MACHINE LEARNING APPLICATION FOR HIGH-SPEED FTIR ABSORPTION SPECTRA ANALYSIS

G. CHIROȘCA[1,a], S. MUSAT[1], D. ISTRATE[1], A. CHIROȘCA[2,3,b]

[1]"INOE 2000" National R&D Institute for Optoelectronics,
Atomiștilor 409, RO-077125, Măgurele-Bucharest, Romania
Corresponding author[a]: gianina.chirosca@inoe.ro
[2]Faculty of Physics, University of Bucharest, Magurele - Ilfov, Romania
Corresponding author[b]: alecsandru.chirosca@unibuc.ro
[3]Networks srl, Cugir - Alba, 5 Biruinței Str., Romania

Received Aprll 1, 2024

Abstract. With this work our main objective is to find the best general (baseline) model for analyzing unknown spectra using Fourier infrared transformed spectroscopy (FT-IR) coupled with machine learning (ML) algorithms. This goal allows us to identify the best methodology applied for inline analysis of different experimental spectra for qualitative structural information obtained with types of structures that generate absorption or emission peaks. This methodology opens new perspectives for automated data processing using flexible algorithms and machine learning to encode experimental data for future applications. The results provide a good perspective on Machine Learning algorithms for applied sciences research. For our case study (FTIR experimental data) our model allows for peak feature extraction with a relative low, close to machine standard deviation, error budget. The best model identified is a specialized model but the standard, fully connected network models are evaluated.

Key words: Fourier transform-infrared spectroscopy, machine learning algorithms, autonomous data processing, data feature extraction.

DOI: https://doi.org/10.59277/RomJPhys.2024.69.115

1. INTRODUCTION

Recent developments focus on machine processing of data structures related to scientific experiments allowing for data augmentation using machine (robotic) processing serving the scientists in correlation processes with other data or easing feature extraction with high impact on a large number of applications from material sciences, medicine and pharmaceutical applications up to cultural heritage analysis enabling faster and more accurate composition identification and decay products characterization [1–3]. Our work focuses on the latter scenario using Machine Learning due to their reduced sensitivity towards data-induced noise and their versatility [4]. Within this work, we focus on spectral data obtained using Fourier Transform Infrared Spectroscopy (FTIR) [5]. Still, it can be easily extended towards other spectral data types (emission - natural or induced spectroscopy - ultraviolet-visible spectrum, Gamma, RAMAN, Fluorescence a.s.o.) once the baseline for model development is established.

Such algorithms and methods can be employed within data processing pipelines aiding scientists in their research activities providing proper feature extraction activities for automated platforms and addressing the requirements of the FAIR data principle. Machine Learning approaches provide us with a way of preserving knowledge related to data processing for specialized models thus addressing the growing issue related to knowledge degradation, these types of models have great potential in encoding the human experience for future reference [4, 7].

Previous works found within the literature [5, 6] provide models applied for specific detection types. In this paper, we are addressing generic samples that can be applied for any FTIR spectra. Our main goal is to identify the base structure that can be applied to any spectrum while retaining compatibility towards applied machine architectures that can be employed for high-speed machine data processing [8, 9] thus optimizing the processing of spectral information applied for Fourier transform absorption measurements (FT-IR). Our main objective is to reduce processing times compared to classical statistical methods. The specific result obtained is given by a set of algorithms specific to the automatic spectra processing. A secondary objective is to identify the methodology used for inline analysis of various spectra obtained experimentally, which allows the experiments to be associated with qualitative structural information obtained with the types of structures that generate absorption or emission peaks for implementing a FAIR methodology in storing scientific information, thus facilitating Findability, Accessibility, Interoperability and optimization of scientific results obtained using that data (Reusability).

