# Extracting Wavelength Information from Laser Speckle Patterns using a CNN

**2d Lt Aaron Bonner**

**WI25 645A Final Project**

E-mail: aaron.bonner.3@us.af.mil

**Abstract**

This project examines the usage of a machine learning model for extracting the wavelength of incoming laser light by analyzing the resulting speckle pattern that is generated. The scope of the project is focused on a scenario involving a satellite in space under threat from an adversary laser of unknown wavelength, and although the data simulation portion is entirely simulated, certain design choices are made to best emulate this scenario to remain relevant to real-life analysis. The value of satellite assets has steadily grown as their use for global communications and surveillance has become more prevalent in modern warfighting for the United States Air Force and Space Force. The rise of high-power ground-based lasers as a low-cost threat to satellites has been recognized by these forces for future conflicts. One promising solution to this problem is the use of a machine learning model to analyze laser speckle patterns. In the machine learning method, a model is formed and trained on labeled speckle patterns. After training, it is expected that the model will detect the small variance and pattern differences that occur as wavelength changes when a speckle pattern is created. It is expected that the model will be able to output with a high degree of accuracy the wavelength used to generate a speckle pattern when provided a speckle pattern of unknown wavelength within its training range. In this project, a Convolutional Neural Network is trained with three convolutional layers and regularization techniques. This model predicted wavelengths of unknown speckle patterns with a mean accuracy error of 2.28, meaning that its prediction was on average 2.28nm off from the true wavelength used. Additionally, it is noted that in general, a machine learning solution relies on training data, which is generated or acquired in an environment with known conditions. For the aforementioned project scope, the transmission of light from ground to space introduces a factor of uncertainty that may impact the viability of these methods for the given scenario.

Keywords: laser speckle, wavelength, machine learning, CNN

---

## 1. Introduction

### *1.1 Problem Statement*

Satellites play a critical role in the United States Air Force and for the United States military. As of 2023, it is reported that the United States operates 247 satellites for military purposes, and 5,176 satellites for any purpose [1]. Additionally, the threat of directed energy weapons against satellites is a known threat to military assets. In a Congressional Research report on directed energy, it states that China "may field higher power systems that extend the threat to the structures of non-optical satellites" and that Russia has conducted research into a High-Energy Laser (HEL) that "some analysts assert is intended to dazzle satellites" [2]. In a conflict, learning more information about incoming threats may prove monumental in developing countermeasures to them. For wavelength information specifically, knowing wavelength information could be used to correlate the laser's origin to known development intelligence gathered through other means. Additionally, this wavelength information could be passed to onboard optical filters to reduce the laser's ability to affect valuable optical sensors on the satellite.

### *1.2 Application of Artificial Neural Networks*

The application of Artificial Neural Networks (ANNs) provides a promising solution for analyzing laser speckle patterns. Specifically, Convolutional Neural Networks are best suited for this task given that our input data can be considered an image. Since we can train the model on labeled data, the goal is to teach the model how to recognize the subtle differences that are caused by a change in wavelength in the laser speckle pattern. For this model, we

treat each different wavelength in the dataset as a different class, and so the model will predict what class is associated with the test and validation sets.

### *1.3 Baseline*

The baseline accuracy of the model was set to be 5nm for this project. This number was not derived from any sort of statistical regression but chosen to reflect the capabilities of optical filters available in the current world. For example, the manufacturer ThorLabs sells the K2XL1 device in their Kurios® Liquid Crystal Tunable Bandpass Filters. This device features a wavelength range of 420nm to 730nm, with a bandwidth of around 7nm to 14nm depending on the specific wavelength chosen as the center point, as seen in Figure 1 [3].

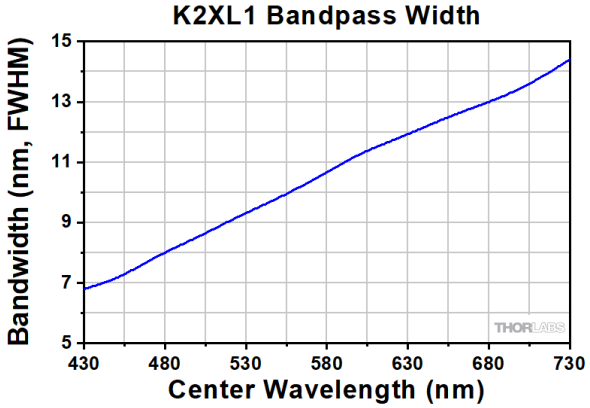


Figure 1. ThorLabs K2XL1 Bandwidth Datasheet. For this device, the bandwidth is interpreted as width of the filter's coverage at any given center point in its range. [3]

Although this project is not focused on one particular optical filter in mind, the choice of 5nm accuracy is reasonable given the current capabilities of publicly available optical bandpass filters.

