AFRL-RY-WP-TR-2022-0140

SEQUENTIAL LEARNING USING IMAGE-BASED CLASSIFICATION

Kevin Lee
The Ohio State University

APRIL 2022
Thesis

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.
See additional restrictions described on inside pages

STINFO COPY

AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7320
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

REPORT DOCUMENTATION PAGE

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
April 2022	Thesis	11 April 2022	11 April 2022

4. TITLE AND SUBTITLE
SEQUENTIAL LEARNING USING IMAGE-BASED CLASSIFICATION

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
N/A	N/A	N/A
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**
N/A	N/A	N/A

6. AUTHOR(S)
Kevin Lee

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Ohio State University
281 W. Lane Ave.
Columbus, OH 43210

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'SREPORT NUMBER(S)
Air Force Research Laboratory, Sensors Directorate Wright-Patterson Air Force Base, OH 45433-7320 Air Force Materiel Command, United States Air Forces	AFRL/RYWE	AFRL-RY-WP-TP-2022-0140

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
PAO case number AFRL-2022-1676, Clearance Date 11 April 2022. This is an Electrical and Computer Engineering Masters project report. This work was funded in whole or in part by Department of the Air Force. The U.S. Government has for itself and others acting on its behalf an unlimited, paid-up, nonexclusive, irrevocable worldwide license to use, modify, reproduce, release, perform, display, or disclose the work by or on behalf of the U. S. Government. Report contains color.

14. ABSTRACT
Machine learning for time series, or sequential learning, has been a growing field due to interests in medicine, weather, stocks, and more. We present an image-based scenario classification solution to a dataset with highly nonuniformly sampled data. Scenario data are obtained through software and are used as a starting point for data processing. First, we fill in data using a number of samples determined from averaging adjacent sampling rates of groups of data, we call this data the “dead zone.” Next, groupings of output data or dead zones are then given a temporal encoding, denoting dead zones with zeros and output data with a linear encoding. Finally, we transform the scenario by feature into 2D channels of a full image using signal processing techniques such as the Constant Q-Transform (CQT).

15. SUBJECT TERMS
time series classification

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
Unclassified	Unclassified	Unclassified	SAR	17

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Christopher Ebersole	

PREVIOUS EDITION IS OBSOLETE.

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

THE OHIO STATE UNIVERSITY

Sequential Learning Using Image-based Classification

Electrical and Computer Engineering: Masters Project Report

Kevin Lee

Submitted to:
Professor Lisa Fiorentini
Professor Philip Schniter

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

Table of Contents

Abstract **3**

Introduction **4**

Data 4

Methods **5**

Data Processing 5

Time encoding 6

Time Series Image Processing 6

TSNE and UMAP Data Visualization 7

Model Architecture Design 8

Hyperparameter Tuning 9

Training 9

Results **10**

Data visualization using t-SNE and UMAP 10

Optuna trial results 11

Manually optimized model results 12

Conclusion **13**

References **14**

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

Abstract

Machine learning for time series, or sequential learning, has been a growing field due to interests in medicine, weather, stocks, and more. We present an image-based scenario classification solution to a dataset with highly nonuniformly sampled data. Scenario data are obtained through software and are used as a starting point for data processing. First, we fill in data using a number of samples determined from averaging adjacent sampling rates of groups of data, we call this data the “dead zone.” Next, groupings of output data or dead zones are then given a temporal encoding, denoting dead zones with zeros and output data with a linear encoding. Finally, we transform the scenario by feature into 2D channels of a full image using signal processing techniques such as the Constant Q-Transform (CQT).

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

Introduction

Sequential Learning is a topic that deals with using machine learning (ML) algorithms to learn features of a time series dataset. In this supervised learning classification problem, we are tasked to classify sequences of samples from a scenario with class labels 0, 1 or 2.

Time series is much more abstract in nature compared to images for machine learning. This project aims to use image classification concepts on time series data by transforming the data into images using signal processing techniques.