2. DATA PROCESSING METHODS

Absorption spectroscopy is used as a chemical analysis tool to determine the presence of a particular species in a sample and to quantify the species concentration within the sample. Our aim towards the FTIR has a considerable range of applications from molecular and atomic physics studies, and astronomical spectroscopy, to remote sensing. The use of spectroscopic techniques using FT-IR is a rapidly expanding field of research, focusing on its usefulness in various diagnostics by generating spectral information. This information is usually represented in the form of images or datasets. The energy of molecular vibrations corresponds to the infrared (IR) region of the electromagnetic spectrum and the position of the absorption bands is indicated in wave numbers, having the unit cm^{-1}. The infrared spectrum can generally be recorded in the range of 450-4000 cm^{-1}. The IR spectrum is obtained by recording the intensity of radiation in the absence of the analysis sample (background spectrum) and the intensity of infrared radiation after it passes through the sample. After passing through the sample, the intensity of the beam will attenuate, attenuation due to radiation absorption by the molecules of the sample. The vibration spectrum of a molecule is considered to be a unique physical property, and the infrared spectrum can be used as a fingerprint for identification. The qualitative aspects of infrared spectroscopy are one of the strongest attributes of this diverse and versatile analytical technique.

2.1. **SPECTRAL DATA SET**

To compare methods and gain obtained by processing using algorithms of machine learning we have established a database containing 12 FTIR (Fig. 1) spectra made for samples containing thin inorganic films. These spectra contain a specific set of molecular species, which facilitates the most accurate evaluation of spectral lines of absorption introducing as little overlap and attenuation as possible.

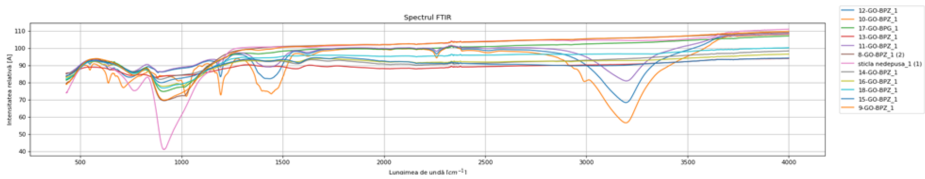


Fig. 1 – Initial data set considered for the two Machine Learning model implementations consists of 12 sample spectra.

To pre-process the data for ML processes a specific pipeline is implemented using Python [10] (3.10.14) as the mostly used language for data processing where standard computational methods like SciPy [11], and Scikit [12], respectively Numpy [13], and for the implementation of ML algorithms, we will use the TensorFlow framework [14] (v2.15.1). The Python programming language is establishing itself as one of the most popular languages for scientific computing, it is a programming language with very simple and concise syntax, thus allowing an easy expression of complex ideas. This language can be adapted to various fields of research, offering a wide range of packages, specialized modules for data analysis, statistics, machine learning, visualization, etc., and being an excellent choice for researchers, the academic community as well as for those who use this language in industry (robot programming, analysis of statistics, automated systems a.s.o.).

To extract spectral lines from spectral (ASCII) files with FT-IR spectra provided by Horiba equipment, we will use a regular expression to identify and extract the content of the spectrum while its header is ignored. The spectra analysis was performed using the FT-IR Spectro-photometer type Spectrum 100 Perkin Elmer with UATR accessory.

The data extraction process is performed using two methods, one static consisting of Fully Connected Networks that can extract a limited set of features and one dynamic that employs recurrent networks performing a sliding window approach on the data allowing for more accurate, dynamic results. For each of the methods, a reinforced learning pattern is performed based on static spectra analysis. The available data is split into training and validation data, after the data set is shuffled for consistency 2 of the spectra are kept for testing and not seen by the training stage.

The extracted data consists of a TensorFlow Dataset with windows (having a shape of (1,100) with 1 being the batch size and 100 de window dimension where the latter has a type of Float32 as the model precision requirements are limited in terms of number representation (in this case the precision is 10^{-14}). The batch size parameter is always an integer and we are using one as the model results need to be estimated each window at a time. The second input parameter is the window's base location in terms of wavelength (but the same approach can be applied for other spectra types) and provides the window base wavelength (expressed in cm^{-1}). This input parameter also has a batch size of 1 (it needs to be consistent with the first input batch size) and a data dimension of 1 while having a data type of Int16.

To process data in a windowed fashion with a large scale of applications being one of the most used Machine Learning methods. This type of architecture requires us to pre-process data by defining windows upon the data that slide through the entire data set. To perform such data preparation we take advantage of another Python package for processing tabular data, namely Pandas [15].

