(The whole code for the project has been done on Google Colab)

3. Proper Orthogonal Decomposition (30 marks)

On the basis of the flow parameters provided to us

Table 1: Flow Parameters

Parameters	Value / Description
Fluid	Air
Fluid Viscosity	$1.7894 \times 10^{-5}\ Kg\ m^{-1}s^{-1}$
Fluid Density	$1.225\ Kg\ m^{-3}$
Inlet Velocity	$0.0025\ ms^{-1}$
Inlet Gauge Pressure	$0\ P\ a$

3.1 Image Generation (3 marks)

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 the images stored as frames:

https://drive.google.com/file/d/1bNBK9uJilZNVE0G9NwC1jd8Y8wLC3U_1/view?usp=sharing

3.2 Execute POD (12 marks)

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

Here's the mathematical formulation:

Given a set of images represented as a stack X where each column vector xi represents a flattened image, the SVD decomposition is computed as follows:

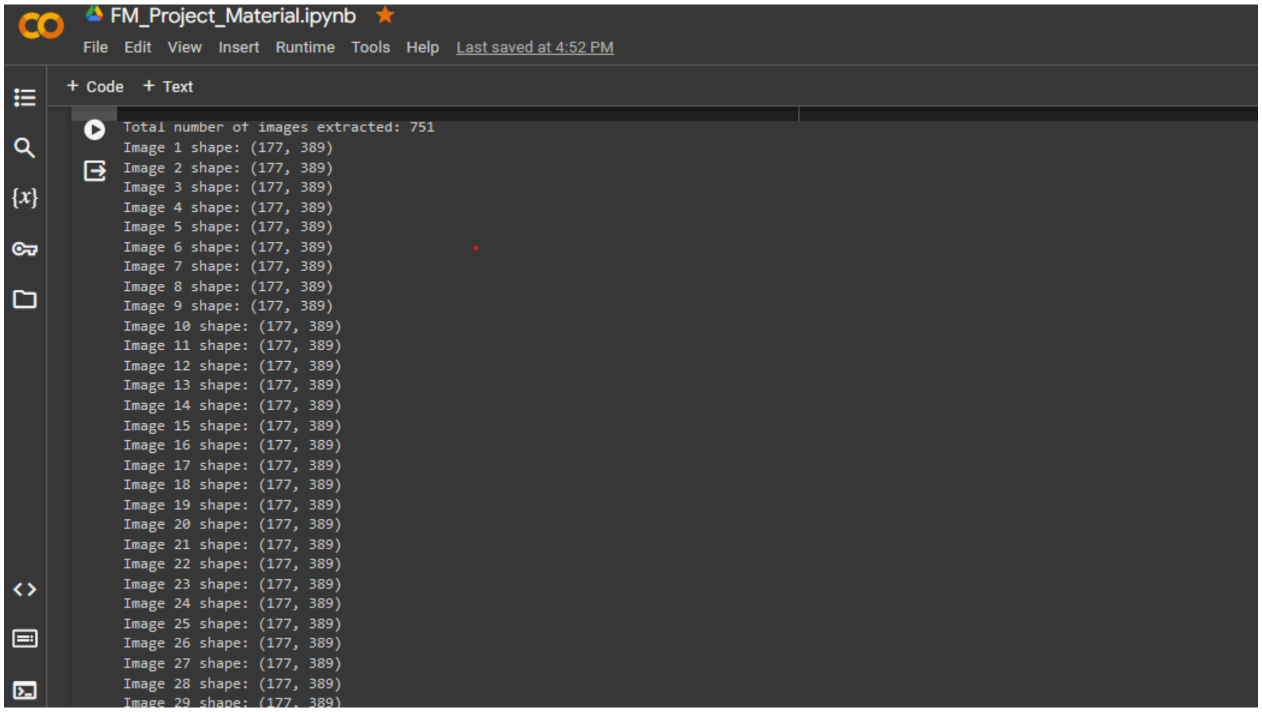
$$X=U\Sigma V^T$$

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.
- V^T 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.

