Development of use-specific high-performance cyber-nanomaterial optical detectors by effective choice of machine learning algorithms

IOP Publishing Mach. Learn.: Sci. Technol. 1 (2020) 025007 https://doi.org/10.1088/2632-2153/ab8967

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
29 December 2019

REVISED
25 March 2020

ACCEPTED FOR PUBLICATION
15 April 2020

PUBLISHED
19 May 2020

PAPER

Development of use-specific high-performance cyber-nanomaterial optical detectors by effective choice of machine learning algorithms

Davoud Hejazi[1], **Shuangjun Liu**[2], **Amirreza Farnoosh**[2], **Sarah Ostadabbas**[2] **and Swastik Kar**[1]

[1] Department of Physics, Northeastern University, Boston, MA, United States of America
[2] Augmented Cognition Lab (ACLab), Electrical and Computer Engineering Department, Northeastern University, Boston, MA, United States of America

E-mail: ostadabbas@ece.neu.edu, s.kar@northeastern.edu

Keywords: 2D materials, artificial neural networks (ANN), Bayesian inference, k-nearest neighbor (kNN), optical wavelength estimation, support vector machine (SVM), transition metal dichalcogenides (TMDs)

Abstract

Due to their inherent variabilities, nanomaterials-based sensors are challenging to translate into real-world applications, where reliability and reproducibility are key. Machine learning can be a powerful approach for obtaining reliable inferences from data generated by such sensors. Here, we show that the best choice of ML algorithm in a cyber-nanomaterial detector is largely determined by the specific use-considerations, including accuracy, computational cost, speed, and resilience against drifts and long-term ageing effects. When sufficient data and computing resources are provided, the highest sensing accuracy can be achieved by the k-nearest neighbors (kNNs) and Bayesian inference algorithms, however, these algorithms can be computationally expensive for real-time applications. In contrast, artificial neural networks (ANNs) are computationally expensive to train (off-line), but they provide the fastest result under testing conditions (on-line) while remaining reasonably accurate. When access to data is limited, support vector machines (SVMs) can perform well even with small training sample sizes, while other algorithms show considerable reduction in accuracy if data is scarce, hence, setting a lower limit on the size of required training data. We also show by tracking and modeling the long-term drifts of the detector performance over a one year time-frame, it is possible to dramatically improve the predictive accuracy without any re-calibration. Our research shows for the first time that if the ML algorithm is chosen specific to the use-case, low-cost solution-processed cyber-nanomaterial detectors can be practically implemented under diverse operational requirements, despite their inherent variabilities.

1. Introduction

Nanomaterials are very attractive for building sensors, and various examples of using 2D nanomaterials, nano-tubes, quantum-dots, *etc*, can be found in the fabrication of optical detectors [1–3], molecular and bio-sensors [4–8], ion and radiation sensors [9], chemical sensors [10, 11], gas sensors [12, 13], temperature sensors [14] and many other cases of detection and sensing. There are many aspects that make nanomaterials promising candidates for these applications compared to the bulk materials. For instance, their enhanced optoelectronic and novel chemical/physical properties make them efficient choices for sensing, while their small dimensions will lead to devices with lower power consumption and smaller size. In many cases, nanomaterials are much more attractive than conventional semiconductor sensors due to their low-cost, earth-abundant availability, and compatibility with affordable solution-processable techniques. Their high surface-to-volume ratio makes them highly sensitive as chemical sensors, whereas their quantum confinement or excitonic processes enables them to be excellent target-specific photodetectors. As a result, over the past decades, there has been a tremendous progress in fundamental understanding and

proof-of-concept demonstrations of chemical, biological, optical, radiological and a variety of other sensors using nanomaterials [1–5, 7–16].

However, there exists many challenges in real-world implementation of sensors made from nanomaterials; above all the difficulties in reproducing them which makes the size and physical location of the fabricated nanomaterials on the substrate unpredictable and uncontrolable. Moreover, the nanomaterials undergo gradual decay in ambient condition called "drift", *i.e.* they are not very stable; also there is often a large noise in their measurement because of their small size due to the fact that nanomaterials not only respond to what they are designed to measure, but also are very sensitive to many other conditions in their environment. These shortcomings, not to mention the gradual decays of nanomaterials, have introduced huge challenges in mass production of reliable devices from them, where predictable and controllable manufacturing processes is essential to the industry.

In recent decades, the emergence of machine learning (ML) has demonstrated a great potential for enhancing statistical analysis in the field of material science. Nowadays, ML provides popular tools for obtaining information from internet of things (IoT) networks [17–21] such as charge-coupled devices (CCDs)[22, 23], complementary metal-oxide-semiconductor (CMOS) detectors [24–26], or regular silicon-based spectrometers, which are examples of sophisticated networks of optical detectors [27, 28]. Advances in deep learning (DL) have paved the way to understand ultra-complex problems beyond physics and engineering where, for instance, by fine-tuning convolutional neural networks parameters on RNA-sequencing and pharmaceutical big data, scientists are trying to model cancer.[29–31] In physics, on one hand, people employ machine learning to analyze, predict, or interpret physical quantities; on the other hand, underlying physical principle has also been employed to facilitate designing effective machine learning tools [32, 33]. ML/DL methods have been successfully applied for accelerated discovery [34–36] and development of materials and metamaterials with targeted properties [37–43], predicting chemical [44–47] and optoelectronic properties of materials [48–51], and synthesizing nanomaterials [52]. The variations in nanomaterial properties are usually considered as "noise" and various experimental or statistical approaches are often pursued to reduce these variations or to capture the useful target data from noisy measurements [53–57]. However, the direct applications of the data analytic approaches have never been sought on the variability of the nanomaterials themselves to utilize these variations as information instead of treating them as noise. In the context of sensing applications, one way to overcome the aforementioned challenges of nanomaterials is to use ML on a multitude of sensors in order to extract relevant response patterns towards achieving accurate, reliable, and reproducible sensing outcomes.

Recently we showed that Bayesian inference can be employed on engineered variability in layered nanomaterials-based optical transmission filters to determine optical wavelengths with ultra-high accuracy and precision. Our previous work (Hejazi *et al* 2019 [58]) demonstrated the power of using advanced data analytics on the measured data from a few uncontrolled low-cost, easy-to-fabricate mixed semiconducting nanomaterial thin films. Mixing of nanomaterial phases has been shown to have interesting mechanical properties.[59] In contrast, we exploited their optical properties in order to estimate the peak wavelength of any incoming monochromatic/near monochromatic light over the spectral range of 351–1100 *nm* with high precision and accuracy, and we created the world's first cyber-physical optical detector. In that work, we applied a Bayesian inference on optical transmittance data of 11 nanomaterial filters fabricated from two transition-metal dichalcogenides, MoS_2 and WS_2 (see figure 1). We were also able to reduce the number of filters to two filters via step-wise elimination of least useful filters and still achieve acceptable results even with two filters. We also discussed that it is possible to choose suitable materials for desired spectrum ranges for optical filter fabrication. In many practical applications, however, the sensing cost/speed and long-term reliability can be equal or more important considerations. Although various machine learning (ML) tools are frequently used on sensor and detector networks to address these considerations and dramatically enhance their functionalities, nonetheless, their effectiveness on nanomaterials-based sensors has not been explored.

In the present work, our aim is to augment our analytical tools by employing various ML techniques, compare their efficacy in color sensing, and finally choose the most suitable ML algorithm for color detection based on the application requirements. We note in doing so, it is important to discuss the data-analytical process of ML techniques within the context of nano-science datasets, so that they can be appropriately utilized in analyzing nano-science data of other types as well. Hence, we provide below a brief outline, using schematic visualizations, of how different ML approaches are analyzing our data. In our target application in this paper, labels are wavelengths that combined with measured transmittance values that we will call filter readings, create the set of sample-label pairs known as the training set. Therefore, we chose our analytical approaches based on the supervised ML algorithms. Apart from the Bayesian inference, we employed k-nearest neighbour (kNN), artificial neural network (ANN), and support vector machine (SVM); the details of each method can be found in the Computational Details section. In the following discussions, we provide a brief overview of each method to clarify their algorithmic steps.