## 2. Background

### *2.1 Data Generation and Visualization*

Starting off, 5120 laser speckle patterns of size 2000 by 2000 pixels were generated for this project using Python simulations. Laser speckle patterns are the result of coherent light reflecting off or through an optical surface that generates a random interference pattern [4]. The resulting random plane is a result of the constructive and destructive interactions between the random phases of the light scattered by the optical material. Overall, the simulation consisted of the following steps. First, a Gaussian beam was generated with a specific wavelength. Next, it was propagated through a phase screen that imposed a uniformly random phase shift at each point. Lastly, the field was propagated to the detector plane, where it was then converted to an intensity field that is the laser speckle pattern. Special parameters that were chosen were a propagation length from the phase screen to detector plane, in this case 10cm, the size of a standard cube satellite. Additionally, the sampling rate at the detector plane was maintained at the same value throughout all wavelengths, as in real life the pixel size of the sensor does not change arbitrarily. As stated, a total of 5120 speckle patterns were generated, which consisted of 20 iterations of the 400nm-656nm spectrum. Between each iteration, a new uniformly random phase was chosen so that each iteration was unique, but shared the same statistical properties between each run. An example of two speckle patterns is shown in Figures 2 and 3, and although these images were center-cropped to be 200 by 200 pixels in order to see the speckles, they are a valid representation of the visuals for a whole image. The two images visually show differences in the number and intensity of the speckle patterns, so it is expected that the machine learning model will detect these subtle differences.

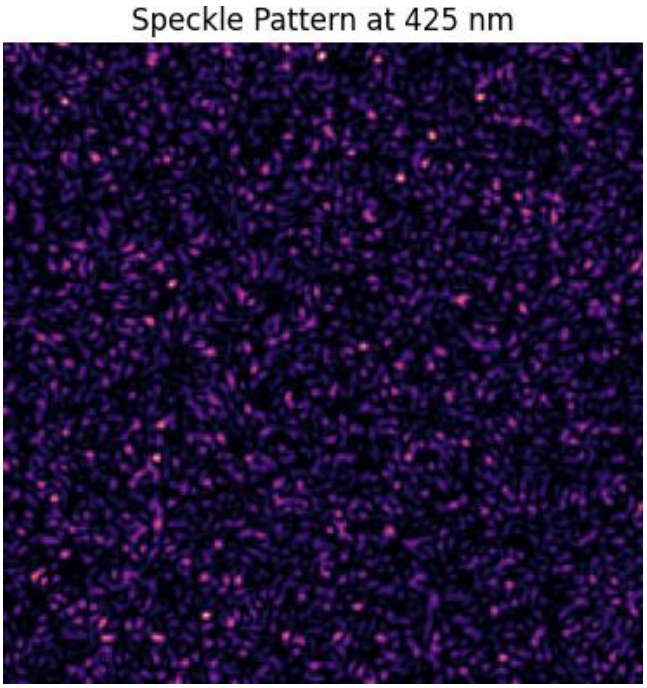


Figure 2. The intensity field of a speckle pattern at 425nm. This image was cropped from 2000x2000 pixels to 200x200 pixels centered to increase feature size, but is an accurate representation of what the image could look like at any cropping location.

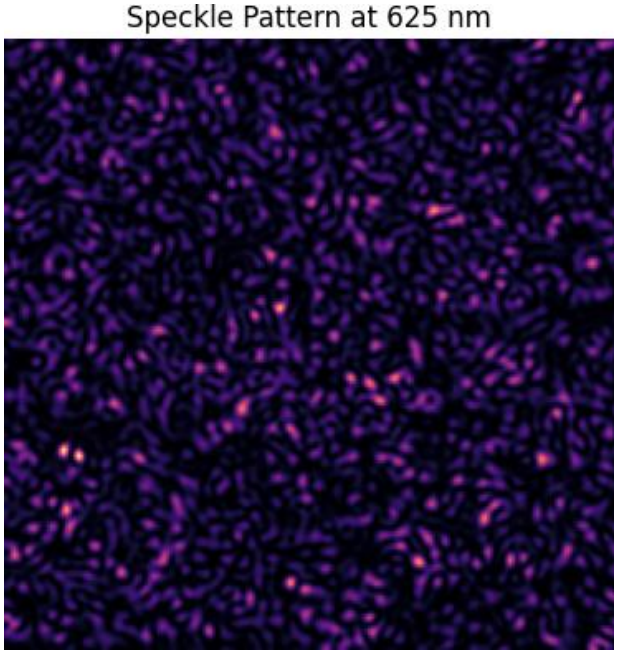


Figure 3. The intensity field of a speckle pattern at 425nm. This image was cropped from 2000x2000 pixels to 200x200 pixels centered to increase feature size, but is an accurate representation of what the image could look like at any cropping location.