This type of pre-processing is applied to the experimental data taking the wave number as the dimension of the data and intensity for its records. The output values.

The data preparation algorithm for the chosen range of 100 the value of vertices using the find peaks function and if it finds vertices it will calculate the best vertex, width, and prominence. If it is not found, mark with 0 the value of the best peak, the energy of the peak will be the initial energy, and mark with 0 the values for width and protrusion thus enabling the implementation of supervised learning techniques.

While our model inference provides us with several outputs (peak wavelength, width, and prominence) and the model uses supervised learning we also need to define output data where all parameters have a data type of Float32 and each of them has a shape of (1,1) meaning that the batch size is consistent with the input data and the output has only one number.

To analyze both inputs, our model concatenates the processed window data with the global location of the window within the spectrum we concatenate the former values with this data before the final processing stage where all data is correlated. The complete data series consists of 41652 windows. This complete data set is split into training (70% - 29156 windows), validation (15% - 6248 windows) and testing (15% - 5247 windows).

2.1.1. Basic model architecture

Both models described within this work have a common architecture design pattern approaching the issue with a multi-headed, multiple output design pattern -

MIMO (Multiple Input Multiple Output). This methodology allows us to approach the sliding window algorithm of the spectral data series as uni-dimensional data. This forces us to provide the system with knowledge about the window base energy allowing the results to be presented concerning spectral line location (wavelength) instead of using relative locations within each window.

For the implementation of the structural stages, we apply two techniques where the window processing stage is implemented using a standard, fully connected network (FCN design) and one RNN (Recurrent Neural Networks) providing a better understanding of the input series.

Each of the implemented network designs is optimized using the Adam Optimizer [17] without exponential momentum techniques that are usually employed for speeding up the training process with the trade-off in model convergence towards the loss functions. Model evaluation is performed using Mean Square Error and Mean Absolute Error metrics with the loss defined similarly as the former one. The overall training process is previously set at 100 epochs focusing on peak description (location, width, and prominence) where peak width and prominence describe its characteristics and its location (wavelength index). While the widths for solid materials as the one analyzed here the width can have values from a few wavelengths up to 500, this algorithm provides us with enough descriptors to function as input for other unsupervised processing techniques like PCA (Principal Component Analysis) or LDA (Least Discriminant Analysis).

Both algorithms provide a minimal network design for such issues and provide a baseline for future developments of more complex architectures. This design was implemented and tested for optimal results convergence while not needing dropout techniques to improve over-fitting situations. This minimalist approach requires specialized training techniques resulting in a slightly more complicated architecture design with parameters being computed in a cross-linked manner.

This modular approach allows us to change the model processing pipeline for each of the implementation methods presented below.

2.1.2. Static model - Fully Connected Network - FCN

The FCN network, one of the oldest architectures consisting of a Fully Connected Neural Network yields acceptable results but the level of network understanding of the data structures within the inputs is limited to the layer dimension meaning that excessive computing resources need to be allocated to obtain more accurate results.

Due to a large number of parameters specific to FCN implementation, the training stage for the whole model presented in Fig. 2 takes about 11 ms/step for A400 CUDA-enabled GPU with 16Gb of GPU RAM for modern (2.15) stable version of TensorFlow. The processing time/epoch is 349 ± 33 s yielding a total of 9.69 ± 0.93 h for 100 epochs.

```
________________________________________________________________________________________________
Layer (type)                  Output Shape                 Param #   Connected to
================================================================================================
series (InputLayer)           [(1, 100)]                   0         []

dense (Dense)                 (1, 128)                     12928     ['series[0][0]']

dense_1 (Dense)               (1, 256)                     33024     ['dense[0][0]']

dense_2 (Dense)               (1, 256)                     65792     ['dense_1[0][0]']

base_line (InputLayer)        [(1, 1)]                     0         []

dense_3 (Dense)               (1, 128)                     32896     ['dense_2[0][0]']

dense_4 (Dense)               (1, 4)                       8         ['base_line[0][0]']

concatenate (Concatenate)     (1, 132)                     0         ['dense_3[0][0]',
                                                                      'dense_4[0][0]']

dense_5 (Dense)               (1, 64)                      8512      ['concatenate[0][0]']

dense_6 (Dense)               (1, 64)                      4160      ['dense_5[0][0]']

dense_7 (Dense)               (1, 64)                      4160      ['dense_6[0][0]']

eidx (Dense)                  (1, 1)                       65        ['dense_7[0][0]']

width (Dense)                 (1, 1)                       65        ['dense_7[0][0]']

proeminence (Dense)           (1, 1)                       65        ['dense_7[0][0]']

================================================================================================
Total params: 161675 (631.54 KB)
Trainable params: 161675 (631.54 KB)
Non-trainable params: 0 (0.00 Byte)
________________________________________________________________________________________________
```