Data

In this problem, we are given data from a software that can be processed using our own crafted methods. The data consists of unique scenarios over a time frame of 20 seconds made by inputting different seed values. For one scenario there are a total of 3 different classes all with 4 outputs, making a total of 3x4=12 total outputs for one scenario.

The training dataset consists of 320 scenarios with 3 different classes all with 4 unique data outputs, for a total of 320x3x4=3840 unique outputs. Both the validation and test datasets consist of 32 scenarios for a total of 32x3x4=384 unique outputs. The training, validation, and test sets all are made of different seeds, where seeds were carefully chosen so that there would be no redundant data. This is to also ensure that the designed classifier can have the most unbiased results as possible.

The data itself poses challenges such as very nonuniformly sampled data points and itself is a small dataset. However, we can show that it is still possible to address this problem and successfully classify with great accuracy through signal processing techniques. Much of the relevant literature only addresses time series machine learning tasks for uniform temporally sampled data such as stocks and ECG signals. Thus, much of the models from these literature are tuned for uniformly sampled data.

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

Methods

Data Processing

The original software outputs data that can be parsed in Python and converted into a Pandas DataFrame object and then a Numpy array for further processing. Important features we considered were features A, B, C, and W. Another important feature is a Coherent Processing Interval (CPI) index that denotes which group of samples belong to a selection of time. For example, the first example is the first grouping of samples in orange corresponds to CPI 1, the next grouping of samples in orange corresponds to CPI 2.

Many machine learning models require standardized features, so each feature is standardized to Gaussian distribution (zero mean and unit variance). A mean and standard deviation is computed for each column (feature) for this calculation.

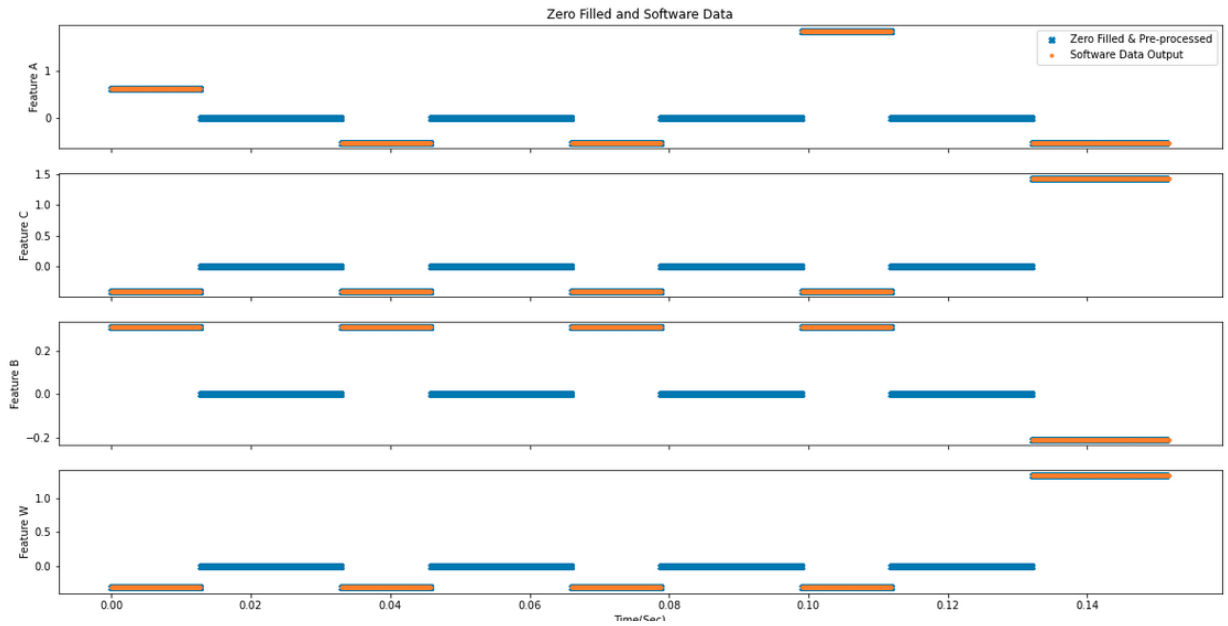

Figure 1: Software output (orange) and zero-filled data (blue)

