LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Technical Note	**LIGO-T1800245–v1**	2018/09/28

Optical loss characterization at the 40m prototype lab

SURF Student: Pooja Sekhar, Mentors: Gautam Venugopalan, Koji Arai

California Institute of Technology
LIGO Project, MS 18-34
Pasadena, CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project, Room NW22-295
Cambridge, MA 02139
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

LIGO Hanford Observatory
Route 10, Mile Marker 2
Richland, WA 99352
Phone (509) 372-8106
Fax (509) 372-8137
E-mail: info@ligo.caltech.edu

LIGO Livingston Observatory
19100 LIGO Lane
Livingston, LA 70754
Phone (225) 686-3100
Fax (225) 686-7189
E-mail: info@ligo.caltech.edu

http://www.ligo.caltech.edu/

Acknowledgements

I would like to thank my mentors Gautam Venugopalan and Koji Arai for their continuous support and guidance throughout the completion of the project. I am really grateful to Prof. Rana Adhikari for enlightening me on this new topic and for guiding me. I would also like to express my sincere gratitude to Gabriele Vajente and Holger Wittel for their constant help and support.

I am also grateful to all the members of the 40m, LIGO Scientific Collaboration (LSC), SURF program and California Institute of Technology for this great experience.

Contents

1 Introduction **5**

2 Objectives **7**

3 Telescopic Lens System **7**

4 Neural network to analyze the video **11**

4.1 Synchronizing captured video with applied signal 11

4.2 Understanding and implementing neural networks 13

4.3 Convolutional Neural Networks (CNN) 21

5 Tracking beam spot motion using OpenCV **25**

6 Interacting with GigE camera **26**

7 Work to be done and Challenges **27**

List of Figures

1 Gravitational wave strain is derived from differential arm motion (DARM) that is read-out from a photodiode downstream of the antisymmetric port [5] 5

2 Schematic of one of the Fabry perot cavity in 40m 6

3 Colorbar plots showing percentage error in magnification for various combinations of lenses 10

4 Block diagram that illustrates the scheme of training neural network to estimate test mass motion from the video of scattered light 11

5 Block diagram that illustrates the scheme of testing the trained neural network 12

6 Synchronizing video signal with applied sinusoidal signal. The top plot gives the time series of the sum of pixels from the captured video of ETMX. The second one gives the variation in background noise after the laser has been switched off. The third and fourth plots show the same time series of transmitted light and the sinusoidal dither signal given to the test mass. 12

7 General topology of artificial neural networks 13

8 Different activation functions 15

9 Time series of applied signal, NN output and residuals for Case 1 17

10 Variation of mean squared error with epochs for case 1 17

11 Time series of applied signal, NN output and residuals for case 2 18

12 Variation of mean squared error for case 2 18

13 Time series of applied signal, NN output and residuals for case 3 19

14 Variation of mean squared error with epochs for case 3 19

15 Time series of applied signal, NN output and residuals for case 4 20

16 Variation of mean squared error with epochs for case 4 20

17 Time series of applied signal, NN output and residuals for case 5. Mean and standard deviation values of the residual error have been shown in blue and green hashed lines respectively. 21

18 Time series of applied signal, NN output and residuals for case 6. Mean and standard deviation values of the residual error have been shown in blue and green hashed lines respectively. 22

19 Basic topology of CNN 23

20 Working of convolution layer 23

21 Max pool layer with filter size 2×2 and stride 2 24

22 Time series of applied signal, CNN output and residuals. Mean and standard deviation values of the residual error have been shown in blue and green hashed lines respectively. 24

23 Time series of output signal from the tracker (KCF) in OpenCV and the input signal. 26

24 Time series of output signal from the tracker (KCF) in OpenCV, filtered signal and the input signal. 26

25 MEDM screen to interact with GigE camera 27

26 Block diagram that illustrates how Medm screen interacts with the Pylon software of GigE camera. 28

List of Tables

1 Telescopic solution for different focal length combination of the lenses 9

2 Hyperparameters considered while training the neural network 14

3 Different cases that I tried while training the neural network 16

1 Introduction

Albert Einstein predicted the existence of gravitational waves (GWs) through his General Theory of Relativity in 1916. As compared to electromagnetic waves, GWs have very low absorption cross-section which makes them very effective in providing information about the universe and various astrophysical events that are distinct from those obtained using electromagnetic spectrum [1]. As a result of the relentless efforts by many scientists over many decades, these waves with extremely small amplitude were discovered on September 14, 2015 by the twin detectors of aLIGO with a signal to noise ratio of 24 [2].

Although the basic technique of Michelson interferometer has been employed, several advancements including four stage suspension of test masses for seismic isolation, highly stable laser that can be boosted upto 200W, control loop mechanisms and many more have been included so that the intensity of light at the photodiode which is a function of the differential arm length ($DARM = \frac{L_X - L_Y}{2}$) of the interferometer as shown in Figure 1 gives the gravitational wave strain [3]. The test masses are made of fused silica to minimize IR absorption and optical coatings are extremely smooth to reduce scatter loss. They are also very large with a diameter of 34cm and weigh around 40kg for vibration isolation [4]. Cameras have been placed facing various optic in the interferometer for continuous observation of the beam spot on the test masses.

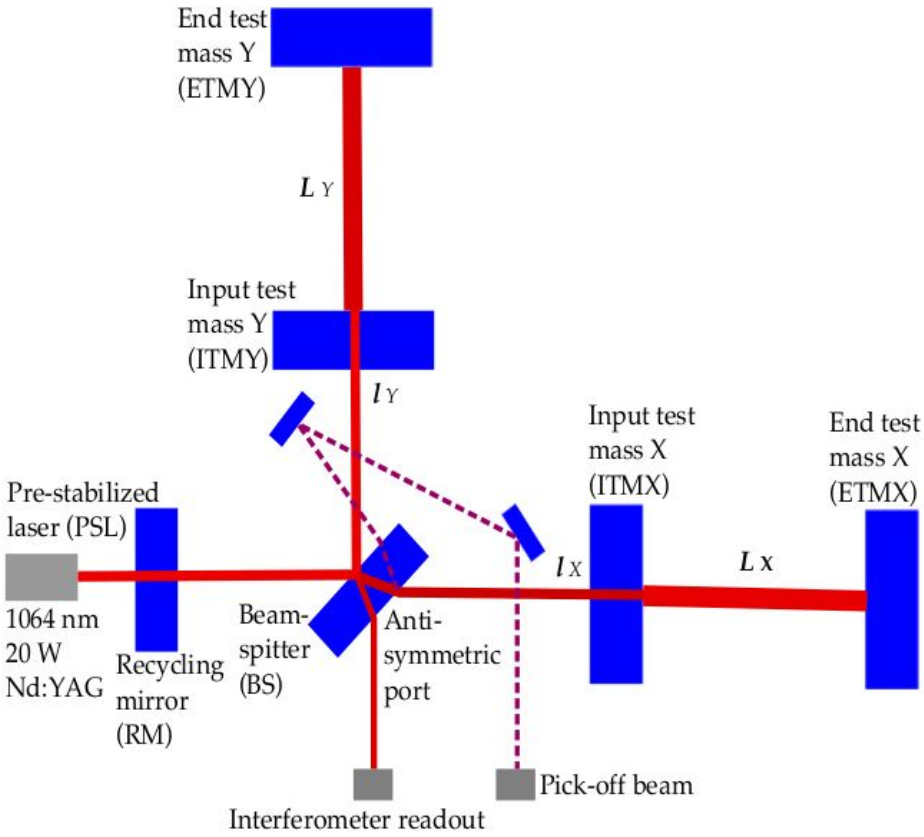


Figure 1: Gravitational wave strain is derived from differential arm motion (DARM) that is read-out from a photodiode downstream of the antisymmetric port [5]

Since closed circuit analog camera networks were associated with many disadvantages like inability to change the exposure times, digital camera networks were proposed and implemented a decade ago [6]. Later cameras with Gigabit Ethernet (GigE) connection were employed for faster data transmission over ethernet from a camera network to the computer. It provides data on position, mode and intensity fluctuations of the laser beam and through integration with the control loops of aLIGO, aid in aligning the laser beam in optical cavities. In this project, GigE cameras will be used to measure the scatter loss in optical cavities. Figure 2 shows how GigE cameras are placed to capture the scattered light from the end test mass.

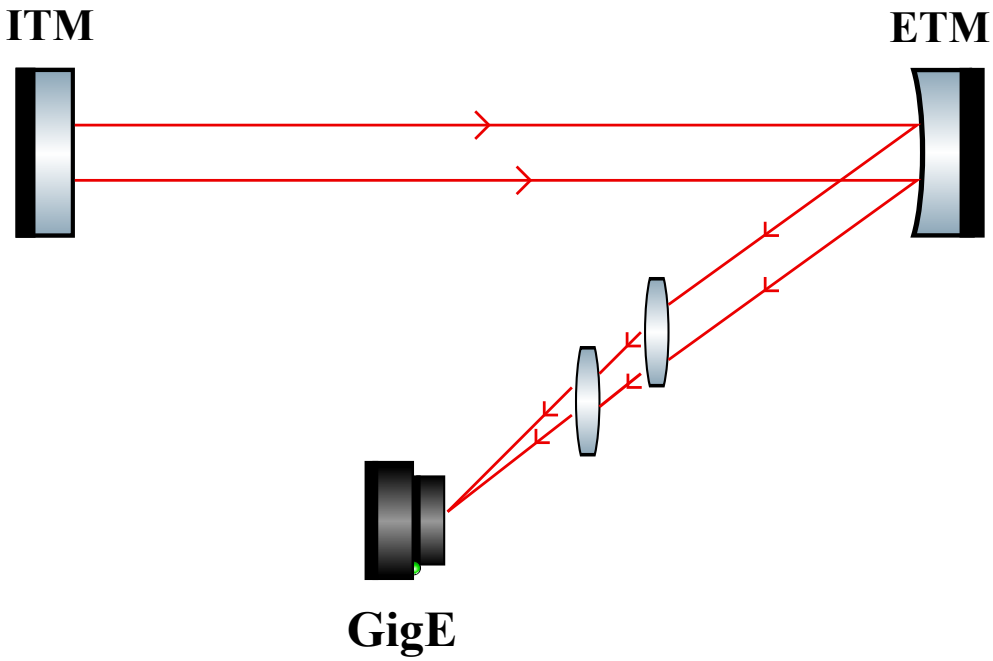


Figure 2: Schematic of one of the Fabry perot cavity in 40m

Being a very sensitive instrument, aLIGO is prone to a wide range of noise sources in spite of these precautions. Scattering of light, one of the noise sources, is the deflection of light from its path defined by specular reflection caused due to the irregularities on the surface of test masses. It reduces the power circulating in the Fabry perot cavities leading to a low signal to noise ratio and scattered light might also couple back into the instrument imparting random phase noise. In order to measure the scattered light, a CCD camera, Basler ace acA640-120gm, with Gigabit Ethernet (GigE) connection will be used such that it can image the test masses. Since intensity of the scattered light is angle dependent, bidirectional reflectance distribution function (BRDF) as defined below, will be employed to calibrate CCD from pixel counts to power units [7].

$$BRDF = \frac{P_s/\Omega}{P_i \cos(\theta_s)} \tag{1}$$

where θ_s is the scattering angle and Ω is the solid angle subtended at the sensor.

