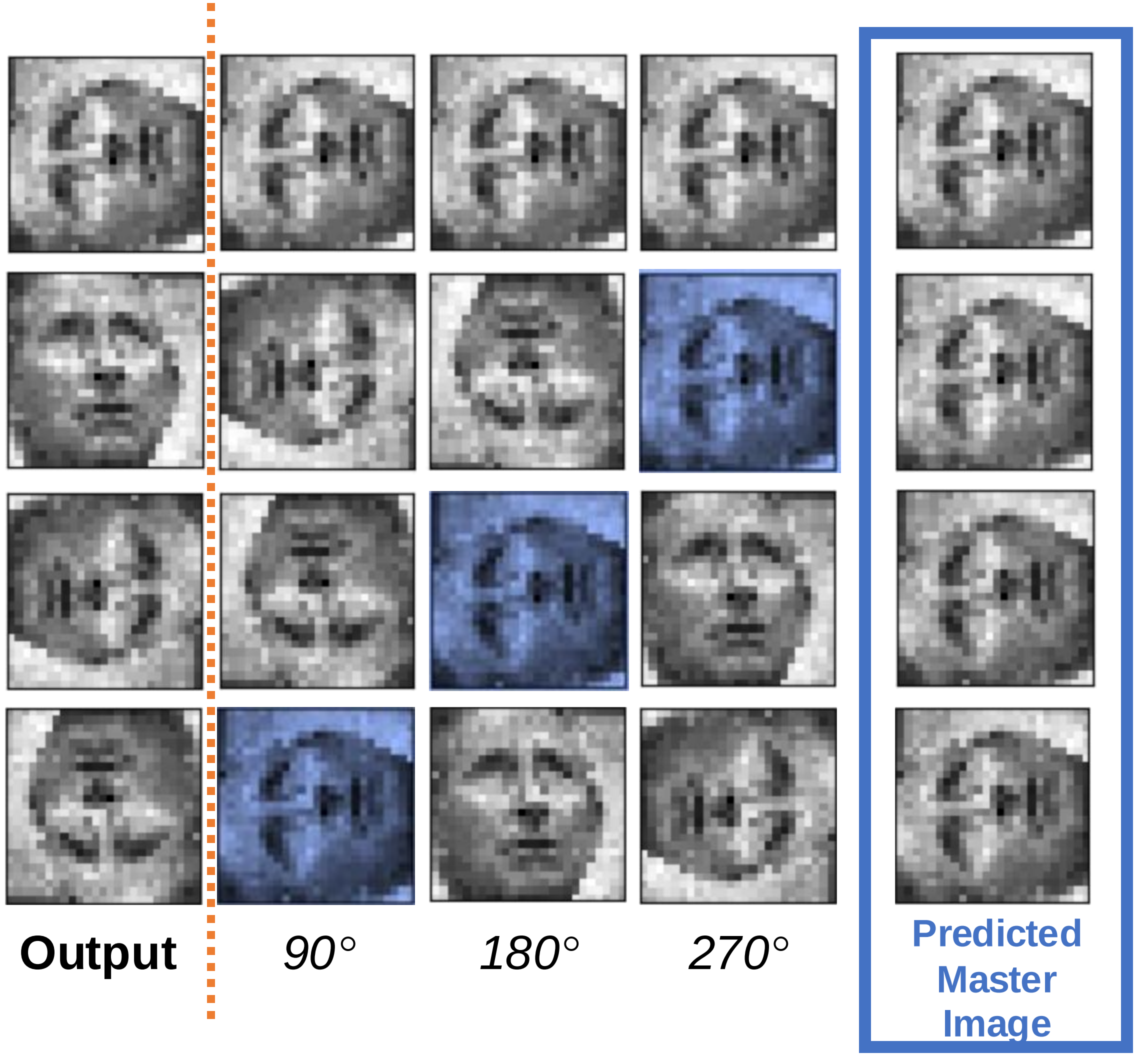


Cosine Similarity Formula

$$\boxed{\cos\theta = \frac{\vec{\mu}_1 \cdot \vec{\mu}_2}{||\vec{\mu}_1|| \, ||\vec{\mu}_2||}}$$