However, before the dataset was finalized, the total size had to be reduced due to hardware constraints and expected training performance. A single speckle image requires 8.52MB on disk, and with 5120, that equates to approximately 42.6GB of storage required. Therefore, when the simulation ran, each time a new set of 256 speckle patterns was generated during one iteration, the speckle patterns were run through a Principal Component Analysis (PCA) function to reduce the size of each 2000x2000 image to 144 principal components. PCA is a linear dimensionality reduction function that reduces the size of a data structure to a set of numbers while preserving as much statistical data as possible [5]. After performing this function, the data on that iteration was converted from 256 speckle patterns of the original size to 256 speckle patterns of size 12 by 12 pixels, which no longer visually were interpretable but statistically were. At the end of the data generation process, the 5120 PCA'd speckle patterns only required 5.62MB of disk space..

### *2.2 Data Pre-Processing for Machine Learning*

Starting off in the machine learning program, the dataset was loaded into the program from a local file. It was then flattened to eliminate one dimension that represented what iteration the data was generated on, so that the first dimension of the dataset was now an index between 0 and 5119, with each index representing the speckle pattern contained in the latter dimensions. The speckle data was then normalized to be between 0 and 1 to aid in the training process. Next, a set of labels was created to be between 0 and 255 and matched to the flattened speckle data. Initially, labeling was marked as the true speckle wavelength, but performance was found to be improved using a 0 index and later adding a 400nm numerical offset when it came to analyzing the output data and performance. Next, the data was split into 80% for the training set, 10% for the validation set, and 10% for the testing set. The numerical counts for these data sets are 4096, 512, and 512 respectively. The split was done using the StratifiedShuffleSplit to ensure that each set had properly distributed data labels, given that the data started off in a uniform format. The StratifiedShuffleSplit randomly distributes the data between sets, while attempting to maintain uniform distributions of labels in each set [6]. For this project, a hardcoded seed of 42 was used in the StratifiedShuffleSplit to keep testing consistent as it progressed.

### *2.3 Measure of Success*

Given that this is a classification problem on labeled images, the evaluation metric was chosen to be Mean Accuracy Error (MAE). In this context, as the model trains to reduce this metric, when it reaches the final state and predicts the wavelength of speckle patterns in the test set, those predictions would have a mean difference of this value. For example, with the goal of 5nm, the absolute value difference between the true and predicted wavelength value would be on average 5nm. For the loss metric, the metric was chosen to be Mean Squared Error (MSE), which calculates the mean square difference of the predicted values compared to the true value. The squaring in this metric more heavily penalizes larger errors so the model can learn what not to do faster.

## 3. Methodology

### *3.1 Model Parameters*

The three different models featured in the project share many of the same parameters and only differ in model structure. Firstly, the optimizer used is the adaptive moment estimation, or Adam optimizer. This optimizer was chosen as it is a modern and computationally efficient adaptive learning rate algorithm [7]. Next, the learning rate was set to start at 0.001, and fitted with a callback function to incrementally reduce the learning rate down to 0.00001 each time a plateau in learning was reached. The number of epochs was set to be 200, which was initially set to a smaller number, but during training it was found that the model still did not plateau on smaller epoch counts and a higher epoch count benefited the training process. As mentioned earlier, the metrics for loss and accuracy were MSE and MAE respectively.

### *3.2 Basic Model*

The first step in forming a model was to form a basic model that fits the dataset. Three convolution layers form the core of the model, with a single MaxPooling2D function in between the first and second layer, and second and third layer. To bridge the gap between the convolutional layers and the output, the output of the convolution layers is flattened and passed to a dense layer before reaching the final output layer. A visual representation of this model can be seen in Figure 4.

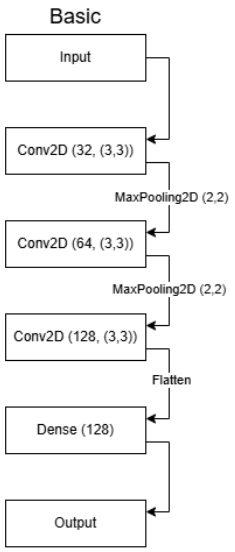


Figure 4. A flow chart style representation of the basic CNN model featured in this project.