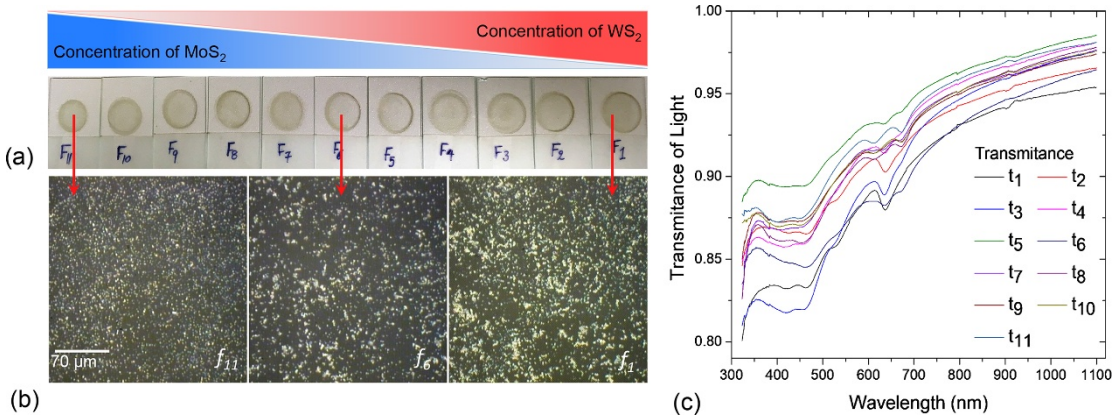

Figure 1. (a) Eleven filters drop-casted on glass slides; f_1 is 100% WS_2, but $f_2, \ldots, f_{10}$ are made by gradually adding MoS_2 and decreasing WS_2, and finally f_{11} is 100% MoS_2. (b) Microscopic image of three filters f_1, f_6, f_{11}: nanomaterials on glass substrate. (c) Background-subtracted transmittance *vs.* wavelength t_1, ..., t_{11} for all 11 filters. The excitonic peaks get modified gradually from f_1 to f_{11} as a result of changing proportion of mixing two TMDs. (The figure is re-plotted from the original Hejazi *et al* 2019[58]).

As for the Bayesian inference, we discussed its underlying statistical approach in details in our previous article [58]. For a given set of known sample-label pairs (i.e. training set), Bayesian inference gathers statistics of the data and uses them later to classify an unknown new sample by maximizing the collective probability of the new sample belonging to corresponding category (see figure 2(a)).

In pattern recognition, the kNN is a non-parametric supervised learning algorithm, where, a new sample is classified into a specific category, if in average, that category's members have smallest distance from the unknown sample (see figure 2(b)).

ANNs are computing models that are inspired by, but not necessarily identical to, the biological neural networks, where, models "learn" to perform tasks by considering samples, generally without being programmed with any task-specific rules. In this work we have used two different fully-connected ANN architectures to investigate their efficacy on optical wavelength estimation (see Computational Details' section). The schematics of a three layered fully-connected ANN model is shown in figure 2(c).

When it comes to supervised classification, SVM algorithms are among the powerful ML inference models [60–62]. In its primary format as a non-probabilistic binary linear classifier, given labeled training data, SVM outputs an optimal hyperplane which categorizes new examples into two classes (see figure 2(d)). This hyperplane is learned based on the "maximum margin" concept in which it divides the (training) examples of separate categories by a clear gap that is as wide as possible [63, 64].

In real-world sensing and other "estimation" applications, the needs (*i.e.* speed, accuracy, low-complexity etc) of the end-use should determine the approach or method. Keeping these in mind, we have compared the efficacy of these ML technique by considering the following main considerations: (a) The average error in estimating wavelength of test samples collected at the same time the training samples were collected; (b) The average absolute error for entire spectrum; (c) The required time for training; (d) The elapsed time for estimating wavelength of one test sample using model/trained parameters; (e) The effect of reducing the training set size on efficacy of each model; and (f) How well the models behave on new set of test samples collected several months after the training. Applying these four ML techniques to our wavelength estimation problem has revealed important facts about their efficacy. The kNN algorithm appears to perform the best in terms of the estimation accuracy, however unlike the other three techniques, kNN time complexity is directly proportional to the size of the training set, which will hinder its use in applications that demand real-time implementation. It is due to the fact that kNN is a non-parametric algorithm, in which the model parameters actually grows with the training set size. k should be considered as hyper-parameter in kNN. On the other hand, ANN models perform fastest in the test time, since all of the model parameters in ANNs are learned from the data during the training time, and the test time is only the classification step, which is simply calculating the output value of an already-learned function. An interesting observation from our results is that the SVM model shows slightly larger estimation errors compared to the rest of the algorithms, however it is not sensitive to data size and is more resistant to time-dependent variations in optoelectronic response of nanomaterials *i.e.* to drift. Bayesian inference turns out to be very accurate, and quite fast as well.

By looking at the outcomes of our estimation problem, we have also discovered another important aspect of the data that we are dealing with in the nanomaterial applications. We noticed a significant nanomaterial measurements drift over time in our dataset, which can be described as "evolving class distributions". This

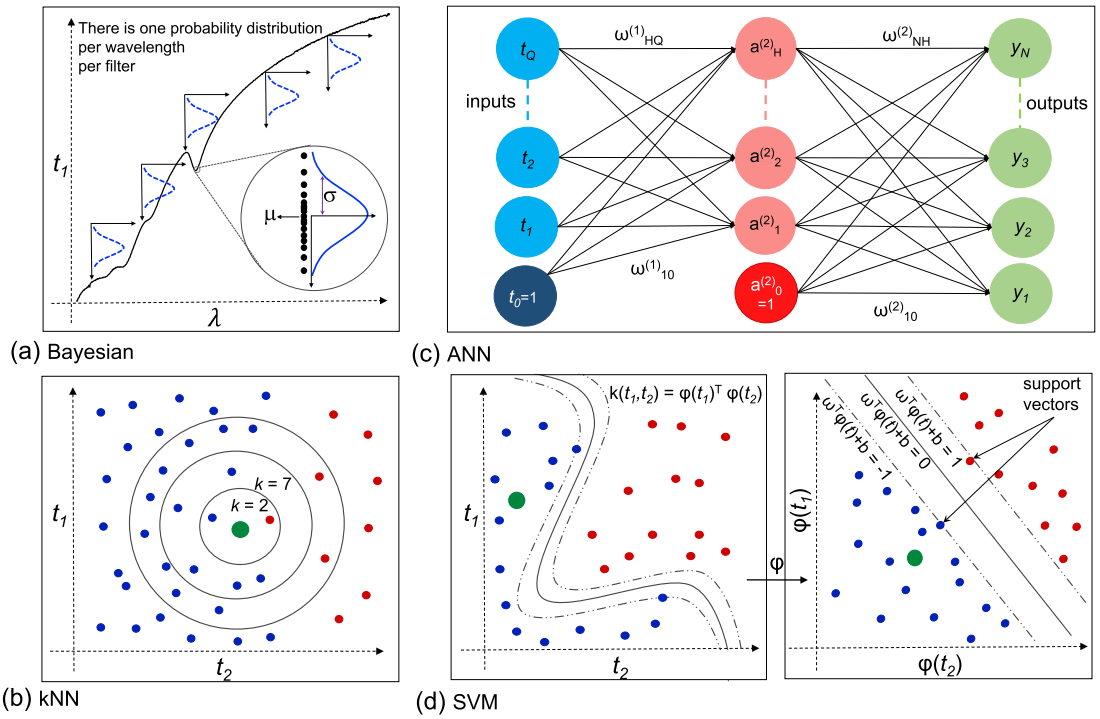

Figure 2. Schematic representations that distinguish the various analysis approaches used in this work: (a) The Bayesian inference, that shows at each wavelength over each filter a probability distribution can be formed from the labeled samples of that given wavelength. (b) The kNN algorithm. Each point represents a sample in an 11-dimensional space (transmittances $t_1, \ldots, t_{11}$), but only two dimensions are shown for convenience. Blue and red points represent samples belonging to two wavelengths that have close transmittance values. The unknown sample in green will be classified depending on the majority votes of the samples encircled in the circles depending on the number of the closest neighbors. With $k = 2$ the choice is not certain but with $k = 7$ the unknown sample obviously belongs to the class of Blue points. (c) The fully-connected three layered ANN model. Each neuron is connected to the neurons in previous layer via weight parameters that must be optimized for the model to correctly estimate the unknown samples class. The bias neurons (in dark blue and dark red) are not connected to previous layers since they are by definition equal to +1. (d) The non-linear SVM algorithm, where the wavelength classes are the same as (b) and the gray solid line draws the barrier between the two classes. The dashed lines indicate the margins. By doing a kernel trick we can transform the data from feature space t to its dual space $\phi(t)$.

means the same object (*i.e.* light ray) will not create the same responses on the nanomaterial filters over time. Therefore, a model trained on a training set may have completely different parameter values compared to the same model trained on another training set collected after a period of time (*e.g.* a couple of months in our case). In attempt to overcome the shifts in the data due to the drift in electronic and spectral transmittance of nanomaterials, we show that it is possible to model the drift of nanomaterial responses over time and combine them with the future estimations where the nanomaterial filters have drifted even more. By observing the transmittance of filters over the period of more than a year, we were able to predict the drift in transmittance after two months and improve the performance in the wavelength estimation. This was however only possible in the kNN and Bayesian algorithms since they employ no other parameters than the transmittance values themselves, while SVM and ANN train their own corresponding parameters. In the next section we will summarize the main findings of our work.

2. Results and discussion

The detailed description of each ML algorithm, the number of parameters to be trained, and the computational complexity of each technique will be discussed in the Computational Details section. The resolution of the collected wavelength samples is 1 *nm*. To discuss the efficacy of our wavelength estimators, we define the estimation error percent as $error\% = \frac{|\lambda_{Groundtruth}(nm) - \lambda_{Estimated}(nm)|}{\lambda_{Groundtruth}(nm)} \times 100$.

