Fluids in Machine Learning

Exploring the intersection of machine learning and fluids dynamics by-

Rutam S Rajhansa, Ojasvi Pandey

1. Review Of Machine Learning in Fluids

1. Recent progress, in fluid dynamics (CFD) has highlighted the merging of machine learning (ML) methods to improve the efficiency and accuracy of simulations.

2. We explored the use of Machine Learning in various areas of Fluid Mechanics

.

3. We read about various techniques such as Physics Guided Neural Networks (PGNNs) and POD and the use of them (specially POD) in refining ROMs.

4.We explored the use of ML and deep neural networks in DNS and turbulence modeling.

Different of these studies show the possibility of ML in reducing computational needs but with high fidelity in the simulation. Based on our study and findings we saw that ML has a huge potential to be incorporated in CFD and other processes

2. New Ideas

Automatic Way of Increasing the Efficiency of Fluid Network Design.

1. Artificial intelligence (AI) could be employed to create more efficient process piping networks self-optimizing themselves, like water supply systems, heating systems, and complex piping in industrial facilities.
2. The idea is to train models based on pipeline architechtures and system efficiencies so that the models can recommend optimum configurations and sizes of pipes, pumps, and valves which have minimal energy consumption and minimal cost while having high flow efficiency.

.The implementation of AI/ML in this field can lead to smarter, more responsive infrastructure systems that meet the needs of modern populations and industries more effectively.

<u>2. New Ideas</u>

Leak Detection in Fluid Systems using Machine Learning

- We may develop a machine learning solution that will identify and localize leaks better than existing methods within homogeneous fluid systems (for example, oil pipelines).
- The system trains models using acoustic, vibration, and pressure sensors data that may pinpoint the subtle anomalies.
- Further those results can be used immediately for monitoring leaks based on the results

Benefits of ML in Leak Detection

- **Improved Accuracy**: Machine learning models can detect subtle signs of leaks that may not be discernible through traditional methods or manual inspection.
- **Early Detection**: The ability to monitor systems in real-time allows for earlier detection of leaks, significantly reducing potential environmental and resource losses.
- **Cost Efficiency:** Reducing the frequency and severity of leaks lowers the overall maintenance and operational costs.
- **Scalability**: Machine learning models can be scaled and adapted to different types of fluid systems and can handle growing data volumes.

2. New Ideas

Increasing Precipitation Accuracy

- Machine learning might just offer major improvement in the prediction of sediment transport in rivers and coastal areas by a considerable sum.

- By making use of actual time data obtained from satellite pictures, water flow gauges and weather reports, ML models are now able to forecast changes in sediment tendencies, which is decisive for navigation safety, flood prevention, and conservation of the environment.

- Machine learning models can process data as it is received, enabling real-time responses to potential sediment-related issues before they become critical.

These enhancements provided by machine learning not only improve safety and environmental protection but also optimize the management of water resources.

2. New Ideas

Flow Modelling With Machine-Learning

- Flow control involves manipulating the movement of fluid to improve efficiency and effectiveness in systems such as pipelines, air conditioning units, and aircraft.
- The goal is to optimize parameters such as speed, pressure, and turbulence to minimize energy use, reduce drag, or prevent undesired behaviors like flow separation.
- Machine learning algorithms can enhance flow information and automatically perform tasks that involve active flow control and optimization.
- Many Machine Learning techniques, such as Reinforcement Learning can be used to obtain the flow field parameters.

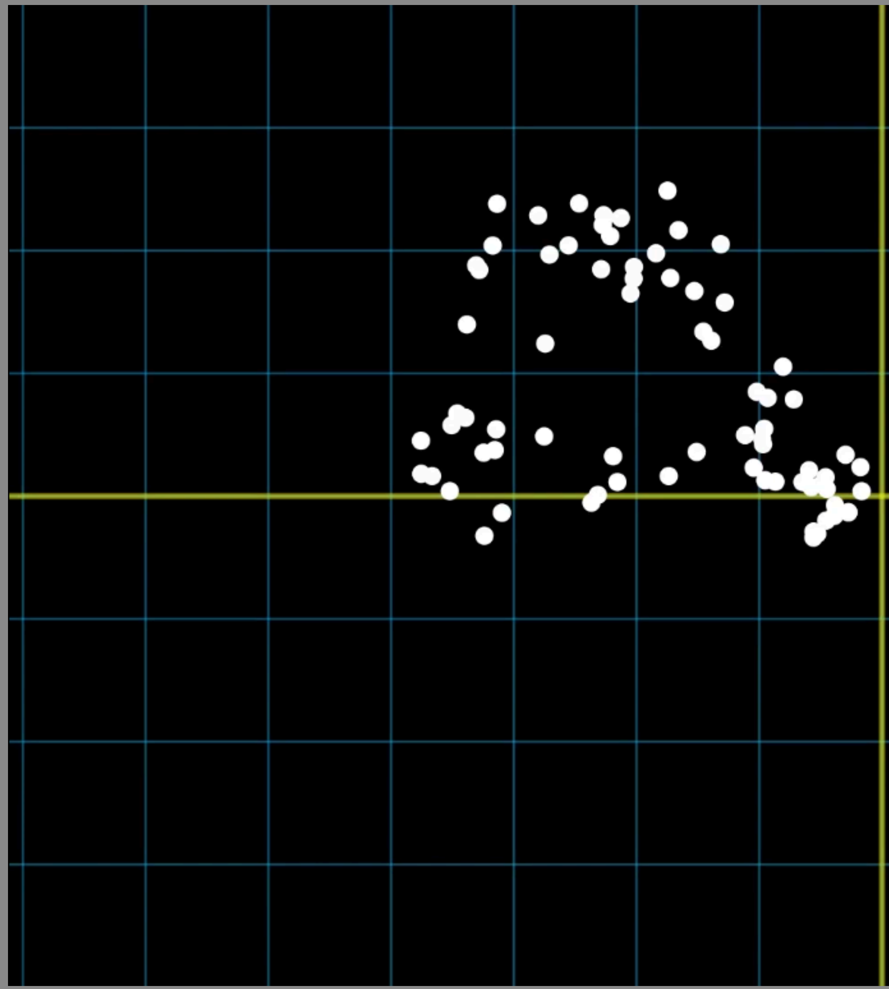
R**L algorithms learn from the environment by continuously interacting with it, receiving feedback in the form of rewards based on the efficacy of their actions. This method is incredibly beneficial for real-time flow control where conditions constantly change.**

3. Proper Orthogonal Decomposition

3.1 Image Generation

Steps:
1. We first mounted the drive with our Google Colab.
2. Further to perform POD we extracted the frames from the video provided and saved it to a folder in the drive named "output folder".

Total frames extracted from the video: 751
Problems faced: Due to the database being large, our system crashed several times.
Solution from our side: We downsampled the images with the **scale factor of 0.5**. Although we found that it was still creating problems when we were working on further tasks like applying POD on after adding noises. So we further grayscale the images.
Example of some of the images stored as frames: https://drive.google.com/file/d/1bNBK9uJilZNVE0G9NwC1jd8Y8wLC3U_1/view?usp=sharing



1. Maximize energy along each basis vector
2. Every next basis vector is orthogonal to all previous ones

$$A = U\Sigma V^T$$

Temporal Correlation Matrix

$$\boxed{AA^T} = U\Sigma \underbrace{V^T V}_{I} \Sigma U^T$$

$$AA^T = U\Sigma^2 U^T$$

$$AA^T U = U\Sigma^2 \underbrace{U^T U}_{I}$$

$$\underbrace{AA^T U = U\Sigma^2}_{\text{Eigenvalue Problem}}$$

Spatial Correlation Matrix

$$\boxed{A^T A} = V\Sigma \underbrace{U^T U}_{I} \Sigma V^T$$

$$A^T A = V\Sigma^2 V^T$$

$$A^T A V = V\Sigma^2 \underbrace{V^T V}_{I}$$

$$\underbrace{A^T A V = V\Sigma^2}_{\text{Eigenvalue Problem}}$$

3.2 Execute POD

For this, we followed the following steps:

1. Converting the images obtained to stack.
2. Pre-processed the image stack obtained (making it ready for POD).
3. Applying POD using SVD:

The mathematical formulation used in the compute pod function is based on the Singular Value Decomposition (SVD) method. SVD is a matrix factorization technique that decomposes a matrix into three other matrices, namely U, Σ (S), and V^T, where U and V are orthogonal matrices, and Σ is a diagonal matrix containing the singular values.