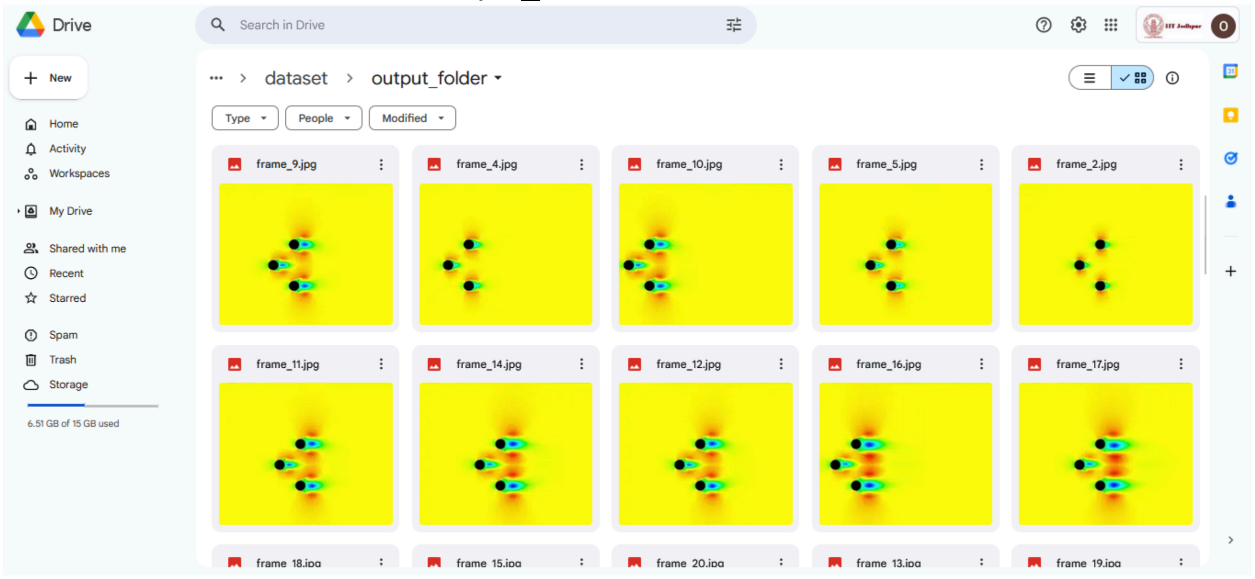
So on using the function we got:


FM_Project_Material.ipynb
File Edit View Insert Runtime Tools Help Last saved at 4:52 PM
+ Code + Text

```
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)
```

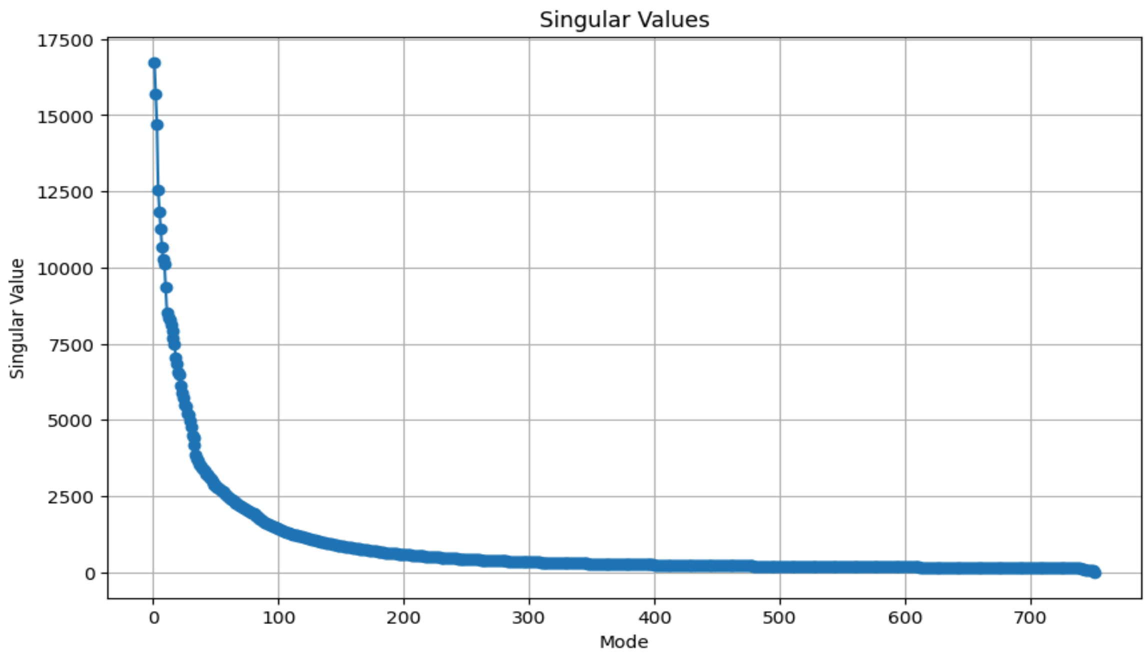
Like this we got all 751 images.