Here Lambertian model with BRDF $= \frac{1}{\pi}$ sr^{-1} has been assumed and CCD has been calibrated accordingly. The scattered power (P_s) has been calculated from the images of the calibrated

CCD according to the following equation,

$$P_s = \text{Calibration Factor(CF)} \times \frac{\sum_{ROI} \text{Pixel Value}}{\text{Exposure Time}} \tag{2}$$

where ROI is the region of interest selected around the beam spot in the captured images. Here the pixel counts have been summed over the ROI and normalized by the camera exposure time [7].

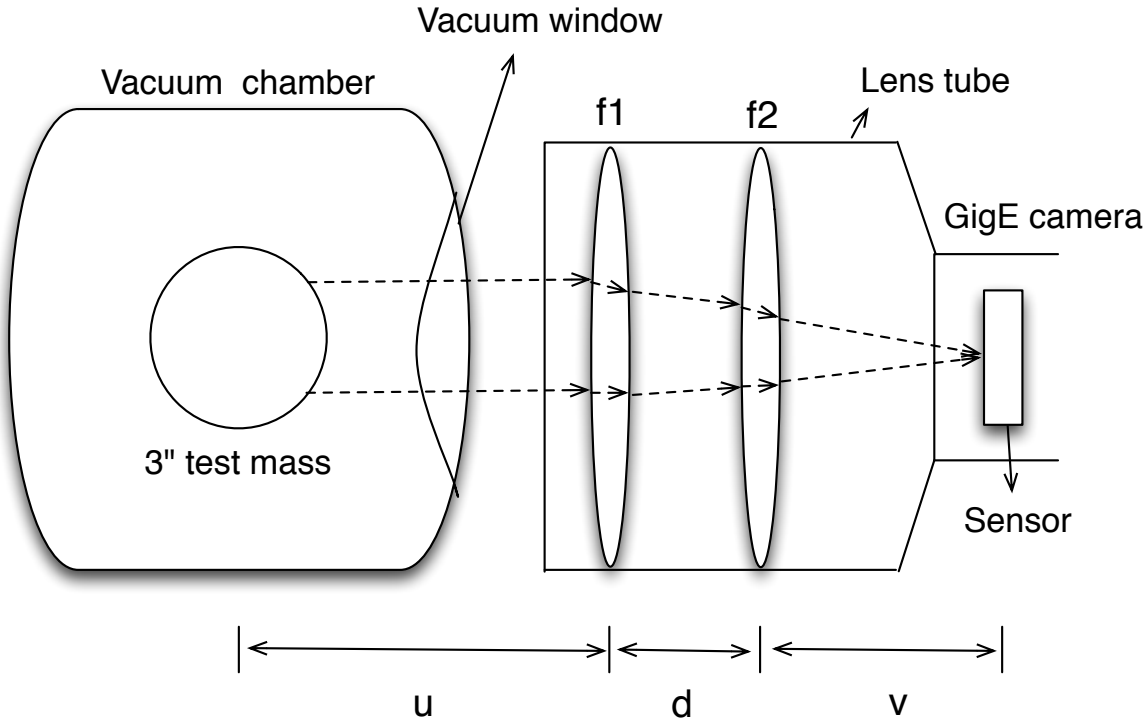
Thus, scattering loss in all the optical cavities can be measured using a GigE camera. The fluctuations observed in light intensity in the videos captured has been of great interest even in aLIGO since on finding the causes behind it, we can feedback loop to control the cavity parameters, centre the beam spot and finally reduce these fluctuations. Thus, it serves an additional alignment tool. In this project, we are trying to correlate the beam spot motion to the angular motion of the test mass.

2 Objectives

- Implement appropriate two lens system that focuses the image of the optic onto the camera.
- Develop a good means to interface Pylon software of GigE camera, Basler ace acA640-120gm, with python to capture images and videos at a frame rate of 30-50fps.
- Develop simulated video of the random motion of beam spot on the test mass.
- Develop a neural network to correlate the motion of the beam spot to the motion of the test masses from ground vibrations and other means and then to analyze the motion of point scatterers like dust particles across the surface of the test masses from the videos captured.
- Develop a convolutional neural network to extract the features i.e. slight movement of the beam spot and to map it to the angular motion of the test mass.
- Develop other image processing techniques to resolve test mass motion from the video of scattered light.

3 Telescopic Lens System

Here, our objective is to focus test mass of size 3 inches and a beam spot of size 1 inch on rectangular sensor of GigE camera placed at a distance of 1m from the test mass. A schematic of the telescopic lens system with two biconvex lenses that has to be placed between the CCD camera and vacuum viewport to focus the beam spot and the optic onto the camera sensor has been shown in the figure below.



Path of light rays shown in dashed lines

ABCD matrix analysis has been employed to calculate the focal lengths of a combination of two biconvex lenses to be placed at a distance d apart so as to focus the beam spot onto the camera sensor [8]. According to this formalism, the object and the image parameters are related by the following equation,

$$\begin{bmatrix} y \\ \theta\prime \end{bmatrix} = \begin{bmatrix} 1 & v \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \frac{-1}{f_1} & 1 \end{bmatrix} \begin{bmatrix} 1 & d \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \frac{-1}{f_2} & 1 \end{bmatrix} \begin{bmatrix} 1 & u \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \theta \end{bmatrix} \tag{3}$$

where, y and x represent the image and object heights respectively, $\theta\prime$ and θ are the angles made by the rays from image and object with the principal axis, f_1 and f_2 are the focal lengths of the lenses, v = distance of the camera sensor from the second lens, d = distance between the lenses and u = distance of the object from the first lens. The system matrix that relates the image and object parameters is as follows,

$$\begin{bmatrix} 1 & v \\ 0 & 1 \end{bmatrix} \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} 1 & u \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} A + Cv & u(A + Cv) + B + Dv \\ C & Cu + D \end{bmatrix} \tag{4}$$

According to the imaging criteria in paraxial approximation, light rays should converge at the image plane irrespective of the incident angle, therefore,

$$u(A + Cv) + B + Dv = 0 \tag{5}$$

$$v = \frac{-Au - B}{Cu + D} \tag{6}$$

Also, magnification (m) of the imaging system is defined as,

$$m = A + Cv = 1 - \frac{d}{f_2} + (\frac{-1}{f_1} - \frac{1}{f_2} + \frac{d}{f_1 f_2})v \tag{7}$$

The optimum image circle diameter is found to be $\frac{1}{4}$ inches such that the image circle just encloses the sensor and thus efficiently utilizes the pixels. [9] So, the required magnification is found to be 0.083 for focusing the entire optic of size 3 inches and 0.25 for the beam spot of size 1 inch. If the magnification values are greater than these respective values, some information will be lost.

Using the above formalism a Python program was developed to select the appropriate combination of focal lengths with the available options from Thorlabs such that the combination can focus the entire optic as well as the beam spot by varying the distance between the lenses (d). Here lenses with 2" diameter have been chosen to collect maximum power since the solid angle subtended on 1" lenses reduces to one-fourth of that on 2" lenses. It was found that three combinations of focal lengths served the purpose. Percentage error in magnification is defined as follows,

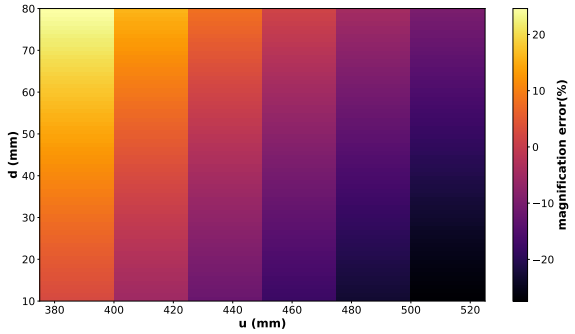
$$\triangle m = \frac{\text{calculated m} - \text{actual m}}{\text{actual m}} \times 100 \tag{8}$$

where actual m values are 0.083 for focusing the test mass and 0.25 for focusing the beam spot. The colorbar plots for the percentage error in magnification of 150mm-150mm, 125mm-150mm and 125mm-125mm lenses to focus the entire optic as well as the beam spot are shown in Figures 3(a)-3(f). A sensitivity analysis was also performed and the results show that an error of 1cm in object distance and 5mm in the distance between lenses leads to a change in the error in magnification less than 3%.

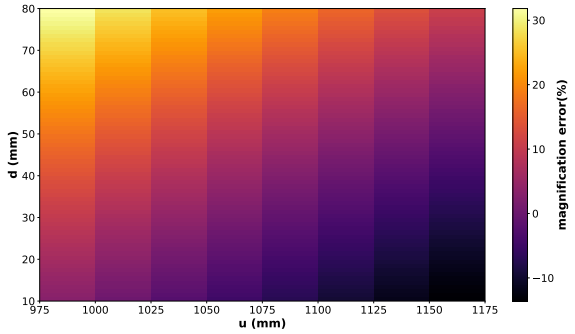
The percentage magnification error along with u,v and d values for some of the cases that serves our purpose have been listed in Table 1.