Fig. 2 – Baseline, reference model for data processing using a simple, flat-model FCN implementation.

Extending the number of epochs has an exponential increase in model performance estimating that for a $< 1\sigma$ accuracy (where $\sigma = 4$ cm^{-1}) is achievable by 10^4 epochs of training.

The data processing stage consists of three Dense (FCN) layers each having 128 neurons that use Rectified Linear Activation Function (relu) and a specialized kernel initializer from literature [16].

Model training checkpoints will save models when minimizing the peak energy regression and when the total loss reaches a minimum value as the model tries to minimize three output parameters while this function has only recently gained traction for the TensorFlow framework (from basic functionality in TensorFlow 2.15 and more advanced on the development 2.16 branch).

While the number of training epochs is kept low (100 epochs - roughly 9 hours of training time) special attention needs to be taken while setting up the network optimizer where, in our case, we are using [18] enabling the AMSGrad algorithm variant of Adam and weights optimization is performed using Exponential Moving Average for rapid convergence. These optimizations bring a training overhead of 4 ± 2 ms for each training step.

2.1.3. Specialized model - Recurrent Neural Network based network design

This approach uses a window processing stage that requires the existence of a short-term memory within its layers and is not, until recently, able to be implemented using programmable digital logic devices. This type of memory is more appropriate when working with sequential data such as the records within our spectra.

Layer (type)	Output Shape	Param #	Connected to
series (InputLayer)	[(1, 100)]	0	[]
tf.expand_dims (TFOpLambda)	(1, 1, 100)	0	['series[0][0]']
lstm (LSTM)	(1, 1, 128)	117248	['tf.expand_dims[0][0]']
lstm_1 (LSTM)	(1, 1, 128)	131584	['lstm[0][0]']
base_line (InputLayer)	[(1, 1)]	0	[]
lstm_2 (LSTM)	(1, 128)	131584	['lstm_1[0][0]']
dense (Dense)	(1, 4)	8	['base_line[0][0]']
concatenate (Concatenate)	(1, 132)	0	['lstm_2[0][0]', 'dense[0][0]']
dense_1 (Dense)	(1, 128)	17024	['concatenate[0][0]']
re_lu (ReLU)	(1, 128)	0	['dense_1[0][0]']
dense_2 (Dense)	(1, 129)	16641	['re_lu[0][0]']
re_lu_1 (ReLU)	(1, 129)	0	['dense_2[0][0]']
dense_3 (Dense)	(1, 128)	16640	['re_lu_1[0][0]']
eidx (Dense)	(1, 1)	129	['dense_3[0][0]']
width (Dense)	(1, 1)	129	['dense_3[0][0]']
proeminence (Dense)	(1, 1)	129	['dense_3[0][0]']

Total params: 431116 (1.64 MB)
Trainable params: 431116 (1.64 MB)
Non-trainable params: 0 (0.00 Byte)

Fig. 3 – Recursive Neural Network proof of concept architecture design optimized for speed allowing for faster processing times.

The generated tabular file occupies about 45 Mb and has 41652 entries (for 12 spectra), with each experimental data file contributing 3471 entries. A set of spectra that have not been processed during method development shall be used to validate the results.