We first present our results comparing the wavelength estimation accuracy from various techniques. Figure 3 shows a comparison of wavelength estimation by different ML techniques performed using the same set of training data comprised of 75,000 samples. The average errors are the average of error percentages for 10 test samples for each wavelength. By comparing the overall values of the average error as a function of wavelength, it is possible to see that the kNN method appears to best estimate the wavelength of an unknown light source, followed by the Bayesian inference method, when the estimation conditions (time, number of filters used, training size etc) are not constrained to low values. The ANN and SVM are in the 3*rd* and 4*th*

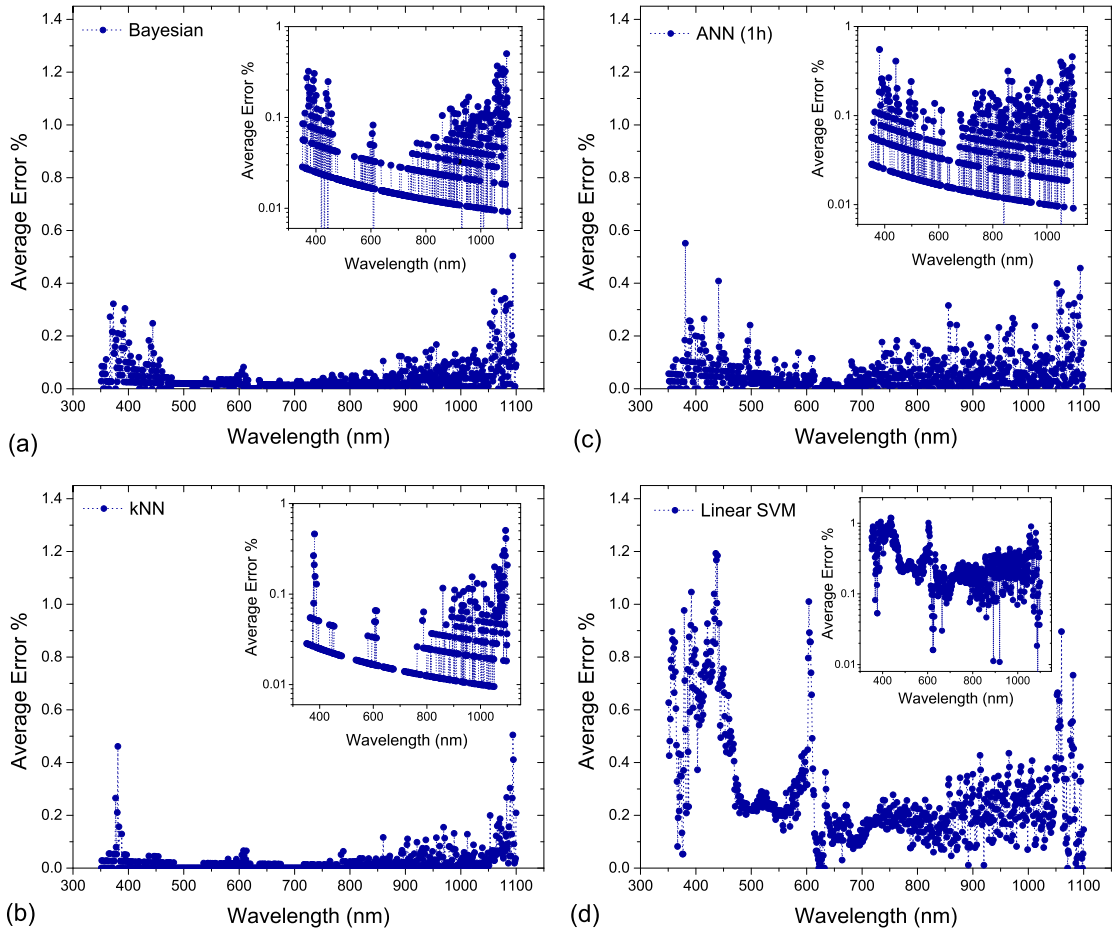

Figure 3. The percent error of estimating wavelength of test samples in a given wavelength averaged over 10 samples of the same wavelength; we define the estimation error percent as $error\% = \frac{|\lambda_{Groundtruth}(nm) - \lambda_{Estimated}(nm)|}{\lambda_{Groundtruth}(nm)} \times 100$. (a) Bayesian inference; (b) kNN algorithm; (c) ANN algorithm; and (d) Linear SVM, using all 11 filters. The insets are semi-log plot of the same figures. The *y*-axis of inset plots have been limited by cutting of the values that are too close to 0, for better visibility, and consequently these points do not show up in semi-log plot.

place in overall performance on estimating wavelength of test samples. In order to perform a more quantifiable comparison between the various approaches, we have calculated the average absolute error of entire spectrum by calculating the absolute error ($| \lambda_{Groundtruth}(nm) - \lambda_{Estimated}(nm) |$) for all 7,500 test samples and averaging them (see figure 4(a)). In addition, we have performed the AAN using both 1 and 2 hidden layers, which has been presented in the comparison data shown in figure 4 and subsequent figures, where we can see a fifth batch of columns for 2 hidden layer ANN shown with AAN(2 h) as opposed to ANN(1 h) with 1 hidden layer.

To investigate the sensitivity of the models to the size of the training set, we randomly picked different portions of the training set to perform the training and testing, *i.e.* by randomly choosing $\frac{1}{5}$, $\frac{2}{5}$, *etc* of the original dataset (see figure 4(a)). As it was expected from theory, the SVM model is least sensitive to the size of training data, followed by the Bayesian inference. However, the ANN and kNN show considerable reduction in performance by reducing the training set size. We can see that figure 4(a) where SVM shows minor changes from one data size to other, while for instance 1 hidden layer ANN shows steep change in error values. Another important fact that we learn from this figure is the minimum size of training set required to perform reasonable estimation. As we see, in each case using only $\frac{1}{5}$ of the training set, the average absolute error tends to increase considerably. The $\frac{1}{5}th$ translates to 20 times the number of different classes (wavelengths in our case), which sets a lower bound on size of the training dataset that must be collected. Another non-trivial and highly interesting observation is the relative errors of 1 hidden layer *vs*. 2 hidden layer ANN model as the error in wavelength estimation rises more sharply with decreasing training sets in the 1 hidden layer ANN, suggesting that ANN with more hidden layers appears to ”learn” better from the available data and yield more accurate estimations. The other consideration is the available data is not exactly enough for this problem even when all data is used. This can be justified by seeing that even from going from $\frac{4}{5}$ to all of the data there is a noticeable change in overall accuracy, while we expect to see minor

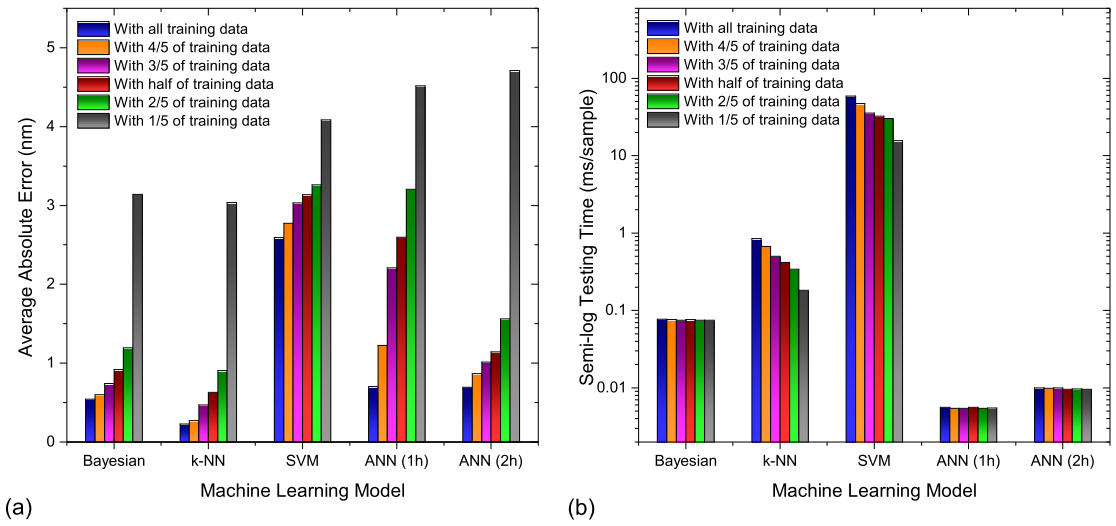

Figure 4. (a) Average absolute error calculated by averaging $abs.err = |\lambda_{Real}(nm) - \lambda_{Estimated}(nm)|$ of all 7500 test samples when different sizes of randomly chosen training data are used for training the model, performed with each of the 4 machine learning methods; (b) The semi-log plot of required time to test each new sample using the trained models after all training steps are completed.

change in accuracy of each model if the supplied data was sufficient. However, collecting large amounts of data in some applications might be expensive and time-consuming some experimental applications, and the user may look for the best ML model regardless of the testing time. In those cases, if we ignore the drift in nanomaterial properties, then the kNN and Bayesian models would give good results with small data sizes; but if the data is scarce and the nanomaterial filters are going to be used for longer periods without re-training them, then SVM model can work the best. We note that for more advanced applications such as multi-wavelength detection, more advanced ML/DL algorithms and larger datasets will be required.