During training of this model, it was observed that even with the increased epoch value, sometimes the model would plateau at a high validation MAE value and would not improve. To reduce the inconsistency of the final model, the training process was run 10 times and the model with the best validation MAE was saved. This looping process was continued for the remaining models. Evidence of this inconsistency can be seen in Figure 5.

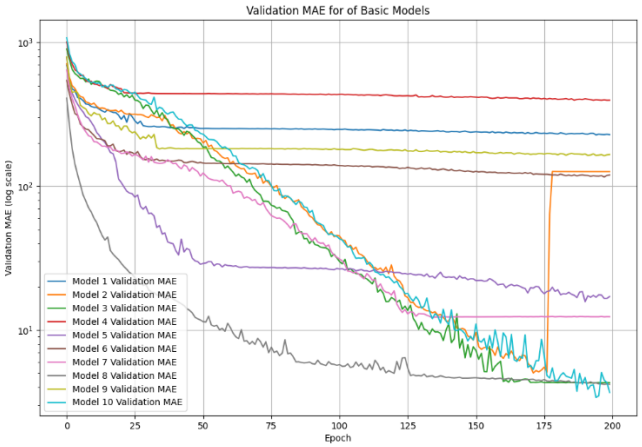


Figure 5. A plot showing training improvements the basic model experienced as the epochs increased. Ten iterations of training are shown to highlight the inconsistencies in training this model.

### *3.3 Overfit Model*

The next step in the model design was to create an overfitting model that would be expected to excel at the validation dataset it was trained on, but perform worse on the test set due to overfitting. To achieve this, each layer of convolution was directly doubled, and the number of filters in each layer was also doubled. A visual representation of this model can be seen in Figure 6.

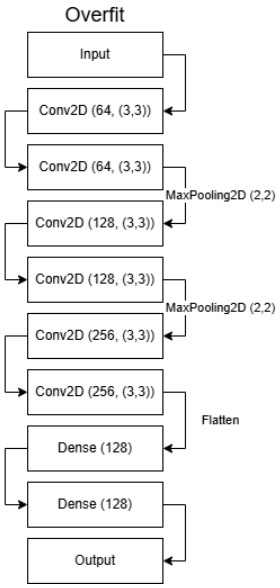


Figure 6. A flow chart style representation of the overfitting CNN model featured in this project.

During the training of this model, it was observed that this model achieved an MAE of 1.13 on the training data, and 3.78 on the validation data. This proves that the model had

overfitted to the training data as intended. A visual representation of the training MAE and validation MAE over epoch count can be seen in Figure 7.

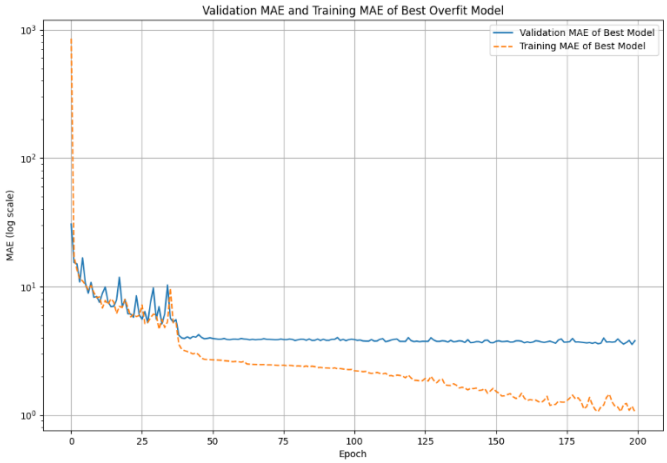


Figure 7. A plot of the training MAE and validation MAE of the best overfitting model as it trained. The noticeable plateau in the validation MAE compared to the dropping training MAE indicates overfitting occurred.

A chart of the model history shown in Figure 8 shows that the model improved more rapidly compared to the basic model, and had more consistency between runs.