Data from the STRiDESim output is not uniform (Figure 1). If this data were input into a neural network, the network would have a much harder time capturing the temporal representation that the data has compared to plotting the samples on a figure. To make it more uniformly sampled in time, zero data is filled in between each CPI (each separated orange plot). This allows the data to capture the temporal representation of each CPI while giving meaning to the "dead zones" in between each CPI. The time sampling of each PDW in the dead zones are calculated by determining the average sampling rate of the adjacent CPIs. Samples near the beginning of each scenario have a much greater sampling rate than that of the samples near the end of each scenario.

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

Time encoding

Encoding words in a sentence is a common method in training deep learning text models. Similarly for time series, we can give the model another feature to help the model learn differences between samples and artificially added samples of the "dead zones." A time encoding algorithm is used to generate a 1D array that can be appended as a column onto the scenario data array.

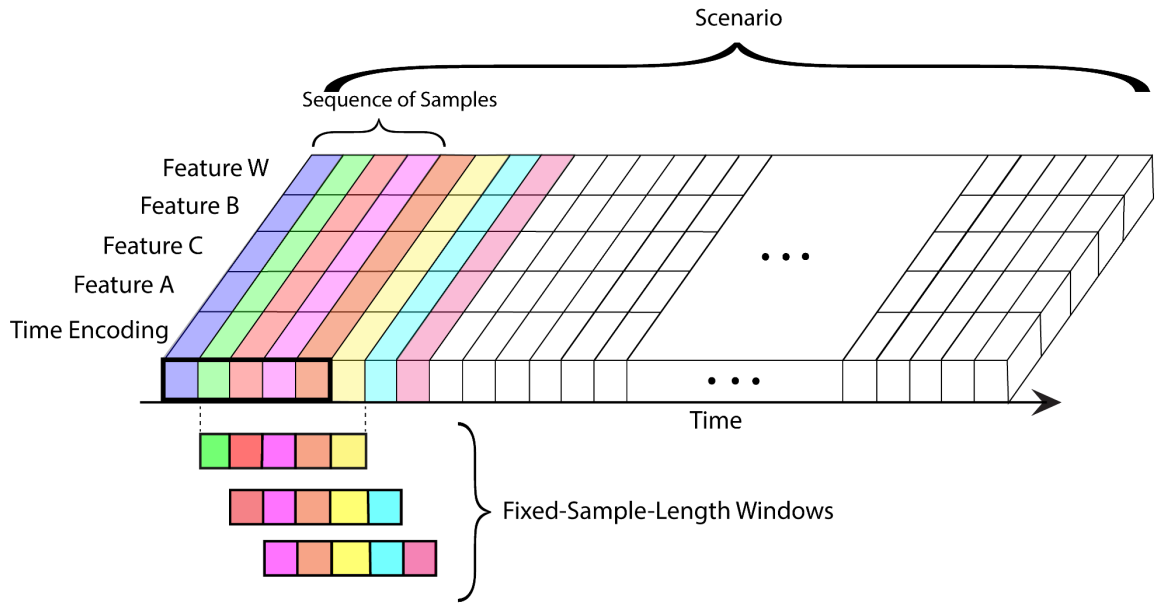

Figure 2: Windowing method for generating inputs using the data array

Time Series Image Processing

One method for creating inputs for a machine learning model is to use a sequence of PDWs (Figure 2). This is analogous to using a sentence of words as an input to a text task. A different approach is to take an entire scenario sequence and to transform it into an "image" using transforms. Each feature is a row vector, or 1D signal, that is transformed into a 2D-array as one channel of a 5-channeled "image" using Librosa's Chroma Constant Q-Transform (chroma_cqt) where each channel represents amplitude, carrier frequency, bandwidth, pulse width, and time encoding. A Chroma Q-Transformed image has a y-axis pitch class representing musical pitches and a x-axis of time (Figure 3).