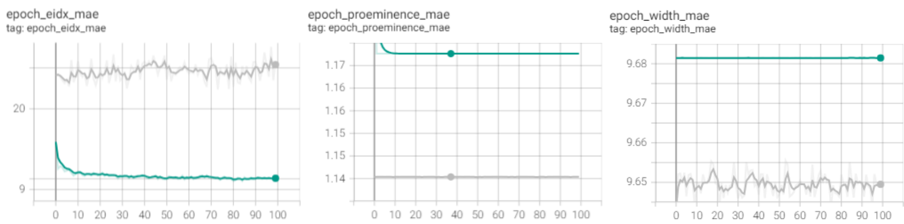
For the RNN network implementation, we started from the already established Long Short-Term Memory (LSTM) [19] developing a specialized model structure (see Fig. 2) with an Adam [17] optimizer function using quadratic error loss function with a window class through which to go through the data set, the class uses the training and testing data set, the size of the input and output sequence, and the shift parameter, the RNN model can be built so that it uses the sequences of 100 from the initial data frame. The network runs 100 epochs at a time of 60-100/ms. The loss values and validation loss reach low values between 0.5 and 0.01.

3. RESULTS AND DISCUSSION

Model performance is evaluated using the validation and testing output values (as they are not previously seen by the model during the training steps) allowing us to evaluate real experimental data metrics. These results aimed implementation is specific to supervised learning techniques.

The compared models within this study require specific steps to be applied for each of the development methods. The FCN model tends to be more accurate but suffers from the large parameter budget leading to exploding and vanishing gradient. To overcome this issue we are forced to introduce normalization layers between the dense stages as weight distribution (although centered) is often sparsely distributed. The limitations of current developments make it harder to compare floats and integers leading towards a higher loss function for integer values (in our case the peak wave number as it can be observed in Fig. 4).

Even with all the model improvements applied the training stage is taking a long time and has many stationarity areas (no improvement can be seen for many training epochs) that require extensive training times.

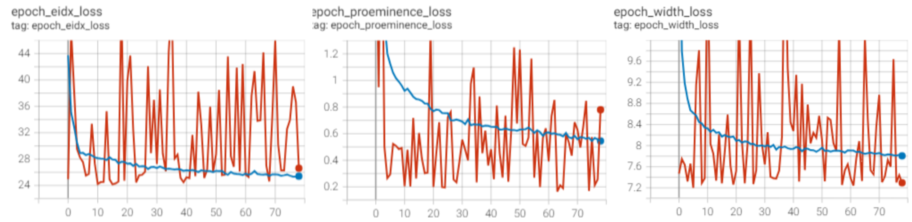


(a) Wavenumber (b) Proeminence (c) Peak width

Fig. 4 – Training process for the proposed FCN (static model) network design concerning iteration number. Gray is the validation while the training line is the blue one.

The recorded results show that the results are close to our aim (36 cm^{-1}) with a small margin thus our implementation for the second model, a much more specialized model for handling data series. The other peak descriptors (width and prominence) provide accurate results even for the smaller training size. These results show that general network types provide a lesser response as they cannot gain knowledge from the ingestion data.

On the other hand, the LSTM approach provides us with more consistent results (see Fig. 5) but suffers from an increase in training time as the LSTM parameters require more computations and transient data storage (each LSTM cell has a memory element that keeps the last seen values thus the name Long Short Term Memory [19]).



(a) Wavenumber (b) Width (c) Proeminence

Fig. 5 – Training process for the proposed RNN network design. The orange line contains the validation data while the blue one is the training.

This specialized model is more intensive on the processing times (training times for 100 epochs estimated at 14h training time using GPU accelerated training even if enough resources are available at the computing level. One optimization that might allow for a smaller training time is using GPUs with higher bandwidth (256 bits) as the board applied for this study uses only 196 bits of bandwidth. The results provide higher precision for the wavelength ($\sigma = 24.17$ cm^{-1}) much better than the ones obtained using the standard FCN-based network. In terms of peak features (width and prominence) have better results (prominence validation mean absolute error is 0.21 in transmittance units while the peak width validation MAE is 7.3 cm^{-1}).

Our aim of finding the minimal baseline models for this type of data is achieved by trial and error processing (increasing or decreasing the number of dense layers for data processing leads to an increase in detection Mean Absolute Error/Mean Squared Error as both are used for quantifying feature detection). Better results are obtained using a feed-forward network design just before the final output layers. To resolve exploding gradient issues common to the ADAGrad variant of the ADAM optimizer the dense layers have to filter out negative values using a Rectifier Linear Unit layers. This behavior can be observed in Fig. 5 where the validation steps have specific higher amplitudes than the training ones.