Reference used to study: https://www.youtube.com/playlist?list=PLnO2sW0-XPrd_D6lV5YBCaUTa-NsMUibJ

Reference used to study with an amazing visualization : https://www.youtube.com/watch?v=axfUYYNd-4Y

The mathematical formulation used in the compute pod function is based on the Singular Value Decomposition (SVD) method. SVD is a matrix factorization technique that decomposes a matrix into three other matrices, namely U, Σ (S), and VT

, where U and V are orthogonal matrices, and Σ is a diagonal matrix containing the singular values.

Here’s the mathematical formulation: Given a set of images represented as
a stack X where each column vector x i represents a flattened image, the SVD
decomposition is computed as follows:
X = UΣVT

Where:

- U is the left singular matrix, containing the left singular vectors. It represents the principal components or the modes of variability in the dataset.

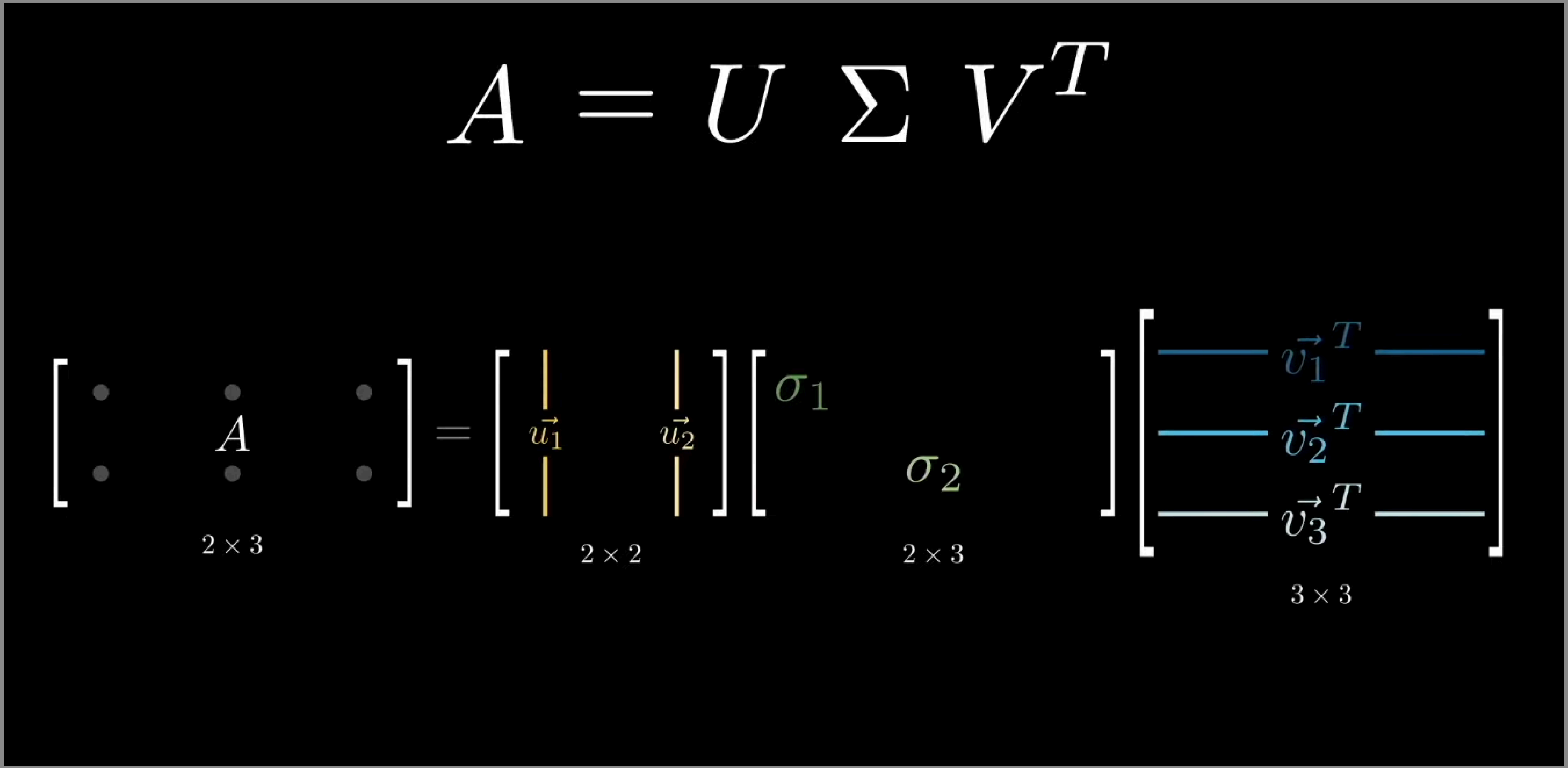
- Σ is the diagonal matrix of singular values, representing the amount of variance captured by each mode.

- VT is the transpose of the right singular matrix, containing the right singular vectors.

So the function defined compute pod takes a stack of images as input and returns the left singular matrix U, the singular values Σ, and the mean image
.
Like this we got all 751 images. The frames were stored in the output folder

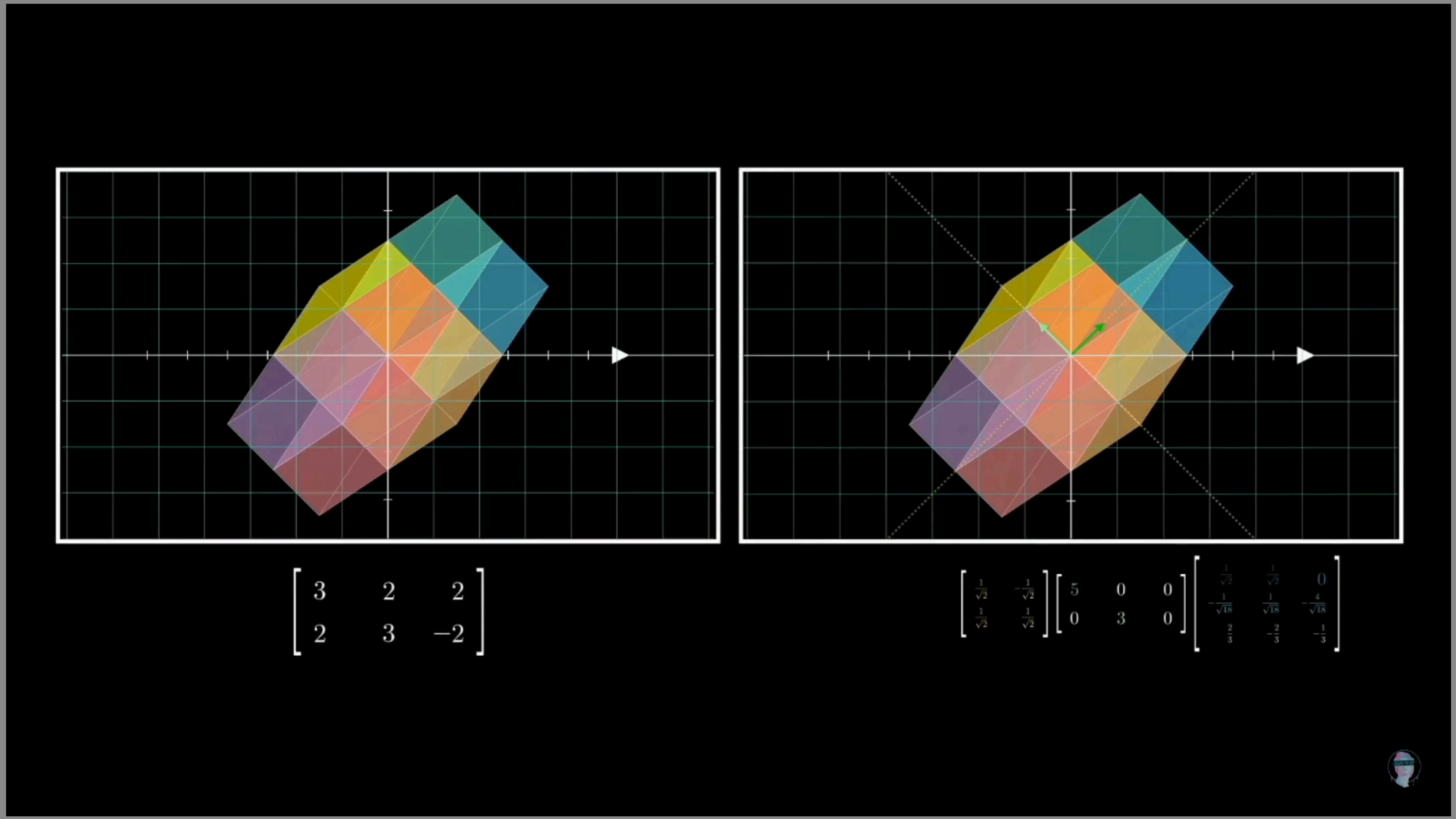
For SVD Visualization:

https://www.youtube.com/watch?v=vSczTbgc8Rc



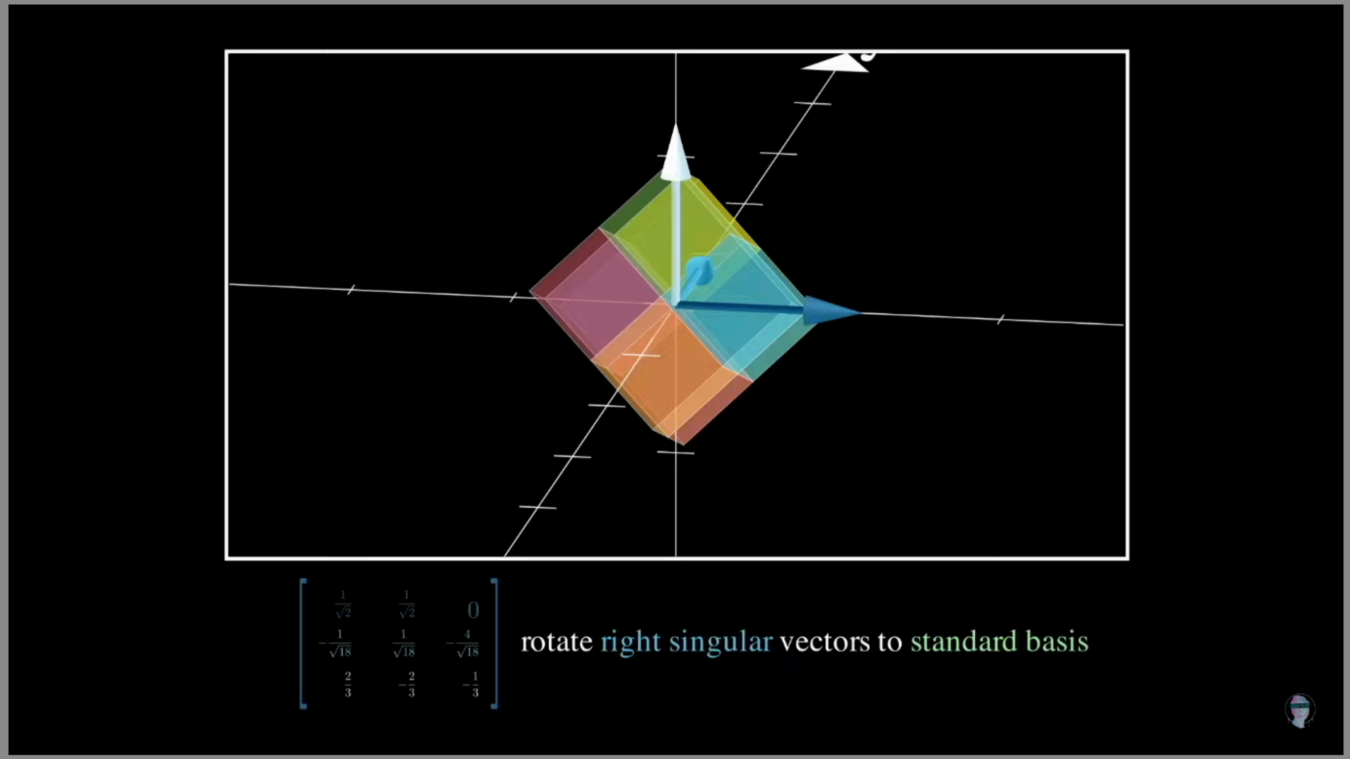
$$A = U \; \Sigma \; V^T$$

$$\underset{2\times 3}{\begin{bmatrix} \bullet & \bullet & \bullet \\ & A & \\ \bullet & \bullet & \bullet \end{bmatrix}} = \underset{2\times 2}{\begin{bmatrix} | & | \\ \vec{u_1} & \vec{u_2} \\ | & | \end{bmatrix}} \underset{2\times 3}{\begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \end{bmatrix}} \underset{3\times 3}{\begin{bmatrix} \text{---} & \vec{v_1}^T & \text{---} \\ \text{---} & \vec{v_2}^T & \text{---} \\ \text{---} & \vec{v_3}^T & \text{---} \end{bmatrix}}$$