Transforming an entire scenario into an image gives the advantage of a model to look at everything at once rather than using fixed-sample length sliding windows that only gives a very small fraction of the scenario at a time. Transforming the scenario into an image still uses the

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

same concepts of the sliding window method in the transformation, a step in the Short Time Fourier Transform (STFT) and Constant Q-Transform (CQT).

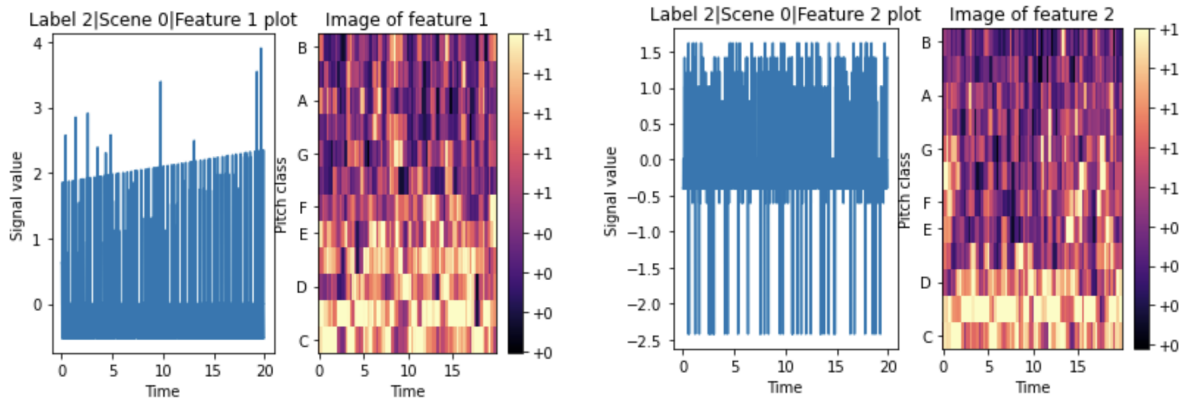

Figure 3: Left: One feature (Feature A) of a scenario and its chroma-cqt time series image. Right: One feature (Feature C) of a scenario and its chroma-cqt time series image.

The resulting 5-channel image contains entries that are floating point precision and are not of the standard 0-255 scale integer pixel value found in images. Each channel is in dB value after the chroma_cqt. To transform the channel into 0-255 scale integers, the following algorithm was used for a given channel array X.

Standardizing algorithm:

1. X' = X - minimum(X) // X is in [0, max(X)]
2. X'' = X' / maximum(X') * 255 // X is in [0, 255] floating point precision
3. Round X''(i,j) to nearest integer // X is in [0, 255] integer precision

Each 5-channel image is then given a label 0,1 or 2 for emitter level low, medium or high, respectively. Each 5-channel image and label are appended onto a features and labels list, respectively, and saved to a PyTorch DataLoader.

TSNE and UMAP Data Visualization

T-distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation Projection (UMAP) tools were used to determine whether the data was possible to be separated and classified by a deep learning design. Unlike traditional image classification tasks of everyday objects such as animals, vehicles, or food, it can be hard for a human to determine which image belongs to a certain class due to the abstract nature of each image. Using t-SNE and UMAP allowed for a preliminary proof-of-concept that it is possible for a deep learning algorithm to classify the images with reasonable accuracy.

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

Model Architecture Design

The model architecture is inspired by preliminary testing using VGG-16. It was later found that VGG-16 did not have a suitable architecture for accurately classifying the time series images, but it provided insights on how to design a suitable convolutional neural network classifier. This is because VGG-16 is designed for RGB images that are 3-channeled, but the model architecture is too large and not necessary for our small dataset.