Table 1: Telescopic solution for different focal length combination of the lenses

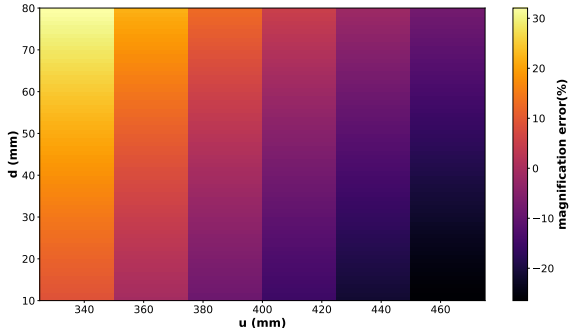
	u (cm)	v (cm)	d (cm)	f_1 (mm)	f_2 (mm)	Δm(%)
test mass	85	6.5	1	125	125	-1.1
3"	92.5	7.2	1	125	150	-1.3
	102.5	7.9	1	150	150	-2.3
beam spot	65	4.6	15	125	125	-8.2
1"	62.5	3.4	15	125	150	-8.2
	65	3.5	15	150	150	-7.6



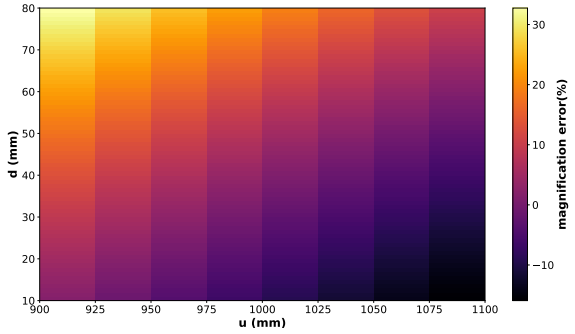
(a) 150mm-150mm combination to image 1" beam spot



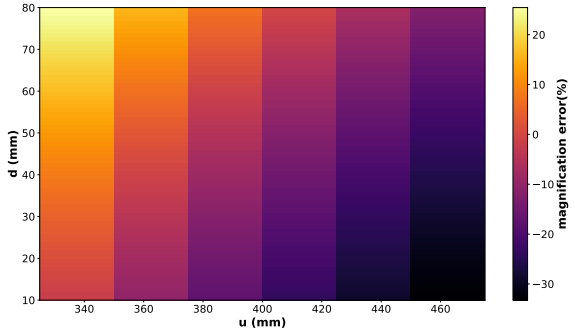
(b) 150mm-150mm combination to image 3" optic spot



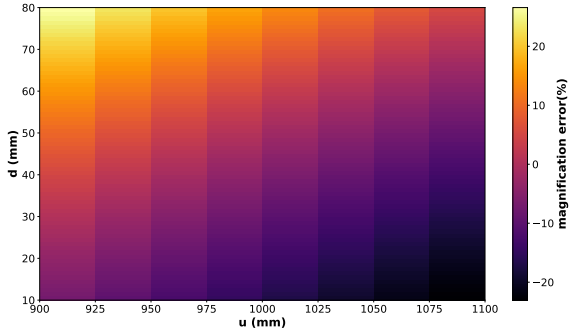
(c) 125mm-150mm combination to image 1" beam spot



(d) 125mm-150mm combination to image 3" optic spot



(e) 125mm-125mm combination to image 1" beam spot



(f) 125mm-125mm combination to image 3" optic spot

Figure 3: Colorbar plots showing percentage error in magnification for various combinations of lenses

Here we have even considered imaging the object to a size lesser than $\frac{1}{4}$ inches by approximately 8%.However, there is a practical difficulty in placing the lenses since there exists difference in the total distance between the test mass and camera of approximately 8cm, 20cm and 28cm in the two cases i.e. on focusing beam spot and test mass for solutions of 125mm-125mm, 125mm-150mm and 150mm-150mm combinations respectively. Thus, 125mm-125mm combination may be appropriate for this purpose. However, this solution requires the lenses to be apart by 15cm and imposes a practical difficulty that we need to join many lens tubes or have a longer lens tube.

4 Neural network to analyze the video

In order to resolve the motion of test mass from the beam spot motion, neural networks (NN) can be used [10]. The block diagram in Figure 4 illustrates the technique of training the neural network to correlate the beam spot motion with the angular motion of test mass like pitch and yaw of the test mass. A sine signal of low frequency is imparted to the test mass such that it dominates over the other causes of test mass motion and a synchronized video is to be taken with GigE camera. The method by which we tried to synchronize video with the applied signal is described later. A Python program is developed to convert the video stream to image frames and the resultant 2D array of pixel values of each image is then converted to a 1D array which is further divided into train and test values. The train values are then given to the neural network implemented in Keras using the Sequential model[11]. In the case of sequential model, we assume the number of layers and its size initially and then the model is compiled and trained to fit to the applied sine signal. The resultant error is then minimized by choosing an appropriate optimization algorithm that updates the weights of inputs via backpropagation. Finally the test dataset of 1D pixel values from a captured video is given to the trained neural network as in Figure 5 and it is compared with the applied signal to give the figure of merit of the process.

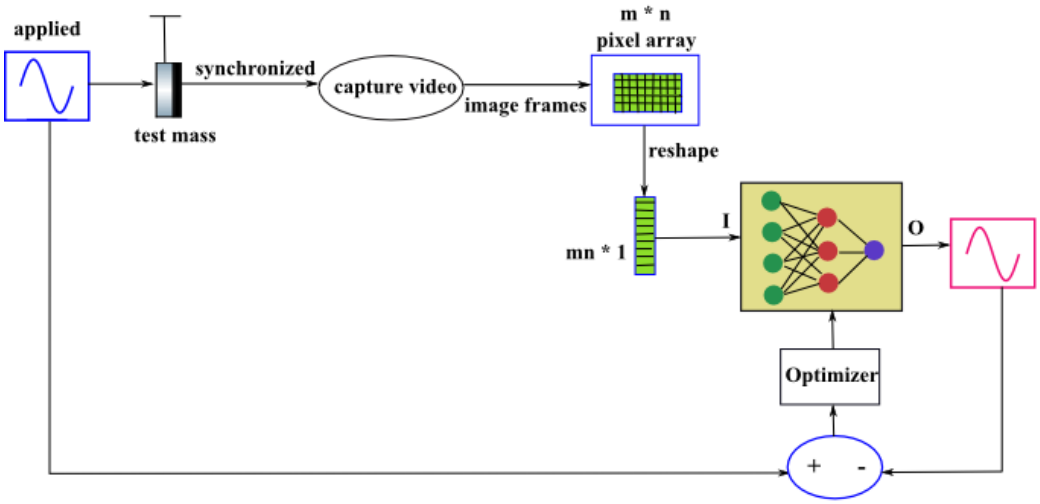


Figure 4: Block diagram that illustrates the scheme of training neural network to estimate test mass motion from the video of scattered light

4.1 Synchronizing captured video with applied signal

A dither signal of frequency 0.2Hz was applied in pitch to X-end test mass (ETMX) and a video was captured for 59 seconds. Then I developed a python program to capture the video and convert it into a time series of the sum of pixel values in each frame using OpenCV to see the variation. Initially we had tried the same with green laser light and signal of approximately 11.12Hz. But in order to see the variation clearly, we repeated with a lower frequency signal after locking IR laser. The plot has been shown in Figure 6. The first graph gives the fluctuations in the sum of pixels from the video. The third and fourth graphs

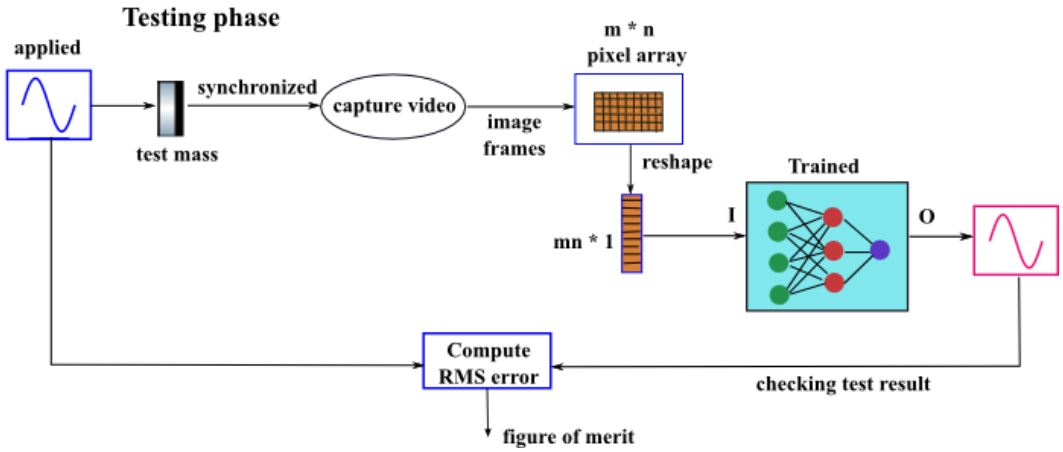


Figure 5: Block diagram that illustrates the scheme of testing the trained neural network

are that of transmitted light and the signal applied to ETMX to shake it. Since the video captured using the camera was very noisy and intensity fluctuations in the scattered light had twice the frequency of the signal applied as expected [12], we captured a video after turning off the laser. The second plot gives the background noise probably by the transmission of signal from the camera. Also, the transmitted light intensity didn't follow the same pattern as the first plot. This may be due to the clipping of the beam onto one of the mirrors that was found later.

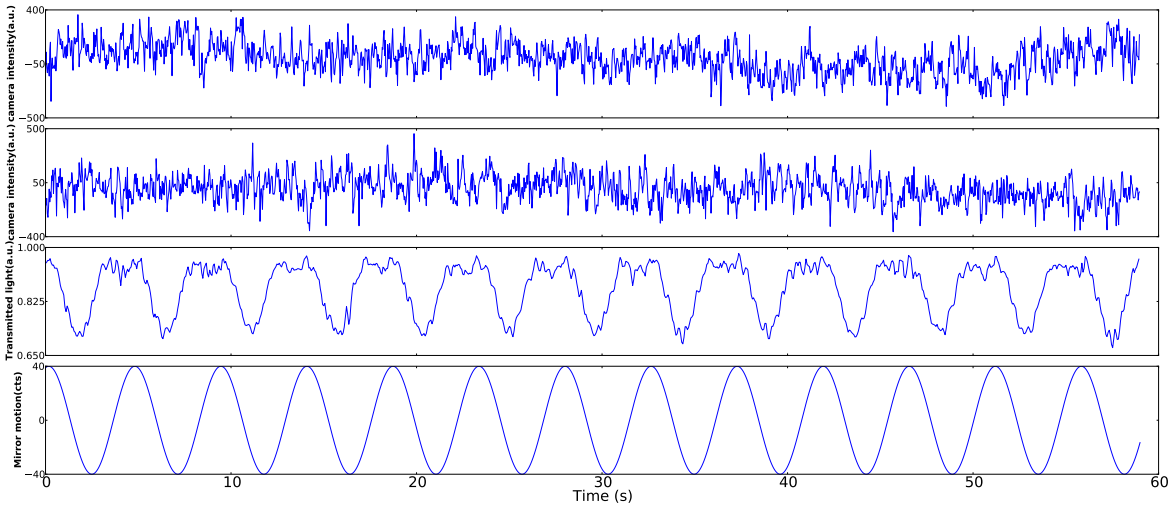
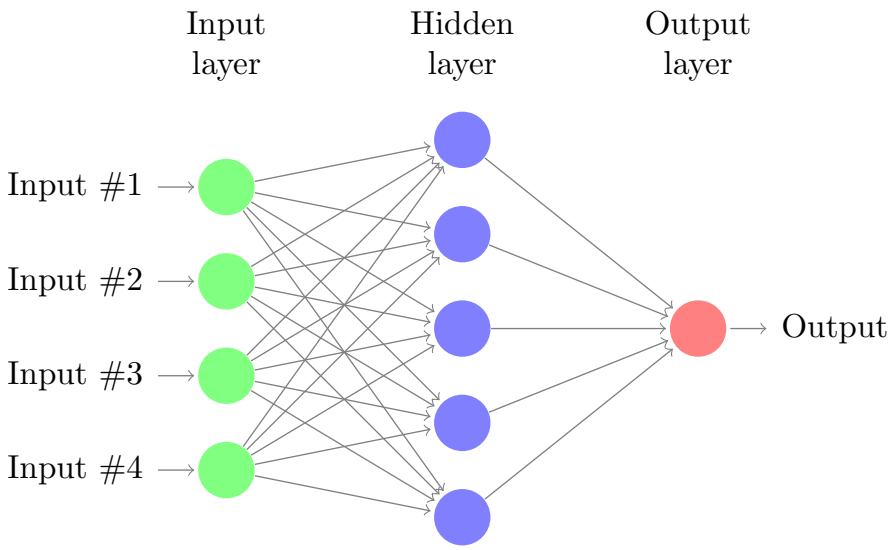