4. CONCLUSIONS

In this study, we examine the traditional method of analyzing unknown spectra, applied to Fourier transform absorption measurements and using these types of processes, we use machine learning techniques to improve our investigation and reduce processing time compared to classical statistical methods. By leveraging dense image time series and machine learning, we overcome these challenges, thereby improving precision in spectra estimation and predicting by linear regression, peak, width, and amplitude (prominence). Special attention was given to the network architecture yielding a more complex architecture with better results as this is a baseline model. The data-series approach is feasible for buffer IoT Programmable Logic Design implementation for systems to be implemented for automated multi-sample analysis that encapsulates machine learning algorithms. The approach and its future developments have a great potential to be extended with any spectroscopy analysis methods applications allowing the development of ML-based physical applications (FPGA based) automating spectra processing pipeline and easing the information analysis procedures thus providing a solution for machine processing of scientific data (FAIR principle). This approach gives more flexibility in terms of input data [20].

Acknowledgements. This work was carried out also, through the Core Program with the National Research Development and Innovation Plan 2022-2027, with the support of MCID, project no. PN 23 05/2023 contract 11N/2023, and Program I – Development of the National R&D System, Subprogram 1.2 Institutional Performance-Projects for Excellence Financing in RDI, contr. 18PFE/2021.

APPENDIX

Figures 6 and 7 provide an overview of the network architectures that better describe the Machine Learning Model architecture with multiple inputs and multiple outputs the specialized encoder is placed just before the output layers. While the encoder is simpler in the case of the reference (FCN) model, the time-series approach requires more information (context) to be provided for the output layers thus the slightly more complicated structure.