The frames were stored in the output_folder:

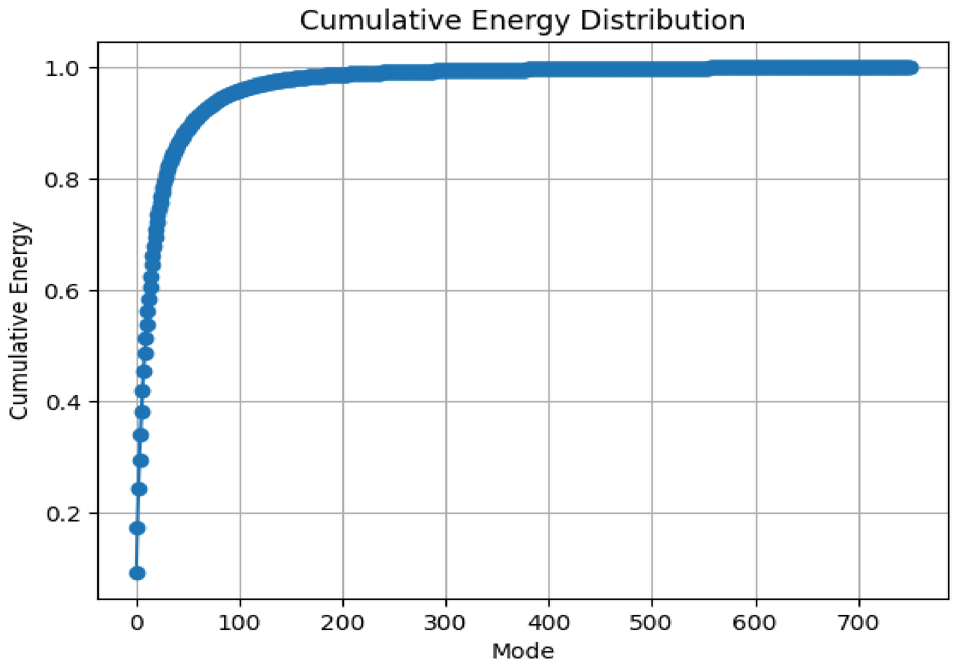


3.3 Analyze POD Modes (15 marks)

After applying the POD we got the following plot:



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



Significance of Mode:

After studying some research papers like:
1. https://www.sciencedirect.com/science/article/pii/S0889974609001352