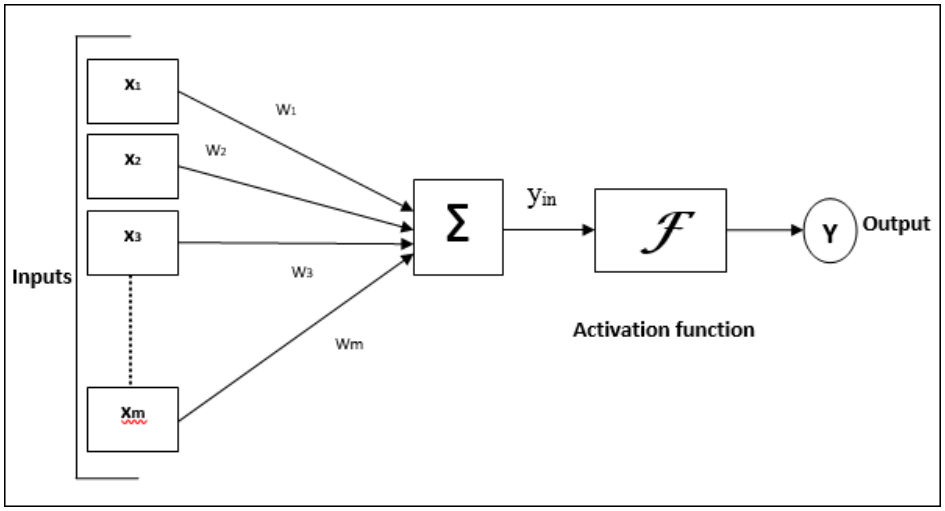
Figure 6: Synchronizing video signal with applied sinusoidal signal. The top plot gives the time series of the sum of pixels from the captured video of ETMX. The second one gives the variation in background noise after the laser has been switched off. The third and fourth plots show the same time series of transmitted light and the sinusoidal dither signal given to the test mass.

4.2 Understanding and implementing neural networks

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. An artificial neuron is the basic building block of ANN and its general model has been shown in Figures 7(a) and 7(b).[13] Figure 7(a) shows an input layer of four nodes which are fully connected



(a) Schematic of Artificial neural networks



(b) Working of ANN

Figure 7: General topology of artificial neural networks

to a hidden layer of five nodes and then they are interconnected to an output layer of one node from which we get the output of the neural network. In Figure 7(b), the input signals (i.e. nodes) $x_1, x_2, ..., x_m$ are multiplied by an initial set of weights $w_1, w_2, ..., w_m$ and then summed over. This linear combination is then compared with a bias value based on which they are acted upon by a nonlinear function to yield an output value. The difference between

this value and the desired output is then minimized by a backpropagation algorithm where the values of weights are updated.

There are two basic types of learning algorithms - Supervised and Reinforcement learning. [14]

- Supervised learning

  It works on a set of labelled data i.e. network is trained on a given set of inputs and their correct outputs. Based on this learning, a trained network can give output for a different set of similar input. Here we deal with supervised learning because we have the labelled data on which neural network can be trained.

- Reinforcement learning

  It usually works in gaming applications. Here a state interacts with the environment by a set of actions and by observing the result, a corresponding reward is calculated. The basic objective is to maximize this reward by finding the best action to perform based on the state.

The model topology i.e. the number of hidden layers and the number of nodes in each layer varies according to the problem [15]. Here we are trying to implement neural networks in Keras which is a high level neural network API that can run on top of Tensorflow, CNTK or Theano. The various parameters to be considered while developing a neural network have been listed in the Table 2.

Table 2: Hyperparameters considered while training the neural network

Loss function	In simple words, it is the difference between the actual output and the predicted output, $(L = y - \hat{y})$ [17].
Optimizer	Optimization algorithm that updates the weights and biases of the neural network to minimize loss [18].
Activation	Calculates the weighted sum of the inputs, adds bias and decides whether it should be activated or not i.e. gives output in the range of 0-1 or -1 to 1 depending on the function [19]. The evolution of different activation functions for a given set of data values have been shown in Figure 8.
Batch size	Number of training inputs in one forward/backward pass i.e. per gradient update.
Epochs	Number of times the entire training dataset passes through the neural network.
Dense	Fully connected NN layer. It implements, output = activation(dot(input, kernel) + bias.
Number of nodes	The number of nodes in a dense layer of NN
Dropout	During the training stage, randomly selected nodes are ignored depending on the dropout specified. It is a regularization technique to reduce overfitting.

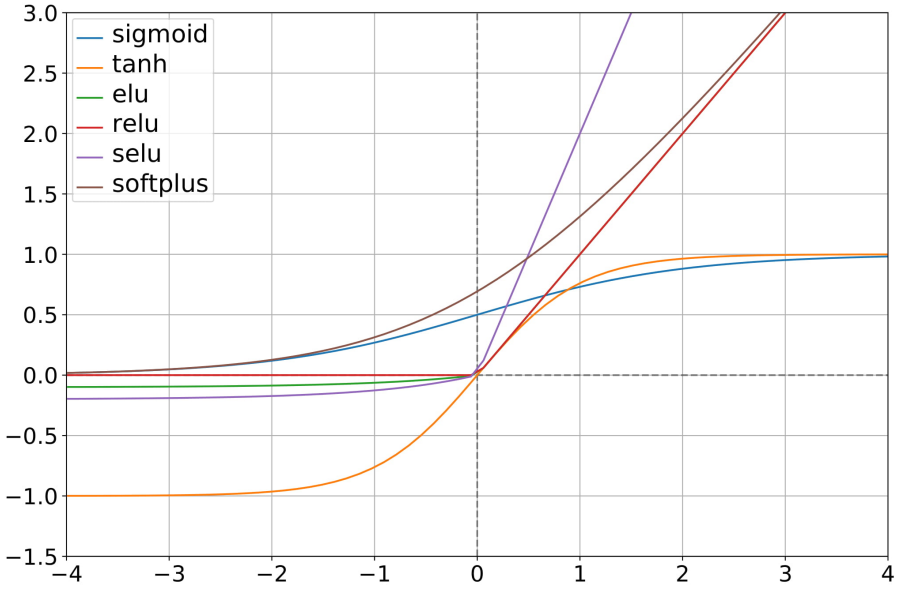


Figure 8: Different activation functions

Neural network training is done on two types of problems: Regression and Classification and this is a regression type problem. So the loss functions available in Keras that can be used are mean squared error, mean absolute error, mean squared logarithmic error, logcosh, poisson and cosine proximity. Out of these, I have chosen mean squared error since it's the most simple and commonly used loss function for regression problems in Keras. There are a large number of optimizers available in Keras. I had used RMSprop with learning rate = 0.00001 for smaller dataset and for faster training initially but then switched to Nadam that is essentially Adam RMSprop with Nesterov momentum since its the advanced optimizer recommended by people in this community . Its parameters include learning rate of 0.001 and other default parameters defining the Nesterev momentum [20]. The activation functions that I tried include Rectified linear unit (relu) and Scaled exponential linear unit (selu) for hidden layers and linear for output layer. The default batch size of 32 have been used in all the cases. I varied the number of epochs for various cases depending on whether loss function curves reached a minima or not. While loss function, optimizer and batch size have been chosen based on recommendations by experts in online sites, activation function, number of epochs, number of nodes and dropout value have been chosen empirically.

Initially I tried to train the network on the video of scattered light captured from the test mass. Since the video was noisy, we were not able to train the network. Then I simulated a video (64 × 64 image frames) of motion of test mass in vertical direction by giving a sinusoidal signal of amplitude 20 pixels and frequency 0.2 Hz at 10 frames per second. This simulated data has an advantage while training since we have control over the data set against which we train the neural network. Thus, the trained neural network would yield high accuracy on similar test data. Moreover, this can be extended to real cases by moving the beam spot in a random fashion in the simulated video and then training the network on

this data. Different approaches that I have tried while training the neural network has been listed in Table 3. The following cases describe the network models that I have tried:

Table 3: Different cases that I tried while training the neural network

Case 1	Smaller training dataset and smaller NN model (Figure 9)
Case 2	Smaller training dataset with uniform noise and smaller NN model (Figure 11)
Case 3	Larger training dataset with uniform noise and larger NN model (Figure 13)
Case 4	Larger training dataset with uniform noise and smaller NN model (Figure 15)
Case 5	Lower resolution training dataset with uniform noise and smaller NN model (Figure 17)
Case 6	Lower resolution training dataset (noise added to amplitudes and frequencies) with uniform noise and smaller NN model (Figure 18)
Case 7	CNN training on the previous case (Figure 22)

1. Smaller dataset and network model:

   Two dense layers of 256 nodes with a dropout of 0.1 each and activation by relu. Optimizer = RMSprop (learning rate = 0.00001). Number of epochs = 1000. Training dataset included 75 frames and test included 25 frames. The output of neural network as well as the input signal given to move the beam spot and the variation in mean squared error with epochs have been shown in Figure 9 and Figure 10. The plots clearly show that the neural network created is biased towards higher values and it is overfitting.

2. Smaller dataset + noise and smaller network model

   The same has been repeated by adding random uniform noise ranging from 0 to 25.5 in each image frame. We got similar results as shown in Figures 11 and 12.

   Then I tried simulating a video of 128 × 128 images in which beam spot moves by the application of sine wave for 1300 cycles, downsampled it to 64 × 64 image frames and then divided it into train (1000 cycles) and test datasets (300 cycles).