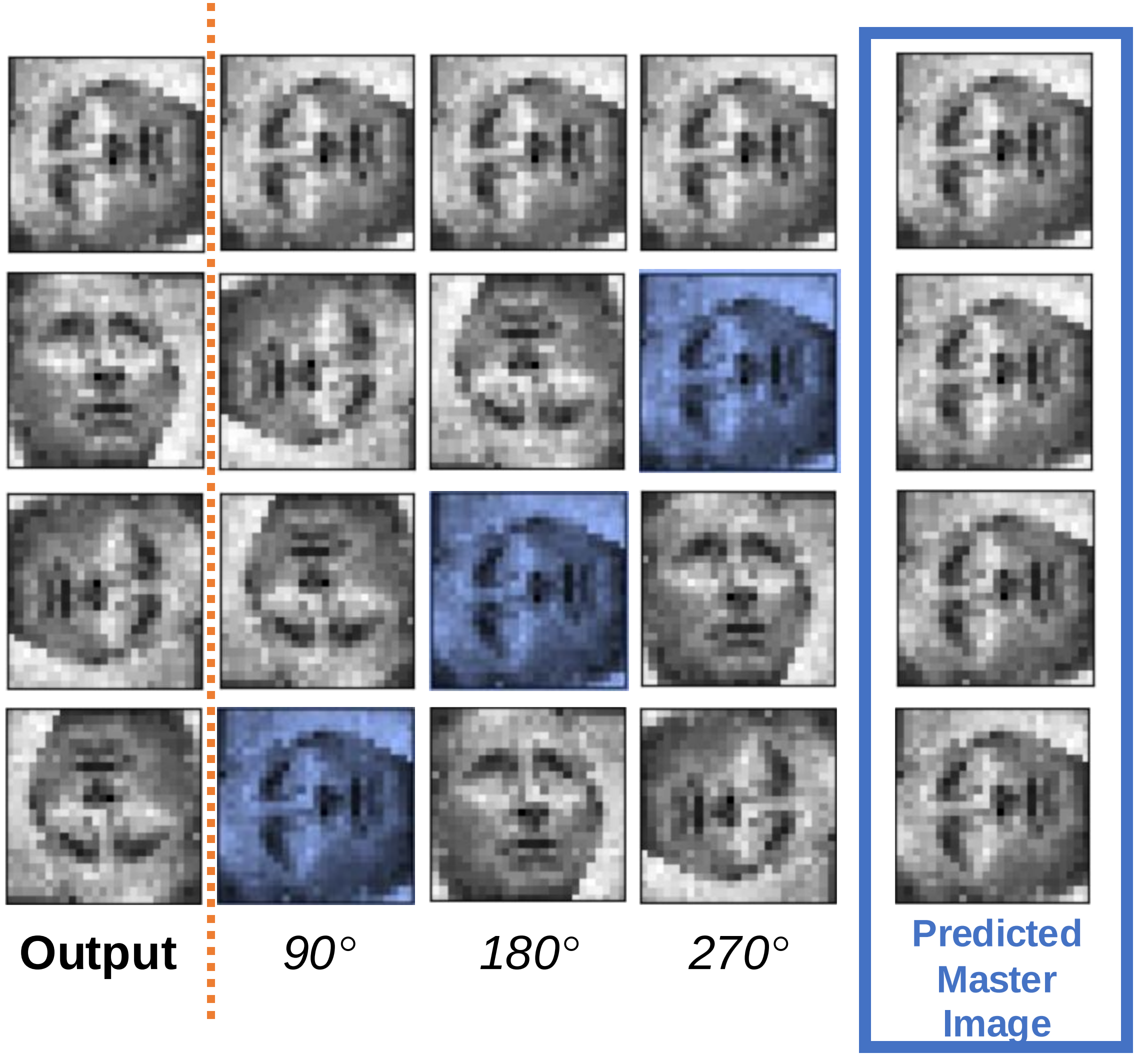
$\vec{\mu}$ = vector representation of an image

Task 7(b)

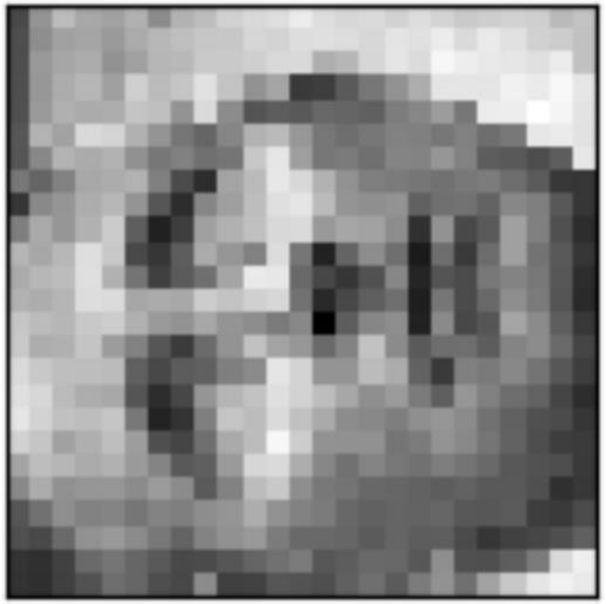


- Finally, we take the average pixel values of each the predicted master images for each orientation.