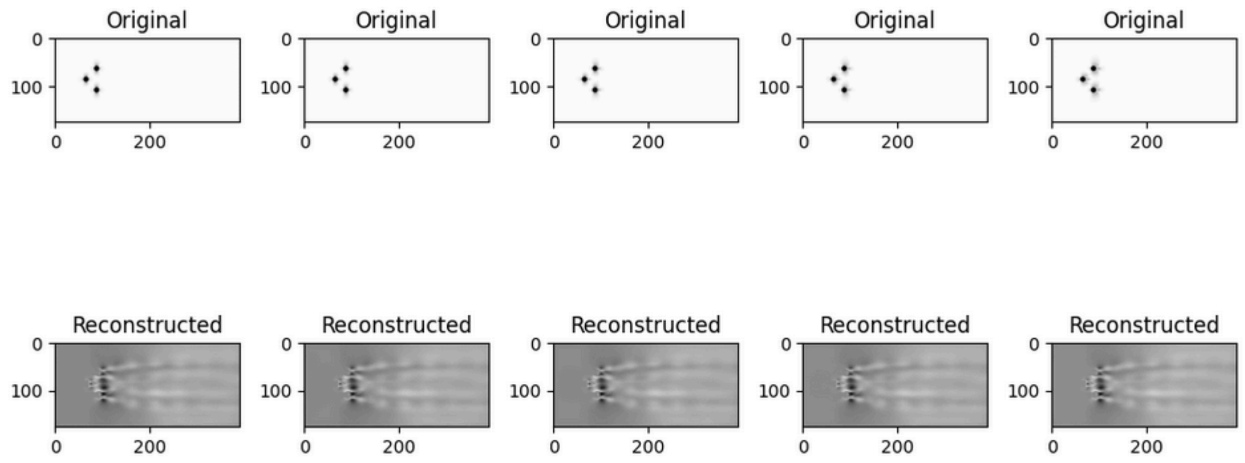
2. https://link.springer.com/article/10.1007/s00162-013-0293-2

We concluded:

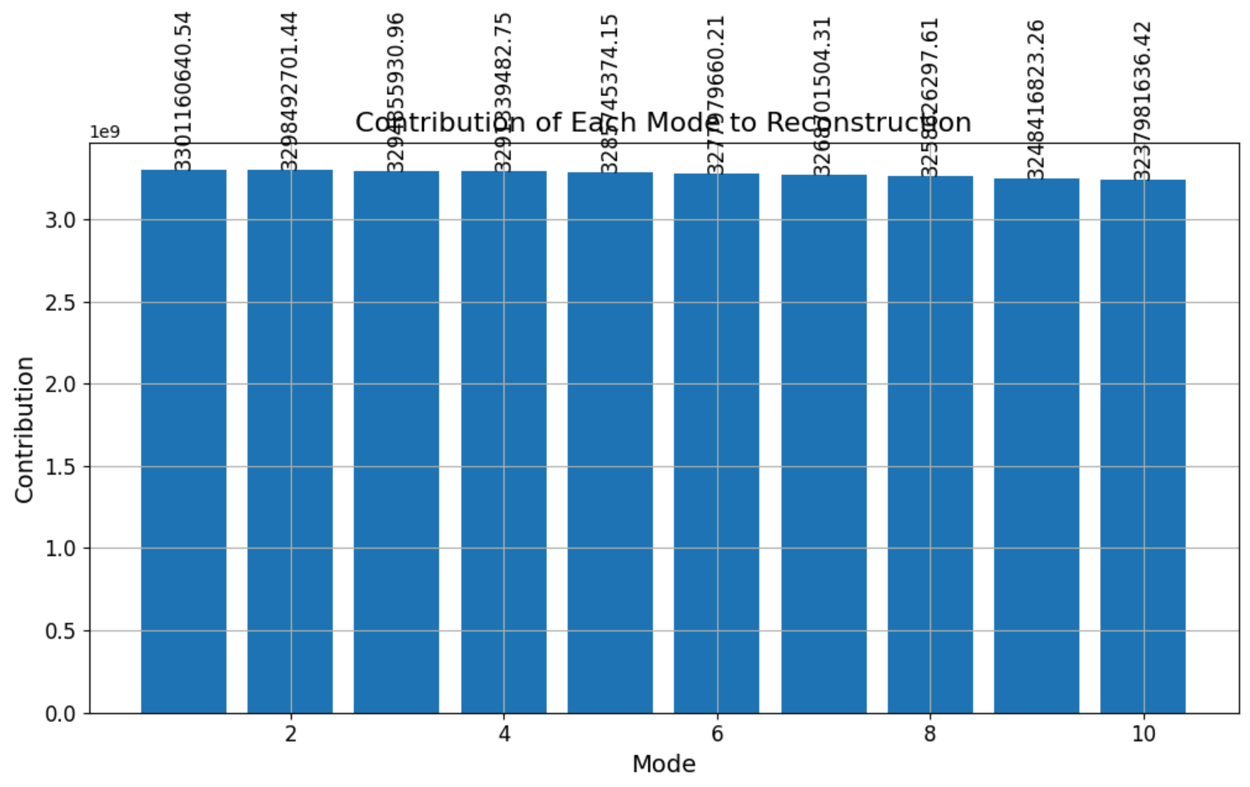
These modes are of high significance as:

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 dynamical 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.

We have tried to depict the significance by reconstructing the original data using these top 10 modes:



We have also plotted a graph showing their contribution :



4. Noise! (25 marks)

4.1 Adding Noise (10 marks)

Here for this we added 2 types of noises each of magnitude = 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

Difference between Gaussian Noise and Speckle Noise:

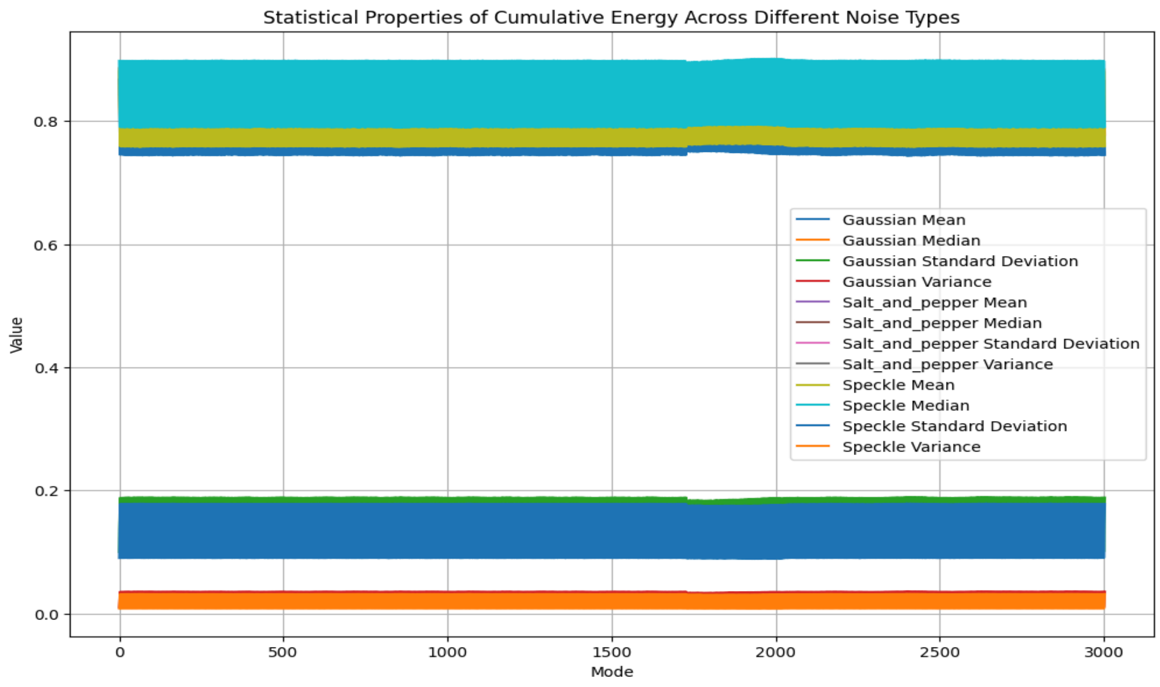
https://medium.com/@sunil7545/speckle-vs-gaussian-noise-7f4f47230d82

4.2 Effect on POD Modes (15 marks)

• How does the energy transcend through modes?