3. Bigger dataset and bigger network model

   In order to get the full information from the pixels, I created a model that included 4096 nodes in the first hidden layer, 1024 in the second, 512, 256, 64, 8 and 1 in the third, fourth, fifth, sixth and the output layer respectively with a dropout of 0.1 in the first three layers. Activation functions used were selu for the hidden layers and linear for the output layer. Optimizer = Nadam (learning rate = 0.001). The run was computationally very expensive and test loss increased after around 50 epochs. In order to reduce this overfitting, I added a dropout of 0.1 to the layer of 256 nodes and eliminated the initial layer of 4096 nodes. The NN output and residual error and the variation in mean squared error have been given in Figures 13 and 14. But still test loss increases after about 60 epochs and it's seen that neural network never gives the highest point value of the applied signal.

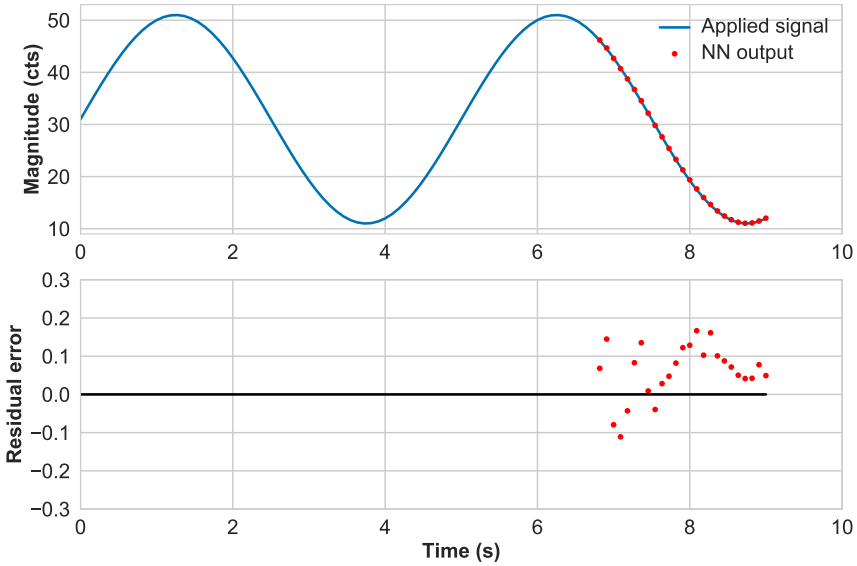


Figure 9: Time series of applied signal, NN output and residuals for Case 1

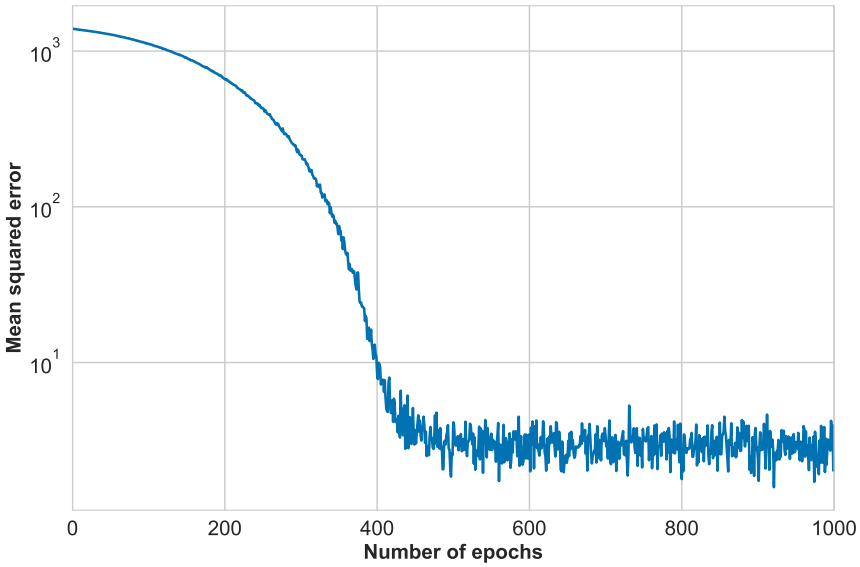


Figure 10: Variation of mean squared error with epochs for case 1

4. Larger dataset and smaller network

   I added random uniform noise to the simulated data. Since most of the cases above were overfitting except that test loss is lower than train loss in the beginning, I trained on 1000 cycles of simulated data and tested on 300 cycles using smaller model network of two densely connected layers of 256 nodes each with no dropout [21]. The output of the neural network, residual error and the variation in mean squared error have been

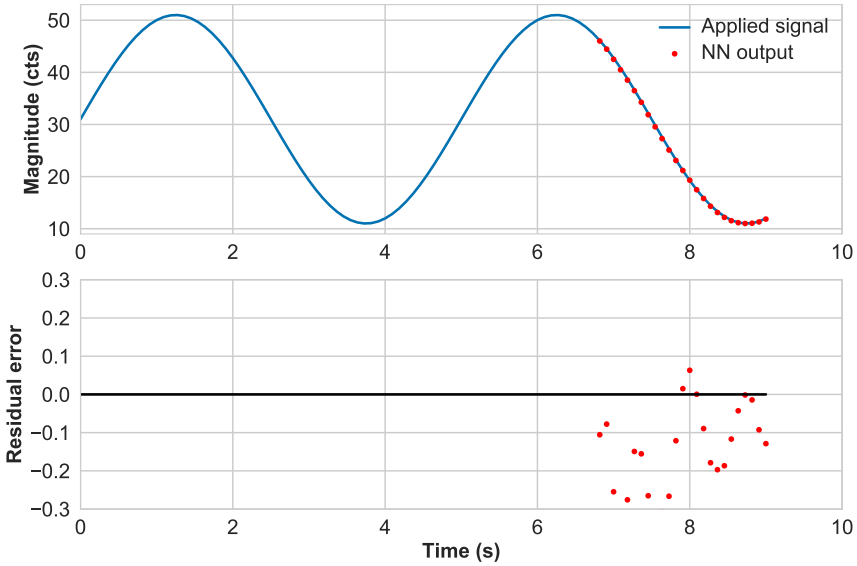


Figure 11: Time series of applied signal, NN output and residuals for case 2

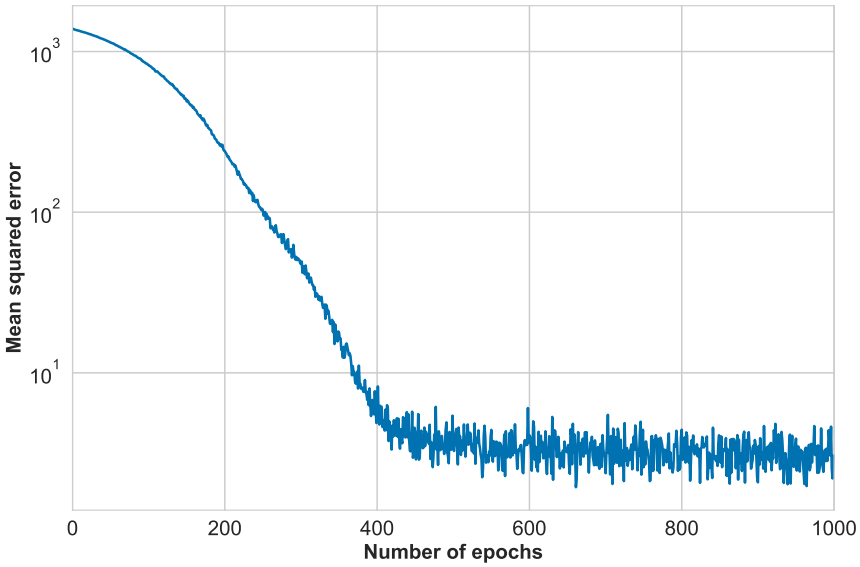


Figure 12: Variation of mean squared error for case 2

given in Figures 15 and 16.

Since all the above cases using large neural networks were overfitting and a lot of computation time was involved, I simulated a video of 32 × 32 image frames and tried using a smaller neural network with a single hidden layer consisting of 8 nodes. Total image frames (300) captured from the video have been divided into train (120), validation (30) and test (150) datasets. The optimizer used is Nadam (learning rate = 0.00001,

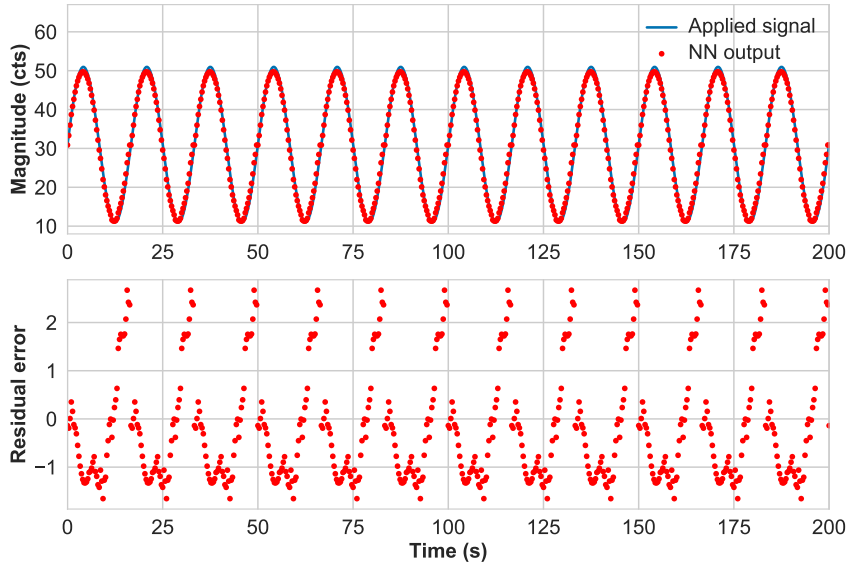


Figure 13: Time series of applied signal, NN output and residuals for case 3

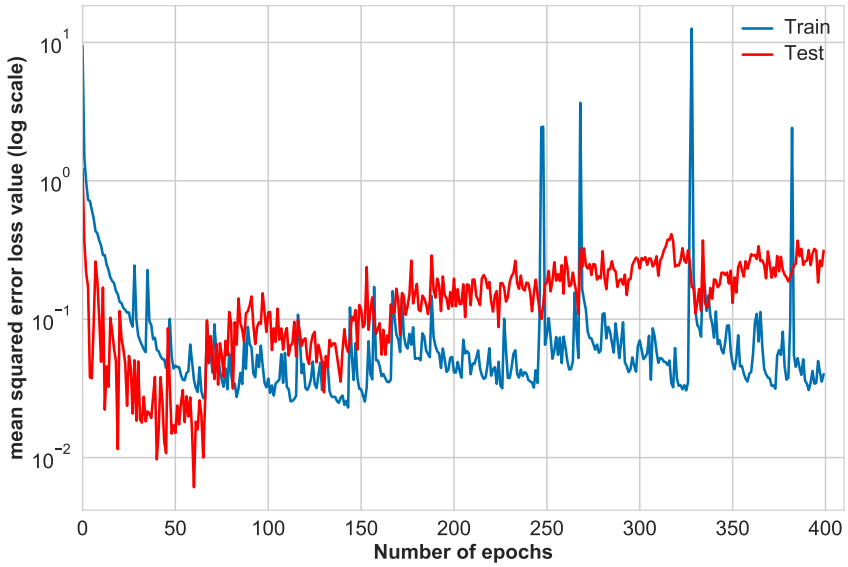


Figure 14: Variation of mean squared error with epochs for case 3

beta1 = 0.8, beta2 = 0.85) where beta parameters control the exponential decay of the first and second moments and have default values of 0.9 and 0.999 respectively. Activation function employed for the hidden layer is selu and that for the output layer is linear for the following cases as in the previous studies. The number of epochs have been changed to 128 keeping the loss function and batch size the same.