We next analyze the performance of each algorithm in terms of the required time for each model to train, and afterwards to test. In kNN and Bayesian models there are no real learning steps, and as a result there is a definitive answer for value of a test sample with a given training set. The kNN model calculates the distance of the test sample from every training samples, which are fixed; so the testing time is directly related to the size of the training set. Given our relatively small dataset the kNN model works rather fast, but most likely it would not be the case if larger dataset were used (see figure 4(b)). As for Bayesian algorithm, the training part is limited to collecting the statistics from training data. In testing step, the model searches through all probability distributions and maximizes the *posteriori*; though it is obviously time consuming but is independent from the training set size. Hence, in both models the main and/or only required time is for testing.

As for ANN and SVM the training step can be dynamically decided by desired conditions. In the case of SVM, the training step is governed by choice of tolerance, kernel type, *etc*. After the support vectors are found, the testing step is carried out by checking which side of the hyperplanes the test sample falls. In our study different choices of kernel/tolerance did not pose meaningful enhancement on the estimation efficacy of the trained SVM models (see Computational Details section).

The situation is quite different for ANN, since one can iterate the training loop infinite times and the results may either improve, converge, or just get stuck in a local minima. Time and computational resources for training are the real costs of the ANN algorithm, but in general ANN can fit very complicated non-linear functions that other models might not have as good performance as ANN. After the end of the training step (decided by the experimenter based on the desired level of accuracy), the testing step is basically a few matrix multiplications only, as explained in Computational Details section. Hence, the testing time of ANN is quite short and independent from the size of training set. In addition, we found that with smaller training sets the ANN model is prone to over-fitting, *i.e*. the model might perform well on the training set itself but not on new test set. The required testing time for each sample when all training steps are completed is shown in figure 4(b), which is the more relevant time-scale for real-world applications. The details of each model and their computational complexity is discussed in Computational Details section.

Owing to their affinity for adsorbing oxygen, and moisture, as well as through creation of defects with exposure to ambient conditions, the electronic and optical properties of the nanomaterials gradually evolve with time, which are reflected in drifts in their spectral transmittance values. Hence, even though they show

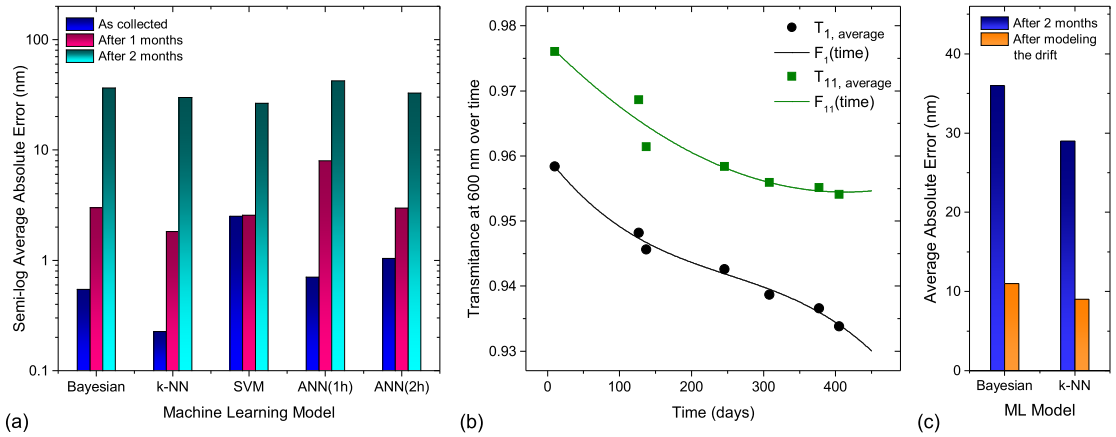

Figure 5. (a) Average absolute error of estimation in semi-log scale using all training data for test samples collected at the same time as the training data, compared to the test samples collected after one and two months; (b) Third order polynomial functions fitted to the average transmittance of filters f_1 and f_{11} over period of ~ 400 days. Scatter plots are the average measured transmittance values and the solid lines indicated the fitted functions; (c) Average absolute error of estimating wavelength of test samples collected two months after training when no modification is applied to the model (Blue bars), and when the training sample-label pairs are corrected using the drift functions (Orange bars) in Bayesian and kNN models.

fair stability in short period of time, the effective transmittance of our nanomaterial filters slowly change over time. This causes slow reduction of accuracy in estimating wavelength over time in later measurements. However, as shown in our previous work [15, 58] by calibrating the filters from time to time it would be possible to continue using these same filters over extended periods of time, and the efficacy of estimations does not suffer from wears or minor scratches, since the re-calibration will overcome the gradual changes of the filters.

In the current work we instead investigate the performance of each ML method over time by testing the efficacy of the trained models on the new test samples collected after two months. The average absolute error (with all training data) of estimating wavelength of newly collected test samples and original test samples are shown in figure 5(a). Quite interestingly, the SVM model shows minimal change in the estimation accuracy having the ratio of ~ 1 in first month, and smaller change later, while all other models show considerable reduction in accuracy. This change is quite obvious in a 1 hidden layered ANN. Next, we discuss how to overcome the effect of transmittance change of filters as a result of drift in optoelectronic response of nanomaterials by modeling the drift over time.

Choosing a proper ML technique that performs more robustly over time is only one way of using the filters over time without the need for re-calibration; but it is also possible to model the drift of nanomaterial. For this purpose we observed the transmittance change over time for our nanomaterial filters in a period of about 400 days, and tried to fit a polynomial curve to the average transmittance values at each wavelength for each filter with respect to number of days after the filters were fabricated. Two examples of these curves shown in figure 5(b) are for filters f_1 and f_{11} at 500 nm, that present the slow decrease of transmittance over time, where a third order polynomial function fairly fits the drift. To check the validity of our claim we calculated the expected transmittance at each wavelength for each filter around the day 450, which was the day that another new set of test samples were collected. In case of Bayesian, we replaced the mean value of transmittance by the calculated transmittance values at the day 450, while for kNN, we multiplied each transmittance t in training set by a corresponding coefficient $\frac{t_{avg}(450)}{t_{avg}(0)} \times t$, then used them for estimation (see figure 5(c)). Applying the drift over time functions is only possible for kNN and Bayesian algorithms since they do not have a distinct learning step, while the two other models have already trained their parameters based on the old training data. The results show that Bayesian model is more compatible with the drift over time function, which is expected since these fitting functions are calculated using the mean transmittance values, as Bayesian algorithm also uses mean values/standard deviations for estimation.

3. Conclusions

In conclusion, we have successfully demonstrated the efficacy of various ML techniques in estimating the wavelength of any narrow-band incident light in spectrum range 351–1100 *nm* with high accuracy using the optical transmittance information collected from a few low-cost nanomaterial filters that require minimal control in fabrication. With the available data the kNN algorithm shows highest accuracy with the average

estimation errors reaching to 0.2 *nm* over the entire 351–1100 *nm* spectrum range, where the training set is collected with 1 *nm* spectral resolution; but this method is not suitable for real-time applications since the required testing time is linearly proportional to the training set size. The situation is almost the same with the Bayesian algorithm which performs very well, but although its speed is not data size dependent, still the process is much slower than the other methods. The real-time speed considerations can be very well satisfied with ANN models where the estimation time can be as low as $10\mu s$, but these models as well as Bayesian and kNN turn out to be more sensitive to drift in spectral transmittance of nanomaterials over time. On the other hand SVM models show a bit lower accuracy compared to the rest but do not suffer from smaller data sizes and are more resilient to drift in spectral transmittance. Even though we have shown in our previous work that re-calibrating the filters will overcome the drifts and wears in nanomaterials, but if the re-calibration is not a readily available option for the user, the SVM model offers acceptable accuracy and longer usability over time. On the other hand if speed is a consideration the ANN models would be the best choice, which turn out to perform well if enough data is provided. We also observed that ANN models with more number of layers seems to learn better from the available data. The choice of model depends on the application; for instance spectroscopy does not demand a fast real-time output but accurate and precise estimations. There are other applications especially in biology, for instance in DNA sequencing [65], where the accuracy of the peak wavelength is not of importance as long as it is estimated close enough, but the time is of vital importance.