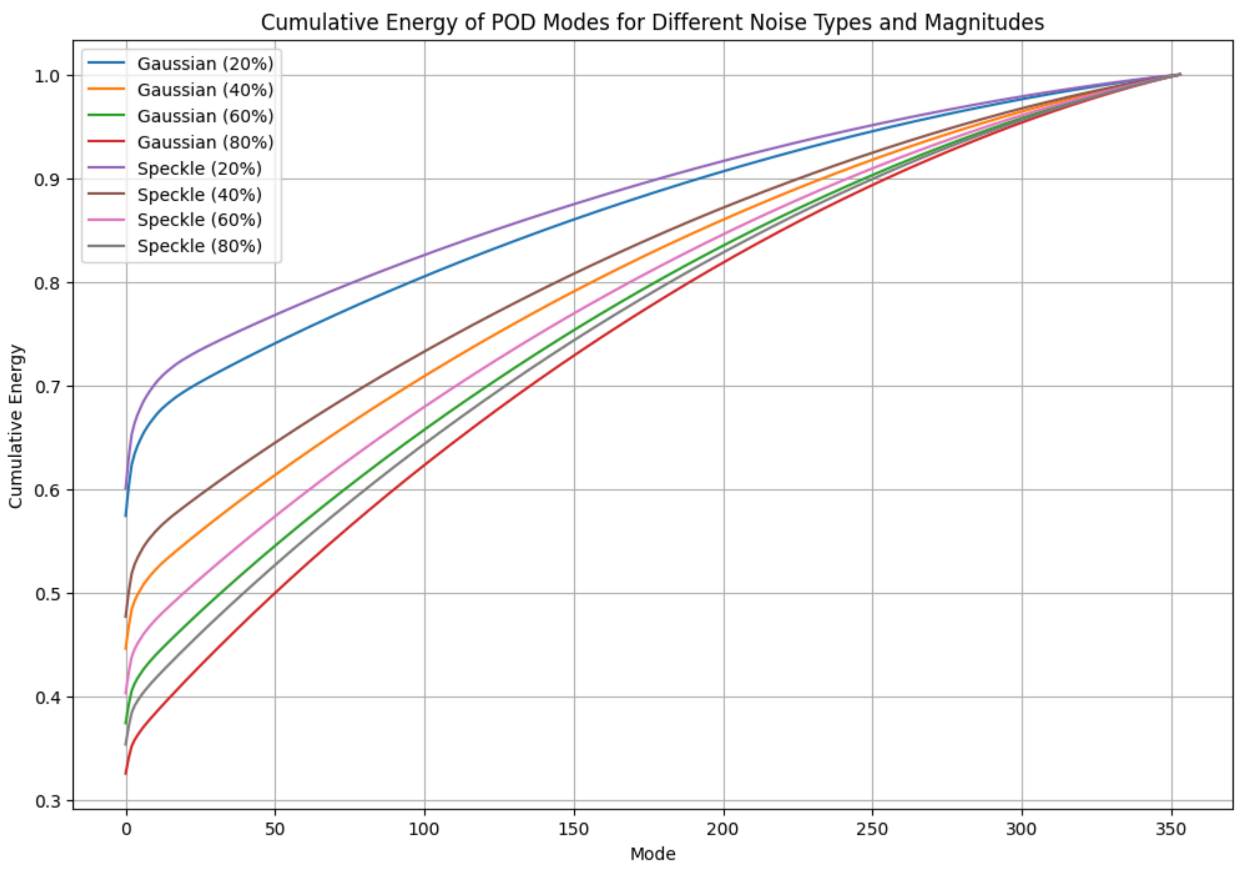
Within the framework of Proper Orthogonal Decomposition (POD), energy is commonly denoted by the singular values derived from Singular Value Decomposition (SVD). Elevated singular values signify modes that encapsulate greater energy or variance within the dataset. Typically, the energy diminishes sequentially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

• Can you give any statistical evidence of the energy changes in the modes? So for this we plotted a graph showing statistical properties of Cumulative Energy energy across Different Noise Types.



Additionally, plotting the cumulative sum of singular values helps visualize

the cumulative energy captured by each mode.



• 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 following observations :

❖ Different types of Noises have different response as here for the same magnitude **Gaussian showed more effect on the data as compared to Speckle.**

• 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.