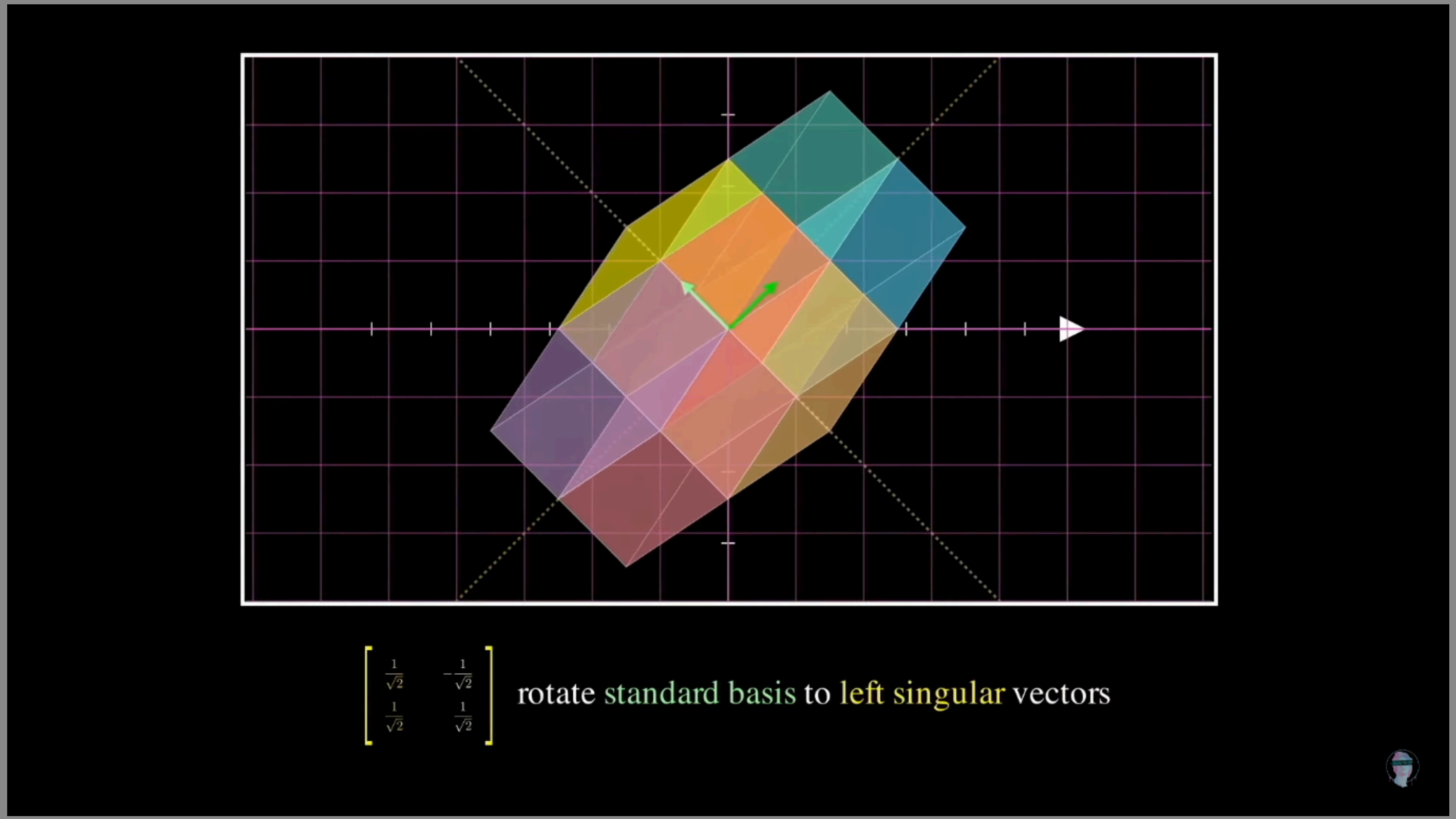


$$\begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix}$$

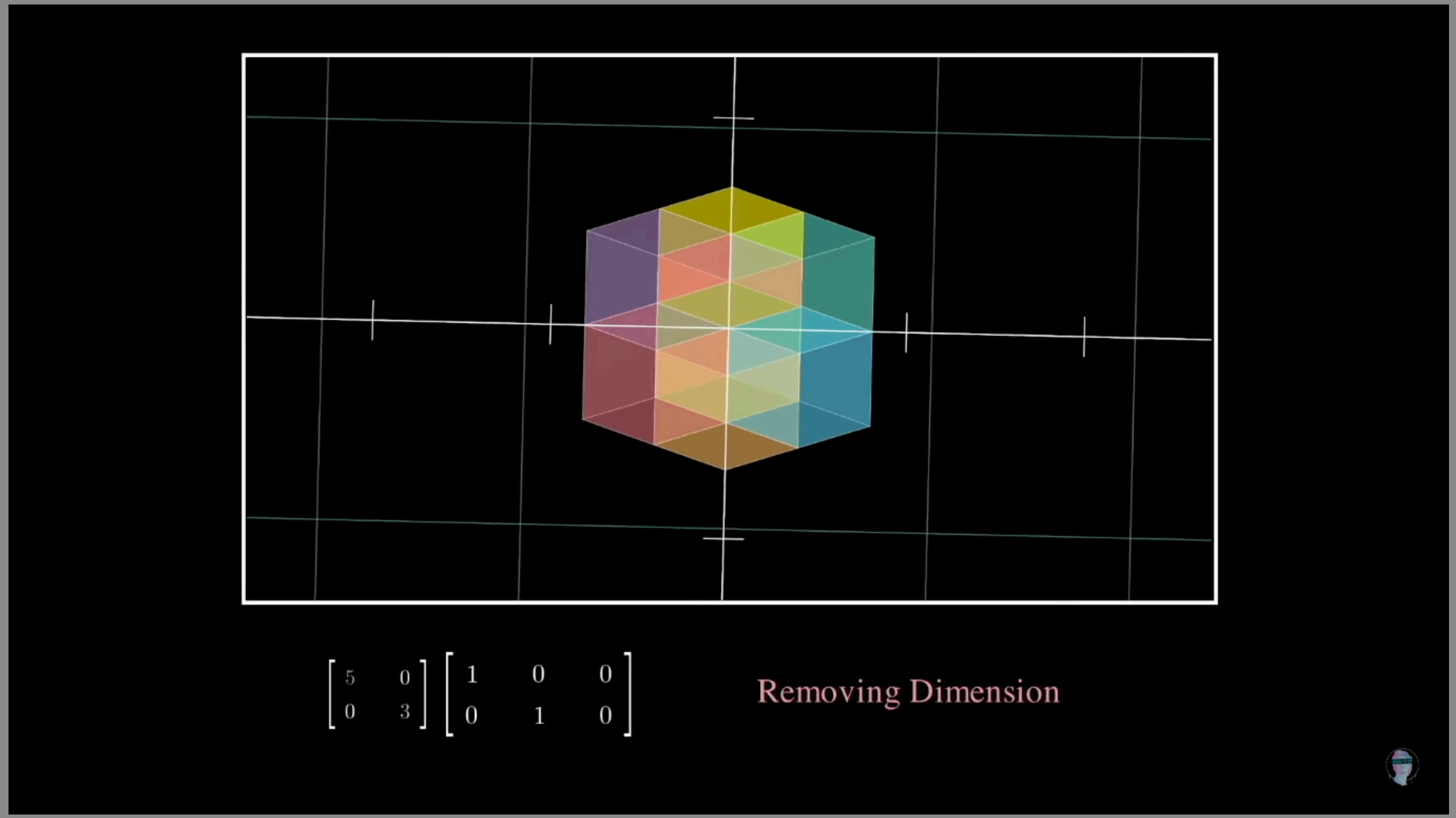
$$\begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}\begin{bmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{bmatrix}\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{18}} & \frac{1}{\sqrt{18}} & -\frac{4}{\sqrt{18}} \\ \frac{2}{3} & -\frac{2}{3} & -\frac{1}{3} \end{bmatrix}$$



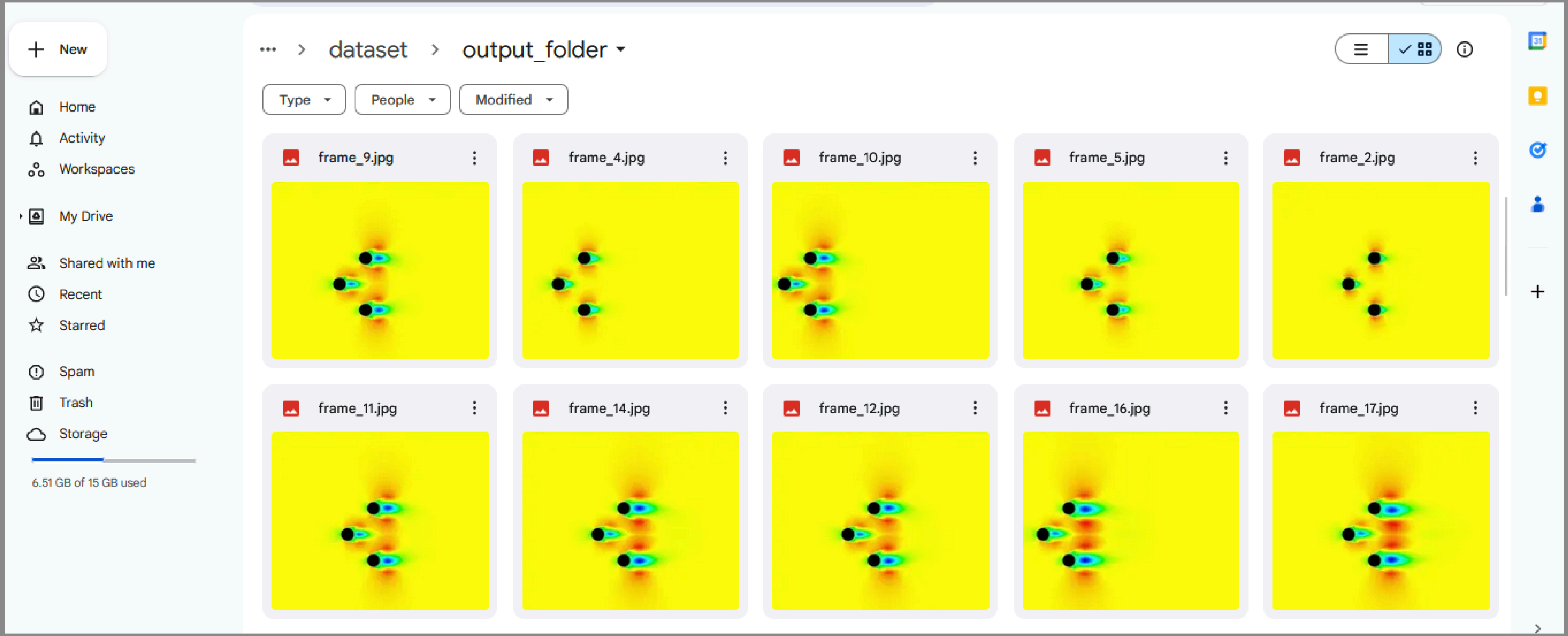
$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{18}} & \frac{1}{\sqrt{18}} & -\frac{4}{\sqrt{18}} \\ \frac{2}{3} & -\frac{2}{3} & -\frac{1}{3} \end{bmatrix}$$ rotate right singular vectors to standard basis