For generality, we have moved the beam spot by applying a linear combination of four

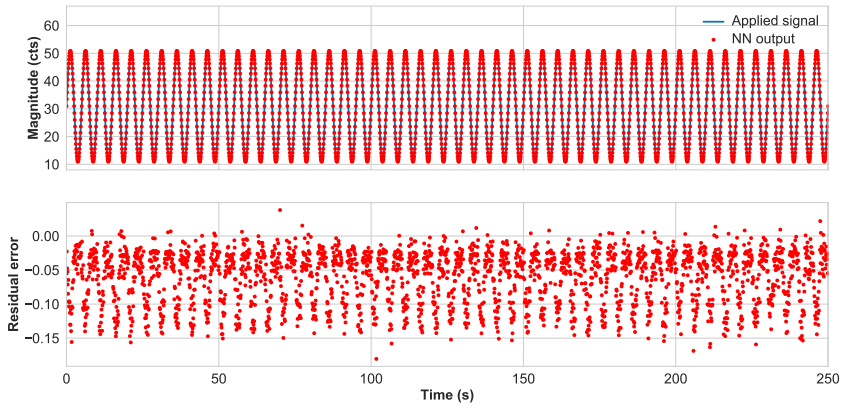


Figure 15: Time series of applied signal, NN output and residuals for case 4

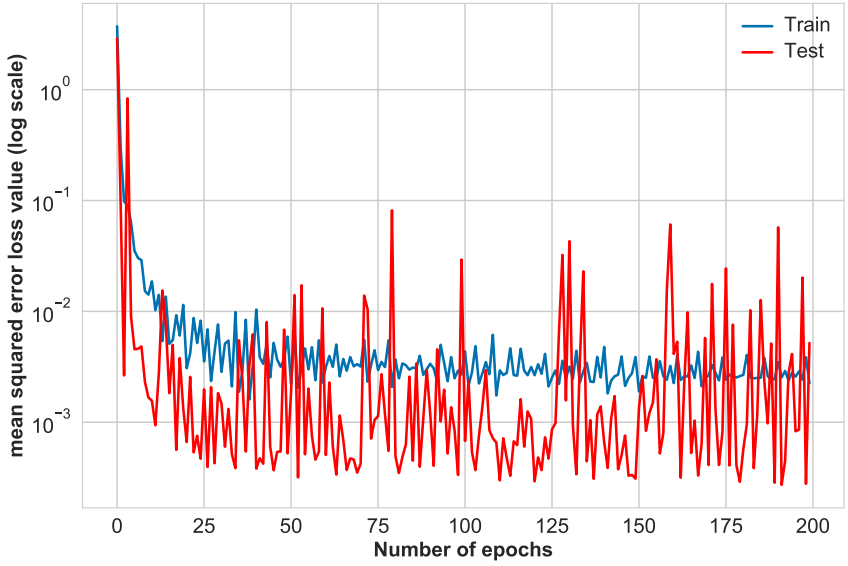


Figure 16: Variation of mean squared error with epochs for case 4

sine waves with frequencies of 0.2, 0.4, 0.1 and 0.3 Hz and amplitudes of 3.2, 1.28, 1.6 and 2.56 respectively. Also random uniform noise ranging from 0 to 5 have been added to all the image frames. It was also found that loss value reduced when this linear combination data against which the network is being trained is normalized such that values range from 0 to 1.

5. Lower resolution dataset and smaller network

   The output of the neural network and residual error and the variation in mean squared error have been given in Figure 17. The mean and standard deviation values of the residual error have also been shown as colored hashed lines. This small network is found to give a good fit even though it's obviously not the best fit.

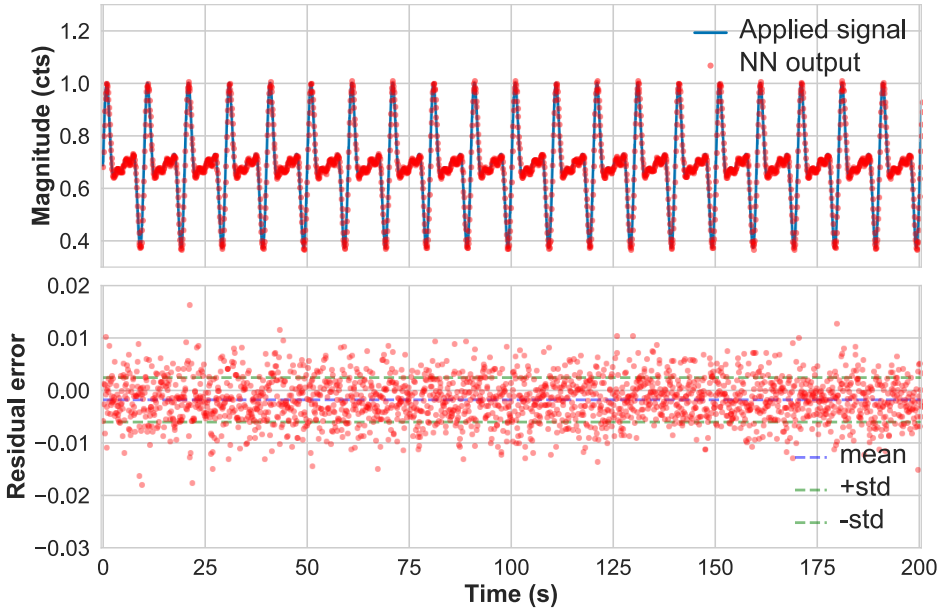


Figure 17: Time series of applied signal, NN output and residuals for case 5. Mean and standard deviation values of the residual error have been shown in blue and green hashed lines respectively.

6. Lower resolution dataset (noisy beam spot motion) and smaller network

   Here random uniform noise ranging from 0 to 0.05 have been added to the above amplitudes and frequencies of the sine waves so that beam spot moves in a random fashion apart from the uniform random noise added to the image frames. The above trained network has been tested on this noisy dataset and the resultant output and residual error have been shown in Figure 18. The beam spot moves beyond the maximum limit of pixel values for which the network has been trained. Therefore, we find that the output of NN has been saturated at its highest point.

4.3 Convolutional Neural Networks (CNN)

In order to extract the slight random movement of beam spot, convolutional neural networks may be required. The basic topology of CNNs has been shown in Figure 19. [22] The input 2D pixel data is being fed to a stack of convolutional, pooling and dense layers. The first two layers have been discussed in detail below:

- Convolutional layer

  In image processing, to calculate convolution at a particular location (x, y), we extract $k \times k$ sized kernel from the image centered at location (x, y). Then multiply the values in this kernel element-by-element with the convolution filter (also sized $k \times k$) and add them all to obtain a single output. This convolution filter is then slid over the entire pixel matrix by a fixed pixel number called stride to obtain an activation map.

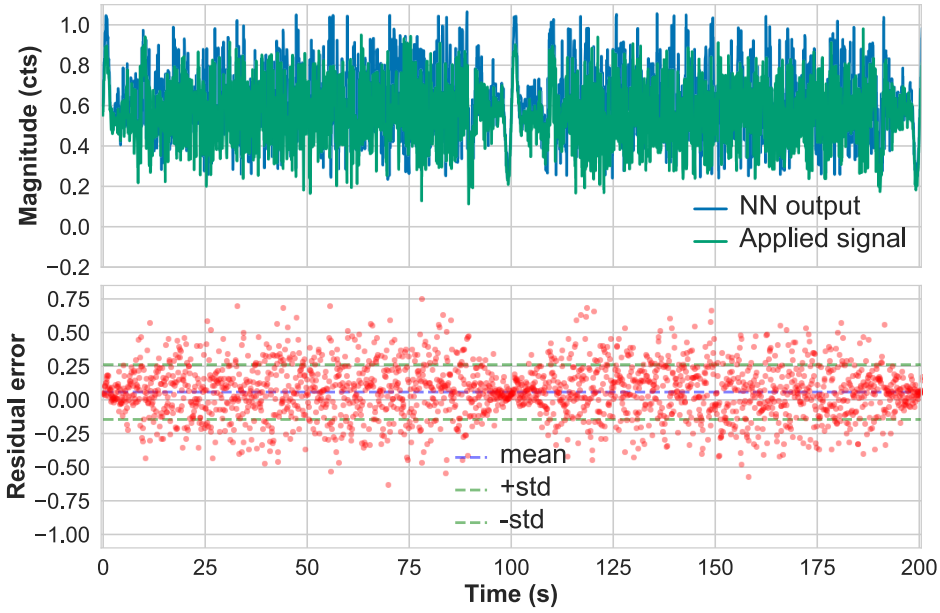


Figure 18: Time series of applied signal, NN output and residuals for case 6. Mean and standard deviation values of the residual error have been shown in blue and green hashed lines respectively.

Figure 20 shows that 32 activation maps of size 30×30 have been produced from input image $32 \times 32 \times 3$ by using 32 convolutional filters of size $3 \times 3 \times 3$. Thus heirarchial features in the input image are identified.

- Pooling layer

  In order to reduce the computational complexity, a pooling layer is used after the convolutional layer. It reduces the number of parameters by reducing the spatial size. The most common form of pooling is Max pooling as shown in Figure 21 where we take a filter of size 2×2 and apply the maximum operation over the sized part of the image.