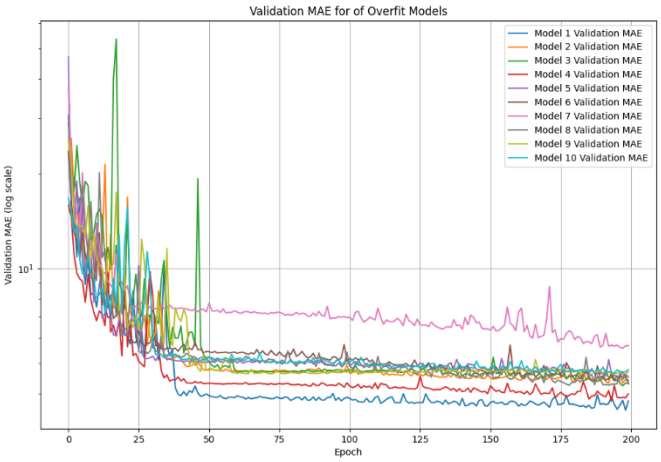


Figure 8. A plot showing training improvements the overfitting model experienced as the epochs increased. The plot shows the model plateaus sooner and is more consistent then the basic model.

### *3.4 Regularization Model*

The last model created was the regularization model, which ended up looking similar to the basic model with extra regularization steps in between layers to prevent overfitting and to generalize the model to the data. The two regularization features implemented were the BatchNormalization function and the Dropout function. The BatchNormalization function normalized the output from the previous layer using the mean and standard deviation, and then scales and shifts it back to a learnable format. Dropout sets a percentage of the input data to zero, forcing the model to not rely on neurons that may have been overtrained on that information. For this model, a dropout value of 0.25, or 25%, was found to be most useful, and was used in the final model. A visual representation of this model can be seen in Figure 9.

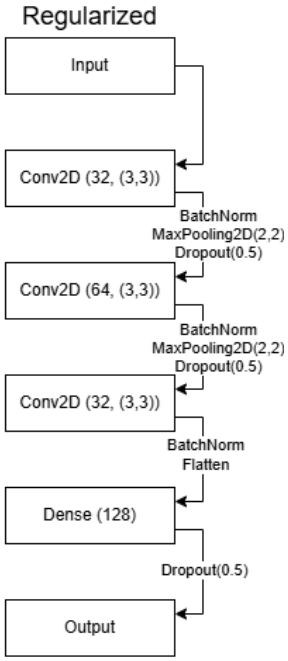


Figure 9. A flow chart style representation of the overfitting CNN model featured in this project.

The training history also showed that the regularized model had the same rapid training rate at the beginning, but leveled off at a much smoother rate compared to the overfitting model. The training history for the regularized model is shown in Figure 10.

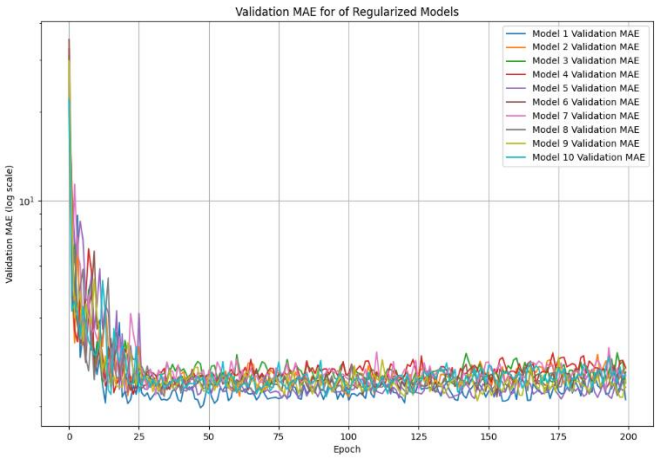


Figure 10. A plot showing training improvements the overfitting model experienced as the epochs increased. The plots shows the model plateaus to a similar level as overfitting model, and is more consistent between iterations.

## 4. Results and Analysis

### *4.1 Basic Model Results*

The basic model yielded promising results, achieving a Test MAE of 3.42 and Test Loss of 22.15. The normalized confusion matrix for this model is shown in Figure 11.

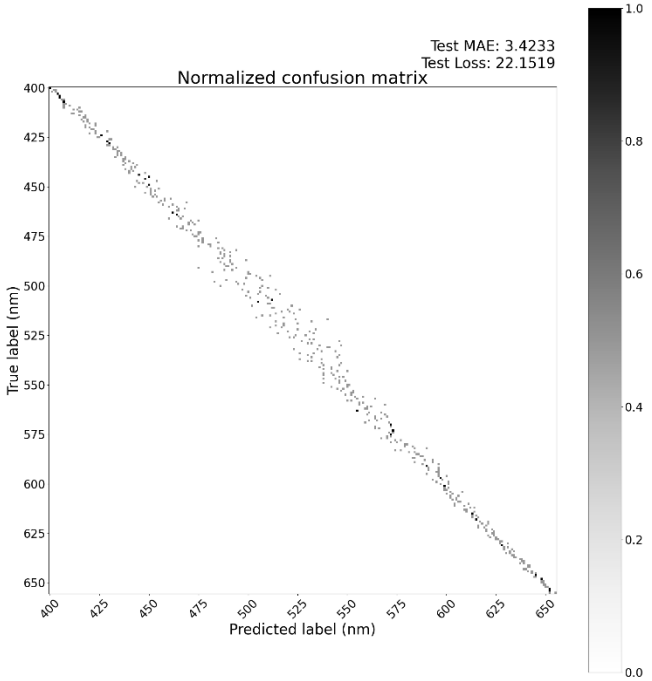


Figure 11. A confusion matrix for the basic model as it is evaluated against the test set.