Furthermore, we have verified the possibility of modeling the drift of nanomaterials over time by observing the gradual changes in the filter functions, hence, being able to predict the filter function at later times, and thereby increase the accuracy of the ML algorithms and usability of the filters over longer periods of time. The efficacy of each ML model in our optical sensing problem reveals some key differences between this problem and other applications of ML in material science and engineering. The drift of nanomaterials properties for instance, which poses an important complication on the problem via evolving class distributions *i.e.* gradually modifying the response function of the filters even though the classes *i.e.* wavelengths remain the same. The other difference is in the feature selection. In optical sensing problem a very small number of features are chosen from optoelectronic properties (transmittance only in this case) of the nanomaterials, while in other areas the feature vector can be huge and very complex. The future work is to generalize the methods of this paper to broad-band optical spectra. All said, we believe that application of advanced data analytic algorithms has been very limited in optical sensing applications, and our findings can open up a new path for designing new generation optical detectors by harnessing advanced data analyzing algorithms/ ML techniques and significantly transform the field of high-accuracy sensing and detection using cyber-physical approaches.

4. Computational Details

When ML is used as a discriminative model in order to distinguish different categories (*e.g.* different optical wavelengths), it comes in one of these two forms: "supervised learning", where new samples are classified into N categories through training based on the existing sample-label pairs; and "unsupervised learning", where the labels are not available, and the algorithm tries to cluster samples of similar kind into their respective categories.

Data Structure. The analysis of our data were performed on transmittance values measured over a wide spectral range, 351 nm $< \lambda <$ 1100 nm) for each of the 11 nanomaterial filters, as well as 110 repetitions of these wavelength-dependent data. As mentioned in previous article, the repeated data was acquired to account for drifts, fluctuations, and other variations commonly observed in physical measurements especially in nanomaterial-based systems, which tend to be sensitive to their environments [15, 16, 58]. On the other hand larger training data usually results in better performance of most ML algorithms. From the mentioned 110 spectra of each nanomaterial filter, 100 of them were labeled as "training data" or sample-label pairs and used for training the models ($M = 750 \times 100 = 75000$ training samples). The other 10 spectra per filter were labeled as initial "test samples" ($M^{'} = 750 \times 10 = 7500$ test samples or original test samples), and were used only for testing the "trained" models. In another words the test samples were not part of the training process and the machine learning models did not "see" these samples until the testing step.

In our classification problem there are N different classes: one per wavelength, and we are trying to classify our transmittance data into these N classes. Here, we will concisely introduce each ML method and give their mathematical equations; also we will mention the number of parameters that are being trained in each model. Computations are carried out in Python 3.7 using a 2.5 GHz Quad-core Intel Core i7.

Bayesian Inference. The filters are not chemically independent from each other, for they are mixtures from different proportions of the same two nanomaterials; so for computational purposes we assume independence between their outcomes, and model them with Naive Bayes algorithm [58, 66]. The Bayesian inference for wavelength estimation problem can be formulated as follows: Let $\Lambda = \{\lambda_1, ..., \lambda_i, ..., \lambda_N\}$ be N

different wavelengths in desired spectral range and with specified granularity (*i.e.* 351–1100 *nm* with 1*nm* step in this study), and $T = \{t_1, ..., t_i, ..., t_Q\}$ be the transmittance vector of Q filter values (*i.e* $Q = 11$ when all of the filters are used in this study). Employing the Bayesian inference, the probability of the monochromatic light having the wavelength λ_j based on the observed/recorded transmittance vector T is called *posterior* probability

$$P(\lambda_j \mid T) = \frac{P(T \mid \lambda_j)P(\lambda_j)}{P(T)}, \tag{1}$$

which is the conditional probability of having wavelength λ_j given transmittance vector T; $P(\lambda_j) = \frac{1}{N}$ is the *prior* probability which is a uniform weight function here since all of the wavelengths are equally-likely to happen; N is the total number of quantifiable wavelengths in the range under study. Moreover, $P(T \mid \lambda_j) = \prod_{i=1}^{Q} P(t_i \mid \lambda_j)$ is the probability of observing transmittance data T given wavelength λ_j, and is called the *likelihood*, which is the probability of having transmittance vector T if wavelength is λ_j; $P(T)$ is the *marginal* probability which is the same for all possible hypotheses that are being considered, so acts as a normalization factor to keep the posterior probability in the range of 0 to 1.

Individual $P(t_i \mid \lambda_j)$ values are assumed to be Gaussian normal distributions for each filter at each wavelength, and their mean values and standard deviations were calculated from the training data (*i.e.* the 100 measured transmittance spectra) collected for each filter at each wavelength. Finally, given the measured transmittance sample T' (a vector of $Q = 11$ elements—one transmittance value per filter at an unknown wavelength), the wavelength λ^* of the unknown monochromatic light is estimated by choosing the value of λ_j that maximizes the posterior probability $P(\lambda_j \mid T')$:

$$\lambda^* = \arg\max_{\lambda_j} P(\lambda_j \mid T'), \tag{2}$$

This optimization called the maximum *a posteriori* (MAP) estimation [67–69]. To clarify the estimation steps further we notice the Bayesian inference finds probability of the combined Q measured test transmittance values named T' in the entire wavelength spectrum. According to MAP estimation the wavelength at which this probability is maximum is indeed the estimated wavelength in Bayesian inference. Though, from machine learning point of view no parameters are being learned in Bayesian inference, but considering the parameters of the Gaussian distribution that we calculate in this method we can say overall $2QN = 16500$ parameters are being learned in this approach, N mean values and N standard deviations from the training data (see figure 2(a)).

k-Nearest Neighbors. In pattern recognition, the kNN is a non-parametric supervised learning algorithm used for classification and regression [70], which searches through all known cases and classifies unknown new cases based on a similarity measure defined as a norm-based distance function (*e.g.* Euclidean distance or norm 2 distance). Basically, a new sample is classified into a specific category when in average that category's members have smallest distance from the unknown sample (see figure 2(b)). Here, k is the number closest cases to the unknown sample, and extra computation is needed to determine the best k value. This method can be very time-consuming if the data size (*i.e.* total number of known sample-label pairs) is large. There are two main categories for kNN: (1) centroid-based, which a new test sample is classified by the distance of its feature values with the average (*i.e.* centroid) of features of all training samples that belong to the same each class, and (2) by-instance-based, which is the standard kNN approach, in which a new case is classified by a majority vote of its neighbors, with the case being assigned to a class that is most common among its k nearest neighbors measured by a distance function. If k is 1, then the case is simply assigned to the class of its nearest neighbor. Since kNN model with small k is prone to over-fitting, usually a finite odd number is chosen for k. There are various kinds of distance functions which from them the four famous distance functions: *Euclidean*, *Manhattan*, *Chebyshev*, and *Minkowski* are used in this study, but only the results of *Euclidean* distance function is presented which is the classical presentation of distance and is given by $df(X, Y) = \sqrt{\sum (x_i - y_i)^2}$. Here, X refers to each sample in the training set and Y refers to the unknown (test) sample. To apply it to our data we need to find distance of a new transmittance vector of $Q = 11$ elements, $T' = \{t'_1, ..., t'_i, ..., t'_Q\}$, with all known transmittance vectors $T = \{t_1, ..., t_i, ..., t_Q\}$ that are already known and labeled in the training set, so the distance function is

$$df(T, T') = \sqrt{\sum_{i=1}^{Q} (t_i - t'_i)^2}. \tag{3}$$

The distance between T' and all M training samples is calculated, and the M calculated distance values are sorted from smallest to largest using a typical sorting algorithm. Afterwards, the k nearest neighbors *i.e.* wavelengths that have smallest distance values from the test T' are found, which are the arguments of the first k numbers of the sorted list. Each nearest neighbor is assigned a uniform weight of $1/k$, and the k neighbors are classified. Then, the test case T' is assigned to the group with largest vote or population. In order to find the best k for our system we tried different values for k in the range $k = [1, 20]$, and picked $k = 7$ which performed the best. As mentioned before, kNN is a non-parametric classification algorithm so, no parameters are being learned in kNN.

Artificial Neural Networks. ANNs are computing models that are inspired by, but not necessarily identical to, the biological neural networks. Such models learn to perform tasks by considering samples, generally without being programmed with any task-specific rules. An ANN is based on a collection of connected units or nodes called artificial neurons, that upon receiving a signal can process it and then pass the processed signal to the additional artificial neurons connected to them. A neural network has always an *input* layer that are the features of each training sample and an *output* layer that are the classes in classification problem, while it can also be only a number in regression problem. However, there are often more than just two layers in an ANN model. The extra layers that are always located between the input and output layers are called *hidden* layers. The number of hidden layers, the number of neurons in each layer, and how these layers are connected form the neural network architecture [71–73]. In general, having more number of hidden layers increases the capacity of the network to learn more details from the available dataset, but having much more layers than necessary can result in overfitting the model to the training set *i.e.* the model might be performing well on the training set but poorly on the unseen test set [72, 74]. In this work we have used two different fully-connected ANN architectures to investigate their efficacy on optical wavelength estimation. The reason behind using the classical structure of ANN *i.e.* its fully connected form, is the small dimensionality of our data compared to other datasets often used in ANN studies -such as long time-series, images, videos, etc The schematics of a three layered fully-connected ANN model is shown in figure 2(c). Backpropagation is the central mechanism by which a neural network learns. An ANN propagates the signal of the input data forward through its parameters called weights towards the moment of decision, and then backpropagates the information about error, in reverse through the network, so that it can alter the parameters. In order to train an ANN and find its parameters using the training set, we give labels to the output layer, and then use backpropagation to correct any mistakes which have been made until the training error becomes in an acceptable range. [61]. Typically, larger training dataset improves ANN's performance since it leads to a model that is more generalizable to an unseen test data.