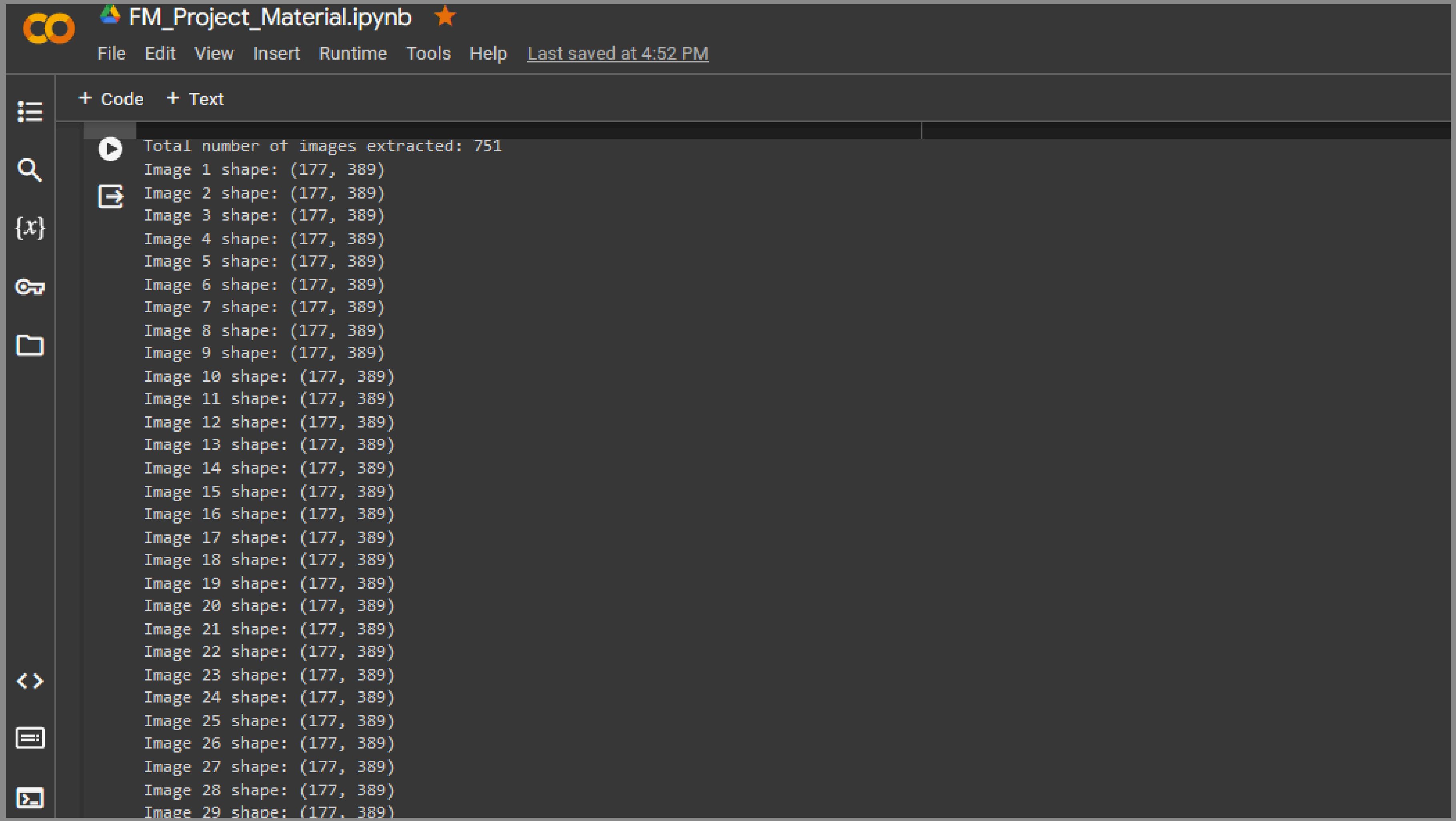


$$\begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$ rotate standard basis to left singular vectors



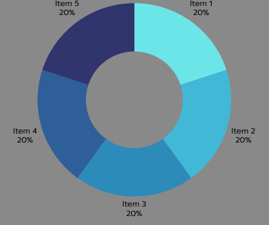
$$\begin{bmatrix} 5 & 0 \\ 0 & 3 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$ Removing Dimension





```
Total number of images extracted: 751
Image 1 shape: (177, 389)
Image 2 shape: (177, 389)
Image 3 shape: (177, 389)
Image 4 shape: (177, 389)
Image 5 shape: (177, 389)
Image 6 shape: (177, 389)
Image 7 shape: (177, 389)
Image 8 shape: (177, 389)
Image 9 shape: (177, 389)
Image 10 shape: (177, 389)
Image 11 shape: (177, 389)
Image 12 shape: (177, 389)
Image 13 shape: (177, 389)
Image 14 shape: (177, 389)
Image 15 shape: (177, 389)
Image 16 shape: (177, 389)
Image 17 shape: (177, 389)
Image 18 shape: (177, 389)
Image 19 shape: (177, 389)
Image 20 shape: (177, 389)
Image 21 shape: (177, 389)
Image 22 shape: (177, 389)
Image 23 shape: (177, 389)
Image 24 shape: (177, 389)
Image 25 shape: (177, 389)
Image 26 shape: (177, 389)
Image 27 shape: (177, 389)
Image 28 shape: (177, 389)
Image 29 shape: (177, 389)
```

3.3 Analyze POD Modes:



- For this, we followed the following steps: This step involves analyzing the modes extracted via POD, where each mode represents a significant pattern or feature in the data. The visualization above shows one such mode extracted from our dataset.