From the confusion matrix, it is inferred that this model performs reasonably well at all of the wavelengths in the dataset, showing slightly better results as the wavelength increased.

### *4.2 Overfit Model Results*

The overfitting model performed slightly worse than the basic model when evaluated on the test set, yielding a MAE of 3.79 and a Loss of 26.63. The normalized confusion matrix is shown in Figure 12.

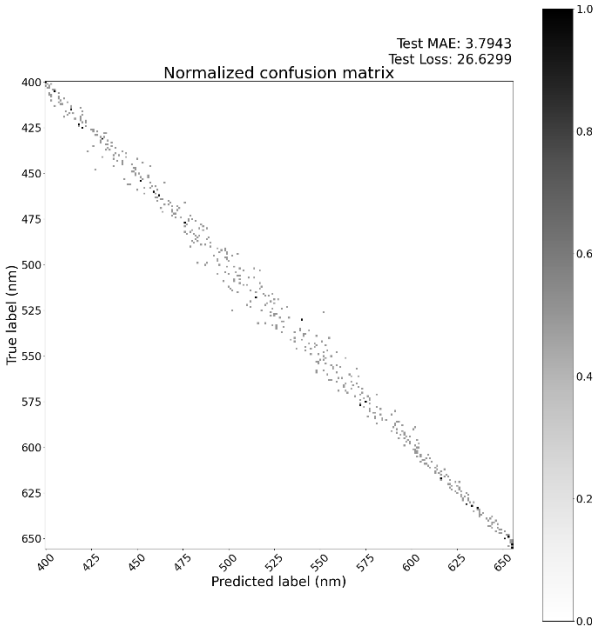


Figure 12. A confusion matrix for the overfitting model as it is evaluated against the test set.

The confusion matrix looks very similar to that of the basic model, with the same general shape, and the outliers looking slightly more exaggerated when compared to those of the basic model.

### *4.3 Regularized Model Results*

The regularized model performed the best, achieving a MAE of 2.28 and a Loss of 11.84. The confusion matrix for this model is seen in Figure 13.

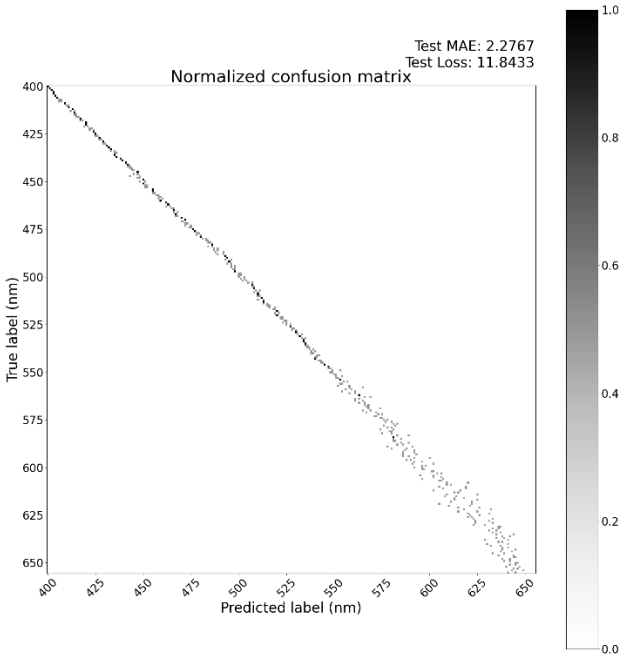


Figure 13. A confusion matrix for the regularized model as it is evaluated against the test set.

The regularized model shows an interesting change in the confusion matrix when compared to the basic and overfit models. In this instance, the model is very accurate up until around 550nm, at which point it starts to struggle to predict an accurate result. This is unlike the basic and overfit models, which struggled more in the center wavelengths in the set.