Task 7(b)



Average Pixel Values



Part 7: Recommendations

Recommendations

- Implement *K*-means clustering with a modification of the cluster-entropy (since there is no way to change the distance function in **sklearn**, and cluster-entropy does not satisfy the conditions for a distance function).
- Implement a measure of convergence (similar to the Frobenius norm for *K*-means clustering) to determine how fast the means do not change (as seen in Task 7).

On Classification of Noiseless and Noisy Pattern Orientations

SIMC Endeavor Team 04
Hans Gabriel D. De Vera
Benjamin L. Jacob
Davis Nicholo A. Magpantay

Republic of the Philippines
Department of Science and Technology
Philippine Science High School – Main Campus
Agham Rd., Diliman, Quezon City, Philippines

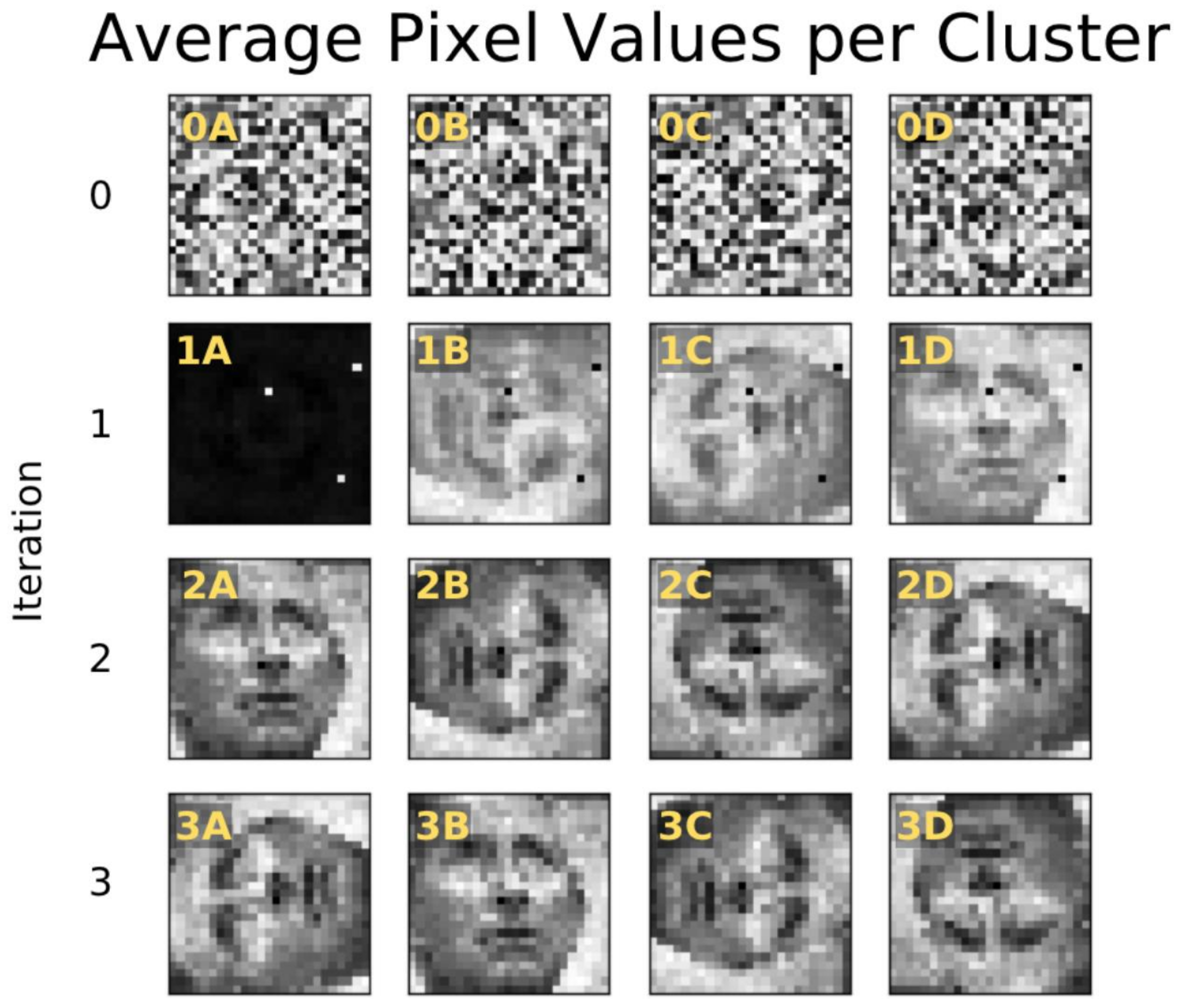
Appendix

Modified K-means Algorithm

Algorithm 1 Modified K-means Algorithm for Task 7b. The number of iterations to run the algorithm for, T, is a hyperparameter.