Lower resolution noisy dataset trained on convolutional neural network

The input dataset for the previous case where the beam spot moves with the application of four sines with random amplitudes and frequencies has been trained on a convolutional neural network with a dense layer of four nodes and an output layer of one node and the following parameters:

- Number of filters = 2
- Kernel size = 2
- Size of pooling windows = 2

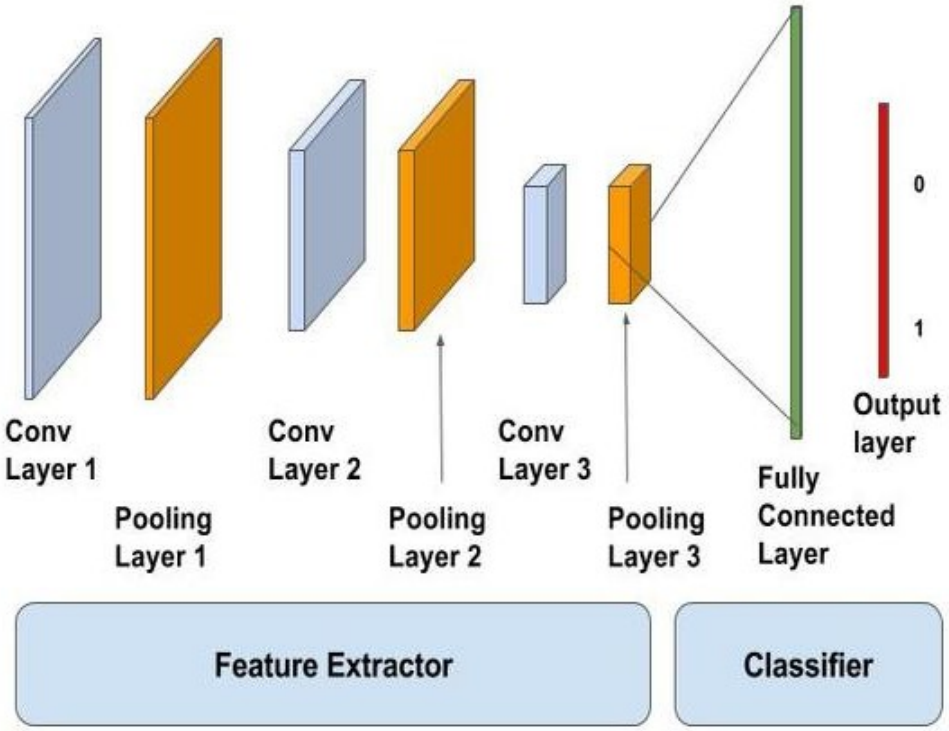


Figure 19: Basic topology of CNN

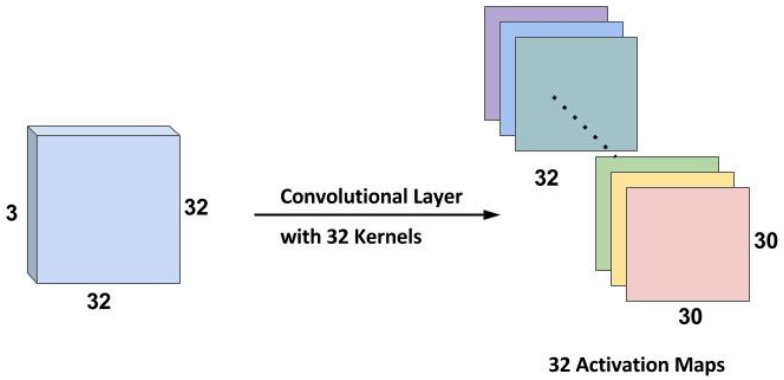


Figure 20: Working of convolution layer

The output of CNN and the residual error along with its mean and standard deviation values have been shown in Figure 22. The residual error values are found to be much lower than the previous cases. These plots reveal a great improvement of CNN over regular neural

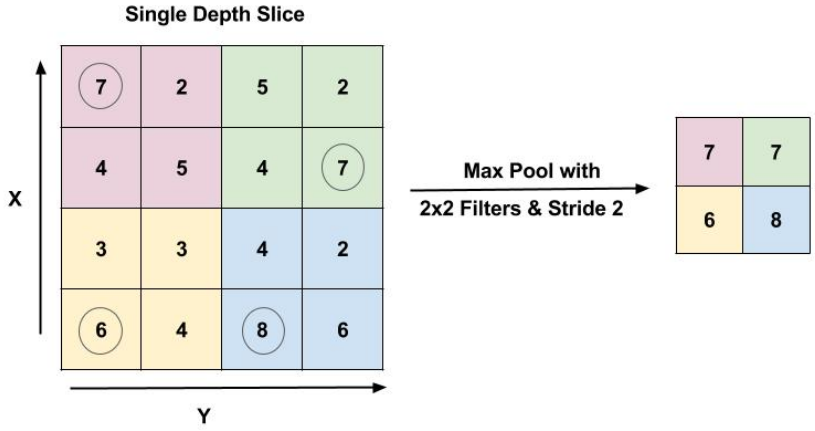


Figure 21: Max pool layer with filter size 2×2 and stride 2

networks.

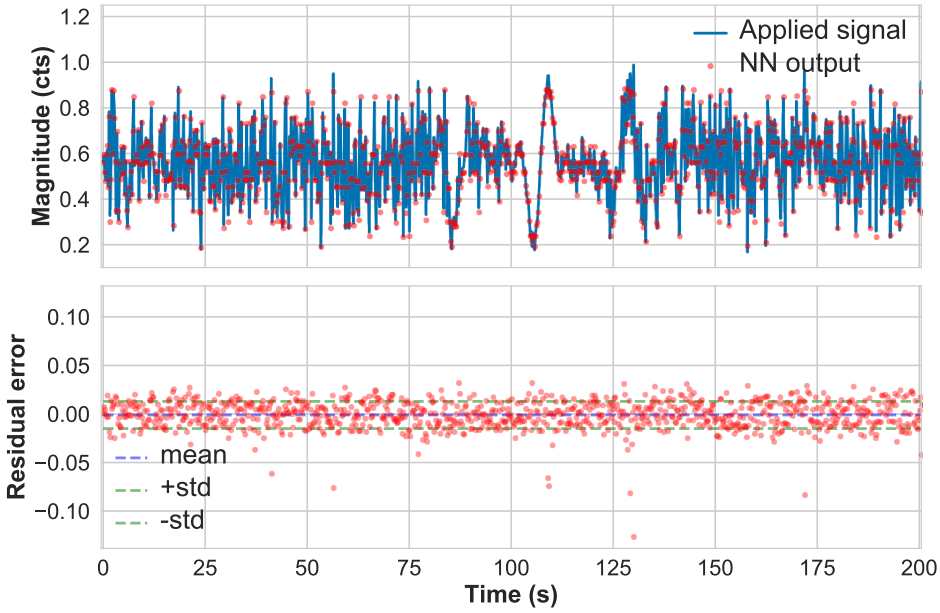


Figure 22: Time series of applied signal, CNN output and residuals. Mean and standard deviation values of the residual error have been shown in blue and green hashed lines respectively.

In short, I have tried tuning the hyperparameters of the neural network and the size of training data set to get optimized results as discussed before. From these observations, I found that my technique of finding the right values for the hyperparameters of neural

network was not fully correct. The right thing to perform will be to code a multiparameter optimization algorithm that finds the right set of parameters for training this simulated data set. I had tried to code a Monte Carlo simulation to do the same but it needs to be further explored.

5 Tracking beam spot motion using OpenCV

I simulated a video that moves the beam spot from the centre of the image frame by a maximum of one pixel by applying a sinusoidal signal of frequency 0.2Hz. In order to try other techniques for tracking the beam spot motion, I used a program that uses Kernelized Correlation Filter (KCF) to track object motion from the video because it is the recommended one. This method is particularly preferred in cases where the training data sizes are limited and when prior knowledge of the data similarities is available [23]. This technique uses deep learning at its backend i.e. using deep architectures to perform kernel machine optimization for computational efficiency and end-to-end referencing.

The main objective of the tracking algorithm is to identify the object in the current frame provided we have tracked the object in all the previous frames.Here is how the tracker works. Since we have been tracking the object, we know its motion and appearance. The initial bounding box defined by the user containing the object is taken as positive and the background i.e. image patches outside the bounding box are taken as negative. Thus positive bags containing image patches with the object and negative bags (background) have been defined and the mathematical properties of the overlapping regions in positive bags have been utilized to track the object.

In the program we can initially define the bounding box (rectangle) that encloses the object we want to track in the video or select the bounding box by dragging in GUI platform. Then I saved the bounding box parameters in the program (x and y coordinates of the left corner point, width and height) and plotted the variation in the y coordinates. The plot of the output of this tracking program and the applied signal has been given in Figure 23. The amplitudes of tracker output and applied signals do not much due to a calibration issue. Also, the output is not found to be exactly sinusoidal because it may not be able to track very slight movement especially at the peaks where the slope = 0. So I passed the tracker output to a low pass filter with threshold of 0.2 Hz and the respective plots have been shown in Figure 24. It was also found that this technique can even track the motion of beam spot in the real video captured by GigE camera where the spot moves due to the dither signal applied to the test mass. However, this technique failed in detecting very slight movement of the beam spot under seismic disturbance. Thus, this may not be a suitable technique for our purpose.

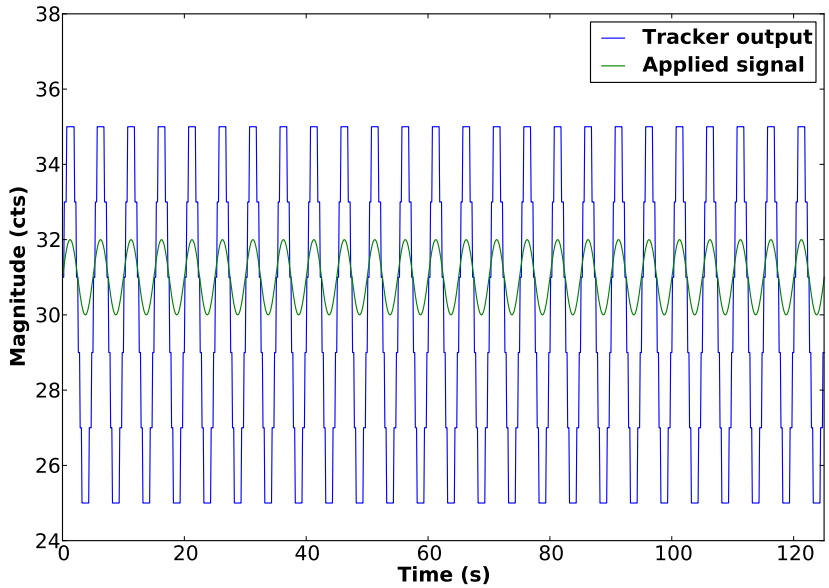


Figure 23: Time series of output signal from the tracker (KCF) in OpenCV and the input signal.