• What sort of noise is best suited to be used for POD for artificial analysis? Provide your comprehensive analysis

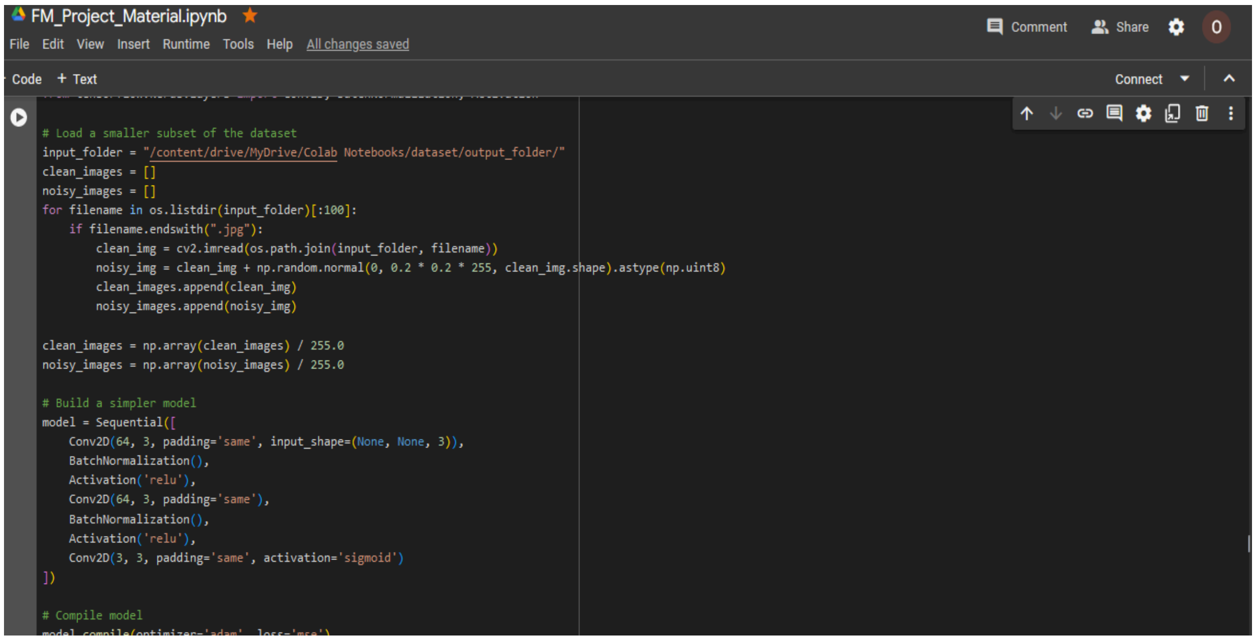
The whole point of adding the noise is to check the robustness of the POD analysis and make as much real noise as possible to simulate real world noise present in the data.

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 measurement 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 statistical properties are well-understood. This makes it easier to analyze and model mathematically.