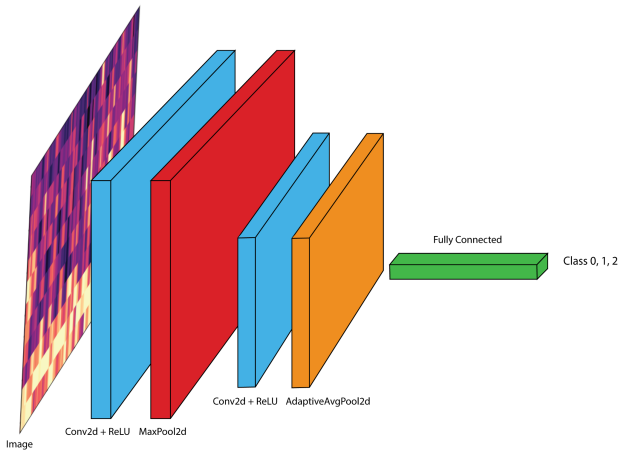

Figure 4: Visualization of the convolutional neural network classifier

```
Sequential(
  (0): Conv2d(5, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU()
  (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (3): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (4): ReLU()
  (5): AdaptiveAvgPool2d(output_size=(2, 2))
  (6): Flatten(start_dim=1, end_dim=-1)
  (7): Linear(in_features=256, out_features=3, bias=True)
)
```

Figure 5: Description of each layer of the network made from PyTorch

The first layer is a 2D-Convolution (Conv2d) of 5 input channels and 128 output channels with a 3x3 kernel and stride of 1 followed by a ReLU activation and max pooling (MaxPool2d). The next Conv2d has a 128 channel input and 64 channel output followed by a ReLU activation and an Adaptive Average Pooling (AdaptiveAvgPool2d) with a 2x2 output size followed by a flattening layer and a linear output layer with 64*2*2=256 inputs and 3 outputs for the number of classes. A flatten layer is used to transition the Conv2D and average pooling to a fully connected linear layer.

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

Hyperparameter Tuning

Optuna was used to optimize the model using the best found hyperparameters from a search space using multiple trials. The first Optuna experiment used a model of 4 Conv2d and a linear layer for 200 trials. The parameter search space was for channel sizes for each Conv2d layer within a search space of [4,512]; AdaptiveAvgPool2d kernel size in a search space of [3,9]; Optimizer type with options Adam, RMSprop, and SGD; and an initial learning rate value in the range of [1e-5,1e-1]. The second Optuna experiment was the same as the first, however the model used only 2 Conv2d where all other search parameters and settings were the same.

Training

The optimized model was trained for 50 epochs for 43.56 seconds on a machine with an Intel Xeon CPU at 3.60GHz and Nvidia Quadro RTX 5000 GPU. The loss function used was cross entropy because the problem is a multiclass classification. The optimizer used was Adaptive Moment Estimation (ADAM). The learning rate was reduced by 0.3 if there was no improvement in the 5 most recent epochs.

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

Results

Data visualization using t-SNE and UMAP

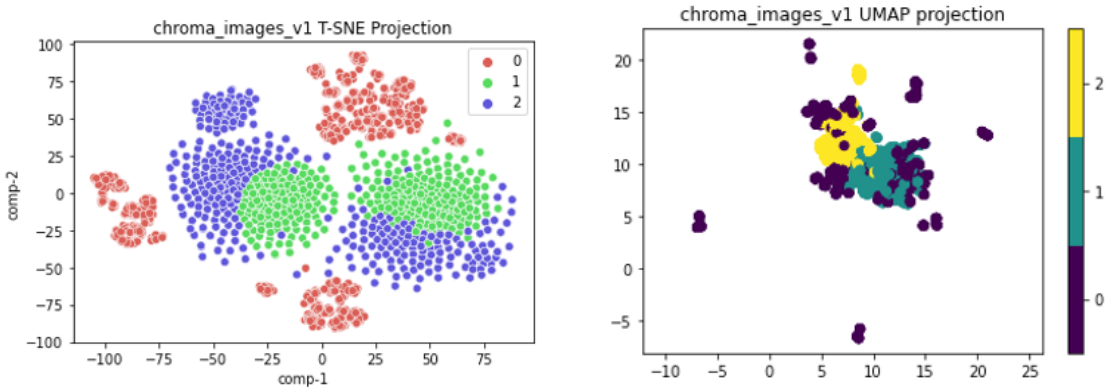

Figure 6: 2D t-SNE Projection (left) and 2D UMAP projection (right) of the Chroma-CQT time series images