### *4.4 Analysis into the PCA Function*

With the results of the training process, additional information on the effectiveness of the PCA function was investigated. Using the data from the beginning of the methodology, where it was still separated by iterations, a correlation test was performed. In this test, the last iteration of speckle patterns was treated as the unknown, and the rest of the data was treated as known data. For every wavelength in the unknown dataset, each speckle pattern in the known dataset performed an individual statistical correlation against it, and the highest correlated wavelength in the known dataset was recorded. This generated the confusion matrix seen in Figure 14.

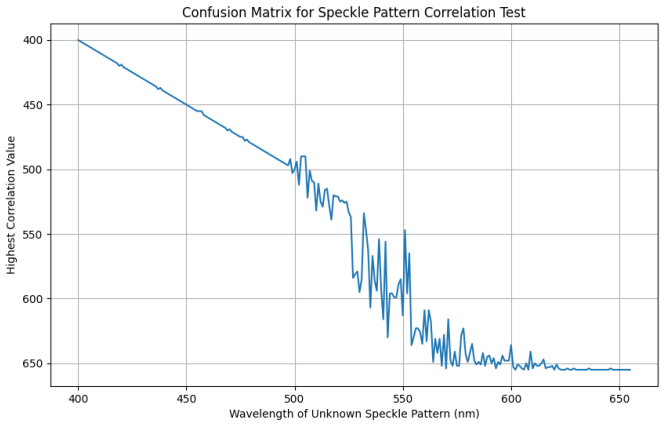


Figure 14. The results of a running a correlation test on the PCA data before it is processed for machine learning. The plot indicates that the PCA process loses statistical information as the wavelength increases.

This test yielded interesting results. In this chart, it is clearly visible that for the wavelengths from 400nm to 500nm, a correlation between unknown data and the known data is found. However, when the wavelength increases beyond that point, a strong correlation is no longer found. This indicates that the PCA function is starting to lose statistical information, which is not as apparent in the machine learning model. This indicates that the model is still able to find some features that allow it to correlate speckle patterns, even after they have been reduced in size by the PCA function.

### *4.5 Analysis on the Impact of Longer Wavelengths*

One trend found in both the regularized machine learning model and the correlation test is that longer wavelengths are more difficult to distinguish from each other compared to shorter wavelengths. One reason this could occur is the oversampling of longer wavelengths during the simulation process. As it was mentioned earlier, the sampling rate at the detector in the simulation maintained a constant sampling rate regardless of the wavelength being used. The result of this feature means that while wavelengths closer to 400nm are sampled at a near-ideal cutoff rate, the longer wavelengths are sampled at rates larger than their cutoff frequencies. This oversampling means that slight noise is introduced as the resolution of the detector exceeds the minimum resolution needed to reconstruct the input signal, leading to additional information being inserted at each pixel that may or may not be helpful in reverse engineering the simulation.

## 5. Conclusion and Future Work

### *5.1 Results Conclusion*

In conclusion, the machine learning approach proved to be a viable contender to extract the wavelength of a speckle pattern, even after it had gone through the PCA process. With the goal of 5nm at the beginning, the basic, overfitting, and regularized models achieved a MAE of 3.71nm, 3.86nm, and 2.39nm respectively. All three models achieved the goal, but ultimately the regularized model proved the most robust and most consistent, as seen in the training history. However, while the regularized model excelled at predicting wavelengths in the first and second thirds portions of the dataset, it struggled to predict the longer wavelengths in the last third of the dataset.

### *5.2 Future Work*

Future work for this project starts off with improvements to the hardware used to generate the speckle patterns. Specifically, the PCA function with 144 components used around 34GB of RAM when converting 256 raw speckle patterns to their PCA equivalents, which was the maximum the computer used in this project could handle. An increase in RAM would allow for a higher principal component count, which may reduce the strange behavior documented in Section 4.4.

Another way to improve the training time would be to adjust the epoch count based on the model's performance. For the regularized model, it is evident in the plot of the training history that the model starts to plateau after around 25 epochs. For this project, the epoch count was kept at 200 throughout all testing to allow the models to reach their plateau, but further testing could justify reducing the epoch count without compromising performance, especially if additional data are used to train the model.

Additionally, increasing the amount of data may provide better insight into how the model behaves when analyzing the confusion matrices generated by the different models. Given that the test set consisted of 512 samples, and there are 256 classes to be categorized, the ratio between test samples and classes is relatively low.

Lastly, additional research should be performed to evaluate if the usage of training data or machine learning models is necessary to solve this problem. Going back to the original speckle patterns shown, if an additional 2D Fourier Transform is applied to the speckle pattern, and a slice of that resulting data is plotted, an interesting plot in the frequency domain is formed. These plots are shown in Figure 15 and Figure 16.