5. Super-Resolving

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

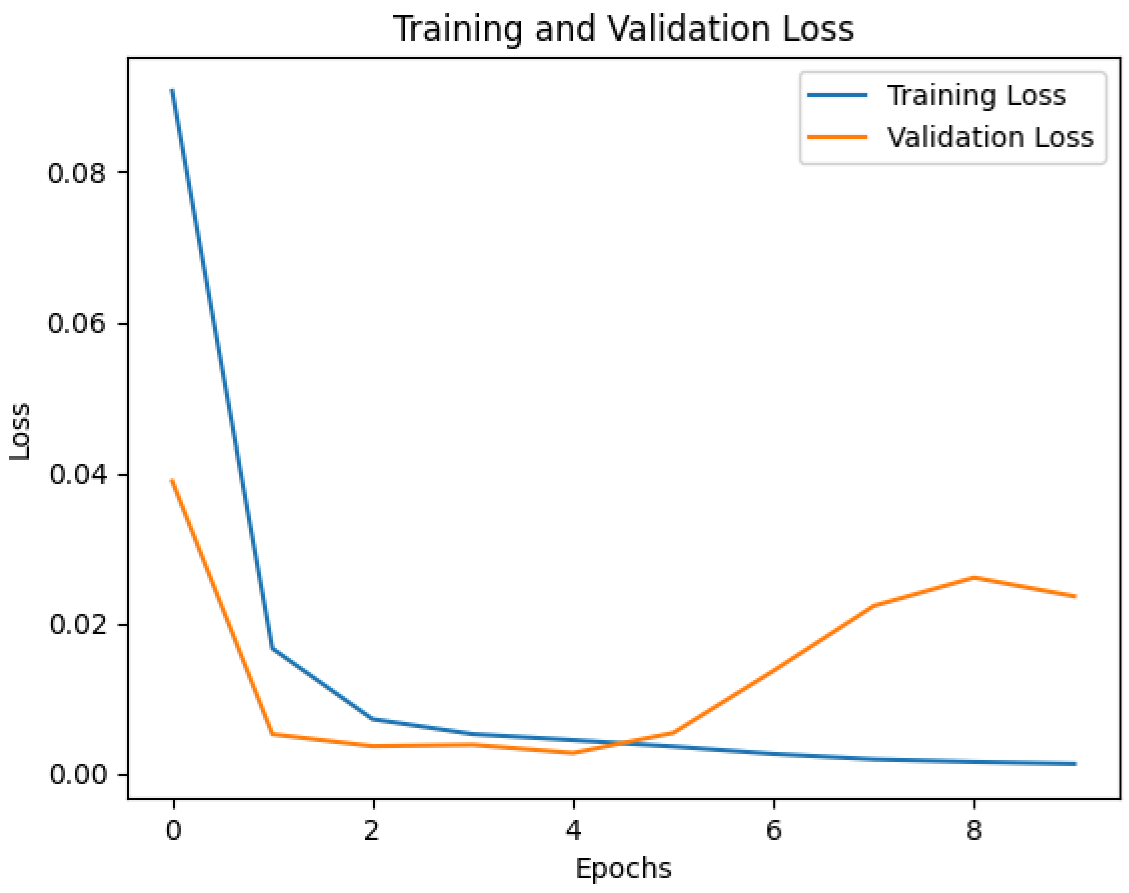

FM_Project_Material.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Code
+ Text
Connect

```
# 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')
```

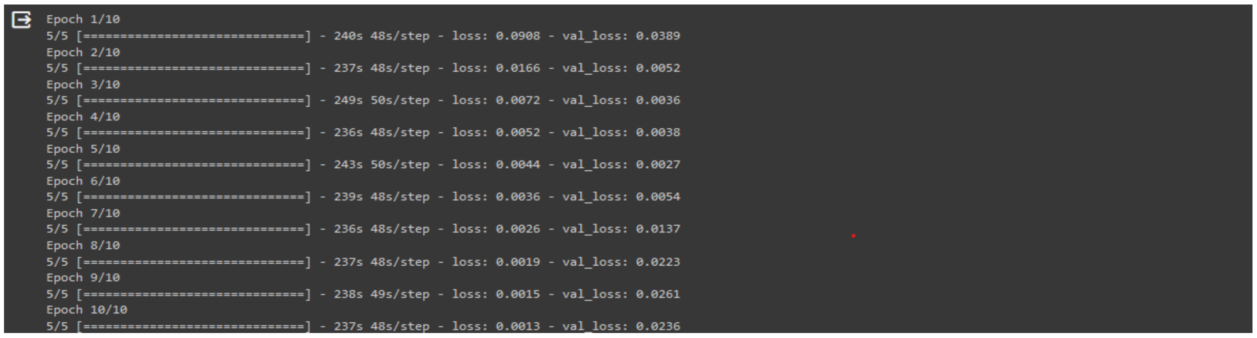


Problems faced:

The system crashed due to a big dataset.

Solution: We applied it on a smaller subset of the dataset

We set the epochs to 10 and then trained the model.



```
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
```