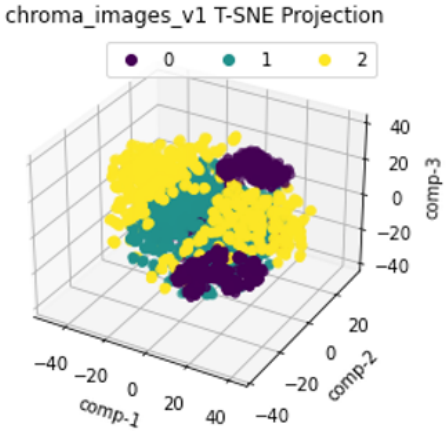

Figure 7: 3D UMAP Projection of the Chroma-CQT time series images

From the t-SNE and UMAP projection data visualization. We gathered that it would be possible for a successful image classification using neural network designs. For both the 2D projections, it can be observed that classes 0, 1 and 2 can be visually separated by drawing a nonlinear boundary between data clusters. Similarly, the 3D UMAP projection can also be separated by drawing a nonlinear boundary between clusters of data.

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

Optuna trial results

```
Study statistics:
  Number of finished trials:  500
  Number of pruned trials:  456
  Number of complete trials:  44
Best trial:
  Value:  100.0
  Params:
    chan1_nodes: 177
    chan2_nodes: 156
    chan3_nodes: 357
    AdpAvgPool2d_kernel_size: 9
    optimizer: Adam
    lr: 0.0006188456306000411
time elapsed: 3013.655634697003 seconds
```