In an ANN model each layer is made from a fixed number of neurons. The output of each neuron is linear combination of corresponding input followed by a non-linear activation function such as logistic *sigmoid* or *softmax*. These layers are connected by weight matrices, so that for an input sample T, by performing layer by layer matrix multiplication we would like to get as close as possible to the real label (y value) of that sample. Let's show a three layer ANN model (with 1 hidden layer) with the layers by $a^{(1)}$, $a^{(2)}$ and $a^{(3)}$. To calculate each layer $a^{(l)}$, $l > 1$), first, a matrix multiplication is performed between previous layer and the hypothesis matrix $\theta^{(l-1)}$ to get $z^{(l)}$; then, an activation function $g(z)$ (usually sigmoid function $g(z) = \frac{1}{1+e^{-z}}$) is applied on $z^{(l)}$ which results in i^{th} layer.

First we construct H linear combinations of the input variables $T = \{t_1, ..., t_i, ..., t_Q\}$ in the form

$$z_j^{(2)} = \sum_{i=1}^{Q} w_{ji}^{(1)} t_i + w_{j0}^{(1)} \tag{4}$$

$$a_j^{(2)} = g(z_j^{(2)}) \tag{5}$$

where, $j = 1, ..., H$, and H is the size of first hidden layer; and the superscript (1), (2) indicate that the corresponding parameters are in the first or second layer of the network. w_{ji} is corresponding weights. $w_{j0}^{(1)}$ is referred as biases; z_j are called activations and $g(z)$ is the mentioned non-linear activation function. At each layer of ANN, there is such a transformation; for example in three layer ANN which includes only 1 hidden layer, the elements of the third layer will take the form

$$z_n^{(3)} = \sum_{j=1}^{H} w_{nj}^{(2)} a_i^{(2)} + w_{n0}^{(2)} \tag{6}$$

$$a_n^{(3)} = g(z_n^{(3)}) \tag{7}$$

where, $n = 1, .., N$ and N is the total number of outputs. $a_n^{(3)}$ is the final output of the hypothesis that is going to be compared with the known target wavelengths The bias parameters can be absorbed into the set of weight parameters by defining an additional input variable t_0 whose value is kept fixed at $t_0 = 1$, and the same for other layers, so we combine these various stages to give the overall network function that, for sigmoidal output unit activation functions, takes the form

$$y_m(T, W) = g\Big(\sum_{j=0}^{H1} w_{kj}^{(2)} g\Big(\sum_{i=0}^{Q} w_{ji}^{(1)} t_i\Big)\Big) \tag{8}$$

Given a training set comprising a set of input vectors T_m, where $m = 1, .., M$, together with a corresponding set of target vectors r_m, we minimize the error function (also called optimization objective or cost function) which for sigmoidal case using Lagrange multiplier method it transforms into

$$E(W) = -\sum_{m=1}^{M} \Big[r_m \ln y_m + (1 - r_m) \ln(1 - y_m)\Big]. \tag{9}$$

or more explicitly

$$E(W) = -\sum_{m=1}^{M}\sum_{n=1}^{N} \Big[r_{mn} \ln y_{mn} + (1 - r_{mn}) \ln(1 - y_{mn})\Big]. \tag{10}$$

where y_{mn} denotes $y_n(X_m, W)$. So far, we have explained the FeedForward propagation. At first there is a large cost because the model is not trained yet. An important step in the ANN learning process is called Backpropagation which, unlike the FeedForward propagation explained above, propagates from last layer and stops on second layer. In Backpropagation, each of the weight parameters are updated a small amount proportional to the gradient of cost (error) function with respect to that weight parameter. The proportion factor is called *learning rate* that defines updating rate for each parameter in Backpropagatoin. The training process happens by iterating many cycles, that in each cycle, we perform the FeedForward propagation, calculate the gradient, update the parameters of θ matrices during Backpropagation, and repeat the loop until the model can classify the training data (in output layer) with desired level of accuracy. Afterwards, the trained model can be used to classify the incoming new sample via a few simple matrix multiplications. The total number of parameters in ANN is equal to elements of the weight matrices that each layer is multiplied into plus a single bias element at each hidden layer. In this study, we examined two architectures of ANN: a three layer network with 1 hidden layer $H1 = 100$ neurons between an input layer of $Q = 11$ and output layer of $N = 750$ neurons, and a four layer network with 2 hidden layers of sizes $H1 = 100$ and $H2 = 400$. In first case the number of parameters is $(Q + 1)H1 + (H1 + 1)N = 76950$. In the second architecture the total number of parameters is $(Q + 1)H1 + (H1 + 1)H2 + (H2 + 1)N = 342350$. In the next section we will estimate computational complexity of each machine learning technique which is an indicator of the testing time. In this project, the Python's PyTorch package is used for building the ANN model. The training step of our ANN models are carried out using Northeastern University's Discovery cluster.

Support Vector Machines. We begin our discussion of SVMs by returning to the two-class classification problem using linear models of the form

$$y(T) = w^{\intercal}\phi(T) + b \tag{11}$$

where $\phi(T)$ denotes a fixed feature-space transformation, and we have made the bias parameter b explicit.[61] The training dataset comprises M input vectors $T_1, ..., T_M$ with corresponding target values $r_1, ..., r_M$ where $r_m \in \{-1, 1\}$, and new data points T' are classified according to the sign of $y(T')$. We shall assume for the moment that the training dataset is linearly separable in feature space, so that by definition there exists at least one choice of the parameters W and b such that a function of the form equation (11) satisfies $y(T_m) > 0$ for points having $r_m = +1$ and $y(T_m) < 0$ for points having $r_m = -1$, so that $r_m y(T_m) > 0$ for all training data points. [61] In support vector machines the decision boundary is chosen to be the one for which the margin is maximized by solving

$$\underset{w,b}{\arg\max} \Big\{\frac{1}{|w|} \min_m \big[r_m(w^{\intercal}\phi(T_m) + b)\big]\Big\} \tag{12}$$

Table 1. SVM model with various kernel types.

SVM Kernel type	Linear	RBF	Polynomial(3rd order)
Total average absolute error (nm)	2.594 1	2.598 8	3.564 4

Table 2. Time Complexity of ML method.

ML Model	Bayesian	kNN	SVM	ANN
Time Complexity	$O(QN)$	$O(kM)$	$O(QM)$	$O(H_LN)$

where we have taken the factor $\frac{1}{|w|}$ outside the optimization over m because W does not depend on m.[61] On the other hand there are different kernel tricks to create a non-linear models, hence create larger feature space by a non-linear kernel function $k(T_i, T_j) = \phi(T_i)^\intercal \phi(T_j)$. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space by replacing the equation (11) with

$$y(T) = \sum_{m=1}^{M} \alpha_m r_m k(T, T_m) + b \tag{13}$$

The transformation may be non-linear and the transformed space high-dimensional.[60, 75, 76] The RBF kernel for example uses a Gaussian distribution for each feature T_i and creates M different features using kernel function $k(\vec{T_i}, \vec{T_j}) = e^{-\gamma|(\vec{T_i}-\vec{T_j}|^2}$ for $\gamma > 0$.

When we have more than two classes, SVM can be used as a combination of several one vs. rest classifiers to find hyperplanes that discriminate one category from the rest of them. SVM can also efficiently perform non-linear classification using what is called the kernel method by implicitly mapping the samples original features set into a higher dimensional feature space, as illustrated in figure 2(d), and a new sample is classified depending on the side of the margin that it falls in. In this work, apart from linear SVM we also tried different kernel functions such as *Polynomial*, *Gaussian Radial Basis Function* (RBF), *sigmoid*, but we only reported the results of the linear model in figure 4, since it performs slightly better than the other models; The absolute error in estimating wavelength averaged over entire test set for a few different kernels are presented in table 1.

The number of parameters to be learned is $N(Q+1) = 9000$ for linear model, and $N(M+1) \sim 56 \times 10^6$ for RBF model, where $N = 750$ is the number of classes, $M = 75000$ is number of training samples and $Q = 11$ is dimension of each training sample. Even though these numbers seem pretty large specially for RBF kernel, but most of these parameters are zero, and the calculation is carried out using sparse matrix of parameters. In fact, SVM kernels are called sparse kernel machines. [61] In this project, the Python's SciKit package is used for building the SVM model.