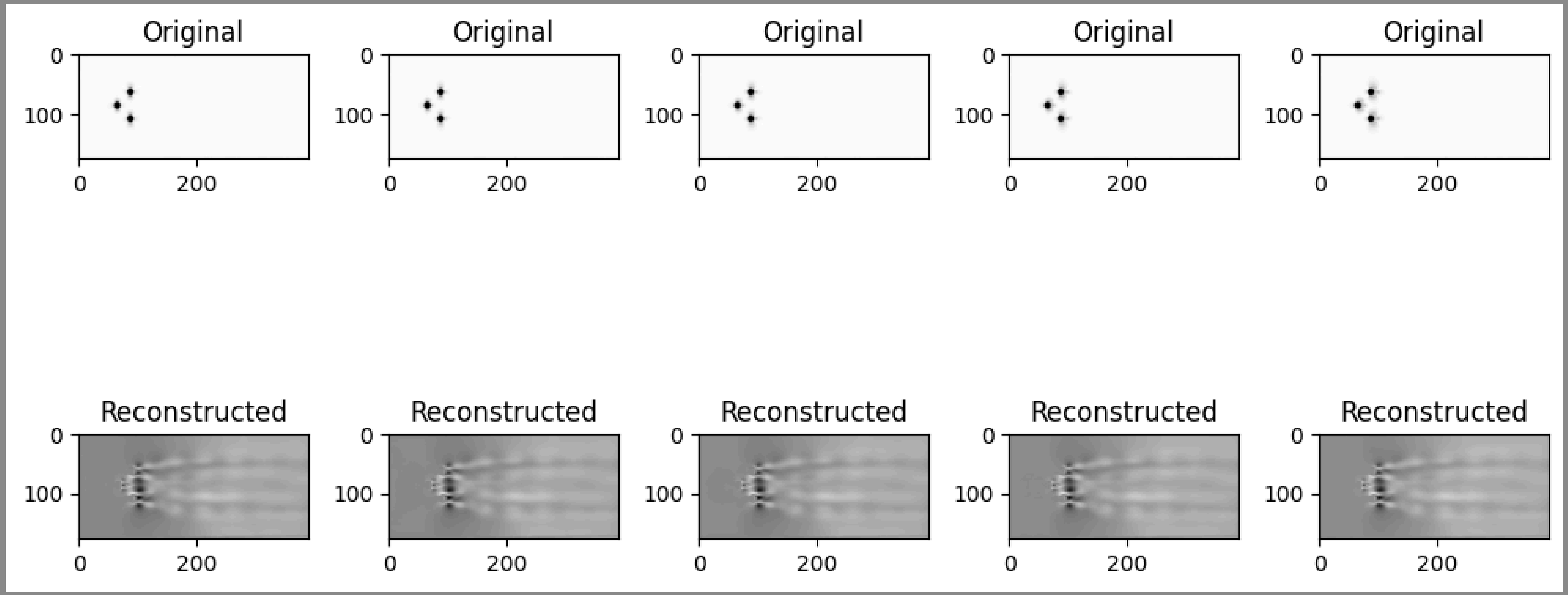
After plotting the graph for top 10 modes corresponding to the highest energy states we got the following trend

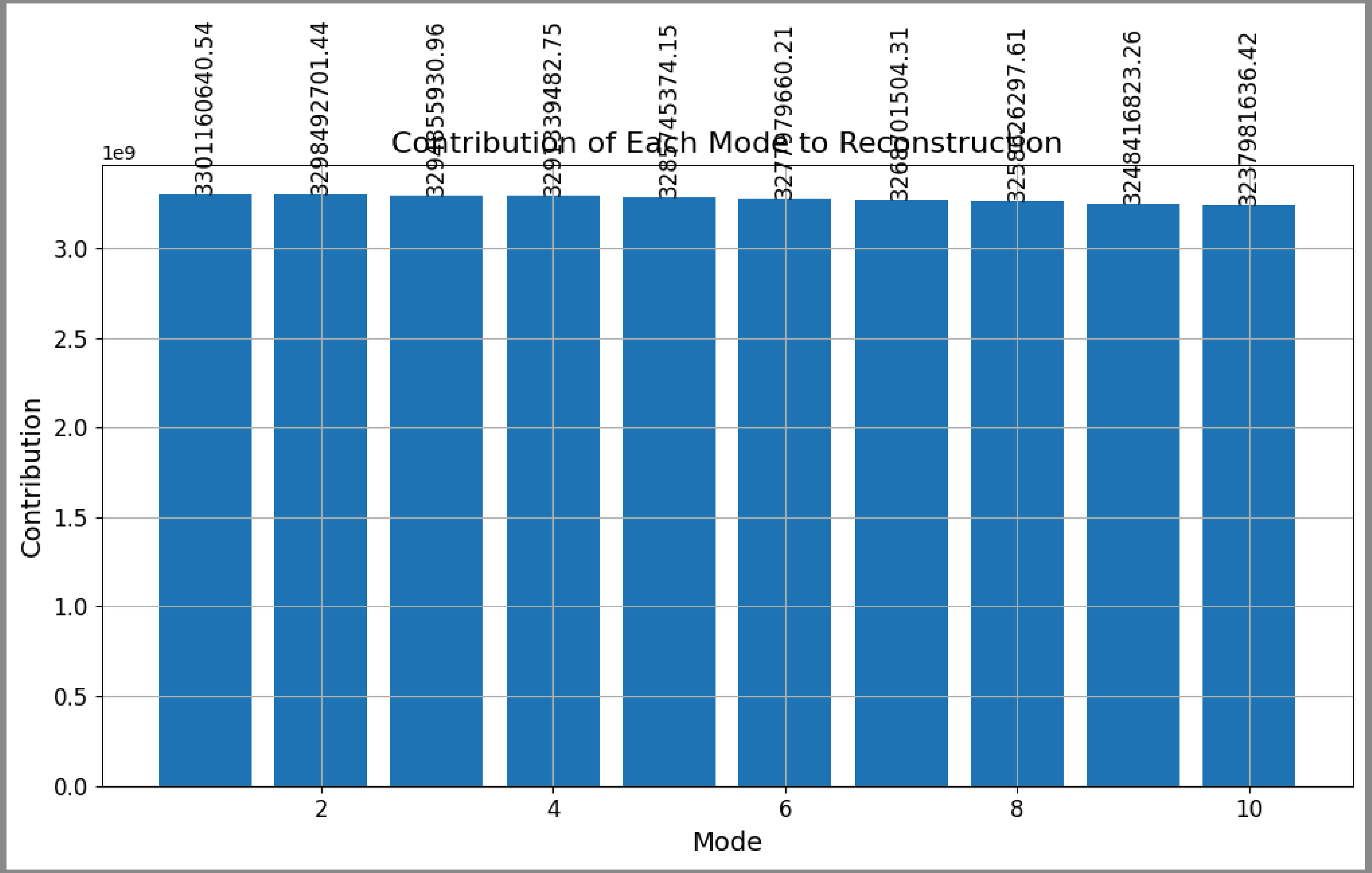
3.4 Significance of Mode:

After studying some research papers like:
1. ScienceDirect Article https://www.sciencedirect.com/science/article/pii/S0889974609001352
2. Springer Link Article https://link.springer.com/article/10.1007/s00162-013-0293-2

we concluded:
These modes are of high significance as:
1. The top modes capture the dominant spatial structures or patterns in the data.
2. These modes represent the most significant modes of variation in the dyamical system.
3. They provide a compact representation of the data and can be used for various purposes such as:

- Dimensionality reduction
- Reconstruction of original data
- Analysis of flow features, turbulence, or other physical phenomena.





4 Noise!

4.1 Adding Noise

Here, we added two types of noises, each with magnitudes of 20%, 40%, 60%, and 80% of the maximum magnitude of the individual frames.

The two types of noises added are:
1. Gaussian Noise
Reference: https://wiki.cloudfactory.com/docs/mp-wiki/augmentations/gaussian-noise
2. Speckle Noise
Reference: https://cames.ippt.pan.pl/index.php/cames/article/view/290

4.2.1 How does the energy transcend through modes?

Within the framework of Proper Orthogonal Decomposition (POD), energy is commonly denoted by the singular values derived from Singular Value Decom position (SVD). Elevated singular values signify modes that encapsulate greater energy or variance within the dataset. Typically, the energy diminishes sequen tially as we transition from the primary mode (housing the highest energy) to subsequent modes.