Figure 8: Optuna hyperparameter tuning summary

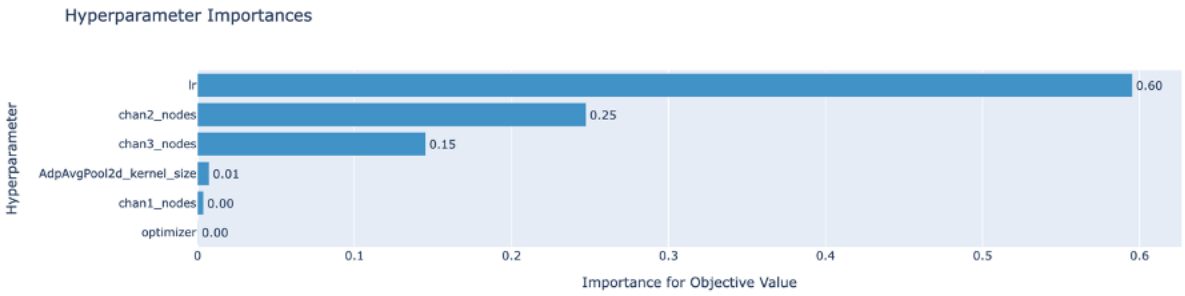

Figure 9: Hyperparameter importances

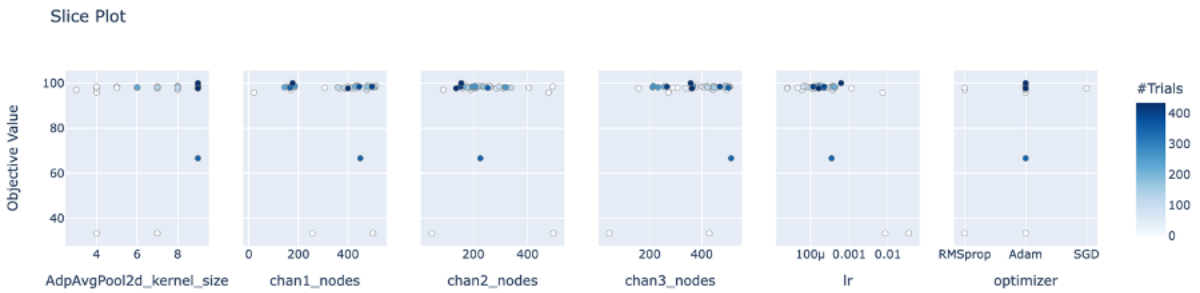

Figure 10: Objective value slice plot vs. each hyperparameter studied

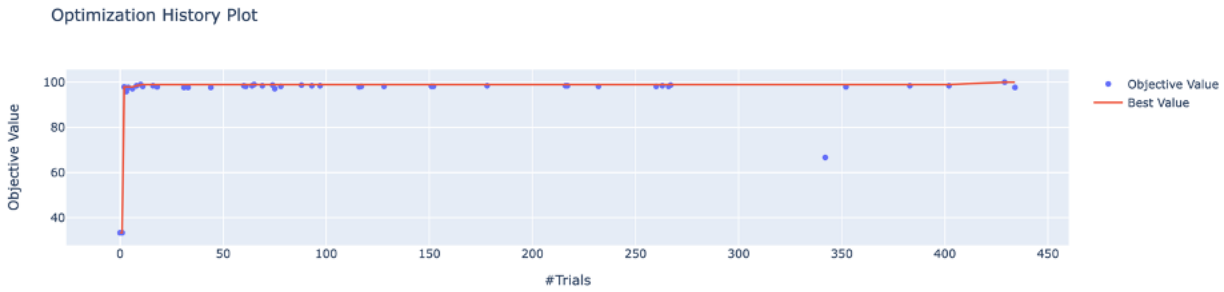

Figure 11: Objective value of the study over total number of trials

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

Optuna optimized model results

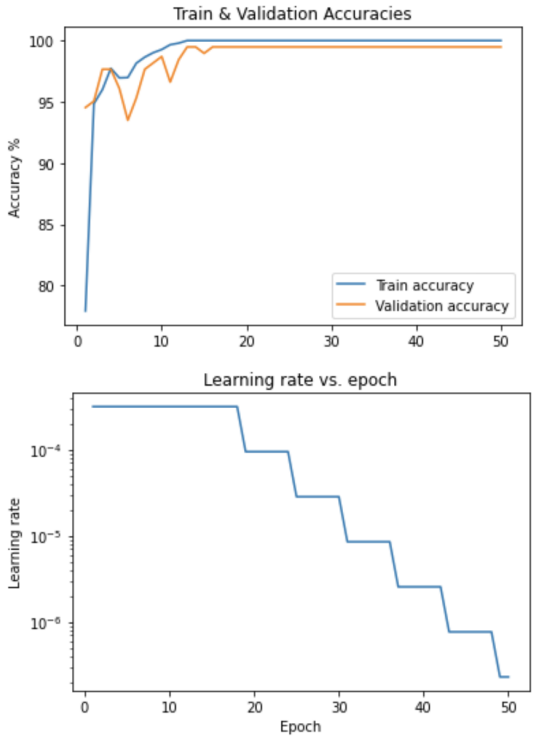

Figure 12: Training and validation accuracies over 50 epochs (Top). Learning rate (log scale) vs epoch (Bottom) of the Optuna optimized model.