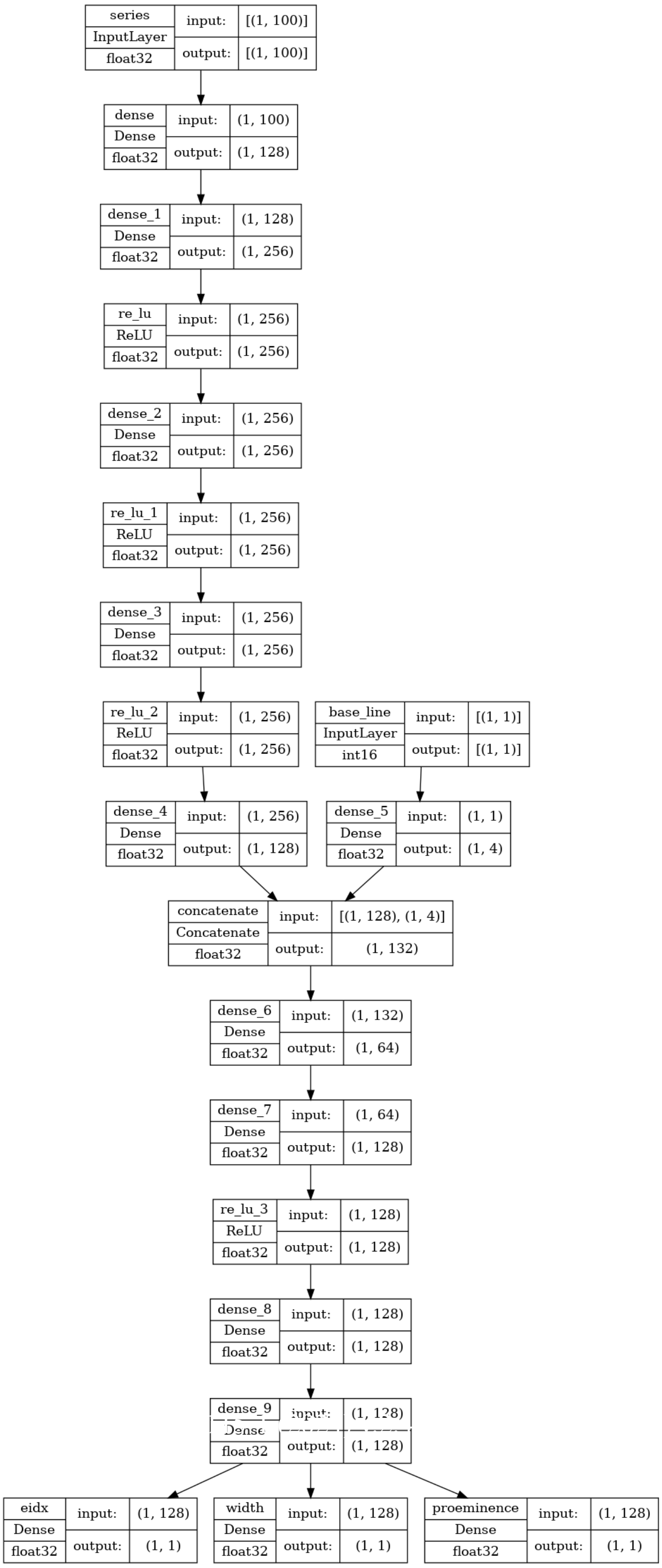


Fig. 6 – Reference model architecture full description.

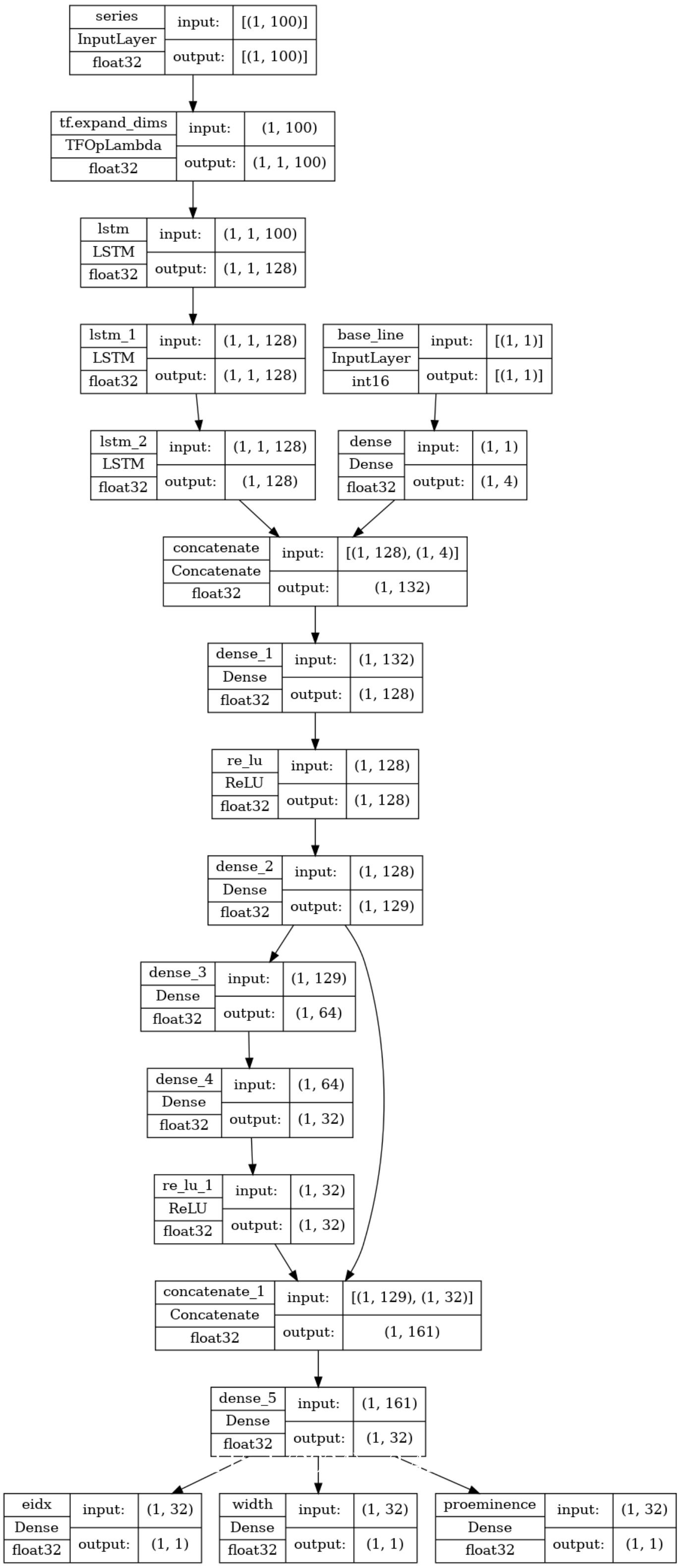


Fig. 7 – Reference model architecture full description.