```
procedure ModifiedKMeans
    Initialize the master image μ'_0 with each entry uniformly sampled between 0 and 1.
    Predict the means of the cluster to be the four rotations μ_{0(0°)}, ..., μ_{0(270°)} of μ'_0.
    for iterations i = 1, 2, 3, ..., T, do
        Implement the k-means algorithm using μ_{(i−1)(0°)}, ..., μ_{(i−1)(270°)} as the initial means.
        Store the four master images M⃗_{i(1)}, ..., M⃗_{i(4)} from the k-means algorithm.
        Select the cluster μ'_i with the highest entropy S(μ⃗') (Eq. 2.7) as the new master image.
        Set the new means μ_i to be the 90° rotations of μ'_i.
    end for
    for images j = 1, 2, 3, 4 do
        Define A⃗_j to be a rotation of the vector M⃗_{T(j)} that maximizes the cosine similarity with M⃗_{T(1)}.
    end for
    Take the predicted master image to be M⃗ = (A⃗_1 + A⃗_2 + A⃗_3 + A⃗_4) / 4
```

Visualizing the Clustering (Task 7)



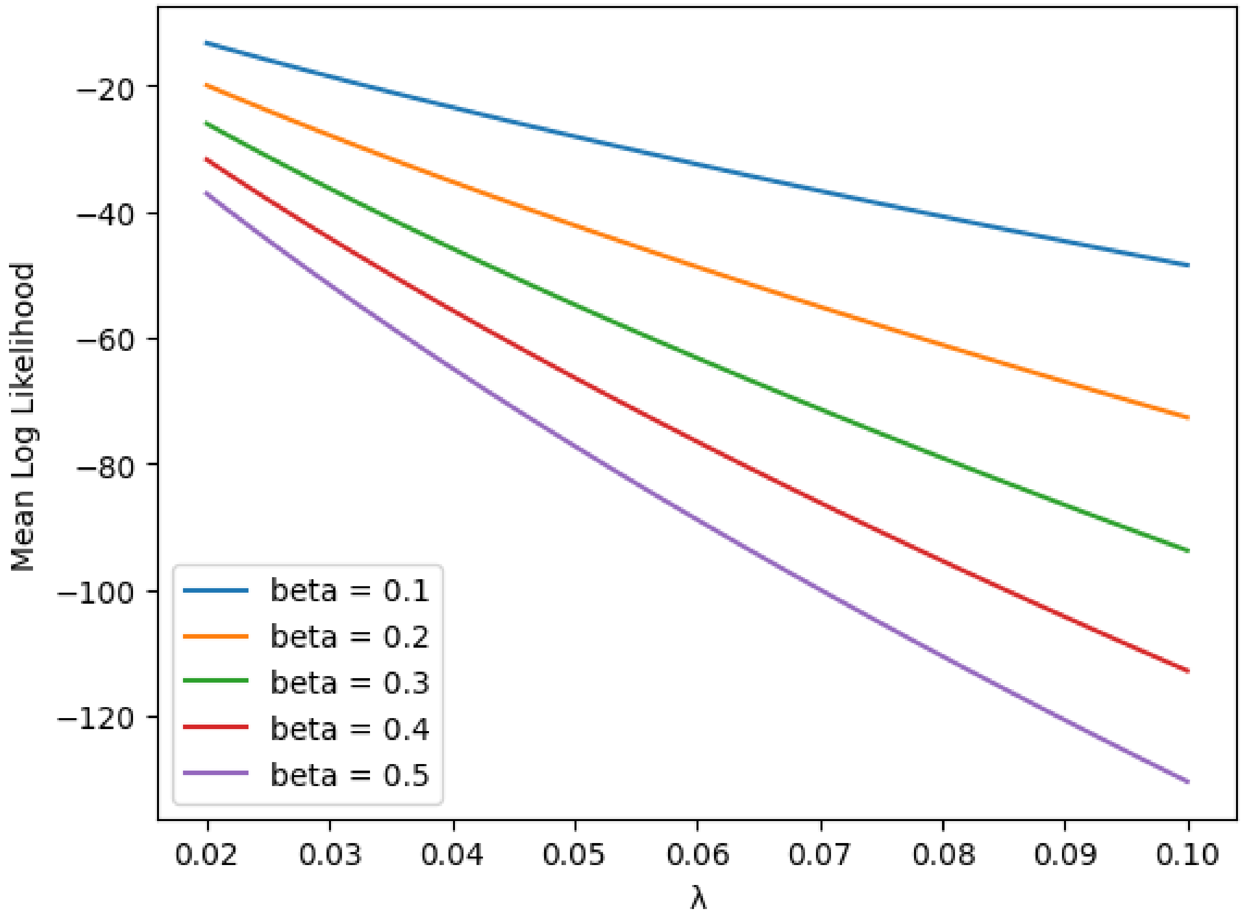
Hashing Function (Code)