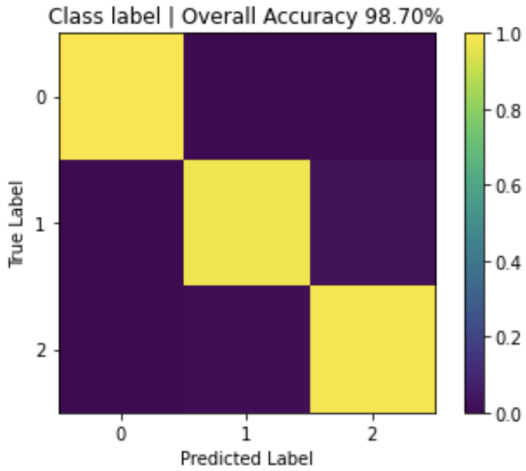

Figure 13: Confusion matrix of the test accuracy of the test set of the Optuna optimized model

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

Manually optimized model results

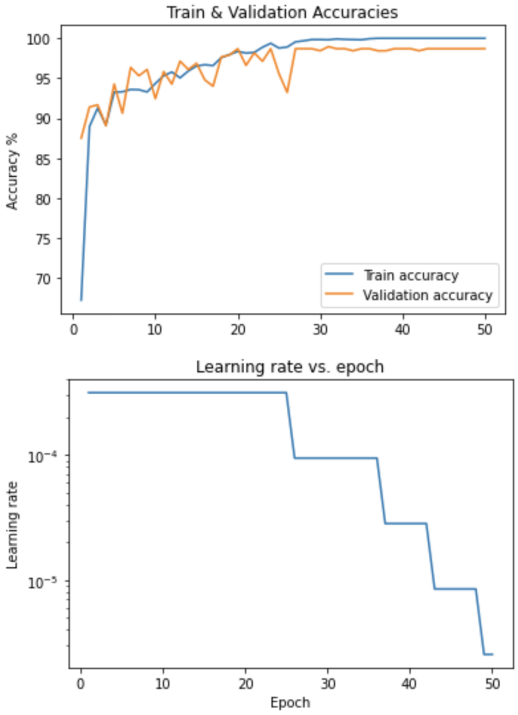

Figure 14: Training and validation accuracies over 50 epochs (Top). Learning rate (log scale) vs epoch (Bottom) of the manually optimized model.

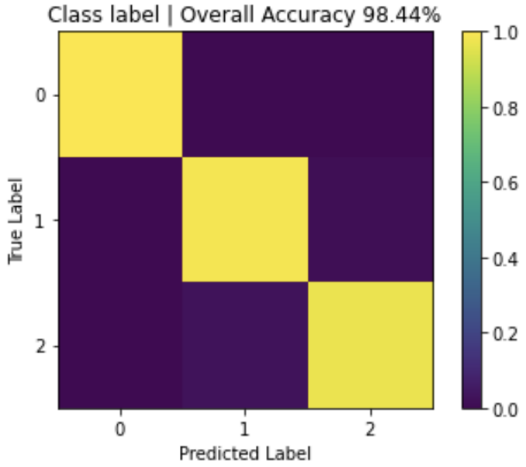

Figure 13: Confusion matrix of the test accuracy of the test set of the manually optimized model

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

Conclusion

The Optuna study yielded a 100% accuracy for a set of determined hyperparameters, but in practice the classifier only yielded 98.70% accuracy. This is likely due to the model overfitting during the trial. The optimized model parameters were found from multiple trials and errors by manually adjusting the network. While Optuna can provide promising results for optimized hyperparameters, short studies such as the one used are not significantly better than taking the time to manually adjust hyperparameters (98.44%). It may prove to yield better results on studies that have much more trials and much longer trials.

We have successfully classified a dataset of very nonuniformly sampled data by filling in gaps of data with zeros, adding a time encoding feature, and transforming each scenario data into a 5-channeled image for a custom designed convolutional neural network. While our work deals with a specific dataset with very nonuniformly sampled data, we conjecture that our method can be extended to other nonuniformly sampled data for similar results.

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

References

1. https://pytorch.org/
2. https://optuna.org/
3. https://librosa.org/doc/latest/index.html#
4. https://librosa.org/doc/main/generated/librosa.feature.chroma_cqt.html
5. https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html
6. https://umap-learn.readthedocs.io/en/latest/
7. http://doc.ml.tu-berlin.de/bbci/material/publications/Bla_constQ.pdf

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.