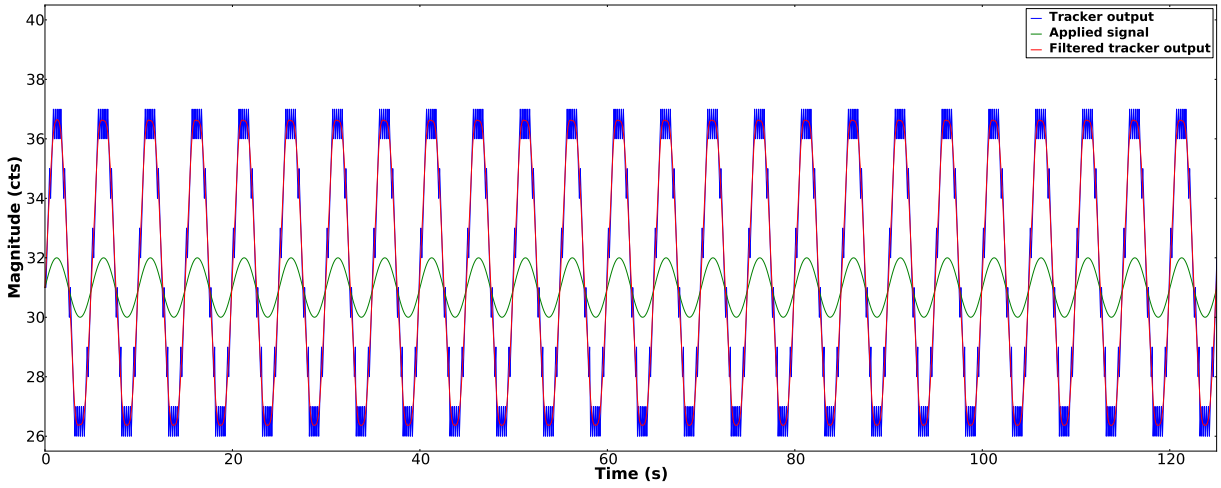


Figure 24: Time series of output signal from the tracker (KCF) in OpenCV, filtered signal and the input signal.

6 Interacting with GigE camera

An MEDM screen (GUI) as given in Figure 25 has been developed to interact with GigE camera. Through this, we can change exposure time, capture images and videos and take a snapshot. The block diagram shown in Figure 26 illustrates the method by which we can

change the GigE camera settings. In order to do this, we interact with the inbuilt pylon software of the GigE using a python wrapper, pypylon. Initially a python code like camera server has been run using the EPICS channel of MEDM screen to enable connection with the pylon software. Then program codes like camera client and camera client movie can be run to launch a live video feed and to save the video feed to an avi file.

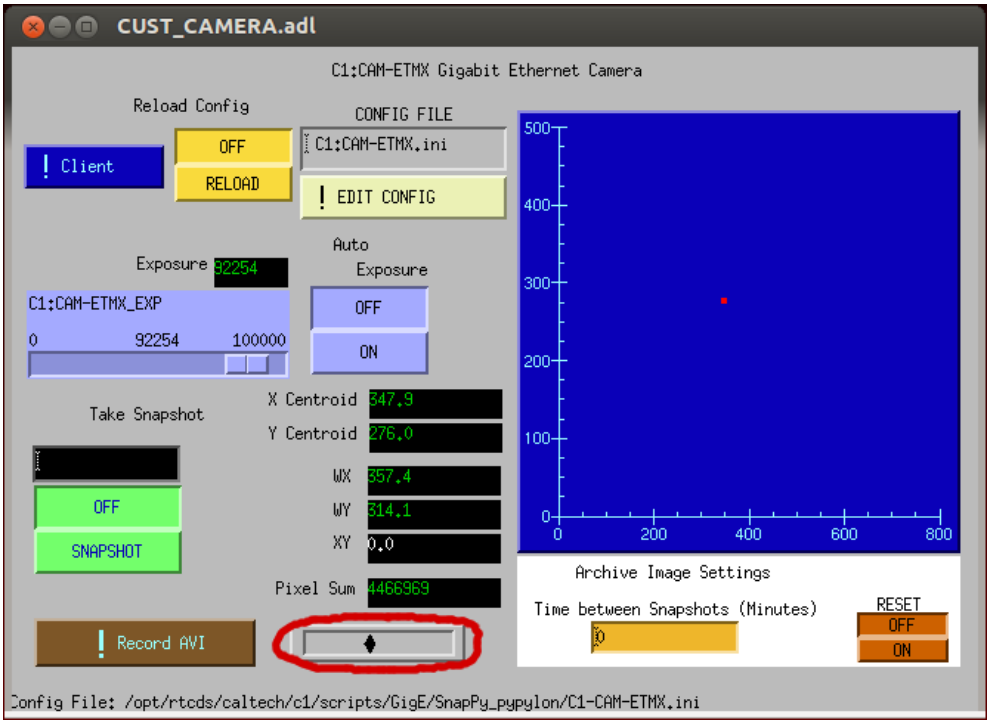


Figure 25: MEDM screen to interact with GigE camera

7 Work to be done and Challenges

- Find the best hyperparameters of the neural network implemented in Keras that minimizes the error or loss value by making a multiparameter optimization algorithm (Monte Carlo simulation) and then to test its accuracy on simulated video. I had already tried this but needs to be further explored.

- Train neural network with a simulated video of random motion of the beam spot in pitch and yaw that resembles the practical case. Till now we had tried with the beam spot motion in only one dimension.

- Further work has to be done in convolution neural networks (CNN) for feature extraction and weight sharing. They can even be combined with Recurrent neural networks (RNNs) since it store information about the previous trial. This may be advantageous over DNNs for tracking the beam spot since it moves in only a confined space under seismic disturbance.

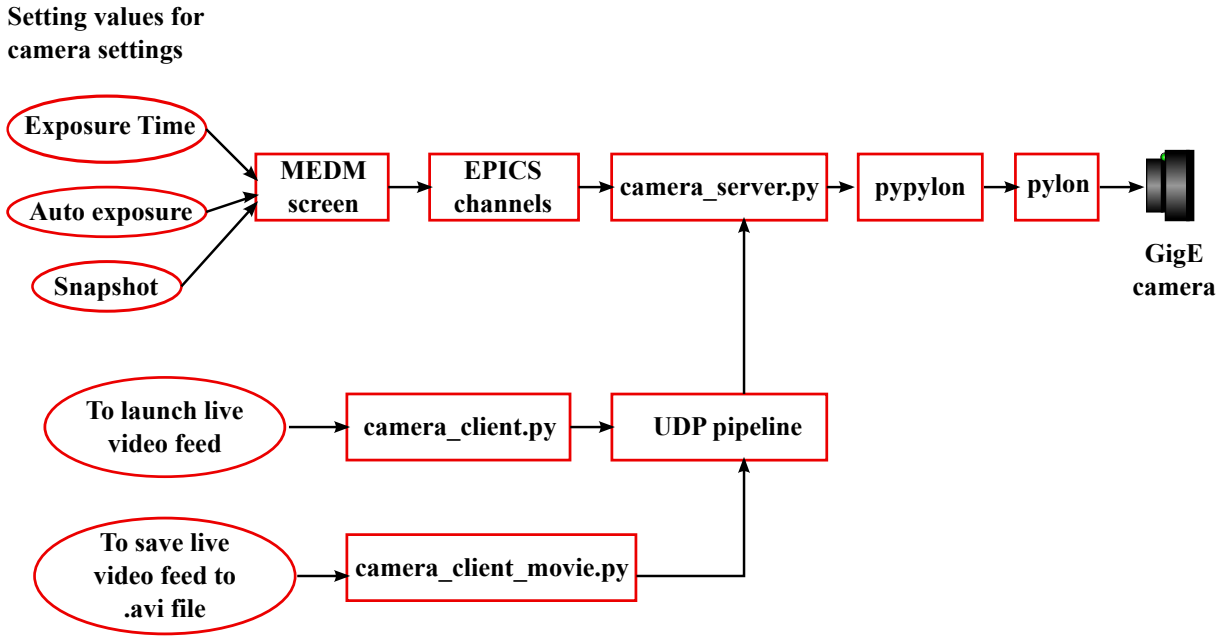


Figure 26: Block diagram that illustrates how Medm screen interacts with the Pylon software of GigE camera.

- Test the accuracy of trained neural network on the actual video captured by the GigE camera and calibrate test mass motion with the beam spot motion for alignment purposes.

References

[1] David G Blair, *The Detection of Gravitational Waves*. Cambridge university press (2005).

[2] B. P. Abbott *et. al.*, *Observation of Gravitational Waves from a Binary Black Hole Merger*. Phys. Rev. Lett. 116, 061102 (2010).

[3] https://www.ligo.caltech.edu/page/ligo-technology

[4] https://www.ligo.caltech.edu/page/optics

[5] Grant David Meadors, Keita Kawabe, Keith Riles *Increasing LIGO sensitivity by feed-forward subtraction of auxiliary length control noise*. Classical and Quantum Gravity 31, 10 (2014).

[6] Rana Adhikari, Joseph Betzwieser, Alan Weinstein *Development of a digital camera network and associated analytical tools for the Caltech 40m LIGO prototype*. Ligo Paper (2008).

[7] Fabian Magana-Sandoval, Rana Adhikari, Valera Frolov, Jan Harms, Jacqueline Lee, Shannon Sankar, Peter R. Saulson and Joshua R. Smith *Large-angle scattered light measurements for quantum-noise filter cavity design studies*. Technical Note LIGO-T1400252-LSC (2014).

[8] Ajoy Ghatak, Optics.4th Edition (2009)

[9] Jigyasa Nigam, Mentors: Gautam Venugopalan, Johannes Eichholz *Characterization of Test Mass Scattering*. LIGO-SURF Report (2017).

[10] http://neuralnetworksanddeeplearning.com/chap1.html

[11] https://keras.io/

[12] Dana Z. Anderson *Alignment of Resonant Optical Cavities* Applied Optics Vol. 23, Issue 17 (1984).

[13] https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_basic_concepts.html

[14] Andrew G. Barto, Thomas G. Dietterich *Reinforcement Learning and its Relationship to Supervised Learning.*

[15] https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

[16] http://www.texample.net/tikz/examples/neural-network/

[17] https://medium.com/data-science-group-iitr/loss-functions-and-optimization-algorithm

[18] https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-netwo

[19] https://medium.com/the-theory-of-everything/understanding-activation-functions-in-ne

[20] Timothy Dozat *Incorporating Nesterov Momentum into Adam.*

[21] https://machinelearningmastery.com/diagnose-overfitting-underfitting-lstm-models/

[22] https://www.learnopencv.com/image-classification-using-convolutional-neural-networks

[23] https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/