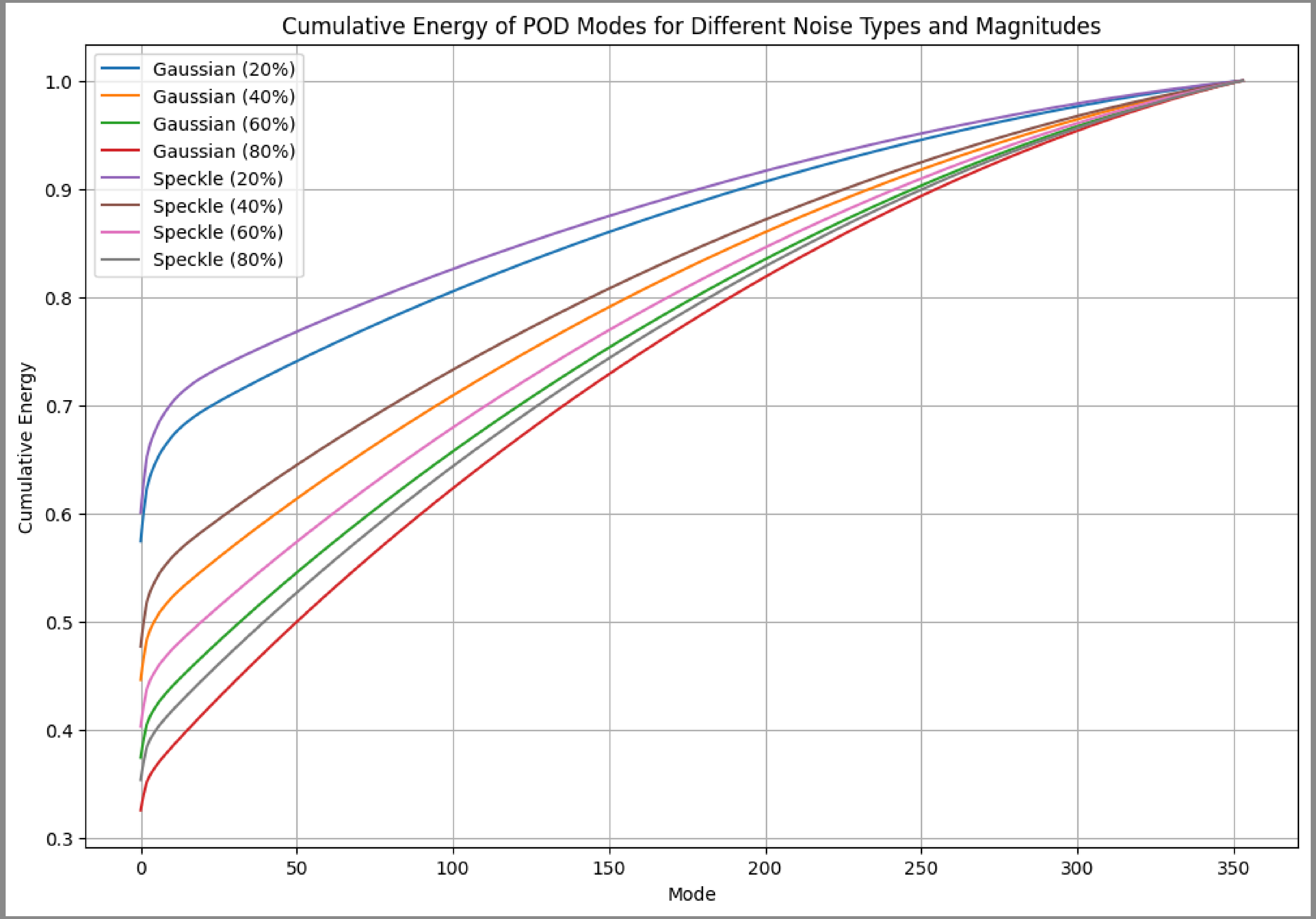
4.2.2 Can you give any statistical evidence of the energy changes in the modes?

For this, we plotted a graph showing statistical properties of Cumulative Energy across Different Noise Types. Additionally, plotting the cumulative sum of singular values helps visualize the cumulative energy captured by each mode.

4.2.3 Do different noise types have different types of response in terms of how the energy of the modes changes ?

From the above graph, we made the following observations:

- Different types of noises have different responses. For the same magnitude, Gaussian noise showed more effect on the data compared to Speckle.



4.2.4 How does the magnitude of noise affect the modal energy?

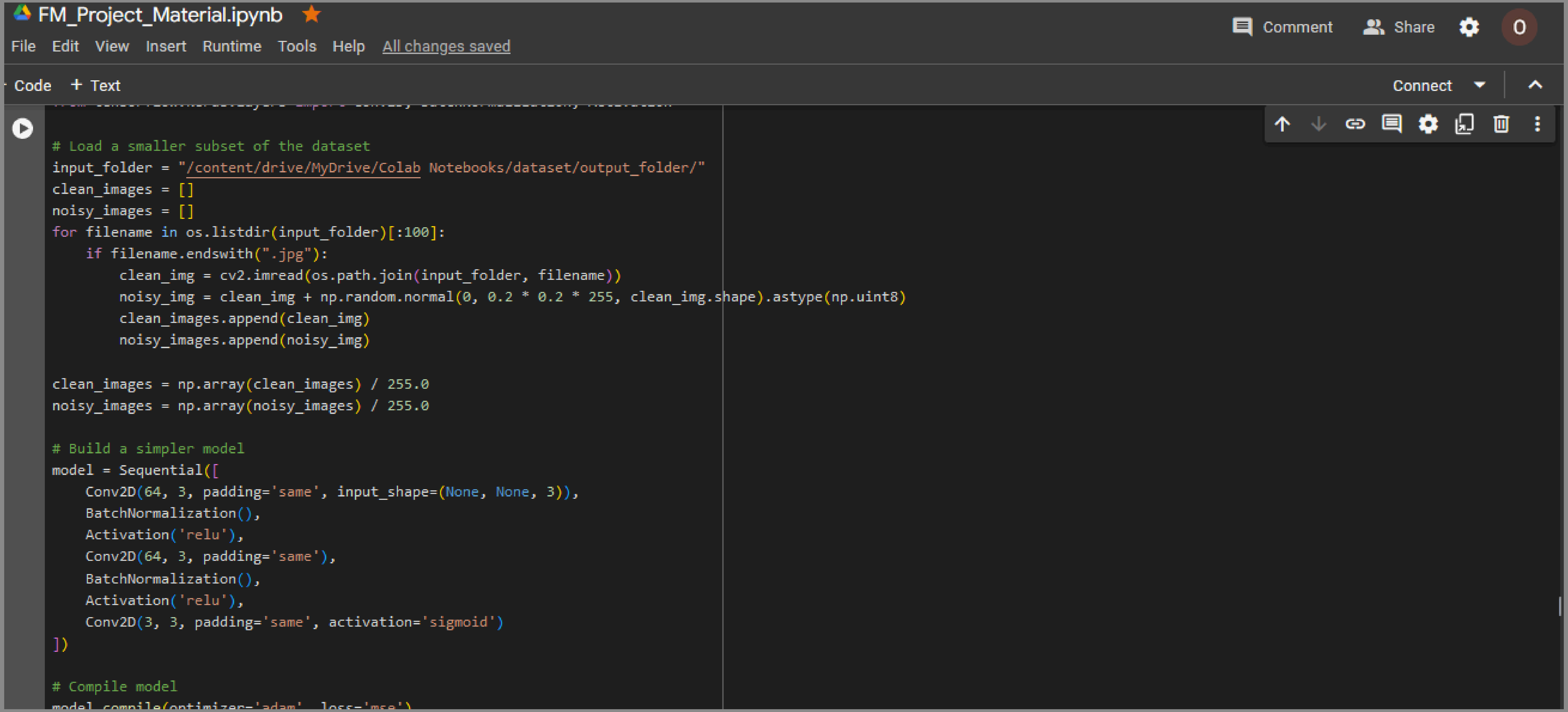
Elevated levels of noise typically result in more pronounced disturbances in the
energy distribution across modes. Stronger noise intensities have the potential
to enhance the variability captured by modes with lower energy levels, thereby
complicating the extraction of meaningful patterns from the dataset.