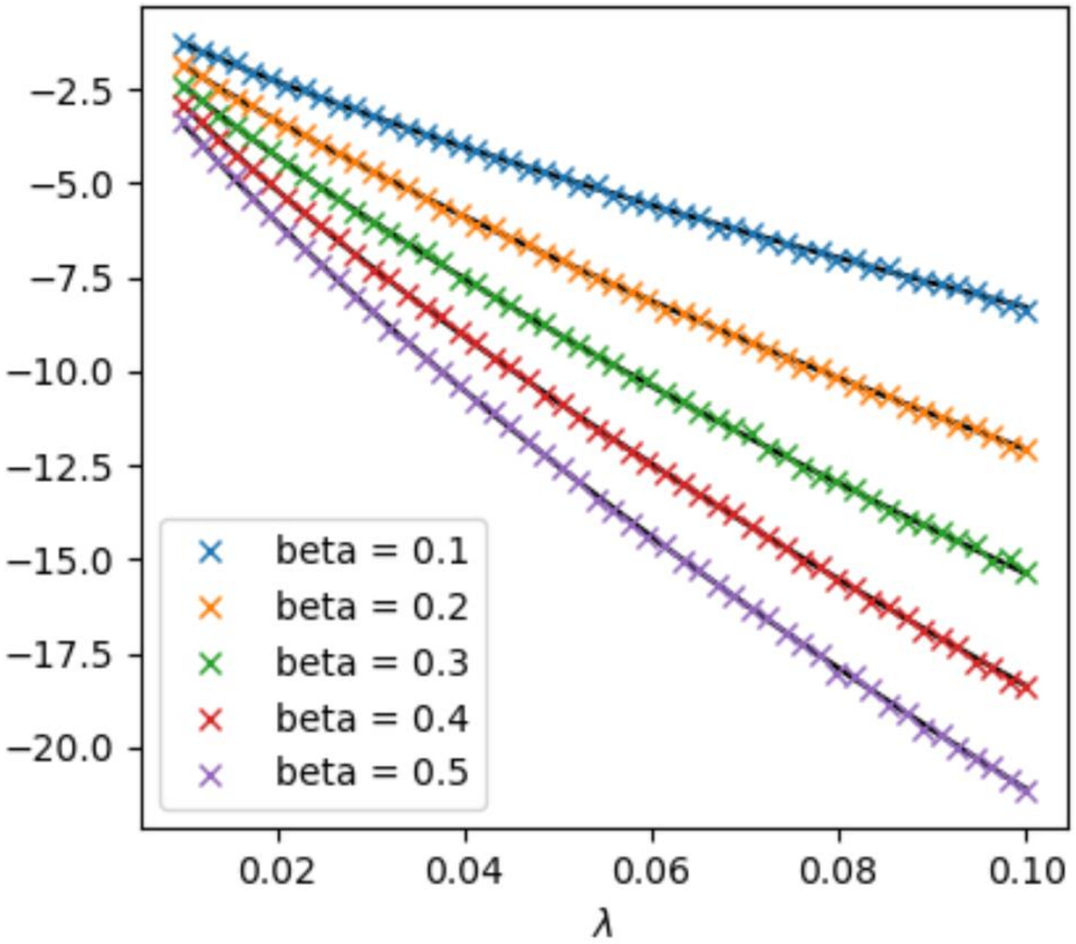
```python
### Precomputing powers of two for more efficient prime hashing
pow_2 = [1]
P_mod = (10 ** 9) + 7
for i in range(200000):
    pow_2.append((pow_2[-1] << 1) % P_mod)

### Hashing function for patterns ('img' is a multidimensional pattern,
### so passing the 2D representation or 1D flattened version would both work)
def hash_img(img):
    img_flattened = img.flatten()
    flat_len = img_flattened.shape[0]

    return np.dot(img_flattened, pow_2[:flat_len]) % P_mod
```

Task (5f) Trends





Task 2 Sorted Design Matrix