Time Complexity Analysis. As given above, we have $N = 750$ classes of all possible wavelengths, $Q = 11$ filters as feature number and totally $M = 75000$ samples for training. Once trained, we care more about their inference efficiency. The time complexity is analyzed as following.

For Bayesian estimation, for each data point, the conditional distribution $P(T|\lambda_j)$ is calculated with all possible wavelengths which is N. The production for joint probability takes N operation as well. However power operation is included in Gaussian distribution density function. As we compute this density across whole spectrum, exponent in this operation will be N related; therefore, each iteration of the implementation takes $O(N)$. So totally, Bayesian takes $O(QN)$ time.

For KNN, calculating the distance with all training data takes $O(QM)$ time. After that, finding the $k = 7$ minimum values and their indices takes $O(kM)$ time, so the overall complexity is in the order of $O(kM)$.

For ANN, the computation cost from layer i to layer j is H_iH_j. For multiple layer version, the time cost can be generalized as $O(QH_1 + \sum_{i=1}^{L} H_iH_{i+1})$, where H_i stands for the hidden neuron numbers at layer i, L stands for the total layer numbers (except input). As the Q and H will be data size-independent, this method is supposed to be much faster than the other methods. In our ANN architecture the number of neurons increases almost by order of magnitude as we go from input to output layer, so the complexity is dominated by the last layer. N is output layer size and if we denote the last hidden layer size by H_L, the time complexity will be in the order of $O(H_LN)$.

For SVM, for each query the kernel operation is across all support vectors within training data. The inference complexity for linear and RBF model will be $O(QM_{sv})$ since we are solving the dual form here; M_{sv} stands for the number of support vectors, which most of the time will be much less than M but it can also be up to M, so we can show its upper limit as $O(QM_{sv})$.

The theoretical complexity estimations presented in table 2 are in agreement with the measured time required for testing each sample (see figure 4(b)).

4. Acknowledgment

DH and SK acknowledges financial support from NSF ECCS 1351424, a Northeastern University internal Gapfund360 award, and a Northeastern University Provost's Tier 1 Interdisciplinary seed grant.

Data Availability Statement

The data that support the findings of this study are openly available at https://github.com/ostadabbas/Machine-learning-for-nanomaterial-based-optical-wavelength-estimation.

ORCID iDs

Davoud Hejazi https://orcid.org/0000-0002-5215-6395
Shuangjun Liu https://orcid.org/0000-0002-2717-5789
Amirreza Farnoosh https://orcid.org/0000-0002-3766-2310
Sarah Ostadabbas https://orcid.org/0000-0002-2216-9988
Swastik Kar https://orcid.org/0000-0001-6478-7082

References

[1] West J L and Halas N J 2003 Engineered nanomaterials for biophotonics applications: improving sensing, imaging and therapeutics *Ann. Rev. Biomed. Eng.* **5** 285–92
[2] Rao C N, Dua P, Kuchhal P, Lu Y, Kale S and Cao P 2019 Enhanced sensitivity of magneto-optical sensor using defect induced perovskite metal oxide nanomaterial *J. Alloys Compd.* **797** 896–901
[3] Wang Y-H, He L-L, Huang K-J, Chen Y-X, Wang S-Y, Liu Z-H and Li D 2019 Recent advances in nanomaterial-based electrochemical and optical sensing platforms for microRNA assays *Analyst* **144** 2849–66
[4] Yang W, Ratinac K R, Ringer S P, Thordarson P, Gooding J J and Braet F 2010 Carbon nanomaterials in biosensors: should you use nanotubes or graphene? *Angew. Chem. Int. Ed.* **49** 2114–38
[5] Wang J 2005 Nanomaterial-based electrochemical biosensors *Analyst* **130** 421–6
[6] Mojtabavi M, VahidMohammadi A, Liang W, Beidaghi M and Wanunu M 2019 Single-molecule sensing using nanopores in two-dimensional transition metal carbide (MXene) membranes *ACS Nano* **13** 3042–53
[7] Liu X, Huang D, Lai C, Qin L, Zeng G, Xu P, Li B, Yi H and Zhang M 2019 Peroxidase-like activity of smart nanomaterials and their advanced application in colorimetric glucose biosensors *Small* **15** 1900133
[8] Alhamoud Y, Yang D, Kenston S S F, Liu G, Liu L, Zhou H, Ahmed F and Zhao J 2019 Advances in biosensors for the detection of ochratoxin A: bio-receptors, nanomaterials and their applications *Biosensors Bioelectron.* **141** 111418
[9] Hao J, Kar S, Jung Y J and Rubin D 2019 Ion and Radiation Detection Devices Based on Carbon Nanomaterials and Two-Dimensional Nanomaterials US Patent App. 16/331,648
[10] Galstyan V, Poli N and Comini E 2019 Highly sensitive and selective H_2S chemical sensor based on ZnO nanomaterial *Appl. Sci.* **9** 1167
[11] Meng Z, Stolz R M, Mendecki L and Mirica K A 2019 Electrically-transduced chemical sensors based on two-dimensional nanomaterials *Chem. Rev.* **119** 478–598
[12] Li W-Y, Xu L-N and Chen J 2005 Co_3O_4 nanomaterials in lithium-ion batteries and gas sensors *Adv. Funct. Mater.* **15** 851–7
[13] Hennighausen Z, Lane C, Benabbas A, Mendez K, Eggenberger M, Champion P M, Robinson J T, Bansil A and Kar S 2019 Oxygen-induced in situ manipulation of the interlayer coupling and exciton recombination in Bi_2Se_3/MoS_2 2D heterostructures *ACS Appl. Mater. Inter.* **11** 15913–21
[14] Yadav B, Srivastava R, Dwivedi C and Pramanik P 2008 Moisture sensor based on ZnO nanomaterial synthesized through oxalate route *Sensors Actuat. B: Chem.* **131** 216–22
[15] Hejazi D, OstadAbbas S and Kar S 2019 Wavelength estimation of light source via machine learning techniques using low cost 2D layered nano-material filters *APS Meeting Abstracts*
[16] Hejazi D, Liu S, Ostadabbas S and Kar S 2019 Bayesian inference-enabled precise optical wavelength estimation using transition metal dichalcogenide thin films *arXiv preprint arXiv:1901.09452*
[17] Lee I and Lee K 2015 The Internet of Things (IoT): applications, investments and challenges for enterprises *Bus. Horiz.* **58** 431–40
[18] Sharma S K and Wang X 2019 Towards massive machine type communications in ultra-dense cellular IoT networks: Current issues and machine learning-assisted solutions *IEEE Commun. Surv. Tutorials* **22** 426–71
[19] Lynggaard P 2019 Controlling interferences in smart building IoT networks using machine learning *Int. J. Sensor Networks* **30** 46–55
[20] Hussain F, Hussain R, Hassan S A and Hossain E 2019 Machine learning in IoT security: current solutions and future challenges *arXiv preprint arXiv:1904.05735*
[21] Yang B, Cao X, Han Z and Qian L 2019 A machine learning enabled MAC framework for heterogeneous internet-of-things networks *IEEE Trans. Wireless Commun.* **18** 3697–712
[22] Carbune V, Keysers D and Deselaers T 2019 Predicted Image-Capture Device Settings through Machine Learning
[23] Nai W, Zhao J, Ma L, Xing Y and Zhu H 2019 A real-time 2-dimentional recovery algorithm for blurred video image filmed by charge-coupled devices in environment sensing for driving safety *2019 IEEE 9th Int. Conf. on Electronics Information and Emergency Communication (ICEIEC)* pp 1–4
[24] Yan T, Yang C and Cui X 2019 A novel energy resolved x-ray semiconductor detector *arXiv preprint arXiv:1907.10796*
[25] Guan W, Wen S, Liu L and Zhang H 2019 High-precision indoor positioning algorithm based on visible light communication using complementary metal–oxide–semiconductor image sensor *Opt. Eng.* **58** 024101
[26] Brunckhorst J, Pirchner A, Naik N R, Sabanayagam M, Paul-Yuan M D K, Paulitschke P and Rauchensteiner C-M M 2019 Machine learning-based image detection for lensless microscopy in life science