4.2.5 What sort of noise is best suited to be used for POD for arti ficial analysis? Provide your comprehensive analysis

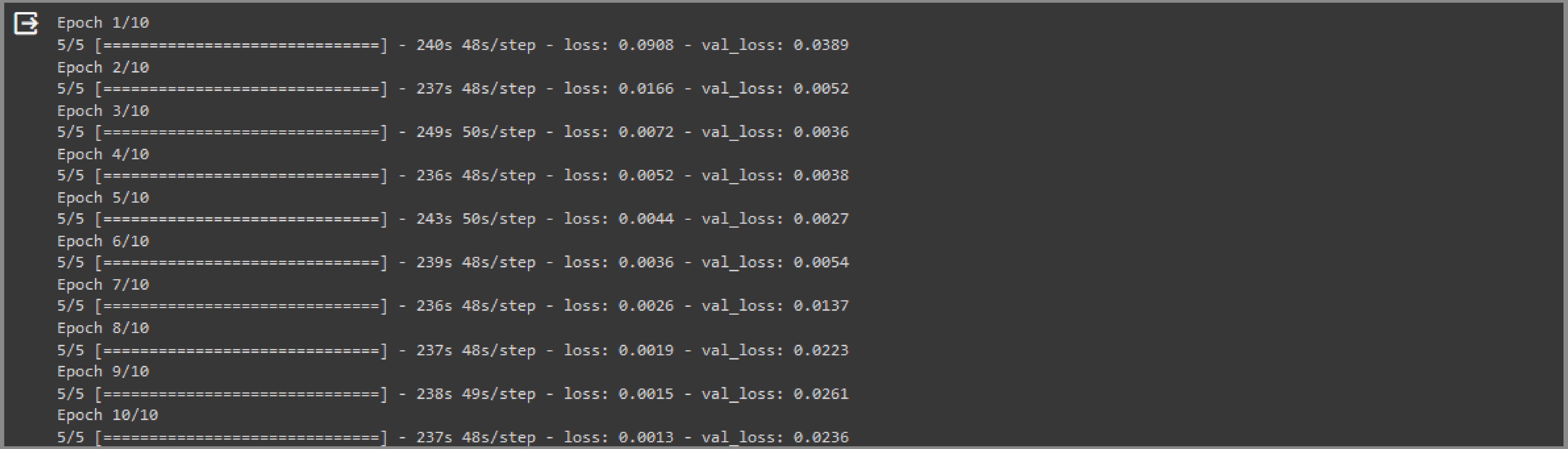
The whole point of adding the noise is to check the robustness of the POD
analysis and make as much real noise as possible to simulate real-world noise
present in the data. From the graphs obtained, we can conclude that Gaussian
noise is best suited to be used for POD for artificial analysis.
• Reasons:
– **Robustness:** Gaussian noise models many natural processes and mea surement errors, making it a robust choice for simulating real-world
conditions.
– **Simplicity:** Gaussian noise is easy to generate and add to datasets.
Its simplicity makes it a convenient choice for conducting experiments
and analyzing algorithms.
– **Statistical Properties:** Gaussian noise is well-studied and its statis tical properties are well-understood. This makes it easier to analyze
and model mathematically.

5. Super-Resolving

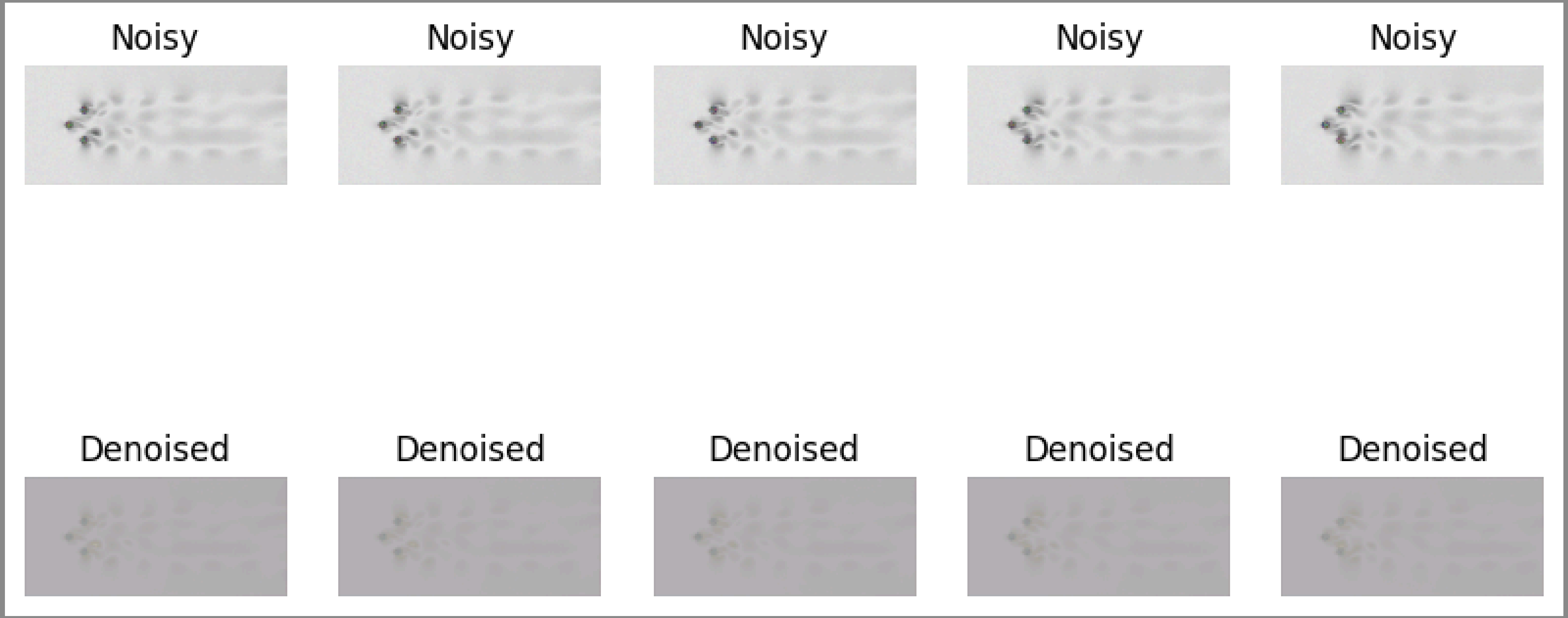
Using the ML model provided in code, we attempted to remove noise.


FM_Project_Material.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Code + Text
Connect

```python
# Load a smaller subset of the dataset
input_folder = "/content/drive/MyDrive/Colab Notebooks/dataset/output_folder/"
clean_images = []
noisy_images = []
for filename in os.listdir(input_folder)[:100]:
    if filename.endswith(".jpg"):
        clean_img = cv2.imread(os.path.join(input_folder, filename))
        noisy_img = clean_img + np.random.normal(0, 0.2 * 0.2 * 255, clean_img.shape).astype(np.uint8)
        clean_images.append(clean_img)
        noisy_images.append(noisy_img)

clean_images = np.array(clean_images) / 255.0
noisy_images = np.array(noisy_images) / 255.0

# Build a simpler model
model = Sequential([
    Conv2D(64, 3, padding='same', input_shape=(None, None, 3)),
    BatchNormalization(),
    Activation('relu'),
    Conv2D(64, 3, padding='same'),
    BatchNormalization(),
    Activation('relu'),
    Conv2D(3, 3, padding='same', activation='sigmoid')
])

# Compile model
model.compile(optimizer='adam', loss='mse')
```



```
Epoch 1/10
5/5 [==============================] - 240s 48s/step - loss: 0.0908 - val_loss: 0.0389
Epoch 2/10
5/5 [==============================] - 237s 48s/step - loss: 0.0166 - val_loss: 0.0052
Epoch 3/10
5/5 [==============================] - 249s 50s/step - loss: 0.0072 - val_loss: 0.0036
Epoch 4/10
5/5 [==============================] - 236s 48s/step - loss: 0.0052 - val_loss: 0.0038
Epoch 5/10
5/5 [==============================] - 243s 50s/step - loss: 0.0044 - val_loss: 0.0027
Epoch 6/10
5/5 [==============================] - 239s 48s/step - loss: 0.0036 - val_loss: 0.0054
Epoch 7/10
5/5 [==============================] - 236s 48s/step - loss: 0.0026 - val_loss: 0.0137
Epoch 8/10
5/5 [==============================] - 237s 48s/step - loss: 0.0019 - val_loss: 0.0223
Epoch 9/10
5/5 [==============================] - 238s 49s/step - loss: 0.0015 - val_loss: 0.0261
Epoch 10/10
5/5 [==============================] - 237s 48s/step - loss: 0.0013 - val_loss: 0.0236
```



THANK YOU!