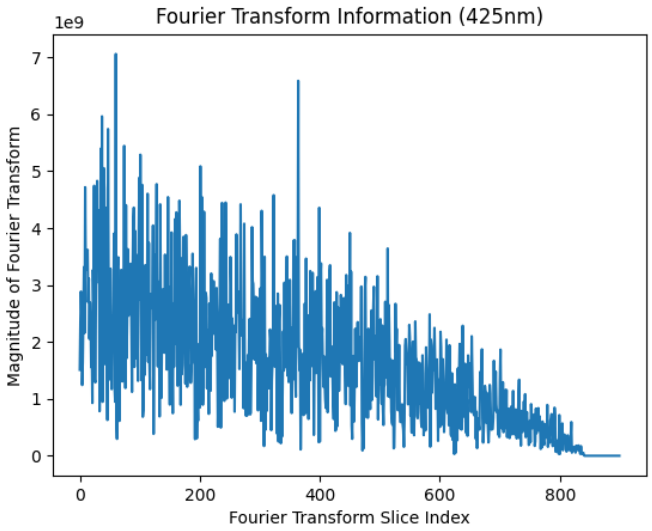


**Figure 15.** A plot of the center slice of an original speckle pattern at 425nm after it experiences an additional 2D Fourier Transform.

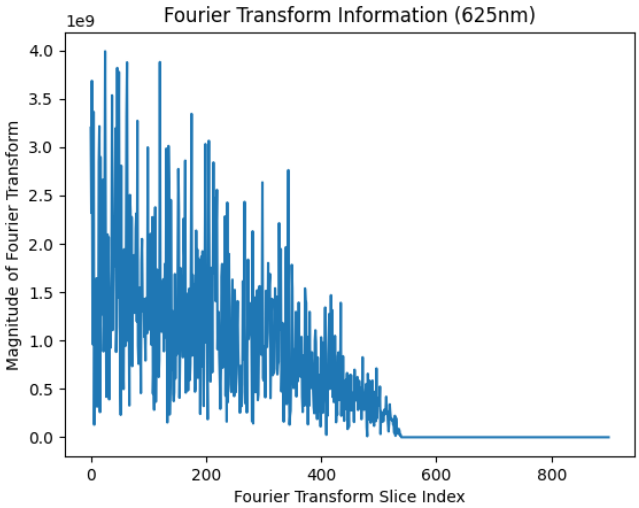


Figure 16. A plot of the center slice of an original speckle pattern at 625nm after it experiences an additional 2D Fourier Transform.

For the scope of the project, the axes are left in a vague format, since the purpose of these charts is to demonstrate the possibility of using this methodology. In these charts, the only difference is the wavelength used to generate the speckle pattern, and yet they show a clear indication that in the original speckle pattern, the frequency domain experiences a cutoff that varies with that wavelength. In reality, it would not be expected to have a cutoff to zero present, since real data would include a multitude of noise. However, if it were possible to develop an algorithm to detect this drop-off in real data, that would prove that training data or machine learning is not required to solve this problem. Additionally, since a solution along these lines would rely on the optical fundamentals of the

system, it should theoretically be more robust to the environment the data is collected in, whereas the machine learning model can only be applied to data gathered in the same environment as its training data. Such a solution only seems theoretically possible, and more research is required to determine its viability.

## References


[1] "World Population Review," Military satellites by country 2024, https://worldpopulationreview.com/country-rankings/military-satellite-by-country (accessed Mar. 13, 2025).

[2] "CRS reports," Defense Primer: Directed-Energy Weapons, https://crsreports.congress.gov/product/pdf/IF/IF11882#:~:text=China%20%5Bhas%20likely%20fielded%5D%20a,structures%20of%20non%2Doptical%20satellites. (accessed Mar. 13, 2025).

[3] "Kurios® Liquid Crystal Tunable Bandpass Filters." Available: https://www.thorlabs.com. [Accessed: Mar. 15, 2025]

[4] F. T. OPTICSTM et al., "Speckle pattern - introduction and applications," AZoOptics, https://www.azooptics.com/Article.aspx?ArticleID=742 (accessed Mar. 15, 2025)

[5] "PCA," scikit-learn. Available: https://scikit-learn/stable/modules/generated/sklearn.decomposition.PCA.html. [Accessed: Mar. 15, 2025]

[6] "StratifiedShuffleSplit," scikit-learn. Available: https://scikit-learn/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html. [Accessed: Mar. 16, 2025]

[7] A. Géron, Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems, Second edition. Beijing [China] ; Sebastopol, CA: O'Reilly Media, Inc, 2019.