[27] Butler K T, Davies D W, Cartwright H, Isayev O and Walsh A 2018 Machine learning for molecular and materials science *Nature* **559** 547–55
[28] Schmidt J, Marques M R, Botti S and Marques M A 2019 Recent advances and applications of machine learning in solid-state materials science *NPJ Computat. Mater.* **5** 36
[29] Mostavi M, Salekin S and Huang Y 2018 Deep-2'-O-Me: Predicting 2'-O-methylation sites by convolutional neural networks *2018 40th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society (EMBC)* pp 2394–7
[30] Chiu Y-C, Chen H-I H, Gorthi A, Mostavi M, Zheng S, Huang Y and Chen Y 2019 Deep learning of pharmacogenomics resources: moving towards precision oncology *Briefings Bioinform.* **00** 1–18
[31] Mostavi M, Chiu Y-C, Huang Y and Chen Y 2019 Convolutional neural network models for cancer type prediction based on gene expression *arXiv preprint arXiv:1906.07794*
[32] Liu S and Ostadabbas S 2019 Seeing Under the Cover: A Physics Guided Learning Approach for In-bed Pose Estimation *Int. Conf. on Medical Image Computing and Computer-Assisted Intervention* pp 236–45
[33] Farnoosh A and Ostadabbas S 2019 Introduction to Indoor GeoNet *CVPR Workshop on 3D Scene Understanding for Vision, Graphics and Robotics*
[34] Xue D, Balachandran P V, Hogden J, Theiler J, Xue D and Lookman T 2016 Accelerated search for materials with targeted properties by adaptive design *Nature Commun.* **7** 11241
[35] Yuan R, Liu Z, Balachandran P V, Xue D, Zhou Y, Ding X, Sun J, Xue D and Lookman T 2018 Accelerated discovery of large electrostrains in $BaTiO_3$-Based piezoelectrics using active learning *Adv. Mater.* **30** 1702884
[36] Raccuglia P *et al* 2016 Machine-learning-assisted materials discovery using failed experiments *Nature* **533** 73
[37] Ley S V, Fitzpatrick D E, Ingham R J and Myers R M 2015 Organic synthesis: march of the machines *Angewandte Chemie Int. edn* **54** 3449–64
[38] Correa-Baena J-P, Hippalgaonkar K, van Duren J, Jaffer S, Chandrasekhar V R, Stevanovic V, Wadia C, Guha S and Buonassisi T 2018 Accelerating materials development via automation, machine learning and high-performance computing *Joule* **2** 1410–20
[39] Iwasaki Y, Sawada R, Stanev V, Ishida M, Kirihara A, Omori Y, Someya H, Takeuchi I, Saitoh E and Shinichi Y 2019 Materials development by interpretable machine learning *arXiv preprint arXiv:1903.02175*
[40] Malkiel I, Mrejen M, Nagler A, Arieli U, Wolf L and Suchowski H 2018 Plasmonic nanostructure design and characterization via deep learning *Light: Sci. Appl.* **7** 60
[41] Mlinar V 2013 Engineered nanomaterials for solar energy conversion *Nanotechnology* **24** 042001
[42] Ma W, Cheng F and Liu Y 2018 Deep-learning-enabled on-demand design of chiral metamaterials *ACS Nano* **12** 6326–34
[43] Bacigalupo A, Gnecco G, Lepidi M and Gambarotta L 2016 Design of acoustic metamaterials through nonlinear programming *Int. Workshop on Machine Learning, Optimization and Big Data* pp 170–81
[44] Coley C W, Barzilay R, Jaakkola T S, Green W H and Jensen K F 2017 Prediction of organic reaction outcomes using machine learning *ACS Central Sci.* **3** 434–43
[45] Padula D, Simpson J D and Troisi A 2019 Combining electronic and structural features in machine learning models to predict organic solar cells properties *Mater. Horizons* **6** 343–9
[46] Hoyt R A, Montemore M M, Fampiou I, Chen W, Tritsaris G and Kaxiras E 2019 Machine learning prediction of H adsorption energies on Ag alloys *J. Chem. Info. Model.* **59** 1357–1365
[47] Schlexer Lamoureux P, Winther K, Garrido Torres J A, Streibel V, Zhao M, Bajdich M, Abild-Pedersen F and Bligaard T 2019 Machine learning for computational heterogeneous catalysis *ChemCatChem* **11** 3581–601
[48] Goldberg E, Scheringer M, Bucheli T D and Hungerbühler K 2015 Prediction of nanoparticle transport behavior from physicochemical properties: machine learning provides insights to guide the next generation of transport models *Environ. Sci.: Nano* **2** 352–60
[49] Ma X-Y, Lewis J P, Yan Q-B and Su G 2019 Accelerated discovery of two-dimensional optoelectronic octahedral oxyhalides via high-throughput ab initio calculations and machine learning *J. Phys. Chem. Lett.* **10** 6734–40
[50] Huang Y Yu C, Chen W, Liu Y, Li C, Niu C, Wang F and Jia Y 2019 Band gap and band alignment prediction of nitride-based semiconductors using machine learning *J. Mater. Chem.* C **7** 3238–45
[51] Khabushev E M, Krasnikov D V, Zaremba O T, Tsapenko A P, Goldt A E and Nasibulin A G 2019 Machine learning for tailoring optoelectronic properties of single-walled carbon nanotube films *J. Phys. Chem. Lett.* (http://dx.doi.org/10.1021/acs.jpclett.9b0277)
[52] Tang B, Lu Y, Zhou J, Wang H, Golani P, Xu M, Xu Q, Guan C and Liu Z 2019 Machine learning-guided synthesis of advanced inorganic materials *arXiv preprint arXiv:1905.03938*
[53] Mihaila M 2004 *Advanced Experimental Methods for Noise Research in Nanoscale Electronic Devices* (Berlin: Springer) pp 19–27
[54] Ru C, Zhang Y, Sun Y, Zhong Y, Sun X, Hoyle D and Cotton I 2010 Automated four-point probe measurement of nanowires inside a scanning electron microscope *IEEE Trans. Nanotechnol.* **10** 674–81
[55] Sikula J and Levinshtein M 2006 *Advanced Experimental Methods for Noise Research in Nanoscale Electronic Devices* Springer Science & Business Media) vol 151
[56] Wu T-T, Chen Y-Y and Chou T-H 2008 A high sensitivity nanomaterial based SAW humidity sensor *J. Phys. D: Appl. Phys.* **41** 085101
[57] Guiot A, Golanski L and Tardif F 2009 Measurement of nanoparticle removal by abrasion *J. Phys.: Conf. Ser.* **170** 012014
[58] Hejazi D, Liu S, Ostadabbas S and Kar S 2019 Transition Metal Dichalcogenide Thin Films for Precise Optical Wavelength Estimation using Bayesian Inference *ACS Appl. Nano Mater.* **2** 4075–84
[59] Benedetti I, Nguyen H, Soler-Crespo R A, Gao W, Mao L, Ghasemi A, Wen J, Nguyen S and Espinosa H D 2018 Formulation and validation of a reduced order model of 2D materials exhibiting a two-phase microstructure as applied to graphene oxide *J. Mech. Phys. Solids* **112** 66–88
[60] Cortes C and Vapnik V 1995 Support-vector networks *Machine Learn.* **20** 273–97
[61] Bishop C M 2006 *Pattern Recognition and Machine Learning* (Berlin: Springer)
[62] Mangasarian O L and Musicant D R 2001 Lagrangian support vector machines *J. Mach. Learning Res.* **1** 161–77
[63] Ng A 2017 Machine Learning coursera [Online] Available at: (org/learn/machine-learning)
[64] Osuna E, Freund R and Girosi F 1997 An improved training algorithm for support vector machines. Neural networks for signal processing VII *Proc. of the 1997 IEEE Signal Processing Society Workshop* pp 276–85
[65] Larkin J, Henley R Y, Jadhav V, Korlach J and Wanunu M 2017 Length-independent DNA packing into nanopore zero-mode waveguides for low-input DNA sequencing *Nat. Nanotechnol.* **12** 1169
[66] Sahami M 1996 Learning Limited Dependence Bayesian Classifiers *KDD'96: Proc. of the 2nd Int. Conf. on Knowledge Discovery and Data Mining* pp 335–8

[67] Bassett R and Deride J 2016 Maximum a posteriori estimators as a limit of Bayes estimators *Math. Program.* 1–16
[68] Bernardo J M and Smith A F 2001 *Bayesian Theory* (Hoboken, NJ: John Wiley & Sons)
[69] Lee P M 1989 *Bayesian Statistics* (London: Oxford University Press)
[70] Altman N S 1992 An introduction to kernel and nearest-neighbor nonparametric regression *Am. Statist.* **46** 175–85
[71] Hsu K L, Gupta H V and Sorooshian S 1995 Artificial neural network modeling of the rainfall-runoff process *Water Res. Res.* **31** 2517–30
[72] Kavzoglu T Determining optimum structure for artificial neural networks *Proc. of the 25th Annual Conf. and Exhibition of the Remote Sensing Society* 8-10 September 1999 pp 675–82
[73] Wilamowski B M, Cotton N, Hewlett J and Kaynak O 2007 Neural network trainer with second order learning algorithms *2007 11th Int. Conf. on Intelligent Engineering Systems* pp 127–32
[74] Montufar G F, Pascanu R, Cho K and Bengio Y 2014 On the number of linear regions of deep neural networks *Adv. Neural Inf. Process. Syst.* 2924–32
[75] Boser B E, Guyon I M and Vapnik V N 1992 A training algorithm for optimal margin classifiers *Proc. of the Fifth Annual Workshop on Computational Learning Theory* pp 144–52
[76] Aiserman M, Braverman E M and Rozonoer L 1964 Theoretical foundations of the potential function method in pattern recognition *Avtomat. i Telemeh.* **25** 917–36