REFERENCES

1. J. Striber, R. Radvan, L.M. Angheluta, *Laser spectroscopy methods for an* 18^{th} *century grisaille painting investigation*, Journal of Optoelectronics and Advanced Materials **11**, 1815–1820 (2009).
2. L.M. Angheluta, A. Moldovan, R. Radvan, *The teleoperation of a LIF Scanning Device*, U.P.B. Sci. Bull. Series A **73**, 193–200 (2011).
3. I.M Cortea, L. Ghervase, O. Tentea, A.C. Parau, R. Radvan, *First Analytical Study on Second-Century Wall Paintings from Ulpia Traiana Sarmizegetusa*: *Insights on the Materials and Painting Technique*, International Journal of Architectural Heritage Conservation, Analysis, and Restoration **14**(5), 751-761 (2019) .
4. A.T. Tunkiel, D. Sui, T. Wiktorski, *Data-driven sensitivity analysis of complex machine learning models: A case study of directional drilling*, Journal of Petroleum Science and Engineering **195**, 107630 (2020).
5. R. Dell'Anna, P. Lazzeri, M. Frisanco, *et al.*, *Pollen discrimination and classification by Fourier transform infrared (FT-IR) microspectroscopy and machine learning*, Anal Bioanal Chem **394**, 1443–1452 (2009).
6. G. Chirosca, M.E. Mihailov, M.I. Tomescu-Chivu, A. Chirosca, *Enhanced machine learning model for meteo-oceanographic time-series prediction*, Romanian Journal of Physics **67**, 815 (2022).
7. B. Kapusuzoglu, S. Mahadevan, *Information fusion and machine learning for sensitivity analysis using physics knowledge and experimental data*, Reliability Engineering & System Safety **214**, 107712 (2021).
8. FastML Team, *fastmachinelearning/hls4ml*, Zenodo (2023).
9. D. Javier *et al.*, *Fast inference of deep neural networks in FPGAs for particle physics*, JINST **13**, 07027 (2013).
10. G. van Rossum, *Python tutorial*, *Technical Report CS-R9526*, ed Centrum voor Wiskunde en Informatica (CWI), Amsterdam (May 1995).
11. P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, *SciPy 1.0*, *Contributors*, *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*, Nature Methods **17**, 261–272 (2020).
12. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel *et al.*, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research **12**, 2825–2830 (2011).
13. C.R. Harris, K.J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, T.E. Oliphant, *Array programming with NumPy*, Nature **585**, 357–362 (2020).
14. M. Abadi, P. Barham, J. Chen, *et al.*, *Tensorflow: A system for large-scale machine learning*, arXiv:1605.08695 (2016).
15. W. McKinney *et al.*, *Data structures for statistical computing in Python*, in Proceedings of the 9th Python in Science Conference, **445**, 51–56 (2010).
16. K. He, X .Zhang, S. Ren, J. Sun, *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*, arXiv:1502.01852 (2015).
17. D.P. Kingma, J. Ba, *Adam: A Method for Stochastic Optimization*, arXiv:cs.LG/1412.6980 (2014).
18. J.S. Reddi, S. Kale, S. Kumar, *On the Convergence of Adam and Beyond*, ICLR 2018, arXiv:1904.09237 (2019).
19. S. Hochreiter, J. Schmidhuber, *Long short-term memory*, Neural Computation, **9**(8), 1735–1780 (1997).
20. M. Simileanu, R. Radvan, *Remote method and set-up for submerged archaeological remains characterization*, Journal of Optoelectronics and Advanced Materials **13**, 528–531 (2011).